Multi-Mission Simulation and Visualization for Real-time Telemetry Display, Playback and EDL Event Reconstruction

AIAA Space 2012
Pasadena CA

Marc Pomerantz,
Christopher Lim, Steven Myint, Gail Woodward,
Bob Balaram, Calvin Kuo

Mobility and Robotic Systems
Robotics Modeling, Simulation and Visualization Group
Jet Propulsion Laboratory,
California Institute of Technology,
Pasadena, CA, 91109

Contact: Marc.I.Pomerantz@jpl.nasa.gov
The research in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.
Outline

Goals
Background
Development
Usage
Movie
Questions
Goals

To develop a physics-based, interactive, multi-mission, 3D visualization and simulation system to be used by mission Entry, Descent and Landing (EDL) engineers for:

- Analysis prior to EDL through display of simulation results
- Real-time telemetry display during EDL by processing and displaying actual mission telemetry
- Post-EDL playback and reconstruction through display of actual and simulated flight performance data and combined with mission imagery and/or other products
- Currently supporting NASA’s MSL EDL Reconstruction effort and NASA’s Low Density Supersonic Decelerator project
Background

Our EDL Reconstruction and Visualization software was built on top of our existing JPL DSENDS EDL simulation framework and our Dspace visualization framework and can support a variety of space mission domains:

- Orbiters
- Landers
- Rovers
- EDL
- Telemetry visualization
- Multiple vehicles
- NEO scenarios
Background – Supported Domains

Roving Vehicle Simulations
Athlete, MSL, LSOS, MER and ROAMS

Orbiter Simulations
MoonRise, Europa Orbiter and Small Body analysis

Lander Simulations
MoonRise, LSOS and Aerobot

NASA Desert Rats Field Trial Support
Background – Supported Domains

Telemetry Visualization

Camera Modeling

GPU-based Engineering Analysis
Background – DSEND5 Simulation Framework

- **DSEND5** is our physics-based, full 6 degree of freedom, space mission simulation framework
  - C++ core with Python wrapped API
  - Support for multiple, articulated spacecraft
    - High fidelity models for thrusters, ACS, RCS, imaging cameras, etc.
    - Third-party models and flight hardware/software in-the-loop
    - Model environmental forces
      - Gravity
      - Atmosphere
  - Monte Carlo simulation support
  - Used by NASA’s MSL and Phoenix missions to help plan vehicle flight paths and to refine landing regions
Background – Dspace Visualization

- **Dspace** is our game engine-based multi-mission visualization system
  - Built on top of the Ogre3D open source game engine library
  - C++ core with Python wrapped API
  - Maintain kinematic accuracy with simulation
    - At run-time, the simulation constructs the visualization “backbone” that describes the spacecraft mechanical system and environment
  - Supports planetary scale and terrestrial terrain/city-scape visualizations
  - Use Ogre Material scripts and GLSL shaders
    - Continuous level of detail terrain rendering
    - Very large texture rendering
    - Real time vehicle shadows
    - Vehicle wheel tracks

Three billion triangle MOLA data set rendered in real time using shader-based clip mapping
System Development – MSL Example

The MSL EDL Telemetry Visualization system (ETV) was developed to support landing night operations and the post landing night flight path reconstruction task

- MSL specific, thin Python software layer built on top of our multi-mission framework using the available Python API
  - Simulation Setup and Execution
    - Spacecraft and planetary body reference frames
    - Telemetry processing pipeline
    - Mission provided base time epoch
    - NASA NAIF Spice kernels for orbiters and planetary bodies
  - Visualization Setup and Execution
    - Load graphical representations of spacecraft and planetary bodies
    - Create viewports, viewing cameras, lighting and all other graphical items
    - Create UI elements

ETV High Level Software Architecture
System Development – MSL Example

Landing night telemetry visualization was performed by processing MSL’s channelized telemetry using our DSEND$S$ Python state machine module. Data flow, simulation execution and visualization views were controlled based on pre-defined state changes.

- Streamed real time data products included
  - Spacecraft position, velocity, orientation, fuel used and flight software states
  - Data rates ranged from .5hz to 4hz for the various products
  - Motion data with respect to Mars centered inertial frame or Mars fixed surface frame
- Processing
  - Time interleaved telemetry was filtered to ensure that simulation time only moved forward
  - Spacecraft motion was displayed based on received data
  - Data dropouts were expected
  - No motion data extrapolation was performed (on mission request). Only actual data was displayed
  - Flight software state data was higher priority than motion data
System Development – MSL Example

More about telemetry reference frames…

• Spacecraft telemetry for motion data arrived in two flavors:
  • With respect to Mars centered inertial frame
    • From start of EDL phase to just after initial ground radar returns (after heat shield separation)
    • Base time (epoch) provided by mission
  • With respect to Mars fixed surface frame (MSF)
    • Constructed on-board the spacecraft using initial ground radar returns and all subsequent motion data with respect to MSF
    • Frame location not transmitted in the channelized telemetry stream
    • Simulation constructed an estimated MSF based on spacecraft position, velocity and flight software mode changes
• Spacecraft graphical representations positioned and rotated based on mission provided reference frames
  • Graphical state of the spacecraft changed based on flight software mode changes
System Usage – MSL Landing Night Data Flow

Telemetry From MSL

UHF Relay
Mars Odyssey

NASA’s Deep Space Network

JPL

EDL Telemetry Visualization System
Questions?