Open-SESSAME Framework

Open-Source, Extensible Spacecraft Simulation And Modeling Environment

A tool for use by engineers and students

Introduction

- Purpose of presentation

  Introduce the user to the concepts, design, and usage of the Open-SESSAME Framework.

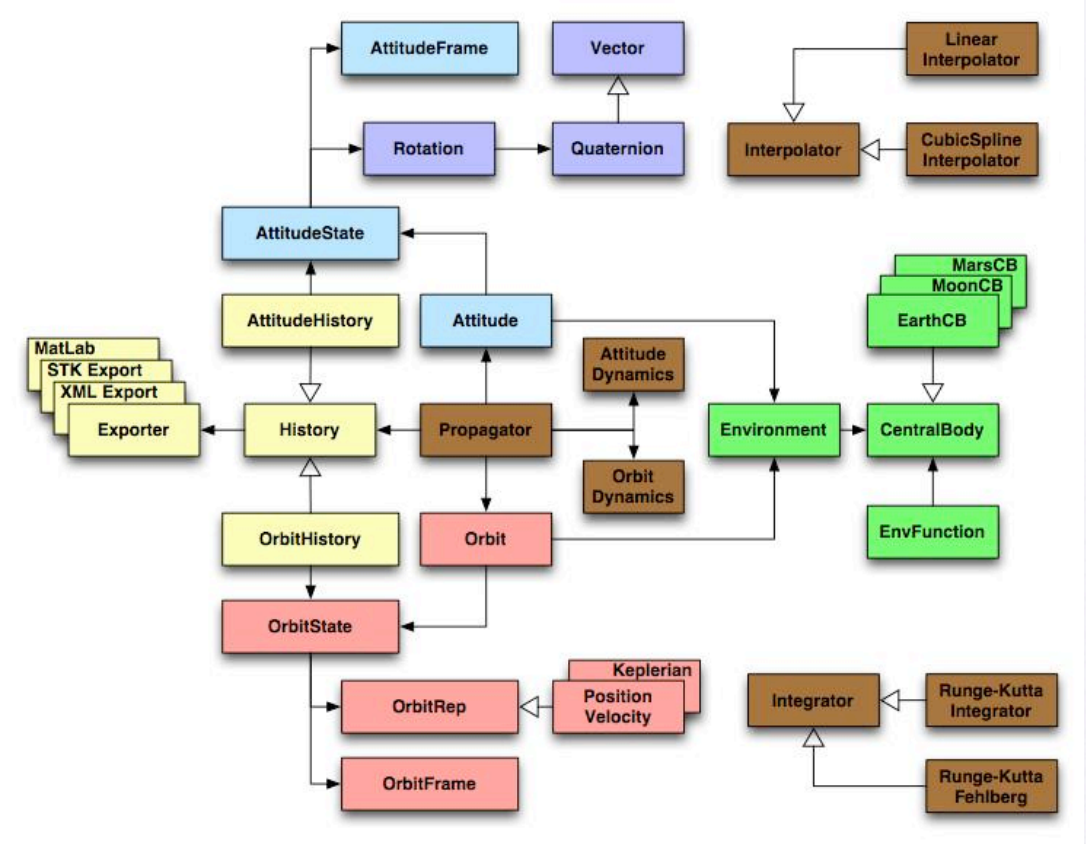
- Outline
  - Goals
  - Framework Architecture
  - Using the Framework
  - Coding Refresher
  - Rotation Kinematics Example
  - Attitude Integration Example
  - Suggested Reading

Goals of Open-SESSAME

- Provide a framework for developing simulation applications
- Includes the tools necessary for analyzing spacecraft and orbits
- Extensible by the user for specific problem domains

Framework Architecture

- Each block is a module or "class"
- Solid arrows denote "*has-a*" relationship (composition)
- Open arrows denote "*is-a*" relationship (inheritance)



Ex: **Rotation** *has-a* **Quaternion**,
which *is-a* **Vector**

Using the Framework

- Because Open-SESSAME is a framework, it is not a stand-alone application, but a collection of modules that can be combined to build an application
- The problem domain is defined by the user, and functionality is added as necessary
- Missing functionality can be added by implementing and attaching to an *extension point*

Coding Refresher (C++)

- Functions: void myFunction(const double& _inVar);
  - returns a *void*
  - takes a *double* as a parameter
  - *const ...&* specifies that is a constant reference. This acts like a normal variable, but because it is passing just a reference and not a copy, it is faster at runtime. The "_" is just a coding name convention.
- Classes
  - Have public *Member Functions* which perform operations on private *Data Members*. Each time there is an object of a class created, it is an *instance*.
  - Act as objects, which have data and can be operated on: A *Car* class has a *MyAudiTT* instance, which has member functions *GetMileage()* and *SetSpeed(const double& _newSpeed)*, with data members *DistanceType m_Mileage* and *double m_SpeedMPH*.
- Call-back functions
  - Setting a pointer to a function to be evaluated later
  - Used for specifying disturbance force functions and dynamics equations
  - Type created that acts as an object: MyAudiTT.*SetSpeedFunction(&SetSpeed)*

Rotation Kinematics

- Attitude kinematics have several common representations: Direction Cosine Matrix, Euler Axis & Angle, Quaternion, Modified Rodriguez Parameters, and Euler Angles.
- A coordinate frame transformation is represented as an abstract **Rotation**, which can be any type of common representation.

```
Quaternion myQuat(0,0.1,0.2,-0.3);
Rotation myRot1(myQuat);
cout << myRot1.GetDCM();
```

Attitude Integration

- Setup the initial **AttitudeState**

```
AttitudeState myAttitudeState;
myAttitudeState.SetRotation(Rotation(Quaternion(0,0,0,1)));
Vector initAngVelVector(3); initAngVelVector(1) = 0.1;
myAttitudeState.SetAngularVelocity(initAngVelVector);
```

- Setup the **Integrator**

```
RungeKuttaIntegrator myIntegrator;
myIntegrator.SetNumSteps(1000);
vector<ssfTime> integrationTimes;
ssfTime begin(0);   ssfTime end(begin + 20);
integrationTimes.push_back(begin); integrationTimes.push_back(end);
```

- Integrate (see testAttitudeIntegration.cpp for full example)

```
Matrix history = myIntegrator.Integrate(
                          integrationTimes,            // seconds
                          &AttituteDynamics,
                          myAttitudeState.GetState(),
                          nonOrbit, nonAttitude,
                          Parameters,
                          AttitudeForcesFunctor );
```

Suggested Reading

- Spacecraft Dynamics
  - David A. Vallado, *Fundamentals of Astrodynamics and Applications*. McGraw-Hill, New York, NY, 1997.
  - James R. Wertz (ed.), *Spacecraft Attitude Determination and Control*. Reidel Publishing, Hingham, MA, 1978.
- Programming
  - J.P. Cohoon, J.W. Davidson. *C++ Program Design: An Introduction to Programming and Object-Oriented Design*. McGraw-Hill, Boston, MA, 2nd Edition, 1999.
  - B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, Boston, MA, 3rd Edition, 1997.