Propagator

De Wiki
Aller à : navigation, rechercher
Propagator

Sommaire

Goal

In this page, we give a relatively concrete example allowing to create an application with its own GUI (using of couse GENIUS and **GENOPUS**!) to propagate an orbit using PATRIUS.

Nevertheless, in order to simplify the example (else you may directly use PSIMU!), we will "only" consider as inputs:

- **Initial orbital parameters**
- **Vehicle characteristics** (only dry mass and simple aerodynamic characteristics)
- Choice of **force models**
  - Only Balmino for Earth potential
  - Atmospheric models

In addition:

- Data will be stored in **INI_suffix.xml** files (or **INIT.xml** by default)
- Some results will be displayed on the GUI console but also in a **EPH_suffix.txt** file (**EPHEM.txt** by default)

Using GDataPanelAbstract to organize input data

To do it, we will have to create a class extending from GDataPanelAbstract and adding GPOrbit, GPVehicle and GPForceModels objects in different tabs.

Note : be careful as GPOrbit extends from Gcontainer and not from GPanel, so it is mansatory to encapsulate it in a GPanel class (privalte class in our example.

```
public class WidPropagatorDataPanel extends GDataPanelAbstract {

    private final WidOrbit widOrbit;
    private final GPVehicle widVehicle;
    private final GPForceModels widForces;
```

```
    /**
     * Constructor.
     * @throws GException   GENIUS exception
     */
    public WidPropagatorDataPanel() throws GException {

        super("Data");

        // Creating an orbit widget
        widOrbit =  new WidOrbit("Initial orbital parameters");

        // Creating a vehicle widget (only with dry mass and simple
aerodynamic properties)
        widVehicle =  new GPVehicle("Vehicle characteristics", true, false,
true, false, false);

        // Creating a force model widget (only with potential [Balmino] and
atmosphere)
        AttractionModelsEnum[] attractionModelsAvailable = {
AttractionModelsEnum.BALMINO };
        widForces =  new GPForceModels("Models",
AttractionModelsEnum.BALMINO, attractionModelsAvailable, false, true, false,
false, false, false);

        // Adding it in a tabbedpane
        this.addTab("Orbit", widOrbit);
        this.addTab("Vehicle", widVehicle);
        this.addTab("Forces", widForces);
        // Adding an output console
        this.addConsoleTab("Console");

    }

    public void clear() throws GException {
        // TODO Auto-generated method stub
    }

    /**
    * Private class needed because GPOrbit is not a GPanel ...
    */
    private class WidOrbit extends GPanel implements GReadWrite {

        private final GPOrbit orbit;
        public WidOrbit ( final String label) {
            orbit = new GPOrbit(label);
        }
        public void display() throws GException { generic(); }
        @Override
        public void generic() throws GException { put(orbit); }
        public void read() throws GException { generic(); }
        public void write() throws GException { generic(); }
```

```
        /**
         * @return the GP orbit widget
         */
        public GPOrbit getGPOrbit() {
            return orbit;
        }

    }

    /**
     * Class to get the PATRIUS orbit object
     * @return PATRIUS orbit object
     * @throws GPOrbitException GENOPUS exception
     */
    public Orbit getOrbit() throws GPOrbitException {
        return widOrbit.getGPOrbit().getPatriusObject();
    }

    /**
     * Class to get the Custom PATRIUS vehicle object
     * @return the widVehicle
     * @throws GPVehicleException GENOPUS exception
     */
    public CustomVehicle getVehicle() throws GPVehicleException {
        return widVehicle.getPatriusObject();
    }

    /**
     * Class to get the Custom PATRIUS force models object
     * @return Custom PATRIUS force models object
     * @throws GException GENIUS exception
     */
    public CustomForceModels getForces() throws GException {
        final Assembly assembly =
getVehicle().getAssembly(FramesFactory.getCIRF());
        return widForces.getPatriusObject(assembly);
    }

}
```

Create the batch computation mode

The basic idea is to develop a class allowing to propagate an orbit and that will be called using the GJavaCommandLauncher class. This class will include:

- reading data in a file (a XML one automatically generated thanks to GENIUS)
- using these data and considering other ones not exposed via the GUI (in order to simplify the example) to build the numerical propagator:
  - Numerical integrator => RungeKutta 4th order with a 5s step
  - Propagation time => 1 orbit

    - StepHandler = 60s
- propagating the orbit
- writing results on the screen and inside the **EPHEM.txt** file.

We may see on the following code inside the compute() method how the PATRIUS propagator is implemented as well as the associated stephandler in a specific private class.

```
public class BatchPropagator {

    /** Initial orbit */
    private Orbit orbit;
    /** Vehicle characteristics */
    private CustomVehicle vehicle;
    /** Forces */
    private CustomForceModels forces;

    /** UTC time scale */
    private TimeScale UTC;

    /** By default EPHEM file names */
    private static final String EPH_FILE = "EPHEM.txt";

    /**
   * Constructor.
   * @param nomFicData    name of the context file.
   * @param nomFicEphem   name of the output file.
   * @throws IOException  GENIUS exception.
     */
    public BatchPropagator (final String nomFicData, final String
nomFicEphem) throws IOException {

        // Patrius dataset initialization
        try {
            PatriusDataset.addResourcesFromPatriusDataset();
        } catch (OrekitException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }

        WidPropagatorDataPanel dataPan = null;
        try {
            dataPan = new WidPropagatorDataPanel();
        } catch (GException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }
        // Data read inside the XML file
        try {
            GFileManipulation.readConfig(nomFicData, "Propagator", dataPan,
true);
        } catch (GFileManipulatorException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }
```

```
        // Orbit initialization
        try {
            orbit = dataPan.getOrbit();
        } catch (GPOrbitException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }
        // Vehicle initialization
        try {
            vehicle = dataPan.getVehicle();
        } catch (GPVehicleException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }
        // Forces initialization
        try {
            forces = dataPan.getForces();
        } catch (GException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }

        // Recovery of the UTC time scale using a "factory"
        try {
            UTC = TimeScalesFactory.getUTC();
        } catch (OrekitException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }

    }

    /**
     * Method to propagate
     * @throws PropagationException PATRIUS exception
     */
    public void compute() throws PropagationException {

        // Getting the mas provider from the vehicle object.
        final MassProvider mm = new
MassModel(vehicle.getAssembly(FramesFactory.getCIRF()));

        // We create a spacecratftstate
        final SpacecraftState iniState = new SpacecraftState(orbit, mm);

        // Initialization of the Runge Kutta integrator with a 5 s step
        final double pasRk = 5.;
        final FirstOrderIntegrator integrator = new
ClassicalRungeKuttaIntegrator(pasRk);

        // Initialization of the propagator
        final NumericalPropagator propagator = new
NumericalPropagator(integrator);
        propagator.resetInitialState(iniState);
```

```
        // Adding additional state
        propagator.setMassProviderEquation(mm);

        // Forcing integration using cartesian equations
        propagator.setOrbitType(OrbitType.CARTESIAN);

        // Adding an attitude law (in case of lift component)
        final AttitudeLaw attitudeLaw = new LofOffset(LOFType.LVLH,
RotationOrder.ZYX, 0., 0., 0.);
        propagator.setAttitudeProvider(attitudeLaw);

        // Adding force models
        List<ForceModel> list = forces.getForceModelsList();
        for (ForceModel forceModel : list) {
            propagator.addForceModel(forceModel);
        }

        // Creation of a fixed step handler
        final ArrayList<SpacecraftState> listOfStates = new
ArrayList<SpacecraftState>();
        OrekitFixedStepHandler myStepHandler = new OrekitFixedStepHandler() {
            private static final long serialVersionUID = 1L;
            public void init(SpacecraftState s0, AbsoluteDate t) {
                // Nothing to do ...
            }
            public void handleStep(SpacecraftState currentState, boolean
isLast)
                    throws PropagationException {
                // Adding S/C to the list
                listOfStates.add(currentState);
            }
        };
        // The handler frequency is set to 60s
        propagator.setMasterMode(60., myStepHandler);

        // Propagating on 1 period
        final double dt = orbit.getKeplerianPeriod();
        final AbsoluteDate finalDate = orbit.getDate().shiftedBy(dt);
        final SpacecraftState finalState = propagator.propagate(finalDate);
        final Orbit finalOrbit = finalState.getOrbit();

        // Printing new date and semi major axis
        System.out.println();
        System.out.println("Initial date = "+orbit.getDate().toString(UTC));
        System.out.println("Initial semi major axis = "+orbit.getA()/1000.+"
km");
        System.out.println("New date = "+finalOrbit.getDate().toString(UTC));
        System.out.println("Final semi major axis =
"+finalOrbit.getA()/1000.+" km");

        // Writing in the EPHEM.txt file
```

```
        try {
            System.out.println();
            System.out.println("EPHEM.txt file writing ...");
            FileWriter ephem = new FileWriter(new File(EPH_FILE));
            Locale.setDefault(Locale.US);
            for (SpacecraftState sc : listOfStates) {
                ephem.write(String.format("%s %22.15e \n", sc.getDate(),
sc.getA()/1000.));
            }
            ephem.close();
        } catch (IOException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }

    }

    /**
     * Main method ...
     * @param args
     */
    public static void main(String[] args) {

        BatchPropagator batch;
        try {
            batch = new BatchPropagator("INIT.xml", "EPHEM.txt");
            batch.compute();
        } catch (IOException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        } catch (PropagationException err) {
            GConsoleLogger.getLogger().log(Level.SEVERE, err.getMessage());
        }

    }

}
```

Using GMainFrameAbstract for the main frame

Once the batch mode is available (see above), we have just to create the last third class extending from GMainFrameAbstract, displaying data and allowing to execute the batch mode ...

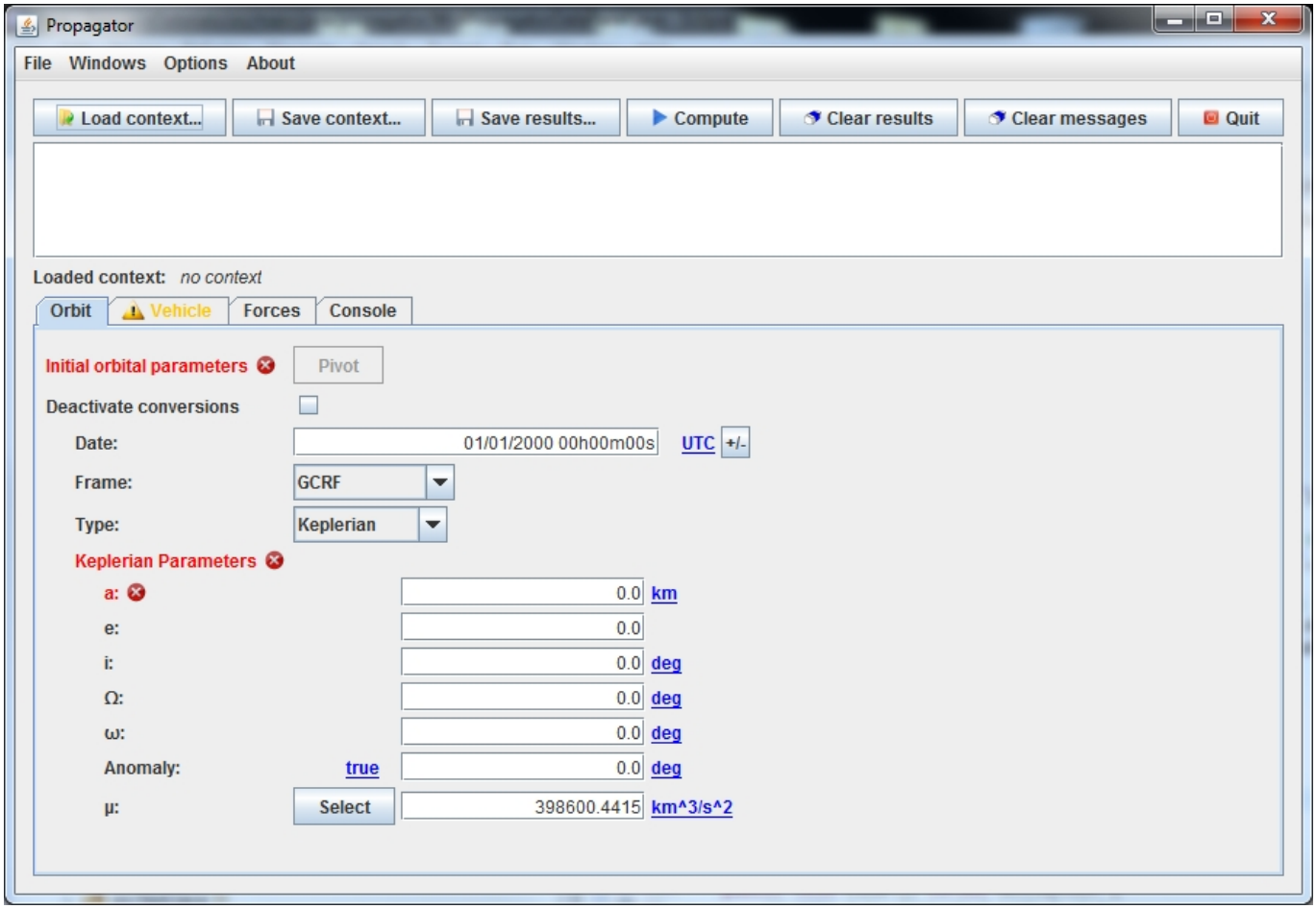
```
public class WidPropagator extends GMainFrameAbstract<WidPropagatorDataPanel>
{

    // FILE NAMES

    /** Prefix for context file names */
    private static final String INI_FILE_PREFIX = "INI_";
    /** By default context file names */
    private static final String INI_FILE = "INIT.xml";
```

```
    /** Prefix for output file names */
    private static final String EPH_FILE_PREFIX = "EPH_";
    /** By default EPHEM file names */
    private static final String EPH_FILE = "EPHEM.txt";

    // SIZES

    /** Data panel height */
    private static final int DATAPANEL_HEIGHT = 400;
    /** Error console height */
    private static final int ERRCONSOLE_HEIGHT = 80;
    /** Icon size */
    private static  final int ICON_SIZE = 12;

    /**
     * Constructor.
     * @throws GException   GENIUS exception
     */
    public WidPropagator() throws GException {

    super("Propagator",
          new WidPropagatorDataPanel(),
          new GContextFileManagement(".", "Propagator data", new
GFileFilter(INI_FILE_PREFIX, ".xml", "Propagator Files") ),
          new GAboutDialog("About S/W", "Propagator using PATRIUS ...",
"CNES", "V1.0 ; 20/12/2017", null),
          new GSaveResults("Saving results", new File("."), ".txt", ".xml"),
          DATAPANEL_HEIGHT, ERRCONSOLE_HEIGHT, ICON_SIZE, false);

    }

    /**
     * Method to manage saving data and results files
     */
    @Override
    protected void saveFilesManagement() throws GException {

        final File ini = new File(INI_FILE);
        final File res = new File(EPH_FILE);

        // The context file INIT.xml will be saved in INI_....xml
        this.getSaveResultsDialog().setContextFile(ini.getName(),
INI_FILE_PREFIX, true);
        // Result files consist in a single one named by default "EPHEM"
        this.getSaveResultsDialog().clearResultFileList();
        this.getSaveResultsDialog().addSingleResultFile(res.getName(),
EPH_FILE_PREFIX, true);

        this.getSaveResultsDialog().show();

    }
```

```
    /**
     * Method to manage what to do before starting the computation
     */
    @Override
    protected void customPreProcessManagement() throws
GFileManipulatorException {

        // We write a context file with data coming from the data panel
        GFileManipulation.writeConfig(INI_FILE, "Propagator",
this.getDataPanel(), true);
        // We initialize the JavaCommandLauncher
       final String classPath = System.getProperty("java.class.path");
        this.getJavaCommandLauncher().setJavaCommand(classPath, new String[]
{"propagator.BatchPropagator"});
        // We display the console above the other tabbedpanes
        this.getDataPanel().selectConsoleTab();
    }

    /**
     * Method to manage what to do after computation
     */
    @Override
    protected void customPostProcessManagement() {
        // Nothing to do ...
    }

    /**
     * Main method.
     * @param args
     * @throws GException
     */
    public static void main(String[] args) throws GException {

        // Patrius dataset initialization
        try {
            PatriusDataset.addResourcesFromPatriusDataset();
        } catch (OrekitException err) {
            err.printStackTrace();
        }

        final WidPropagator pan = new WidPropagator();
        pan.displayMainFrame();

    }
    }
```

Final result

Now you are able to load this GUI and to use it!



Récupérée de « http://genopus.cnes.fr/index.php?title=Propagator&oldid=269 »

Menu de navigation

Outils personnels

- 3.133.141.6
- Discussion avec cette adresse IP
- Créer un compte
- Se connecter

Espaces de noms

- Page
- Discussion

Variantes

Affichages

- Lire
- Voir le texte source
- Historique

- Exporter en PDF

Plus

Rechercher

Rechercher Lire

GENOPUS

- Welcome
- Quick Start
- News

User Manual

- BasicPrinciples
- GPAbsoluteDate
- GPOrbit
- GPFramesConfiguration
- GPVehicle
- GPForceModels
- GPManeuverSequence
- GPAttitudeSequence
- GPIntegrator
- GPAxisCoordinates
- GPGeodeticPoint
- GPOneAxisEllipsoid
- GPRotation
- GPConstants
- Events
- GPCorrelation

Evolutions

- Main differences between V2.4.1 and V2.4.2
- Main differences between V2.3.3 and V2.4.1
- Main differences between V2.2.1 and V2.3.3
- Main differences between V2.2 and V2.2.1
- Main differences between V2.1.1 and V2.2
- Main differences between V2.1 and V2.1.1
- Main differences between V2.0.1 and V2.1
- Main differences between V2.0 and V2.0.1
- Main differences between V1.3.1 and V2.0

- Main differences between V1.3 and V1.3.1
- Main differences between V1.2.1 and V1.3

Training

- Make your own propagator tool!
- Tutorials package for V2.4.1
- Tutorials package for V2.3.3
- Tutorials package for V2.2
- Tutorials package for V2.1.1
- Tutorials package for V2.0 and V2.0.1
- Tutorials package for V1.3 and V1.3.1
- Training slides

Links

- CNES freeware server

Outils

- Pages liées
- Suivi des pages liées
- Pages spéciales
- Adresse de cette version
- Information sur la page
- Citer cette page

- Dernière modification de cette page le 21 décembre 2017 à 12:45.

- Politique de confidentialité
- À propos de Wiki
- Avertissements