UNIVERSIDAD DE MÁLAGA

EPS
Escuela Politécnica Superior
Universidad de Málaga

TRABAJO FIN DE GRADO

CREACIÓN DE ENTORNO DE SIMULACIÓN EN INGRAVIDEZ CON INTERACCIÓN ENTRE SATÉLITE Y ROBOT

Grado en

Ingeniería en Diseño Industrial y Desarrollo del Producto

Autor: RAFAEL CERVILLA RIVAS

Tutor: CARLOS JESÚS PÉREZ DEL PULGAR MANCEBO

Cotutor -

Diciembre de 2.016

UNIVERSIDAD DE MÁLAGA

ESCUELA POLITÉCNICA SUPERIOR

EPS
Escuela Politécnica Superior
Universidad de Málaga

TRABAJO FIN DE GRADO

CREACIÓN DE ENTORNO DE SIMULACIÓN EN INGRAVIDEZ CON INTERACCIÓN ENTRE SATÉLITE Y ROBOT

AGRADECIMIENTOS

A mis padres por su apoyo incondicional, su cariño, y su devoción por mí, cuyo esfuerzo y entrega han sido imprescindibles en mi camino hasta la consecución de esta obra. Gracias a ellos estoy logrando mis sueños tanto profesionales como personales, no es fácil contar con unos padres así de implicados y devotos por las ambiciones de sus hijos, y es algo de lo cual me siento afortunado.

A mi amor por todo su afecto, su apoyo y su comprensión desde siempre; no ha sido nada fácil para ella acompañarme durante estos años; ha sido una fuerza constante que me ha ido empujando a cada paso que he dado hasta aquí. Me ha aportado la fortaleza, la madurez y la felicidad necesarias para mantenerme firme. Es una mujer extraordinaria.

A mis amigos por su apego a lo largo de estos años; a pesar de no haber podido estar siempre presente, me han tenido en cuenta; aún sin verlos un tiempo, me han hecho sentir como si no me hubiese alejado ni un sólo día. Su comprensión y esos pequeños ratos de felicidad que me han aportado a lo largo de este camino han sido trascendentales..

A los compañeros que han recorrido conmigo este viaje y que saben de primera mano todo lo que se sufre; apoyar el hombro los unos en los otros hace llevadero el recorrido a lo largo de la carrera universitaria. Su respaldo emocional y la comprensión mutua es algo que sin lugar a dudas me ha servido para afrontar y sobrellevar la travesía con ánimo y buen humor.

A mis queridos vecinos Esperanza Varo y José Manuel Corrales padre e hijo, por su ternura y su atención a lo largo de estos años; tanto en lo bueno como en lo malo siempre han estado presentes y han cuidado de mí, y por ello no podían faltar aquí para expresarles mi gratitud.

A los profesores que me han aportado los conocimientos adquiridos a lo largo de mi formación y que me han inspirado tanto a la hora de proseguir mis estudios, no sólo por su sabiduría sino por su pasión por aprender; entre todos ellos tengo que destacar a doña Patricia Mora Segado por su cariño y su apoyo, y a don Carlos Jesús Pérez del Pulgar Mancebo por su amabilidad, su confianza, su paciencia y su comprensión.

Al personal de la Biblioteca de Industriales y Politécnica de la Universidad de Málaga por su complicidad con el alumnado haciendo más fácil el acceso a los recursos bibliográficos necesarios para su formación, en especial a Rafaela Díaz Domínguez, Rocío Fernández Guerra y Mª Carmen Díaz Pérez por su cariño, su altruismo y su bondad.

Gracias a todos de corazón.

CREACIÓN DE ENTORNO DE SIMULACIÓN EN INGRAVIDEZ CON INTERACCIÓN ENTRE SATÉLITE Y ROBOT

Rafael Cervilla Rivas | Estudiante Carlos Jesús Pérez del Pulgar Mancebo | Profesor

Grado en Ingeniería en Diseño Industrial y Desarrollo del Producto

Trabajo Fin de Grado

Palabras clave | Escena, dinámica, reaccionable, simulación, objeto, forma, script, satélite.

RESUMEN

El presente trabajo fin de grado nace del especial interés acaecido a lo largo de estos años por la supresión de escombros espaciales, particularmente de satélites muertos, debido a la congestión que producen en las órbitas terrestres promoviendo el riesgo de colisión con otros satélites en funcionamiento.

A este fin, se han sucedido diversas investigaciones donde apuestan por usar como solución manipuladores robóticos y a las cuales se ha sumado la Universidad de Málaga requiriendo, en consecuencia, la creación de un entorno de simulación ingrávido donde concurren un satélite y un robot que ofrece la posibilidad de interactuar entre ellos.

El objetivo de este trabajo es crear una simulación espacial, donde hay que diseñar el entorno teniendo en cuenta los parámetros dinámicos tanto del espacio como el de los objetos, usando un software y las correspondientes secuencias de comandos de programación para poder controlar los manipuladores.

La simulación se presenta en un espacio ingrávido con un satélite y un robot provisto de dos brazos robóticos y un propulsor. Al iniciarse ésta, el robot se desplaza una distancia desde una posición de partida hasta aproximarse frente al satélite donde se frena y queda en reposo; a continuación, mediante unos controles que permiten mover los manipuladores a voluntad por el usuario, se interactúa con el satélite para obtener un resultado visual de la reacción debida a la interacción llevada a cabo entre ellos.

Este entorno de simulación pretende ser un punto de partida para proseguir dichas investigaciones de la Universidad de Málaga, partiendo del diseño dinámico de este entorno y la programación de los manipuladores robóticos que han permitido obtener diferentes resultados de posibles interacciones entre un satélite y un robot.

IMPLEMENTATION OF A WEIGHTLESSNESS SIMULATION ENVIRONMENT TO PERFORM INTERACTION BETWEEN SATELLITES AND ROBOTS

Rafael Cervilla Rivas | Student Carlos Jesús Pérez del Pulgar Mancebo | Professor

Industrial Design Engineering and Product Development Degree

Final Degree Project

Key words | Scene, dynamic, respondable, simulation, object, shape, script, satellite.

ABSTRACT

This final degree project born of special interest befallen over the years by the removal of space debris, particularly from dead satellites, due to congestion that occur in Earth orbits promoting the risk of collision with other satellites in operation.

For that purpose, there have performed several investigations where opt for use the robotic manipulators as solution and ones which the University of Málaga has joined requiring therfore creating a weightless simulation environment where concur a satellite and a robot that offers the possibility to interact with each other.

The aim of this work is to create a space simulation where it has to design the environment taking into account dynamic parameters both the space and the objects, using software and related programming scripts to control the manipulators.

The simulation is showed in a weightless space with a satellite and a robot equipped with two robotic arms and a thruster. When this one is run, the robot moves a distance from a start position to approach in face of the satellite where it slow down and comes to rest; then, via controls that allow to handle the manipulators at will by the user, it is interacted with the satellite to obtain a visual result of the reaction due to the interaction triggered off with each other.

This simulation environment is meant to be a point of departure for pursuing these investigations of the Unversity of Málaga, as of the diynamic design of this environment and the programming of the robotic manipulators that they have allowed to achieve different results of posible interactions between satellites and robots.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN.

A lo largo de estos años ha acaecido un especial interés por la supresión de escombros espaciales, particularmente de satélites muertos, debido al riesgo de colisión con otros satélites en funcionamiento.

Las diversas investigaciones surgidas a raíz de este problema apuestan por usar como solución manipuladores robóticos, a las cuales se ha sumado la Universidad de Málaga requiriendo, en consecuencia, la creación de un entorno de simulación ingrávido donde concurren un satélite y un robot que ofrece la posibilidad de interactuar entre ellos.

1.1. Objeto.

Este trabajo tiene por objeto establecer los requisitos físicos de carácter general de un entorno de simulación espacial como herramienta para las investigaciones llevadas a cabo en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga.

A través de un proceso de diseño dinámico de los elementos y el espacio donde se ubican éstos, se pretende culminar con un entorno de simulación que permita reproducir fielmente el comportamiento físico que se puede dar en el espacio exterior entre un satélite y un robot hipotéticamente usado en una misión de servicio en órbita controlado desde la Tierra.

1.2. Motivación.

Desde mediados de los noventa [1], [2] el paradigma de usar manipuladores espaciales de servicio en órbita ha atraído a muchos investigadores, cuyos trabajos fueron motivados por misiones nacionales e internacionales no sólo para reparar, rescatar y reabastecer de combustible los satélites; sino también para la eliminación de escombros y de satélites muertos.

La supresión de escombros espaciales usando manipuladores espaciales ha llegado a ser de particular interés a medida que los escombros espaciales han ido en aumento hasta alcanzar un punto en el que las órbitas se hayan tan congestionadas que suponen un riesgo de colisión con los escombros de mayores dimensiones. De hecho, varios estudios dirigidos por agencias nacionales espaciales concluyeron que el ambiente espacial sólo puede ser preservado con la supresión continua de los escombros orbitales más grandes, tales como los satélites muertos.

Todas las misiones de servicio robótico requieren que un brazo robot capture todos los satélites

inactivos de una manera fiable y segura dadas las restricciones ambientales y operacionales. Ya que los satélites inactivos no tienen un sistema de control de posición funcional normalmente, es común que se hallen a la deriva; esto es porque el momento angular que se almacena en los volantes y giroscopios del satélite comienzan a migrar de su cuerpo tan pronto como ocurre el fallo y dejan de funcionar. Las observaciones terrestres también confirman que muchos escombros tienen un movimiento a la deriva, haciendo que la tarea de la captura robótica sea una tarea difícil.

En realidad, aunque varias misiones de captura en ingravidez usando un brazo robótico en órbita han sido frustradas, la captura robótica de satélites a la deriva aún no ha sido intentada. Los manipuladores robóticos que sean usados en misiones de servicio en órbita en el futuro serán operados desde tierra o autónomamente dependiendo de las restricciones de la misión, los requerimientos y el nivel de la preparación tecnológica a tal efecto. Sin embargo, en presencia de del respaldo de un robot, incluso un operador entrenado no puede realizar con éxito una captura en ingravidez a menos que la velocidad angular sea extremadamente baja; por tanto, incrementar la autonomía en los sistemas robóticos para las misiones de servicio en órbita es una de las claves de la tecnología identificadas por muchas agencias espaciales.

La ilustración nº 1 muestra lo que sería una típica operación de un satélite de servicio –denominado *Servicer* en inglés– en órbita que lleva un brazo robótico (con un mecanismo de agarre incorporado), que es conducido mediante un sistema de visión para capturar un satélite cliente –satélite inactivo que es objetivo de la captura espacial denominado en inglés *Client satellite*–. El satélite de servicio está bajo su propio control de posición, mientras que se asume al satélite cliente un satélite no cooperativo con dinámica incierta.

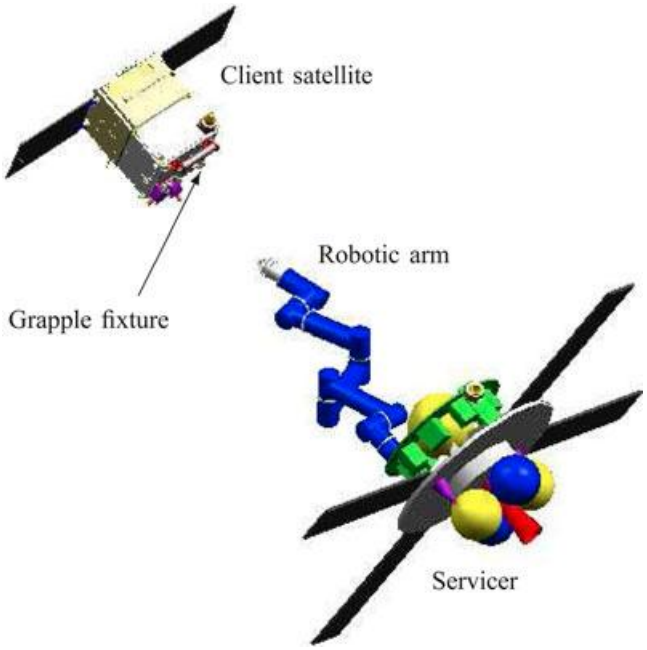


Ilustración 1. Satélite de servicio intentando capturar un satélite cliente [1].

Claramente, solo después de capturar y estabilizar con éxito el satélite que se encontraba a la deriva, puede realizarse una operación de reparación, rescate o de devolver a órbita. Por consiguiente, una tarea común para dicho servicio en órbita consistiría en dos operaciones robóticas triviales: la fase de preagarre y la fase de postagarre. En la primera fase, el brazo manipulador se mueve desde una posición inicial hasta interceptar el accesorio de agarre incorporado en el satélite cliente –denominado *Grapple fixture* en inglés– en un punto de encuentro con velocidad relativa nula. En la segunda fase, el manipulador espacial tiene que llevar el satélite cliente a reposo de tal modo que la interacción de fuerza/par permanezca por debajo de un valor seguro.

Satisfacer estos requerimientos de captura no es posible sin la predicción del movimiento del objetivo para que el movimiento del efector final pueda ser cuidadosamente planeado y ejecutado. Además, los parámetros del satélite cliente son usualmente inciertos porque no hay manera práctica de medir el combustible propulsor restante en gravedad cero. En consecuencia, cuando la dinámica del satélite objetivo no es conocida de antemano, no solo sus velocidades lineal y angular sino también sus parámetros de inercia deben ser estimados con precisión en tiempo real para que el satélite servidor pueda predecir de forma fiable los estados del objetivo en el futuro.

La mayoría de las técnicas que se desarrollan para la planificación de movimiento en tiempo real y la intercepción robótica del objetivo en movimiento han sido estipuladas para un punto de masa; y como es lógico, la intercepción de un satélite errante es una tarea más difícil debido a la complejidad dinámica de los sólidos rígidos en rotación. Por ello, ya ha sido propuesto un algoritmo iterativo para un caso plano a la hora de resolver una trayectoria óptima para un consumo de tiempo y combustible mínimos; un esquema de planificación de movimiento similar con la inclusión de límites tanto en las articulaciones como en la velocidad; e incluso sistemas de visión guiadas, las cuales están predominando sobre las demás opciones debido a que son capaces de estimar la posición de objetos en movimiento. Entre los sistemas de visión guiadas, un sistema de visión activo tal como el sistema de cámara laser (LCS) es la que se está optando por usar debido a su robustez frente a las duras condiciones de iluminación; de hecho, durante la misión espacial STS-105 se corroboró esto con una tecnología de imagen 3D usada en el LCS.

1.3. Justificación del proyecto.

El artículo que inspira este trabajo, denominado *Un esquema de predicción y planificación de movimiento para la captura robótica guiada visualmente de objetos flotando libremente a la deriva con dinámica incierta* [1], se centra en un esquema guiado de robot para la fase de preagarre donde se emplea una teoría de control y estimación óptima para la planificación del movimiento de un manipulador robótico, con el fin de interceptar un satélite objetivo no cooperativo con dinámica desconocida usando un filtro de Kalman con un valor de umbral

definido para la detección de convergencia dada para una matriz característica del filtro mediante el cual se predice el movimiento del objetivo.

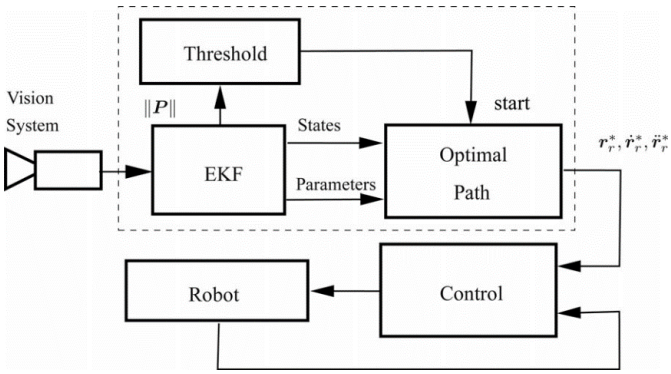


Ilustración 2. Control de planificación y predicción propuesto por Aghili Farhad [1].

El movimiento del robot es planificado para minimizar una función de coste sujeto a la restricción de la llegada simultánea del efector final del robot y el accesorio de agarre al mismo punto y a la misma velocidad, de modo que los experimentos se presentan para demostrar que la aplicación del método de orientación visual que se propone puede ser usado para la captura robótica de un satélite errante. En dichos experimentos, se emplean dos manipuladores robóticos: uno brazo rota y lanza a la deriva la maqueta de un satélite, y el otro que está equipado con una mano robótica trata de capturar el accesorio de agarre dispuesto en el satélite cliente para ser agarrado mediante el sistema de guía visual comentado.

En este trabajo, se pretende reproducir el primer experimento de dicho artículo equipando al satélite servidor de dos brazos robóticos en vez de uno, acercándose al satélite cliente con velocidad relativa nula, y lanzarlo a la deriva.

1.4. Normativa considerada.

El presente trabajo ha sido redactado conforme a las Normas para la Presentación Formal del TFG [3] y la norma UNE 157001/2002 [4].

1.5. Definiciones y abreviaturas.

A efectos del presente trabajo, se entenderá por:

1) «TFG»: trabajo fin de grado.
2) «script»: secuencia de comandos de programación.
3) «API»: conjunto de subrutinas, funciones y procedimientos de programación.
4) «plugin»: complemento de un software considerado como una aplicación adicional ejecutable por medio del API.

5) «redundante»: dícese de un robot con un número de grados de libertad controlables mayor que el total disponible.
6) «escena»: región de la interfaz de usuario destinado a reproducir el entorno de simulación.
7) «efector final»: elemento terminal de un robot.

2. ENTORNO DE SIMULACIÓN.

El simulador de robots V-REP [5], con entorno de desarrollo integrado, se basa en una arquitectura de control distribuida: cada objeto/modelo puede ser controlado individualmente mediante un script incrustado, un plugin, un nodo ROS, un cliente API remoto o una solución personalizada. Esto hace que V-REP sea muy versátil e ideal para aplicaciones multirobot. Los controladores se pueden escribir en C/C++, Python, Java, Lua, Matlab, Octave o Urbi.

Las aplicaciones principales de este software son: la simulación de sistemas de automatización de fábrica; monitoreo remoto; control de hardware; verificación y prototipado rápido; seguimiento de seguridad; desarrollo rápido de algoritmos; educación relacionada con la robótica; y la presentación de producto.

2.1. Introducción.

La simulación y las simulaciones son completamente personalizables con V-REP [5], ya que soporta 6 enfoques de programación, como un script incrustado, un plugin, un nodo ROS, etc.; y 7 lenguajes de programación que son mutuamente compatibles y pueden trabajar mano a mano a través de las APIs.

Ilustración 3. Presentación ilustrativa de los motores de física, y de las funcionalidades de cálculo cinemático y simulación de partículas (adaptado de [5]).

Para establecer los requisitos físicos de la simulación, el software ofrece 4 motores de física, que son: *Bullet Physics*, *ODE*, *Vortex Dynamics* y *Newton Dynamics*; donde los dos primeros son considerados como «motores de física de juego»; el tercero es un motor de física de alta fidelidad utilizado en aplicaciones industriales y de investigación de alto rendimiento y precisión; y el último motor de física se trata de una biblioteca de simulaciones de física de tipo multiplataforma empleado no sólo para juegos sino también para simulaciones físicas en tiempo real. No obstante, todos son motores de suficientes prestaciones para simular con fidelidad el presente trabajo, con la salvedad de que el motor Bullet es más rápido que cualquier otro y presenta dos versiones (2.78 y 2.83) sin diferencias significativas; el Vortex sólo ofrece 20 minutos de simulación en su versión gratuita; y la implementación plugin actual del motor Newton es una versión beta.

Las funciones características que soporta V-REP son [5]:

- Cálculo cinemático directo/inverso para cualquier tipo de mecanismo.
- Reproducción de partículas personalizables usadas para simular chorros de aire o agua, motores a reacción, hélices, etc.
- Rápida verificación de interferencias entre cualquier forma, malla, o nube de puntos.
- Cálculo rápido y exacto de la distancia mínima entre cualquier forma, malla o nube de puntos.
- Cálculo de distancia mínima exacta dentro de un volumen de detección personalizable a través de un sensor de proximidad configurable.
- Simulación de sensores de visión con muchos filtros incorporados de procesamiento de imágenes, totalmente personalizables y extensibles.
- Cada elemento incluido en el entorno de simulación puede tener su propio script incrustado.
- La planificación de rutas/movimientos a través de un plugin.
- La representación gráfica en 2D y 3D de gran variedad de datos capaz de ser registrados en las simulaciones.
- Diseño y presentación de interfaces de usuario personalizables para la definición y el control de los elementos de la simulación.
- Modos de edición de malla especiales, como los métodos semiautomáticos de extracción de formas primitivas, descomposición convexa, control de triangulación de mallas, etc.
- Composición del entorno visualizada intuitivamente en una vista de jerarquía indicando nombres y tipos de objetos, scripts asociados, lazos cinemáticos, estados de selección y visibilidad, advertencias, etc.
- Presentación de un navegador integrado de modelos que admiten operaciones de arrastrar y soltar en el entorno, incluso durante una simulación. La biblioteca de modelos disponible puede ser ampliada por el usuario.
- Simulación de espejos y comunicaciones inalámbricas.

De todo lo expuesto, en este trabajo ha sido empleado el lenguaje de programación Lua, un enfoque de programación de scripts incrustados, el motor de física *Bullet Physics* –versión 2.78–, y las funcionalidades empleadas de entre las descritas son el cálculo cinemático inverso de mecanismo y la reproducción de partículas para el motor de reacción del robot, es decir, del satélite cliente.

2.2. Interfaz de usuario.

El software de simulación V-REP presenta una serie de elementos característicos que debemos conocer para saber manejar la interfaz de usuario adecuadamente, la cual se describe a continuación [5].

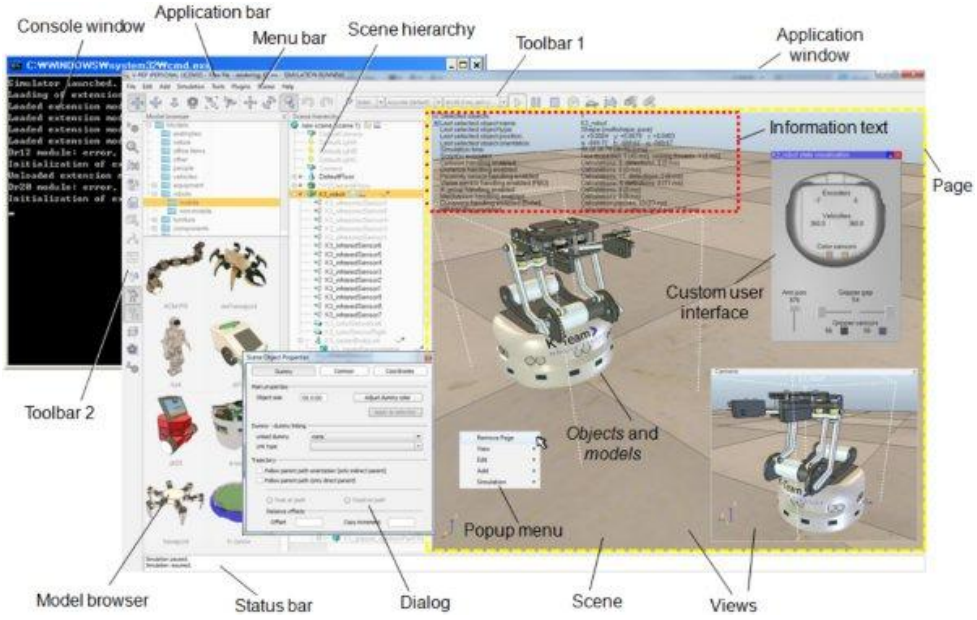


Ilustración 4. Interfaz de usuario del programa V-REP [5].

- Ventana de consola (*Console window*).

  Cuando se inicia el programa, se crea una ventana de consola pero se oculta de nuevo. Este es un comportamiento predeterminado que se puede modificar en el cuadro de diálogo de configuración de usuario. La consola no es interactiva, sólo se utiliza para generar o emitir información cuando el usuario lo estime a través de los comandos de programación oportunos.

- Ventana de aplicación (*Application window*).

  Esta es la ventana principal del programa, y se utiliza para visualizar, editar, simular e interactuar con una escena. Los botones izquierdo y derecho del ratón, la rueda de éste, así como el teclado tienen funciones específicas cuando se emplean en la ventana de

aplicación. Dentro de ésta, las funciones de los dispositivos de entrada (ratón y teclado) pueden variar según el contexto o la ubicación de su activación dentro del programa.

- Cuadros de diálogo (*Dialogs*).

  Junto a la ventana de aplicación, el usuario también puede interactuar con una escena y editarla mediante el ajuste de parámetros o el diálogo de ajustes. Cada cuadro de diálogo agrupa un conjunto de funciones relacionadas a los elementos en general, o funciones que se aplican al objeto de destino seleccionado.

Al iniciar el software, se mostrará una escena (*Scene*) predeterminada. El usuario es libre de abrir varias escenas en paralelo. Cada escena comparte la ventana de aplicación y los cuadros de diálogo con las otras escenas, pero sólo el contenido de la escena activa estará visible en la ventana de aplicación o en los cuadros de diálogo (sólo una escena es visible en un momento dato). En lo sucesivo, se dará una breve descripción de los elementos de la ventana de aplicación [5].

- Barra de aplicaciones (*Application bar*).

  Esta barra indica el tiempo de licencia de la copia de V-REP usada, el nombre del archivo de la escena que se está visualizando actualmente, el tiempo utilizado en un paso de reproducción y el estado actual del simulador.

- Barra de menú (*Menu bar*).

  Esta barra permite acceder a casi todas las funciones del simulador. La mayoría de las veces, los elementos de la barra de menú activan un diálogo. El contenido de dicha barra es sensible al contexto; es decir, dependerá del estado actual del simulador. También se puede acceder a la mayoría de las funciones de la barra de menú a través de un menú emergente (*Popup menu*), hacer doble clic en un icono de la vista de jerarquía de escenas o hacer clic en un botón de la barra de herramientas.

- Barras de herramientas (*Toolbars*).

  Esta barra presenta las funciones más frecuentes de empleo, como por ejemplo cambiar el modo de navegación, seleccionar otra página, ejecutar la simulación, etc. Algunas funciones de la barra de herramientas nº 1 y todas las funciones de la nº 2 también se pueden acceder a través de la barra de menús o del menú emergente.

- Navegador de modelos (*Model browser*).

  Este panel está visible de forma predeterminada, pero se puede alternar con el co-

rrespondiente botón de la barra de herramientas nº 2. Muestra en su parte superior una estructura de carpetas de modelos V-REP, y en su parte inferior miniaturas o imágenes representativas de dichos modelos contenidos en la carpeta seleccionada. Las miniaturas se pueden arrastrar y soltar en la escena para cargar automáticamente el modelo relacionado.

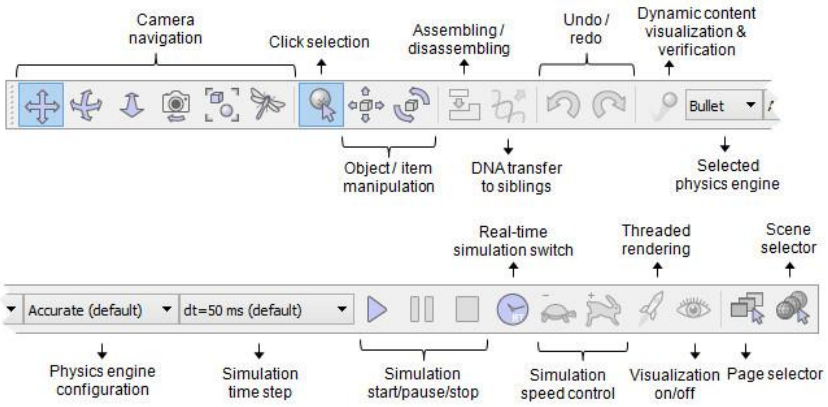


Ilustración 5. Barra de herramientas nº 1 [5].

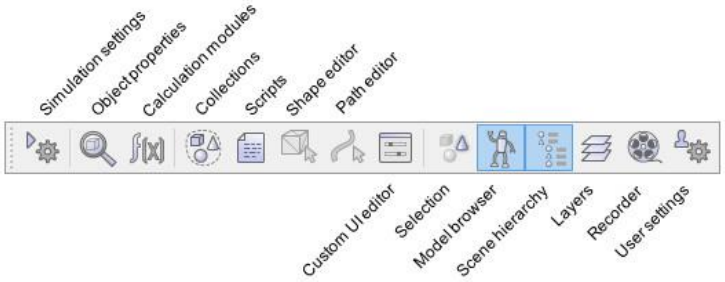


Ilustración 6. Barra de herramientas nº 2 [5].

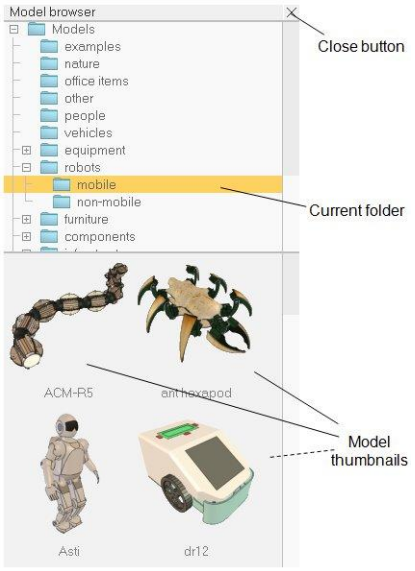


Ilustración 7. Navegador de modelos [5].

- Jerarquía de escena (*Scene hierarchy*).

Este panel está visible de forma predeterminada al igual que el navegador de modelos, y al igual que éste se puede alternar con el botón correspondiente de la barra de herramientas. Muestra el contenido de una escena, es decir, todos los objetos que componen el entorno de simulación. Dado que los objetos de escena se construyen según una estructura jerárquica, la jerarquía de escena muestra un árbol de esta jerarquía y los elementos individuales se pueden expandir o contraer. Un doble clic en un icono abre/cierra un diálogo de propiedad relacionado con el icono que se ha hecho clic. Un doble clic en un nombre de objeto permite editarlo. La rueda del ratón, así como las barras de deslizamiento laterales que presenta permite navegar de arriba abajo por su contenido. Los objetos de la jerarquía de escena pueden arrastrarse y soltarse sobre otro objeto para crear una relación padre-hijo. El panel de jerarquía es sensible al contexto, es decir, mostrará un contenido diferente si el simulador está en un estado de modo de edición u otro.

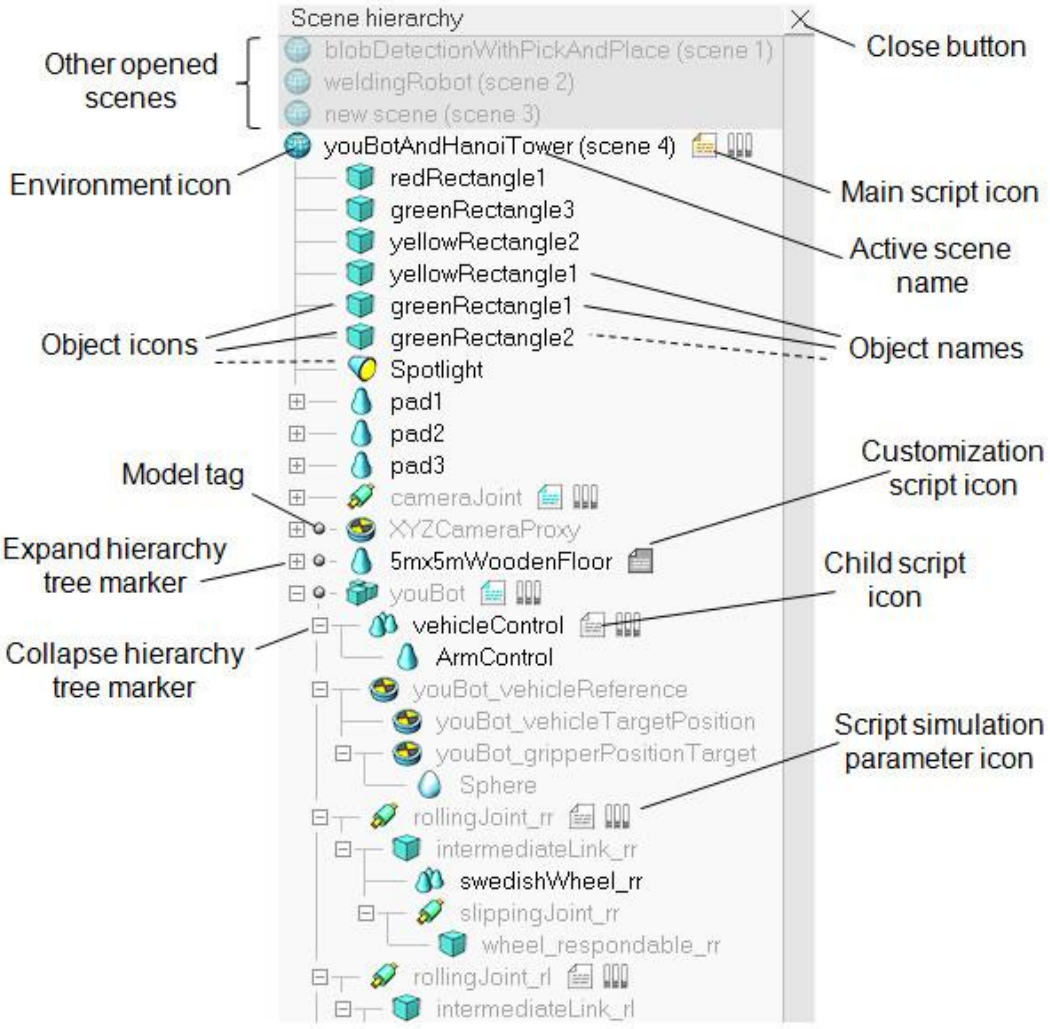


Ilustración 8. Jerarquía de escena típica en V-REP [5].

- Página (*Page*).

  Cada escena puede contener hasta ocho páginas, las cuales pueden contener un número ilimitado de vistas. Una página se puede ver como un contenedor para vistas.

- Vistas (*Views*).

  Se utilizan para mostrar la escena vista a través de objetos visibles tales como cámaras, gráficos o sensores de visión.

- Texto de información (*Information text*).

  Muestra información relacionada con la selección de objeto/elemento actual y con los parámetros o estados de la simulación en ejecución. La pantalla de texto se puede alternar con uno de los dos pequeños botones presentes en el lado superior izquierdo de una página. El otro botón se puede utilizar para alternar un fondo blanco, dando un mejor contraste dependiendo del color de fondo de una escena.

- Barra de estado (*Status bar*).

  Esta barra muestra información relacionada con las operaciones realizadas, comandos y también muestra mensajes de error del intérprete de Lua. Desde dentro de una secuencia de comandos el usuario puede emitir información cuando el usuario lo estime a través de los comandos de programación oportunos.

- Interfaces de usuario personalizadas (*Custom user interface*).

  Se tratan de superficies definidas por el usuario que pueden utilizarse para mostrar información o un diálogo personalizado que permite al usuario interactuar con la simulación de forma personalizada.

- Menú emergente (*Popup menú*).

  Se trata de un menú activable por medio del botón derecho sensible al contexto, por lo que mostrará funciones diferentes según el modo de edición o el estado actual de la simulación. La mayoría de las funciones que ofrece pueden ser accedidas mediante la barra de menú.

2.3. Propiedades dinámicas generales.

Se tratan de las propiedades que definen la dinámica del sistema que estamos considerando, y se haya en el apartado de dinámica (*Dynamics*) dentro del cuadro de diálogo de propiedades del

módulo de cálculo (*Calculation modules*) accesible a través de la barra de herramientas nº 2 y la barra de menú.

Según las especificaciones establecidas previamente, en este apartado se establecen tanto el motor de física a usar como la amplitud y dirección de la gravedad aplicada a todos los elementos; por tanto, aquí se establecerá que nuestro motor de física es *Bullet Physics* y que nuestra gravedad es nula para todos los ejes espaciales ya que el entorno de simulación reproduce el espacio exterior.

Además, existen algunas especificaciones más que se pueden establecer como son la representación de los puntos de contacto entre los cuerpos susceptibles de interactuar físicamente entre sí, la cual dejaremos desactivada; y el ajuste de parámetros del motor de física, que no podemos modificar debido a que la versión de software educativa empleada es limitada.

La ilustración 9 muestra la configuración establecida para el entorno de simulación del presente trabajo.

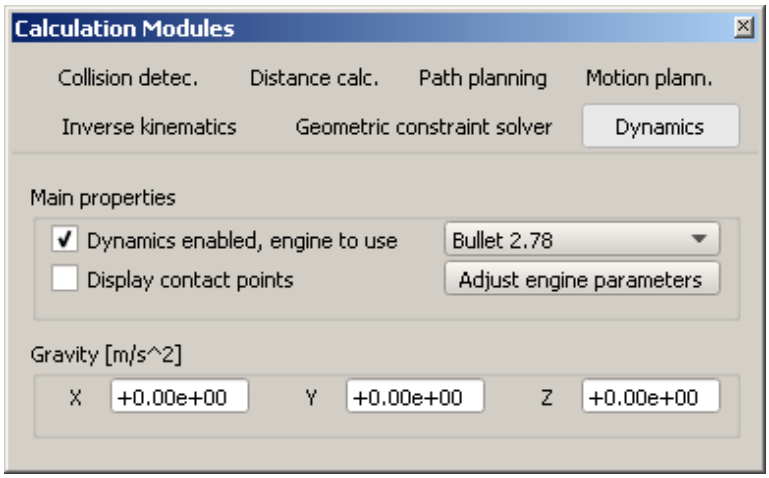


Ilustración 9. Diálogo de propiedades dinámicas generales del TFG.

2.4. Ajustes de usuario.

Se tratan de las características de configuración generales relacionadas con la interfaz de usuario, y es accesible mediante la barra de herramientas nº 2 a través del botón denominado *User settings*. De este cuadro de diálogo sólo se necesitan dos ajustes, dejando los demás por defecto, y son: la desactivación de la herramienta deshacer/repetir (*Undo/redo enabled*) y la de la opción de ocultar la ventana de consola (*Hide console window*) que vienen activas por defecto.

El objetivo de estos ajustes es, en primer lugar, acelerar el funcionamiento del software debido a que no almacena las acciones realizadas al desactivar la herramienta deshacer/repetir; y en segundo lugar, mostrar la consola que se usará para mostrar información implícita de los elemen-

tos de la simulación. La ilustración 10 muestra la configuración establecida para el entorno de simulación del presente trabajo.

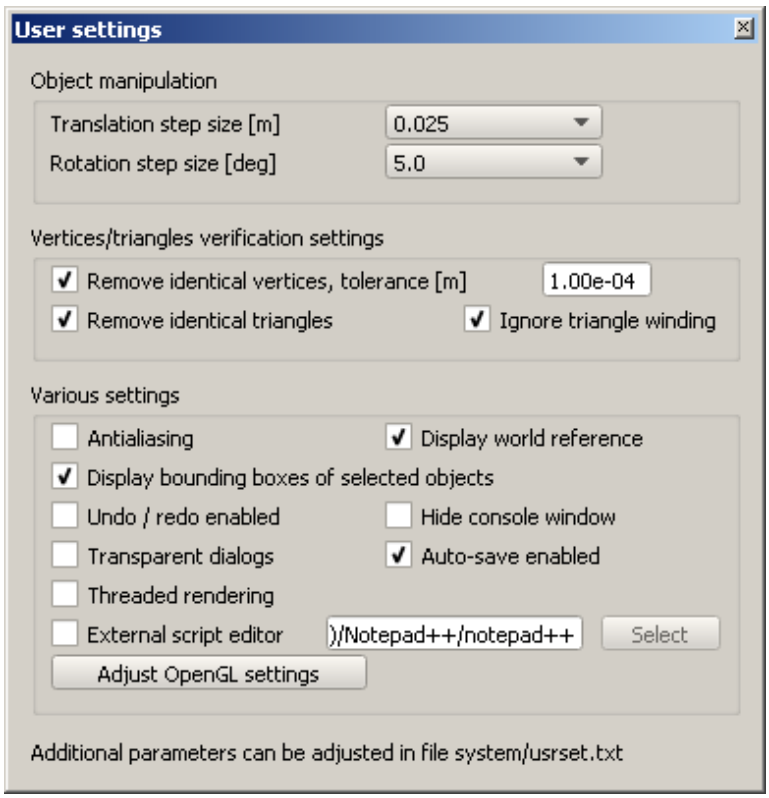


Ilustración 10. Ajustes de usuario del TFG.

2.5. Propiedades ambientales.

El ambiente en V-REP define propiedades y parámetros que son parte de una escena, lo cual no incluye a los elementos que lo componen; de hecho, dichas propiedades no se guardan cuando se guarda un modelo, sino cuando se guarda una escena.

Un ambiente define las siguientes propiedades y parámetros en general [5]:

- Colores de fondo.
- Parámetros de niebla.
- Luz ambiental.
- Información de creación de escena.
- Ajustes adicionales.

Las propiedades ambientales son accesibles a través de la barra de herramientas o haciendo doble clic en el icono presente en la jerarquía de escena junto al nombre de la escena. De este cuadro de diálogo sólo se necesita definir la información de creación de la escena, dejando lo demás por defecto. En el cuadro reservado a tal efecto se ha escrito en inglés:

Final Degree Project entitled: "Implementation of a weightlessness simulation environment to perform interaction between satellites and robots".

Que viene a decir en español:

Trabajo fin de grado titulado: "Creación de entorno de simulación en ingravidez con interacción entre satélite y robot".

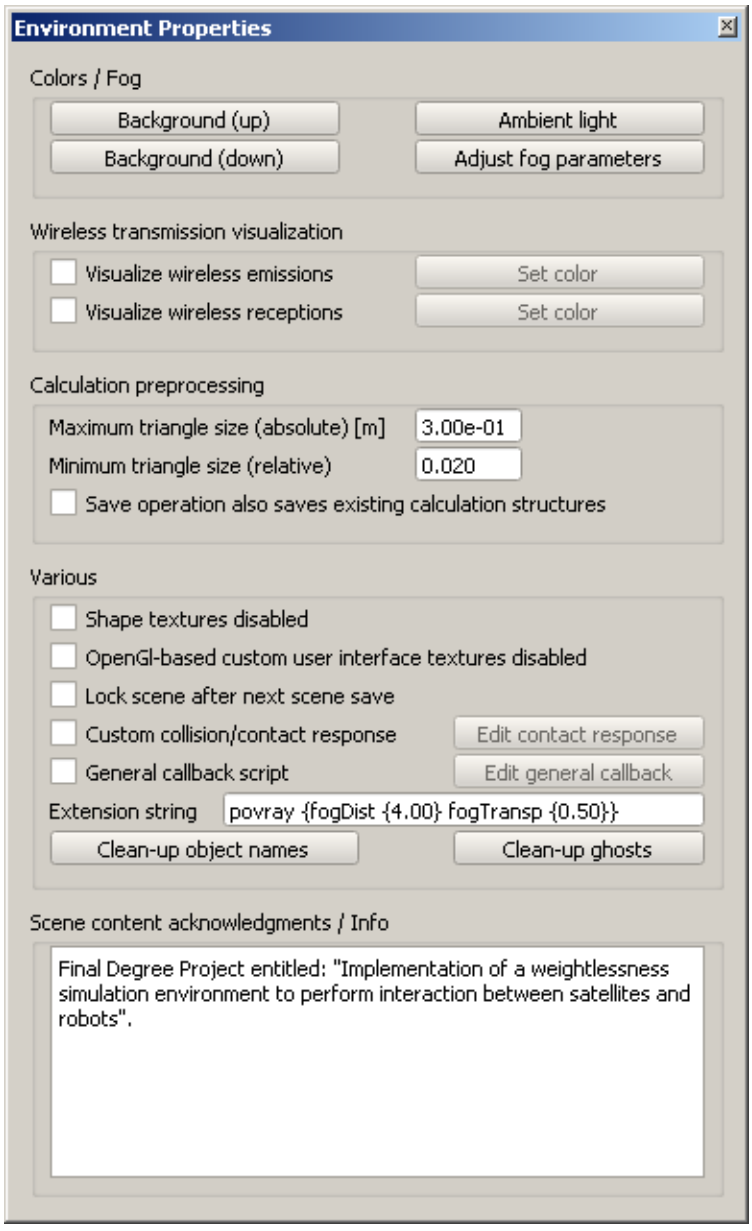


Ilustración 11. Propiedades ambientales del TFG.

De este modo, queda definida la escena y cuando se abra el archivo se mostrará esta información automáticamente –expresada en inglés por ser el idioma de comunicación europeo e internacional por excelencia–. La ilustración 11 muestra la configuración establecida para el entorno de simulación del presente trabajo.

2.6. Modelos.

Un modelo es un subelemento de una escena que por sí solo no puede existir –excepto en un archivo– ni se puede simular por sí mismo; en consecuencia, tiene que estar contenido en una escena para ser operativo [5].

Los modelos se definen mediante una selección de objetos de escena construidos en un mismo árbol de jerarquía, donde la base del árbol tiene que ser un objeto marcado como "objeto es modelo base" en las propiedades de dicho objeto, distinguiéndose del resto mediante un icono a la izquierda de su nombre caracterizado por un círculo gris considerado como la marca o etiqueta que identifica a un modelo [5].

Cuando la base del objeto es seleccionado, un cuadro de selección definido mediante aristas discontinuas abarcando el modelo por completo es mostrado, permitiéndose seleccionar los objetos individuales que lo componen pulsando los botones de teclado *Ctrl*/*Shift* durante su selección o mediante la selección directa en el árbol de jerarquía del modelo.

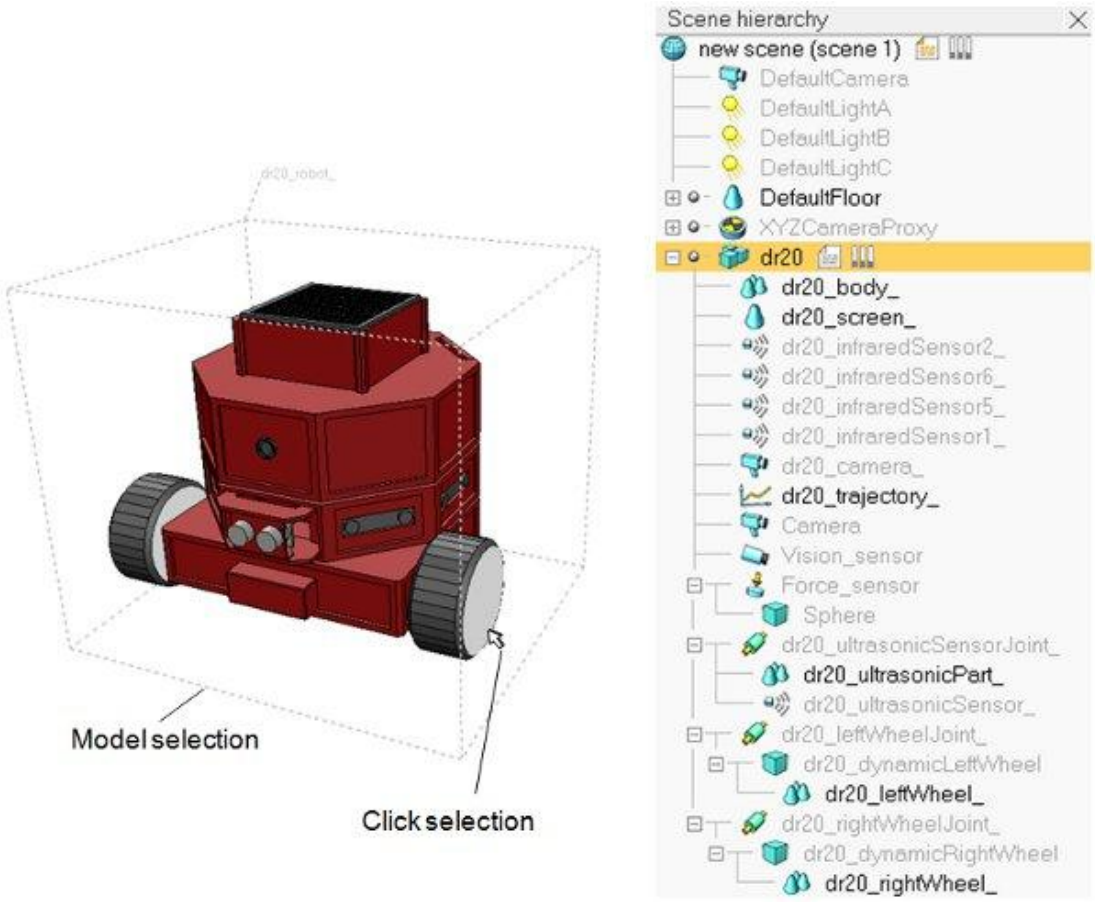


Ilustración 12. Vista de escena y vista correspondiente de la escena de jerarquía. El elemento seleccionado en el panel se corresponde con el objeto base considerado como modelo (círculo gris a la izquierda) [5].

Un doble clic en una etiqueta de modelo abre su cuadro de diálogo de propiedades disponible para ser ajustadas. Además, es recomendable contraer la jerarquía de un modelo una vez que el modelo sea editado con el fin de identificar fácilmente el número de elementos/modelos agrupados lógicamente.

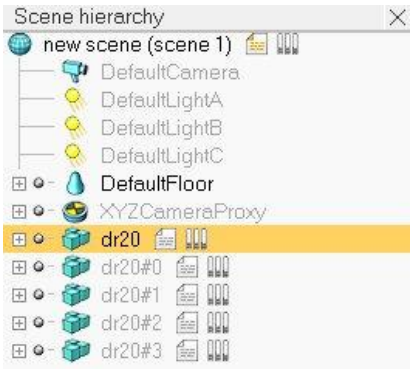


Ilustración 13. Jerarquía de escena donde se muestran 5 árboles de jerarquías de modelos contraídas [5].

La agrupación de varios objetos como modelo también es importante cuando un script accede a los objetos mediante programación, recomendándose asociarlo con la base de éste en consecuencia.

Para que los modelos se puedan combinar fácilmente (es decir, incorporados unos a otros) sin ninguna modificación adicional, es importante considerar qué papel jugará el modelo. ¿Se simulará dinámicamente? ¿Se unirá a otros modelos o aceptará otros modelos? Las respuestas a estas preguntas permitirán seleccionar el mejor tipo de objeto para trabajar como base de modelo, las cuales se obtendrán del siguiente apartado.

Los modelos se pueden copiar de una escena a otra como cualquier otro objeto. Los archivos de modelo (archivos "*.ttm") también admiten operaciones de arrastrar y soltar entre la ventana del explorador y la ventana de aplicación.

2.2.1. objetos.

En la escena sólo un número limitado de objetos se simularán dinámicamente, dependiendo de las propiedades de los elementos y la estructura de la escena, éstos serán formas, articulaciones y uniones de fuerza –accesibles desde el menú "Añadir" o *Add*–.

- Formas.

  Se tratan de objetos de malla rígida que están compuestos por caras triangulares [5]. Pueden ser creados, importados, editados y exportados. Existen cuatro subtipos diferentes: formas aleatorias simples/compuestas, formas convexas simples/compuestas, formas puras simples/compuestas, y formas malladas; todas ellas representadas en la siguiente ilustración de izquierda a derecha según han sido mencionadas.

Ilustración 14. Tipos de formas en V-REP [5].

Las formas aleatorias no son recomendadas para el cálculo dinámico, pues debido a su estructura irregular resultan en simulaciones lentas e inestables. Las formas convexas son formas optimizadas respecto a las aleatorias, resultando un menor tiempo computacional a la hora de simularlas. Las formas puras representan formas primitivas y son las adecuadas para el cálculo de la respuesta de colisión dinámica en una simulación, ya que ejecutan ésta de forma rápida y estable. Las formas malladas se usan para representar terrenos irregulares donde las alturas a lo largo de él cambian.

Las formas compuestas representan una agrupación de formas simples y actúa como una sola, lo cual es útil cuando un conjunto de formas simples que comparten las mismas propiedades se van a comportar como una entidad rígida.

En este trabajo, se usarán formas puras siempre que sea posible a fin de reducir al máximo el coste computacional de la simulación; cuando ello sea imposible, se usarán formas convexas.

Las formas pueden clasificarse en 4 grupos dependiendo de su comportamiento durante la simulación dinámica: estáticas o no dinámicas (*Static*), no estáticas o dinámicas (*Non-static*), reaccionable (*Respondable*) y no reaccionable (*Non-respondable*).

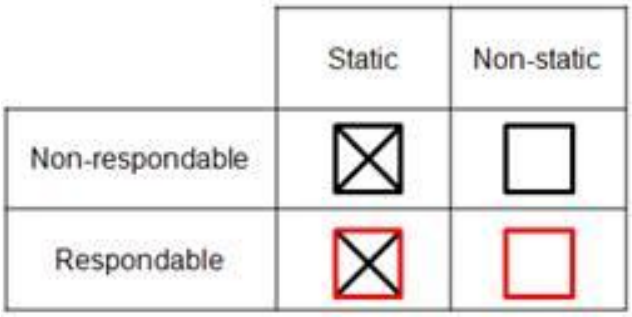


Ilustración 15. Tipos de formas presentes en una simulación dinámica [5].

Durante la simulación dinámica, las formas estáticas no serán influenciadas, es decir, su posición relativa a su objeto primario o padre es fija; mientras que las formas no estáticas estarán directamente influenciadas por la gravedad u otras restricciones. Las formas reaccionables se influyen mutuamente durante la colisión dinámica, o sea, producen una reacción de colisión mutua, rebotan entre sí. La ilustración siguiente representa los comportamientos estáticos/dinámicos y los reaccionables/no reaccionables.

Dos formas reaccionables siempre producirán una reacción de colisión, a menos que sus respectivas capas de colisión no se superpongan. Las propiedades de forma estática/dinámica, reaccionable/no reaccionable, así como las capas de colisión pueden definirse en el diálogo de propiedades dinámicas de la forma, el cual veremos en el siguiente apartado.

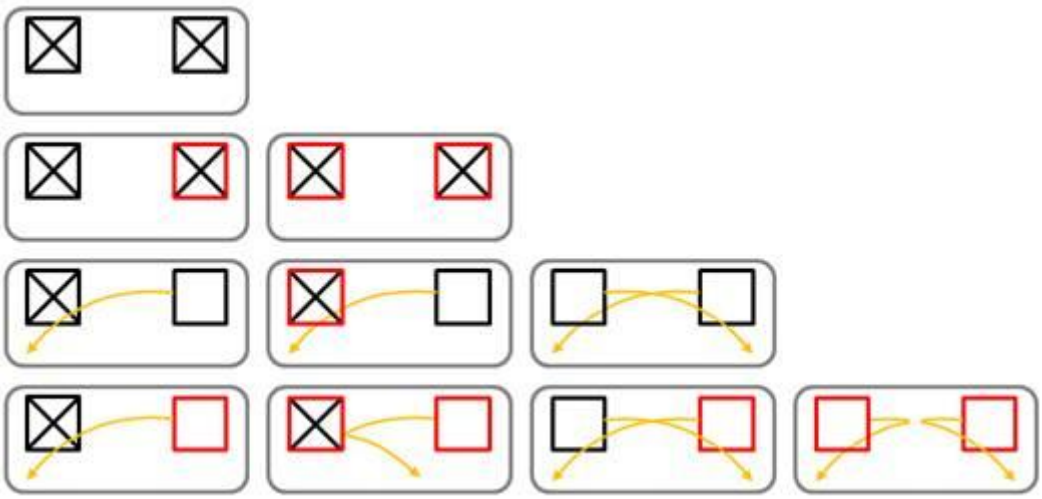


Ilustración 16. Comportamientos e interacciones entre formas estáticas/dinamicas y reaccionables/no reaccionables [5].

- Articulaciones.

Se tratan de objetos que tienen al menos un grado intrínseco de libertad (GdL) utilizados para construir mecanismos y mover objetos [5]. Existen tres subtipos diferentes: articulaciones de revolución (1 GdL) –que son los que se usarán exclusivamente en este trabajo–, prismáticas (2 GdL) y esféricas (3 GdL); todas ellas representadas en la siguiente ilustración de izquierda a derecha según han sido mencionadas.

Ilustración 17. Tipos de articulaciones en V-REP [5].

Comparado con otro objeto, una articulación tiene dos sistemas de referencia, sólo visibles cuando es seleccionado. El primero es el estándar, el cual es fijo al igual que cualquier otro objeto pues define su posición en la escena; el segundo no es fijo, y se desplazará con respecto al primer sistema de referencia dependiendo de la posición de la articulación, ya que se moverá relativamente respecto a éste según los grados de libertad que posea.

Una articulación se utiliza para permitir un movimiento relativo entre su padre y sus hijos. Cuando se construye una relación padre-hijo entre una articulación y un objeto, el objeto se une al segundo sistema de referencia de la articulación; por lo que un cambio en la configuración de la posición se reflejará directamente en sus hijos.

Las formas dinámicas caerán, es decir, estarán influenciadas por la gravedad, si no están limitadas de otro modo. Las restricciones dinámicas entre las formas se pueden configurar conectando dos formas con una articulación habilitada dinámicamente.

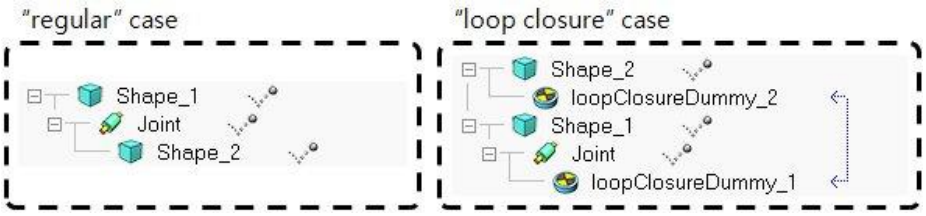


Ilustración 18. Formas dinámicas unidas por medio de una articulación [5].

Las articulaciones habilitadas dinámicamente son aquellas que operan en modo de fuerza/par o híbrido, que tienen un objeto forma como padre, y exactamente un objeto forma dinámico como hijo –que es el caso habitual (*"regular" case*)–. Además, es posible implicar a una articulación en una configuración de cierre de cadena (*"loop closure" case*) en el caso de crear una cadena cinemática, en cuyo caso la articulación tiene que conectarse a las dos formas a través de un vínculo dummy-dummy[1].

- Uniones de fuerza.

Se tratan de objetos destinados a vincular rígidamente dos formas dinámicas, es decir, capaces de trasmitir fuerzas y momentos [5]. La rigidez de estos elementos puede estar condicionada, o sea, se rompen si se supera un umbral de fuerza/par especificado o si se cumplen condiciones establecidas por el usuario.

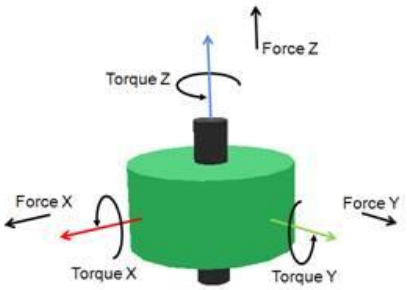


Ilustración 19. Fuerzas y momentos medidos por una unión de fuerza [5].

Una unión de fuerza además mide un par de 3 valores que representan la fuerza a lo largo de los ejes X, Y y Z; y el momento de fuerza alrededor de dichos ejes. Por ello, son denominados por el programa como "sensores de fuerza" en inglés (*Force sensor*); sin embargo, en este texto se denominarán "uniones de fuerza" debido a que se adecua mejor a la función principal que desempeña, pues la medición es una función secundaria aquí.

Es útil cuando se tiene una entidad rígida con propiedades dinámicas diferenciadas, por lo que las formas tendrán que estar rígidamente unidas a través de la unión de fuerza.

[1] Un *dummy* es un objeto cuyo fin no es otro que el de poder posicionar o relacionar determinados objetos entre sí. En nuestro caso, se utilizará para calcular la cinemática inversa del modelo «8 DoF Manipulator» [24].

Las uniones de fuerza habilitadas dinámicamente son aquellas que tienen un objeto forma como padre y exactamente un objeto forma dinámico como hijo. Al igual que las articulaciones, es posible involucrarlos en una configuración de cierre de cadena con analogía a lo explicado para éstas.

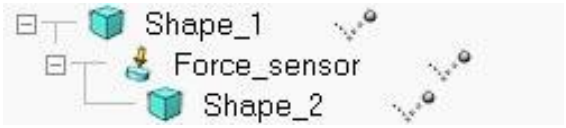


Ilustración 20. Formas dinámicas unidas por medio de una unión de fuerza (caso habitual) [5].

2.6.2. Consideraciones de diseño.

Los objetos que son simulados dinámicamente pueden reconocerse fácilmente durante la simulación, ya que el icono mostrado en la siguiente ilustración aparecerá junto al nombre del objeto en la jerarquía de escena:

Ilustración 21. Icono que indica los objetos simulados dinámicamente [5].

Al hacer doble clic en dicho icono presente en la jerarquía de escena junto a cada uno de los objetos simulados dinámicamente, se mostrará información relacionada con su comportamiento dinámico. Aquellos objetos destinados a simularse dinámicamente y que por alguna razón no puedan mostrarán el siguiente icono:

Ilustración 22. Icono de advertencia que indica que un objeto dinámico no puede ser simulado como tal [5].

Normalmente, dicho icono de alerta aparece cuando V-REP descubre que las articulaciones o uniones de fuerza no están habilitadas dinámicamente como debiesen; aunque pueden existir más razones. Por ello, se citan a continuación una serie de consideraciones de diseño que se han tenido en cuenta para construir jerarquía de un modelo correctamente [5].

- Empleo de formas puras.

  Siempre que sea posible, se debe utilizar formas puras como formas reaccionables ya que son mucho más estables y rápidas durante la simulación dinámica. En lugar de utilizar la complicada malla triangular de un modelo de robot como forma reaccionable, o su repre-

sentación convexa ligeramente mejor, se debe tratar de aproximar la malla triangular con varias formas puras y luego agruparlas, para ello: [*Menu bar* > *Edit* > *Grouping/Merging* > *Group selected shapes*].

Luego, para la visualización del modelo se usa la forma mallada como representación de su aspecto, considerándola como forma estática; y para el comportamiento dinámico del modelo se debe usar formas puras, que se hallarán ocultas por defecto en la capa 9 (en el siguiente apartado se explicarán las capas de visibilidad).

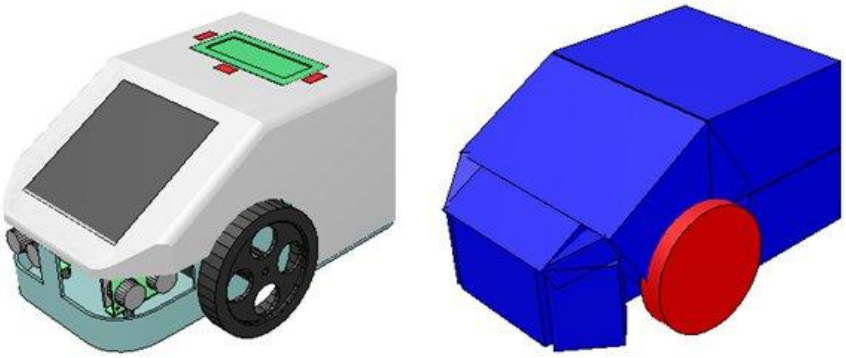


Ilustración 23. Modelo de robot dinámico (izquierda) y formas puras dinámicas subyacentes utilizadas para la simulación (derecha) [5].

Cuando un cuerpo es susceptible de colisionar, pero no constantemente, o no juega un papel importante en la estabilidad de un mecanismo/robot, entonces no es absolutamente necesario utilizar formas puras, y las formas convexas también podrían ser una alternativa viable.

❖ Empleo de formas convexas en lugar de formas aleatorias.

No siempre se puede usar formas puras o no resulta práctico su empleo. En ese caso, se puede generar una forma convexa. Este tipo de formas emplean menor tiempo de computación que las formas aleatorias, las cuales nunca se deben usar.

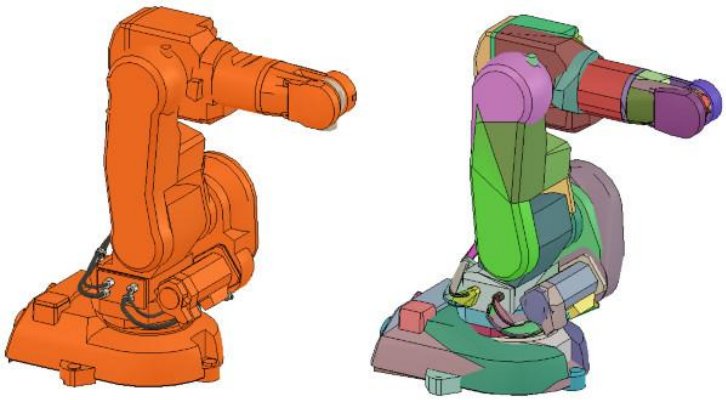


Ilustración 24. Modelo no convexo (izquierda) y correspondiente modelo convexo (derecha) [5].

Cuando se desea generar una forma convexa, se selecciona las formas de las que se quieren crear ésta, y luego: [*Menu bar > Edit > Morph selected shapes into convex shapes*]. La anterior ilustración muestra un ejemplo de dicha generación.

❖ Inspección rigurosa del contenido dinámico de la escena.

A veces, puede resultar un poco confuso trabajar con formas ocultas destinadas a desempeñar un papel activo en la simulación, como ocurre con las formas dinámicamente habilitadas, las articulaciones o las uniones de fuerza. Sin embargo, el contenido dinámico siempre puede ser examinado durante una simulación, activando el botón de visualización de contenido dinámico presente en la barra de herramientas nº 1.

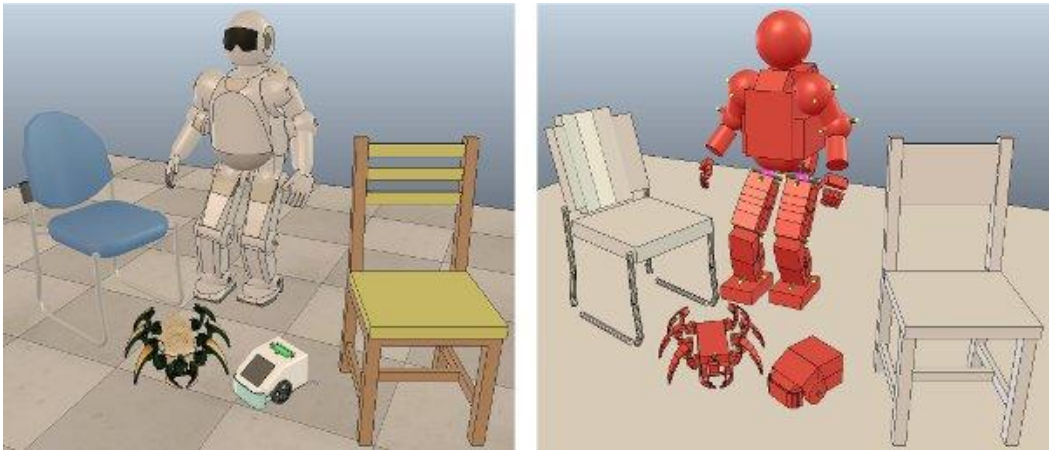


Ilustración 25. Visualización normal de la escena (izquierda) y visualización del contenido dinámico subyacente (derecha) [5].

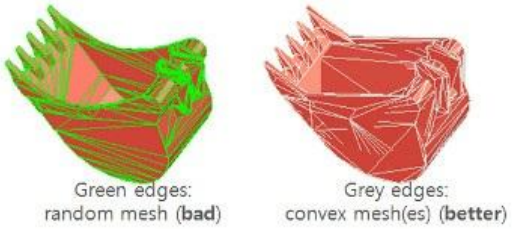


Ilustración 26. Codificación de colores para formas aleatorias y convexas en modo de visualización de contenido dinámico (adaptado de [5]).

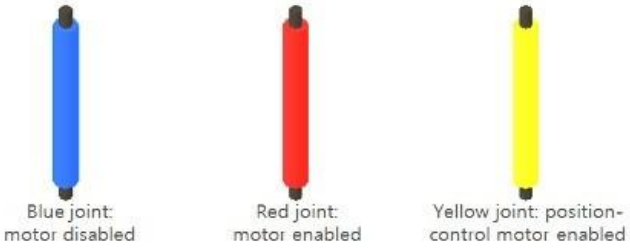


Ilustración 27. Codificación de colores para las articulaciones habilitadas dinámicamente en modo de visualización de contenido dinámico (adaptado de [5]).

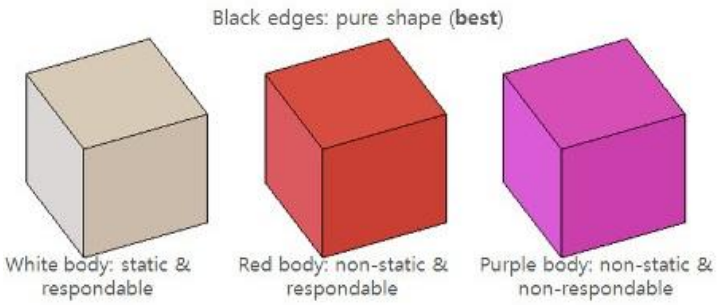


Ilustración 28. Codificación de colores para formas puras en modo de visualización de contenido dinámico (adaptado de [5]).

Los objetos dinámicos aparecerán en varios colores, dependiendo de su función o configuración como se muestra en las ilustraciones anteriores.

❖ Empleo de una estructura jerárquica simple.

Al construir un modelo que se pretende simular dinámicamente, se debe adjuntar todos los objetos estáticos a los objetos dinámicos (formas no estáticas y articulaciones/uniones de fuerza habilitadas dinámicamente). A continuación se ilustra como ejemplo la jerarquía de modelo del robot con ruedas mostrado en la ilustración 23:

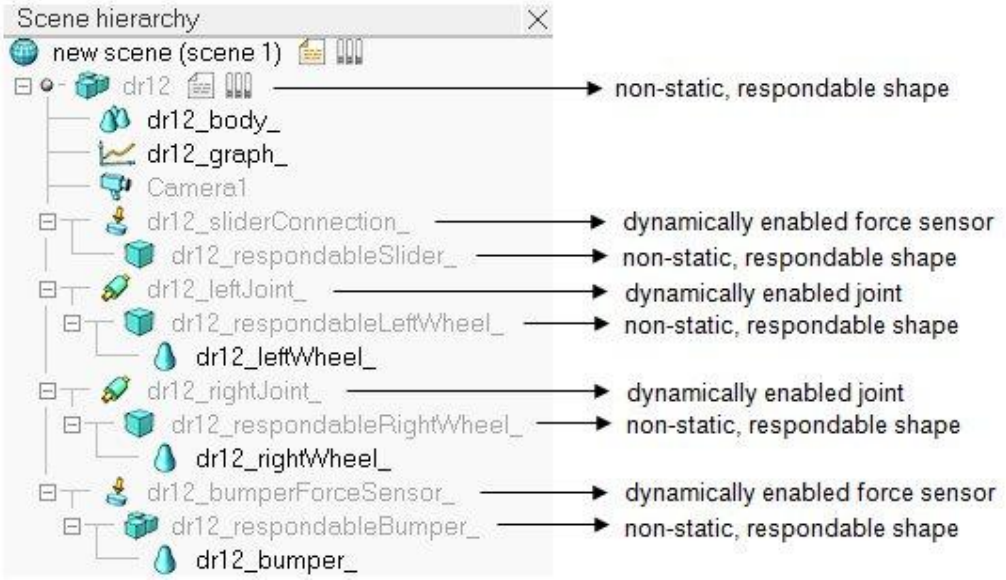


Ilustración 29. Elementos dinámicos del modelo robot de la ilustración 23 [5].

❖ Elección sopesada del objeto base del modelo.

Cuando se construye un modelo dinámico, incluso uno estático, siempre hay que sopesar qué rol jugará. ¿Se usará por sí solo? ¿Puede ser incorporado a otro modelo u objeto? ¿O puede aceptar otros modelos u objetos que se anexionar a él? Por ejemplo, si consideramos un modelo de robot y otro de pinza (*gripper*), es evidente que el de pinza

irá conectado como el efector final del robot, y no al revés (adjuntado el modelo de robot en el efector final del modelo de pinza). También puede darse que se deseen su uso por sí mismos. A continuación se muestran la visión de la jerarquía de sendos modelos dentro de la misma jerarquía de escena:

Ilustración 30. Ejemplo de definición del modelo de un robot y una pinza [5].

Obsérvese los objetos que cuelgan de la etiqueta del modelo de robot y pinza respectivamente. Esto indica que todos esos objetos están incorporados como parte de la definición del correspondiente modelo.

Ambos modelos funcionan correctamente por sí mismos; sin embargo, si se intenta anexar la pinza al efector final del robot haciendo clic en [*Menu bar > Edit > Make last selected object parent*], la pinza no permanecerá fija al robot durante la simulación. A continuación se presenta la jerarquía de escena anterior pero con la pinza conectada al robot como se ha descrito:

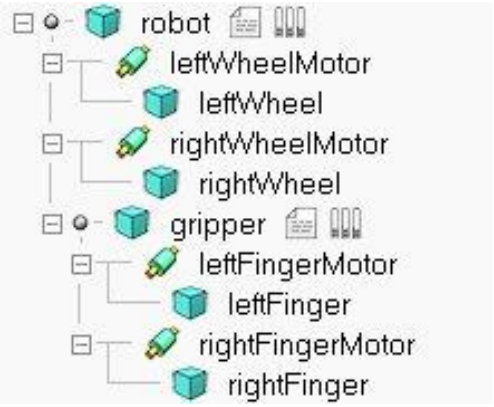


Ilustración 31. Ensamblaje inadecuado del modelo de pinza en el modelo robot [5].

La pinza se caerá por efecto de la gravedad durante la simulación si no se limita por una unión de fuerza a fin de vincularlo rígidamente al robot. Por ello, la definición correcta del modelo de robot debería incluir un punto de fijación para la pinza como se presenta en la ilustración 32, donde éste es una simple unión de fuerza el cual se expuso previamente. El

Ilustración 32. Ejemplo de definición del modelo de un robot con un punto de unión (attachmentPoint) [5].

montaje del modelo de pinza con el modelo de robot anterior da como resultado una pinza que permanece fija con respecto al robot como se expone en la próxima ilustración:

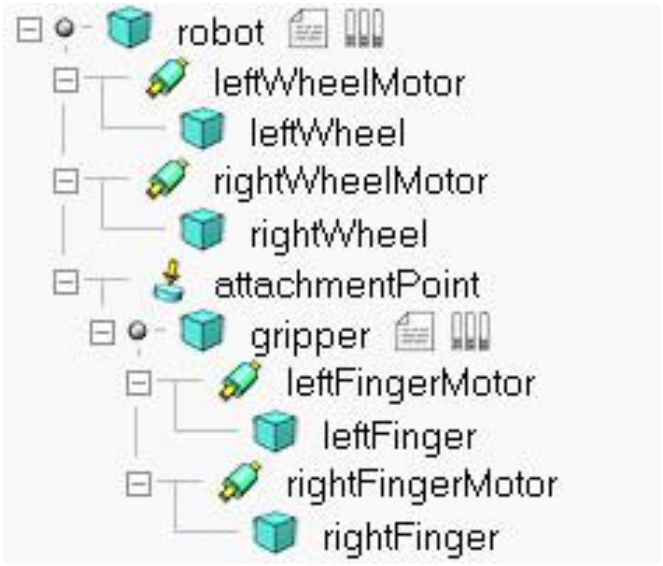


Ilustración 33. Ensamblaje funcional del modelo de robot-pinza [5].

- Empleo de tamaños razonables.

  Las formas que son muy largas y delgadas, o que son demasiado pequeñas podrían comportarse de manera extraña. Se debe de tratar de mantener los tamaños de cada dimensión respecto a las otras por encima de tres centímetros si es posible.

- Empleo de masas similares entre objetos vinculados.

  Cuando se unen dos formas por medio de una articulación o unión de fuerza habilitada dinámicamente, hay que cerciorarse de que las masas de las ambas formas no sean demasiado diferentes ($m_1 < 10 \cdot m_2$ y $m_2 < 10 \cdot m_1$); de lo contrario, la articulación o unión podría resultar poco firme y tambalear presentando grandes errores de posición/orientación. Además, se deben evitar formas de masas muy inferiores ya que no podrían ejercer grandes fuerzas sobre otras formas.

- Empleo de momentos de inercia relativamente grandes.

  Se debe procurar establecer las masas y momentos principales de inercia relativamente suficientemente grandes; de otro modo, las cadenas cinemáticas podrían ser difíciles de controlar y/o comportarse de manera extraña.

- Ubicación correcta de las formas estáticas.

  Jamás se debe anexar una forma estática reaccionable a un elemento dinámico. “Estático”

significa que la trayectoria de la forma no puede ser influenciada por ninguna colisión; pero si al mismo tiempo es reaccionable, significa que puede influir en las trayectorias de otras formas a través de la colisión resultando en simulaciones impredecibles.

Además, nunca se debe ensamblar una forma estática entre dos elementos dinámicos, pues interrumpiría el comportamiento lógico de la cadena cinemática. En todo caso, se debe agrupar a una de las formas dinámicas.

Ilustración 34. Ubicación incorrecta y correcta de una forma estática en una cadena cinemática [5].

2.6.3. Propiedades de objeto.

Las características de los objetos presentes en la escena se pueden visualizar por medio del diálogo de propiedades de objeto, accesible haciendo clic en [*Menu bar > Tools > Scene object properties*]. También se puede abrir dicho diálogo con un doble clic sobre el icono del objeto correspondiente en la jerarquía de escena, o a través de la barra de herramientas nº 2.

Se trata de unas propiedades sensibles al contexto cuyo contenido dependerá principalmente del estado de selección del objeto en escena, o sea, sólo se mostrarán las propiedades del último objeto seleccionado. Estas propiedades se dividen en dos: específicas y comunes; las primeras dependen del tipo de objeto seleccionado; y las segundas se tratan de propiedades universales a todos los tipos de objetos. Para su acceso, se disponen de sendos botones en la parte superior del cuadro de diálogo. Si no hubiese objeto alguno seleccionado, todos los elementos de dicho diálogo estarían inactivos [5].

1.3.1.1. Propiedades específicas.

Dependiendo del tipo de objeto seleccionado, tendríamos unas propiedades específicas u otras. En el desarrollo de este trabajo, solo se necesitarán conocer, interpretar y/o modificar las características de las formas, las articulaciones, las uniones de fuerza, las gráficas y los dummys.

2.6.3.1.1. Formas.

El diálogo de propiedades de forma muestra los ajustes y parámetros de la última forma seleccionada. Si se selecciona más de una, algunos parámetros se pueden copiar de la última seleccionada a las otras usando los botones de “Aplicar selección” (*Apply to selection*). A continuación se muestra una imagen que ilustra dicho diálogo:

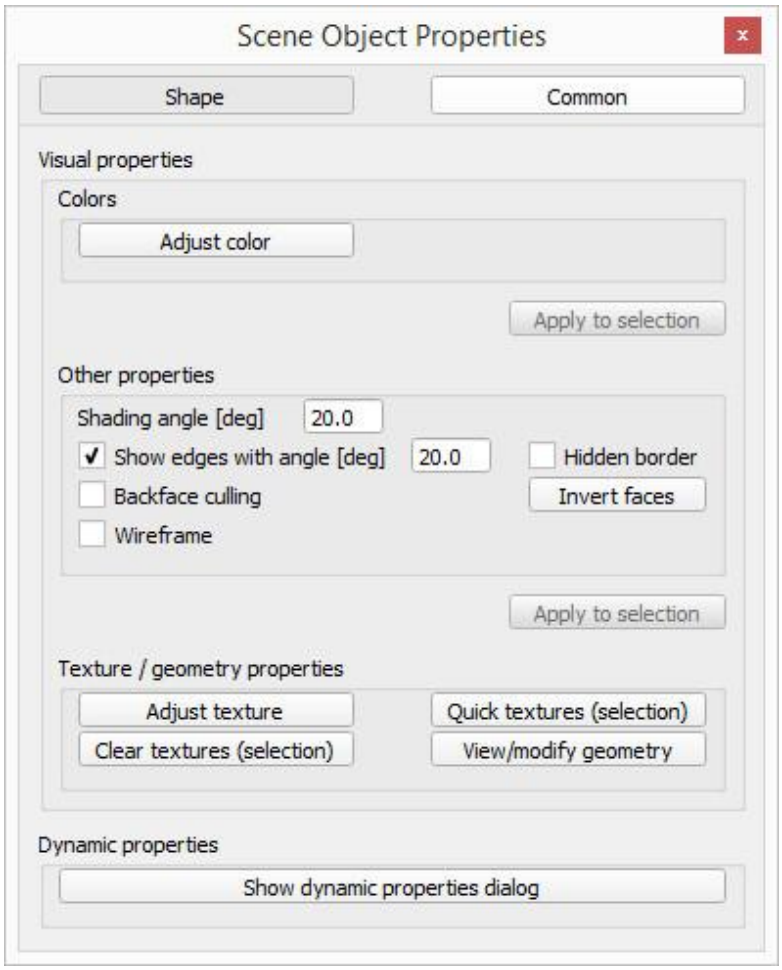


Ilustración 35. Diálogo de propiedades específicas de forma [5].

De este cuadro sólo se necesitan acceder a dos ajustes, dejando los demás por defecto, y son: el ajuste de color (*Adjust color*), para establecer los colores RGB de las formas; y los ajustes de las propiedades dinámicas (*Dynamic properties*), presentes en otro cuadro de diálogo accesibles a través del botón pertinente y cuyo diálogo se presenta en la ilustración 36.

El cuadro de diálogo de propiedades dinámicas es fundamental en el proceso de simulación, pues define el comportamiento dinámico de cada objeto en la escena. De este cuadro se requerirá modificar cinco ajustes, los cuales se describen a continuación [5]:

- El cuerpo es reaccionable (*Body is respondable*).

  Si está habilitado, entonces la forma producirá una reacción de colisión con otras formas reaccionables; pero sólo si las respectivas capas se superponen.

- Capas de colisión (*Respondable mask*).

  Indican cuándo se genera una respuesta de colisión, están disponibles siempre y cuando se halle el ajuste anterior marcado. Las capas están compuestas por dos valores de 8 bits,

local y global. Si dos formas que colisionan comparten cualquiera de sus padres directa o indirectamente, entonces las máscaras locales se utilizarán; de lo contrario, serán las máscaras globales las que se usen. La colisión se generará cuando las formas que se hallen interactuando compartan las mismas capas de colisión local o global según el caso de compartir padres o no.

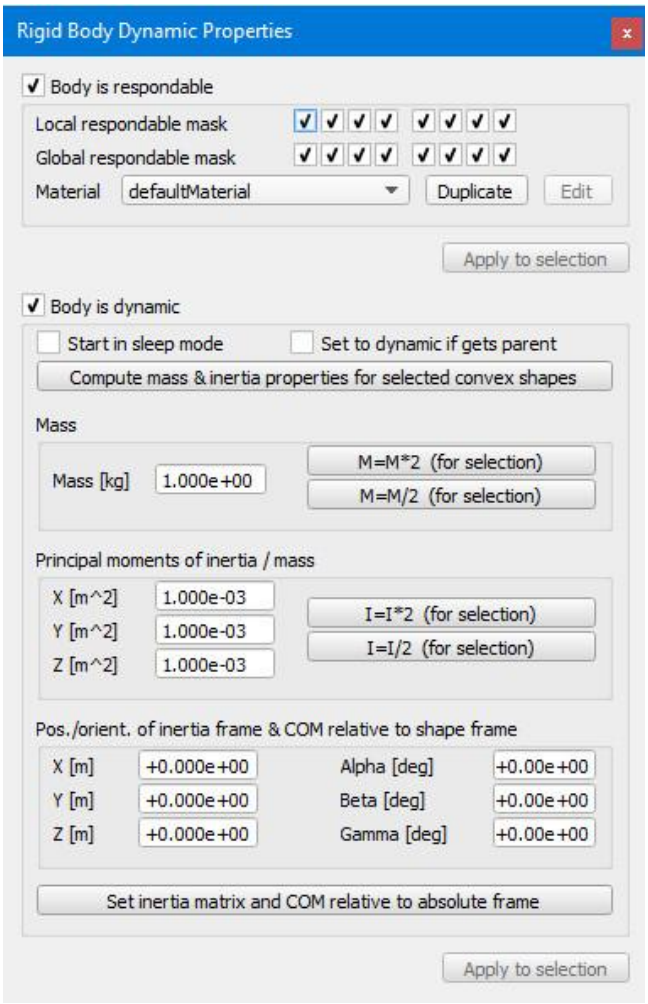


Ilustración 36. Diálogo de propiedades dinámicas de forma [5].

❖ El cuerpo es dinámico (*Body is dynamic*).

Cuando se encuentre activado, la posición y orientación de la forma se verán influenciadas en una simulación dinámica.

❖ Masa (*Mass*).

Las formas seleccionadas pueden tener sus masas fácilmente aumentadas o disminuidas por un factor de 2 con los botones dispuestos para ello, lo cual es muy conveniente a la hora de encontrar rápidamente parámetros de simulación estables mediante ensayo y error.

- Momentos principales de inercia/masa (*Principal moments of inertia/mass*).

  Esta característica se define como los momentos principales de inercia sin masa, es decir, dividido por la masa de la forma. Las formas seleccionadas pueden ajustar sus valores de inercia de forma análoga que la masa con el fin de encontrar los parámetros de simulación más estables.

2.6.3.1.2. Articulaciones.

El diálogo de propiedades de articulación muestra los ajustes y parámetros de la última articulación seleccionada. Si se selecciona más de una, algunos parámetros se pueden copiar de la última seleccionada a las otras usando los botones de "Aplicar selección" (*Apply to selection*). En la ilustración 37 se muestra dicho diálogo.

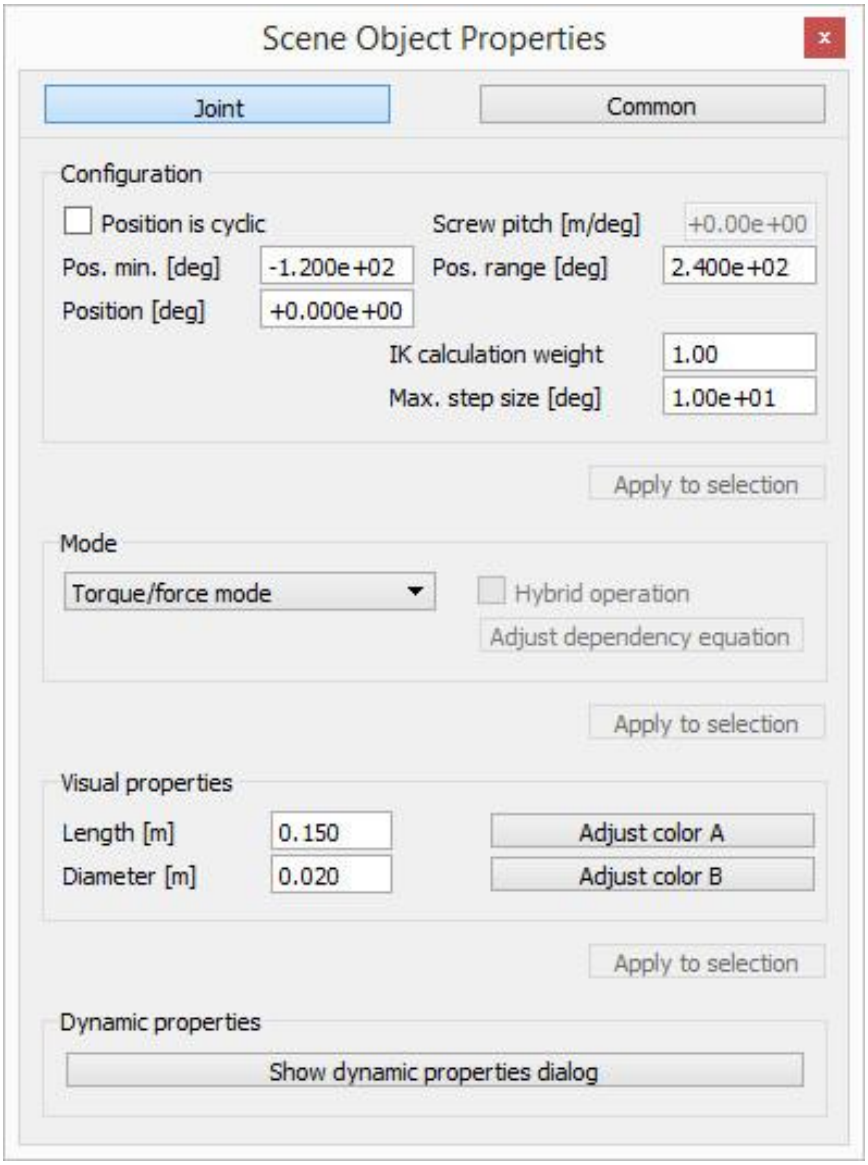


Ilustración 37. Diálogo de propiedades específicas de articulación [5].

Las propiedades que proporciona este diálogo determinan principalmente el modo y el rango de movimiento que tendrán los objetos ensamblados a través de las articulaciones. En nuestro caso, nos harán falta para el modelo del brazo robótico ajustar ocho propiedades de este cuadro de diálogo [5]:

- La posición es cíclica (*Position is cyclic*).

  Indica si la posición de la articulación es cíclica (varía entre -180º y +180º grados sin limitación). Solamente las articulaciones de revolución pueden ser cíclicas.

- Posición mínima (*Position mínimum*).

  Se trata del valor mínimo permitido de una articulación de revolución no cíclica o una articulación prismática.

- Rango de posición (*Position range*).

  Esta propiedad establece el rango de variación máximo permitido de una articulación de revolución no cíclica o una articulación prismática. Luego la posición de la articulación se encontrará limitada entre la posición mínima y la posición mínima más la posición.

- Modo (*Mode*).

  Este ajuste establece el modo de control de la articulación: pasivo, de cinemática inversa, dependiente, movimiento, y par/fuerza. En el presente trabajo se usarán tan solo los tres primeros.

- Funcionamiento híbrido (*Hybrid operation*).

  Cuando la articulación está en modo pasivo, cinemática inversa o dependiente; también existe la posibilidad de operar de forma híbrida. La operación híbrida permite que la articulación funcione de manera regular, pero además, justo antes de los cálculos dinámicos. En consecuencia, la posición actual de la articulación se copiará a la posición de la articulación objetivo y, a continuación, durante los cálculos dinámicos, la articulación será manejado como un motor en control de posición (si y solo si está habilitado dinámicamente).

- Ajustar la ecuación de dependencia (*Adjust dependency equation*).

  Si la articulación está en modo dependiente, entonces se puede especificar una ecuación lineal que enlace la articulación a otra. Los valores en esta sección del cuadro de diálogo vienen indicados en metros o radianes.

- Longitud (*Length*).

  Establece la longitud de la articulación, no tiene sentido funcional; tan sólo sirve para su representación visual, por lo que es útil en la visualización del contenido dinámico subyacente.

- Diámetro (*Diameter*).

  Establece el diámetro de la articulación, no tiene sentido funcional, tan sólo sirve para su representación visual al igual que en el punto anterior.

Cabe pensar que pudiera ser necesario los ajustes correspondientes a las propiedades dinámicas, pero como una articulación es una elemento dinámico de por sí, los ajustes específicos vistos hacen ese papel. Sin embargo, el apartado de propiedades dinámicas en este cuadro de diálogo viene a profundizar aún más en caso de elegir un modo de control de movimiento o de par/fuerza.

2.6.3.1.3. Uniones de fuerza.

El diálogo de propiedades de unión de fuerza muestra los ajustes y parámetros de la última unión seleccionada. Si se selecciona más de una, algunos parámetros se pueden copiar de la última seleccionada a las otras usando los botones de "Aplicar selección" (*Apply to selection*). A continuación se muestra una imagen que ilustra dicho diálogo:

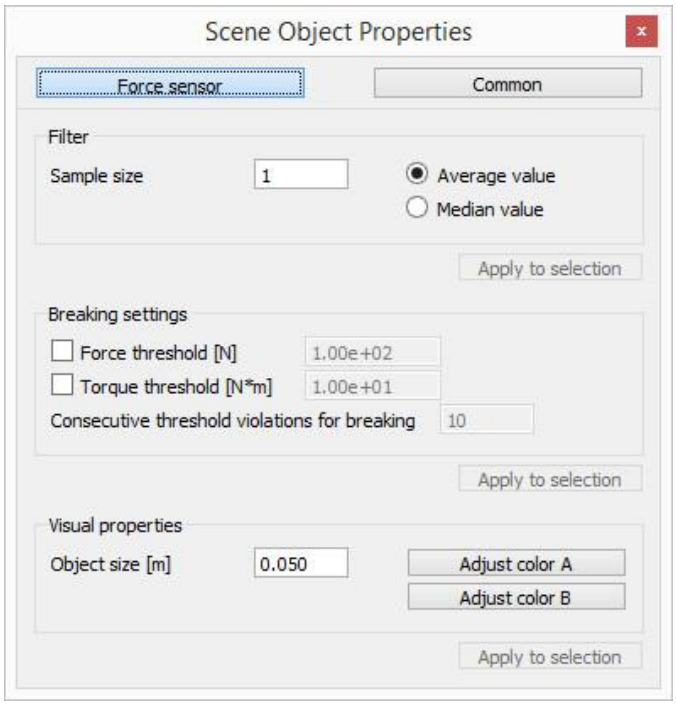


Ilustración 38. Diálogo de propiedades específicas de unión de fuerza [5].

La importancia de este cuadro de diálogo reside en la posibilidad de establecer unos umbrales de

fuerza/par límites a partir de los cuales la unión entre los elementos dinámicos se rompe. Esos ajustes se establecen activando las pestañas correspondientes (Force/Torque threshold) y estableciendo el valor que se requiera en el cuadro habilitado a tal efecto en valores de Newton y Newton por metro para fuerza y par respectivamente.

En nuestro caso no se usará ninguna propiedad y se dejarán todas por defecto; no obstante, debía resaltarse la importancia de los ajustes mencionados debido a que está el presente trabajo dispuesto para que en cualquier momento pudiese ser necesario en futuros experimentos.

2.6.3.1.4. Dummys.

Un dummy es un objeto ficticio, el más simple disponible, se define como un punto con orientación, y puede ser visto como un marco de referencia. Utilizados solos, no son tan útiles; sin embargo, cuando se utilizan junto a otros objetos o módulos de cálculo como veremos, pueden ser de crucial importancia. En consecuencia, puede decirse que son objetos multiusos o auxiliares. La figura siguiente muestra el icono del dummy en la escena de jerarquía y su representación visual en la escena:

Ilustración 39. Representación de un dummy en la jerarquía de escena y en la escena (adaptado de [5]).

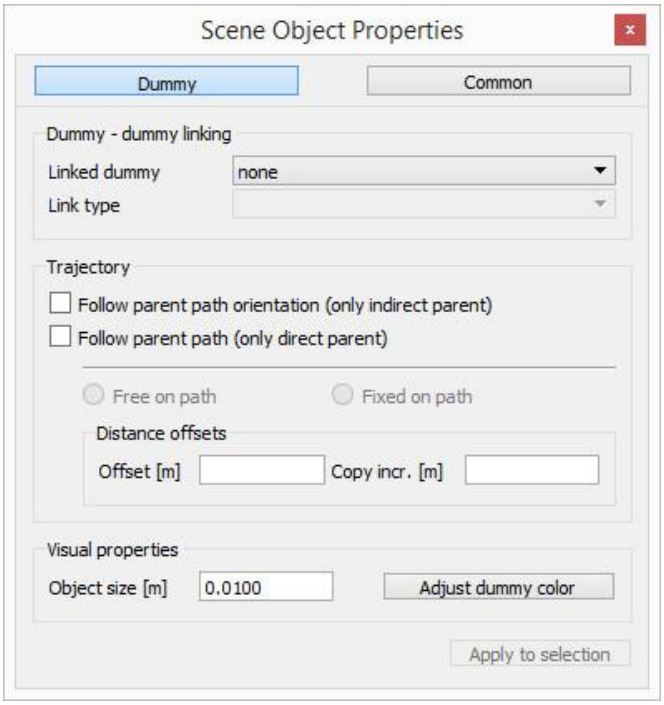


Ilustración 40. Diálogo de propiedades específicas de dummy [5].

Entre las múltiples funciones que puede desempeñar, nosotros los usaremos como apoyo para la ubicación de modelos y para especificar las posiciones/orientaciones finales del efector final en cálculos de cinemática inversa a través de un vínculo dummy-dummy.

El diálogo de propiedades de dummy muestra los ajustes y parámetros del último seleccionado, y puede apreciarse en la ilustración 40. De los ajustes que ofrece, sólo se requerirán en el presente trabajo la que concierte a la vinculación dummy-dummy para los cálculos mencionados previamente.

2.6.3.2. Propiedades comunes.

En este apartado se tratan con propiedades universales a todos los objetos, luego cualquier objeto seleccionado sea del tipo que sea presentará las características adjuntas de este cuadro de diálogo.

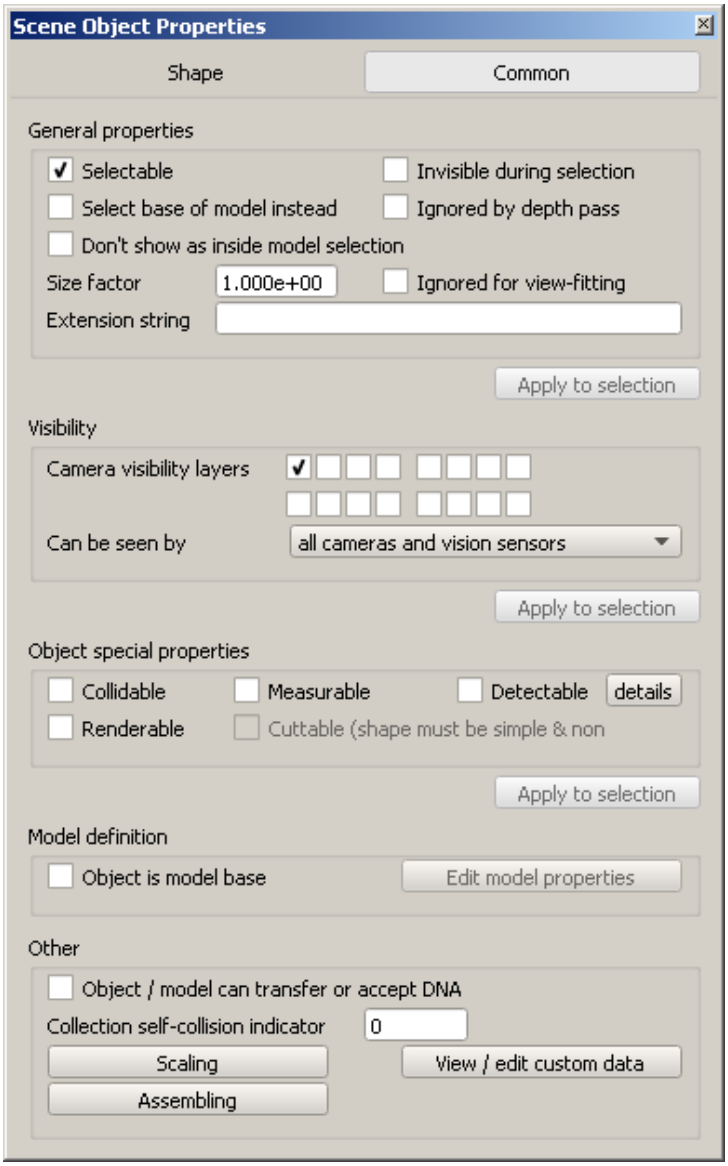


Ilustración 41. Diálogo de propiedades comunes de objeto.

Al igual que todos los mencionados anteriormente, presenta la posibilidad de copiar algunos parámetros del último objeto seleccionado con ayuda de los botones correspondientes de "Aplicar selección" (*Apply to selection*) de los subapartados disponibles de dicho diálogo.

Las propiedades comunes de un objeto son accesibles haciendo doble clic en su icono representativo del panel de jerarquía – denominado *Object icon* en la ilustración nº 8– para acceder al cuadro de diálogo de propiedades del objeto, seleccionando la sección correspondiente (*Common*). Dichas propiedades nos permitirán crear los modelos de la escena, por lo que previamente debemos conocer los ajustes que puedan resultarnos necesarios para ello, que para lo que se requiere en el presente trabajo serán las seis que se citan a continuación, dejando en consecuencia las demás desactivadas [5].

- Seleccionable (*Selectable*).

  Indica si se puede seleccionar el objeto en la escena. A pesar de ello, un objeto siempre se puede seleccionar en la jerarquía de escenas o pulsando los botones de teclado *Ctrl*/*Shift* durante su selección

- Seleccione la base del modelo en su lugar (*Select base of model instead*).

  Cuando se halle activada esta opción, al seleccionar el objeto en la escena se seleccionará su primer objeto padre marcado como "el objeto es la base del modelo" (véase más adelante). Esta propiedad es conveniente cuando se protege un modelo de manipulaciones defectuosas, permitiendo que sea manejado como una sola entidad junto a otros objetos.

- No mostrar como dentro de la selección del modelo (*Don't show as inside model selection*).

  Cuando se selecciona esta propiedad y el objeto forma parte de un modelo, el cuadro de selección del modelo no abarcará ese objeto, lo cual es útil para objetos invisibles que podrían hacer que dicho cuadro apareciera demasiado grande. Aun así, esta propiedad no tiene ningún efecto funcional.

- Capas de visibilidad de la cámara (*Camera visibility layers*).

  Cada objeto en V-REP puede asignarse a una o varias capas de visibilidad por lo que esta herramienta es una manera conveniente de mostrar u ocultar partes específicas de una escena. Cada objeto de escena se puede asignar a cualquiera de las 16 capas de visibilidad disponibles. Si los objetos son visibles, estarán asignados según el tipo, de la capa 1 a la 8 contando desde la izquierda; si por el contrario son ocultos, como ocurre con los objetos simulados dinámicamente, se asignan a la capa de la 2ª fila dentro de la misma columna correspondiente desde la 9 hasta la 16. De forma predeterminada, según sean los objetos

visibles/ocultas, se asigna una forma a la primera/novena capa, una articulación a la segunda/décima capa, un dummy a la tercera/undécima capa, y una unión de fuerza a la cuarta/duodécima capa.

❖ El objeto es la base del modelo (*Object is model base*).

Esta propiedad indica si el objeto debe actuar como la base de un modelo, aportándole de este modo propiedades especiales; como por ejemplo es la característica única de guardar o copiar el objeto que ofrece, pues también guardará/copiará automáticamente todos sus hijos y los hijos de sus hijos, los scripts asociados, los paneles de usuario, etc.

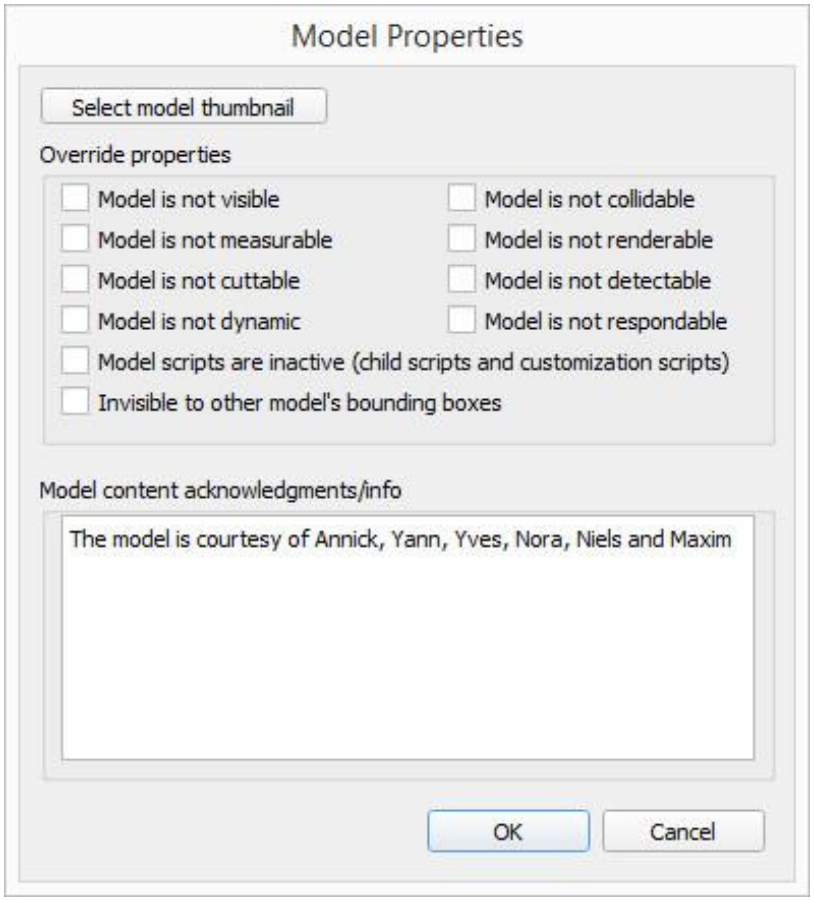


Ilustración 42. Diálogo de propiedades de modelo [5].

Además, permite el acceso al botón "Editar propiedades del modelo" (*Edit model properties*) que abre un cuadro de diálogo subyacente (mostrado en la ilustración 42), donde se puede definir propiedades especiales de anulación en el cuadro de diálogo de modelo que se abre y al cual se puede acceder también haciendo doble clic en la etiqueta de modelo en la escena de jerarquía; por ejemplo, hacer que todo el modelo sea invisible, no medible, estático, etc. Esto permite deshabilitar rápidamente algunas propiedades para todos los objetos definidos en el modelo, que aunque en nuestro caso no será necesario su acceso a él en principio, es importante tenerlo en consideración por si es necesario en

un momento dado para los experimentos que se puedan proponer a futuro.

El cuadro de diálogo de propiedades de modelo permite también seleccionar la miniatura del modelo que representará al modelo en el navegador de modelos, y añadir información relacionada que se mostrará automáticamente cuando se abra el archivo relacionado o se arrastre y se suelte en la escena para cargarlo automáticamente.

- Objeto/modelo puede transferir o aceptar ADN (*Object/model can transfer or accept DNA*).

  Cuando esta característica está habilitada para un objeto o un modelo, comparte un mismo identificador con todas sus copias posibilitando "transferir su ADN" a todos sus hermanos; de este modo, cuando en una de las copias se realice una modificación, haciendo clic en el botón de "Transferencia de ADN a los hermanos" (*Transfer DNA to siblings*) disponible en la barra de herramientas nº 1, se modificarán todas las demás copias.

En consecuencia, un modelo se define teniendo en cuenta los pasos siguientes [5]:

a) Adjuntar todos los objetos, que pertenecen lógicamente al modelo, a un modelo base; de modo que el objeto base sea la base del árbol del modelo.
b) Dentro de las propiedades comunes del objeto base, en el apartado de definición de modelo (*Model definition*) debe marcarse la pestaña *Object is model base*.
c) En el apartado posterior (*Other*), se debe marcar la casilla correspondiente a "el objeto/modelo puede transferir o aceptar ADN" (*Object/model can transfer or accept DNA*). Esto simplificará las copias del modelo si se modificase en una etapa posterior.
d) Para todos los objetos del modelo, excepto para el objeto base, debe mar-carse la opción "seleccionar la base del modelo en su lugar" (*Select base of model instead*) del apartado de propiedades generales (*General properties*). Esto protegerá el modelo al no permitir la selección individual de los objetos que lo componen, y se podrá manipular el modelo como una única entidad.
e) Para todos los objetos que normalmente no son visibles, debe marcarse la casilla de "No mostrar como dentro del modelo" (*Don't show as inside model selection*), lo cual hará que el cuadro de selección del modelo aparezca en el tamaño correcto alrededor de él.
f) En general, todos los objetos del modelo que sean estáticos y no reaccionables serán objetos visibles, mientras que los objetos dinámicos y/o reaccionables se hallarán en su capa de invisibilidad correspondiente.

En resumen, en función de la visibilidad e invisibilidad del objeto que se trate, sea o no modelo; y en función de si el modelo forma parte de otro o no, se representa a continuación una tabla que simplifica las propiedades generales (*General properties*) a considerar:

		Selectable	***Select base of model instead***	***Don't show as inside model selection***
Modelo	Único	☑	☐	☐
	Hijo de otro	☑	☑	☑
Objeto	Visible	☑	☑	☐
	Oculto	☐	☐	☑

Tabla 1. Resumen de las propiedades generales universales según la naturaleza del objeto.

2.7. Scripts.

V-REP es un simulador altamente personalizable pues cada aspecto de una simulación puede ser modificado a voluntad [5]. Además, el propio simulador puede ser personalizado y adaptado para comportarse exactamente como se desee, lo cual se consigue a través de la API, las cuales se apoyan en los scripts como medio para el control de la escena, el modelo o el mismo simulador.

Este enfoque de programación usa el lenguaje Lua, definido como un lenguaje programación extensible diseñado para una programación procedimental general con utilidades para la descripción de datos [6]. Gracias a ese carácter extensible, se puede combinar con la API de V-REP compuesta por varios cientos de funciones reconocidas por sus prefijos "sim-".

Un script incrustado es un script embebido en una escena o modelo, es decir, un script que forma parte de la escena y que se guardará y cargará junto con el resto de la escena o modelo. Existen dos grupos diferenciados: los scripts que se ejecutan solo durante la simulación y los que se pueden ejecutar además fuera de ella; de los cuales, nosotros usaremos el primer grupo. Los scripts de simulación a su vez se subdividen en más grupos, de los cuales serán necesarios conocer el script principal y los scripts secundarios.

- Script principal (*Main script*).

  Por defecto, cada escena tiene un script de este tipo que maneja toda la funcionalidad de la simulación, o sea, se encarga de llamar a los otros tipos de scripts de simulación; por ello, sin script principal una simulación no se puede ejecutar. Puede personalizarse, pero es recomendable hacer todo el trabajo de personalización en un script secundario y por tanto no se modificará en absoluto.

- Scripts secundarios (*Child scripts*).

  Cada objeto de escena puede estar asociado con un script secundario que maneje una

parte específica de la simulación, por ello el uso común que poseen en V-REP es el control de modelos. Pueden tener una lista de parámetros de simulación conectados a ellos (*Script simulation parameters*) que se pueden utilizar como una forma rápida de ajustar valores de un modelo de simulación específico sin tener que abrir el script y escribirlo, como por ejemplo la velocidad máxima de un robot o la densidad de las partículas que genera; de modo que los scripts reaccionan a esa modificación y ajustan su comportamiento en consecuencia.

Ilustración 43. Script principal [5].

Ilustración 44. Lista de parámetros de simulación de script secundario vacía (izquierda) y no vacía (derecha).

Ilustración 45. Script secundario no secuencial (izquierda) y secuencial (derecha) [5].

Los scripts secundarios pueden dividirse en dos tipos:

a) <u>No secuencial.</u>

Se denominan así a los scripts de paso a través, lo cual significa que cada vez que se les llama, debe realizar alguna tarea y luego devolver el control. Estos scripts se ejecutan como funciones que son llamados por el script principal, y siempre que sea posible son los que se deben usar. En nuestro caso, serán usados en los modelos «8 DoF Manipulator» y «SimulationTimeDisplay», y en el objeto «propeller»; todos ellos tratados más adelante.

b) <u>Secuencial.</u>

Se definen como scripts lanzados en un subproceso, cuyo lanzamiento es manejado por el código del script principal por defecto. Al contrario que los scripts no secuenciales, toman el control de la simulación y una vez que terminan lo devuelven al script principal; por tanto, son scripts que hay que manejar con cuidados pues si no devuelven el control como es debido, la simulación completa se detiene. En nuestro caso, será usado en el modelo «Pop-Up» del cual se mencionará a posteriori.

Entre los scripts que se pueden ejecutar tanto dentro como fuera de simulación, necesitamos conocer los scripts de personalización (*Cutomization scripts*), que se tratan de scripts incrustados al igual que los anteriores y que se pueden utilizar para personalizar una escena de simulación en gran medida. Se encuentran unidos o asociados a objetos de escena, en nuestro caso al modelo que usaremos denominado «Center of mass visualization tool» que veremos más adelante.

Ilustración 46. Script de personalización [5].

Las características más destacables de este tipo de script son:

- No es secuencial.
- Se ejecuta todo el tiempo, o sea, cuando la simulación se está ejecutando así como cuando no.
- Están conectados o asociados a objetos de escena.

Todos los scripts mencionados se escriben en el lenguaje Lua como se aludió anteriormente, para lo cual en este TFG se ha requerido de su aprendizaje a través de la literatura [6] y [7], junto con el de las series de funciones API de V-REP provenientes de la web [8] para lograr construir las sentencias de control necesarias que permitan el control de la simulación de los diferentes modelos que en el siguiente capítulo se pretenden describir.

3. MODELOS DESARROLLADOS.

En este capítulo, tras configurar la escena y presentar las propiedades y consideraciones de diseño a tener en cuenta, se presentan los modelos como resultado de ello. La reproducción 3D del satélite y el cuerpo del propulsor se han realizado con ayuda del software SolidWorks, mientras que los demás se han obtenido gracias a V-REP.

La solución de diseño en 3D completa SolidWorks [9], permite crear, simular, comunicar y gestionar los diseños de los productos. Incorpora potentes herramientas de diseño entre las que se incluyen funciones de piezas, ensamblaje y dibujo, junto con la simulación, la estimación de costes, el renderizado, la animación y la gestión de datos integrados.

3.1. Introducción.

Los modelos de los satélites artificiales han sido desarrollados acorde a la misión expresada en

el apartado 1.3 apoyándose de las referencias disponibles hoy en día para poder reproducir unos modelos lo más reales posibles en cuanto a dimensiones, pesos y características generales que logren simular lo mejor posible el entorno de simulación planteado.

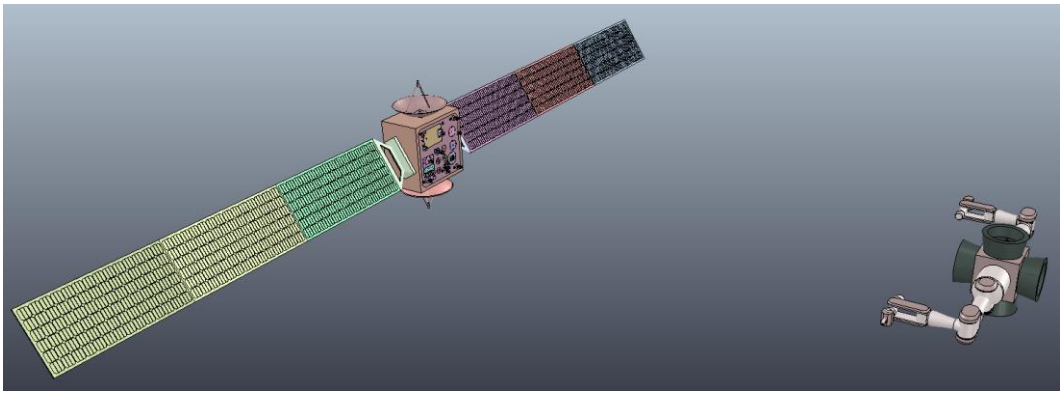


Ilustración 47. Satélite cliente (izquierda) y satélite servidor (derecha) en la escena.

La determinación de dichas características no ha sido suficiente a la hora de definir el entorno, ya que se ha debido ajustar además los momentos de inercia experimentalmente en ausencia de datos y conforme a los requerimientos de la escena para lograr la estabilidad dinámica de los modelos en el espacio simulado.

3.2. Satélite cliente.

Representa el satélite que ha finalizado su vida útil y vaga a la deriva en una órbita operacional donde se ubican satélites en funcionamiento a riesgo de impactar con él, por lo que debe ser retirado como se explicó en el primer capítulo.

Ilustración 48. Reproducción en 3D del satélite cliente en SolidWorks.

Puesto que la intención de este trabajo es reproducir la fase de preagarre además de interactuar con él, no se ha añadido al satélite el elemento de agarre al no ser necesario para esta simulación.

En la construcción y reproducción 3D del satélite se ha usado las funciones de piezas, ensamblaje y renderizado[2] sirviéndose de la literatura [10], [11] y [12] para el aprendizaje y el manejo del programa SolidWorks. Tras culminar, se ha importado a V-REP donde se ha escalado a 1/2.

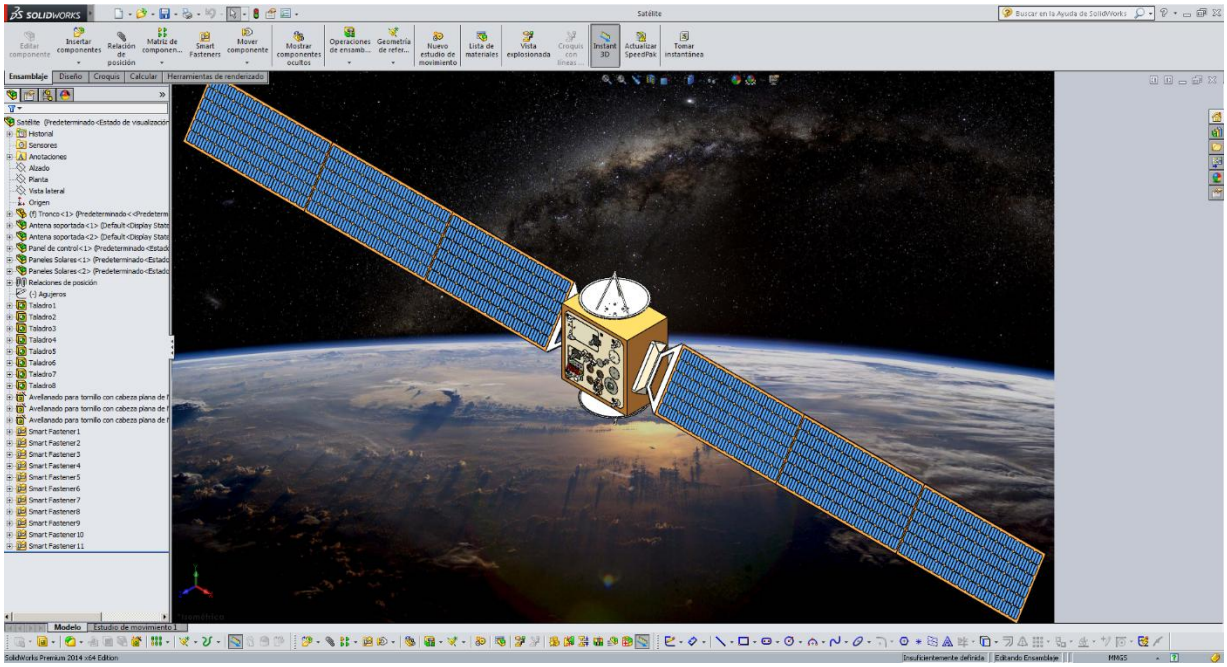


Ilustración 49. Interfaz de usuario de SolidWorks: gestor de objetos de ensamblaje (izquierda), pestañas de funciones (superior) y reproducción 3D del satélite (centro).

La realización del modelo en SolidWorks parte de la creación, mediante la función de piezas, de los paneles solares, las antenas, el panel de control y el tronco al que van anexados los tres primeros. A continuación, en la función de ensamblaje se han unido dichos elementos usando el tronco como cuerpo principal –la notación "(f)" que aparece en el gestor de objetos de la ilustración 49 lo denota– y se ha agregado los cierres al ensamblaje utilizando la biblioteca *SolidWorks ToolBox* de componentes mecánicos estándar mediante la aplicación *Smart Fasteners*. Finalmente, mediante la función de renderizado se ha realizado la reproducción realista del modelo lográndose el resultado de la ilustración 48.

Las características del satélite se han supuesto en orden de las referencias disponibles con un peso de 750 kg como el de los satélites "THEOS" [13] y "Rocsat 2" [14], y unas dimensiones de 0,82 x 0,48 x 0,35 m con una longitud total con los paneles solares extendidos de 5,12 m proporcionales todas ellas a las de los satélites "Venesat-1" [15] y los de la serie TDRS [16].

Para alcanzar la estabilidad dinámica deseada del modelo, se ha considerado los momentos de

[2] Denominación que recibe en los programas de diseño asistido por computador (CAD) la reproducción realista de un objeto.

inercia, según el ajuste experimental establecido en V-REP para hallar los parámetros de simulación más estables, siguientes: I_x = 1783 mm^2, I_y = 8449 mm^2 y I_z = 7293 mm^2.

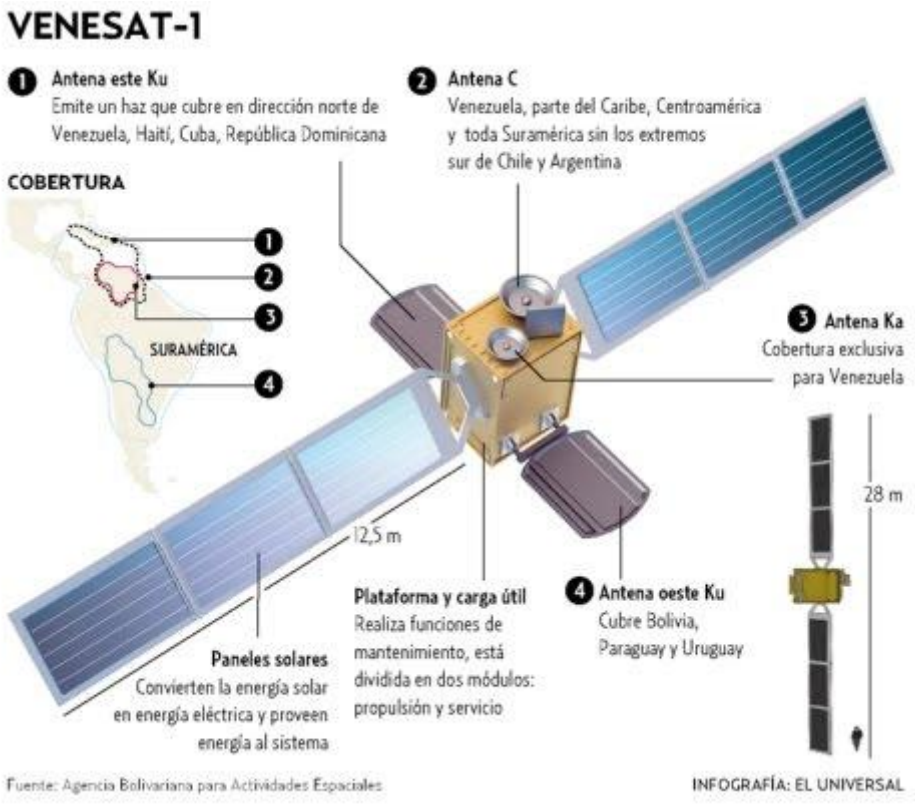


Ilustración 50. Esquema del Venesat-1 [15].

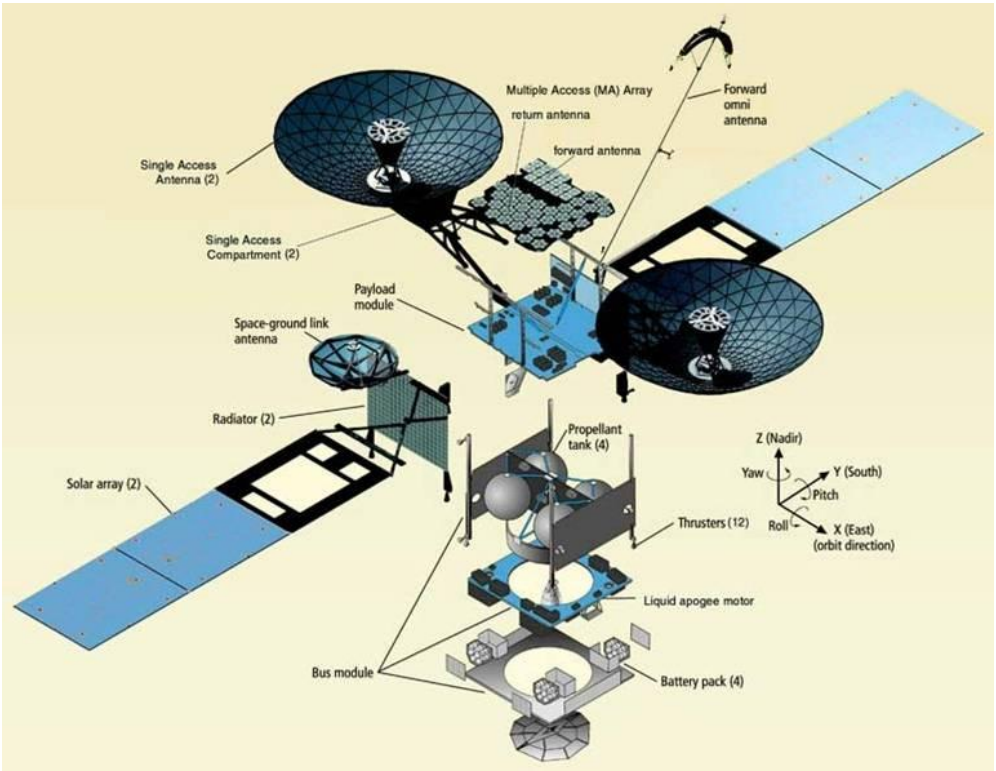


Ilustración 51. Esquema de los satélites TDRS-K, L, M [16].

3.2.1. Paneles solares.

Este elemento ensamblado a un lado y a otro del satélite cliente se ha realizado mediante el ensamblaje de los diferentes paneles solares que lo componen alcanzando una longitud total de 2,32 m con la ayuda de [17] además de las referencias bibliográficas señaladas anteriormente.

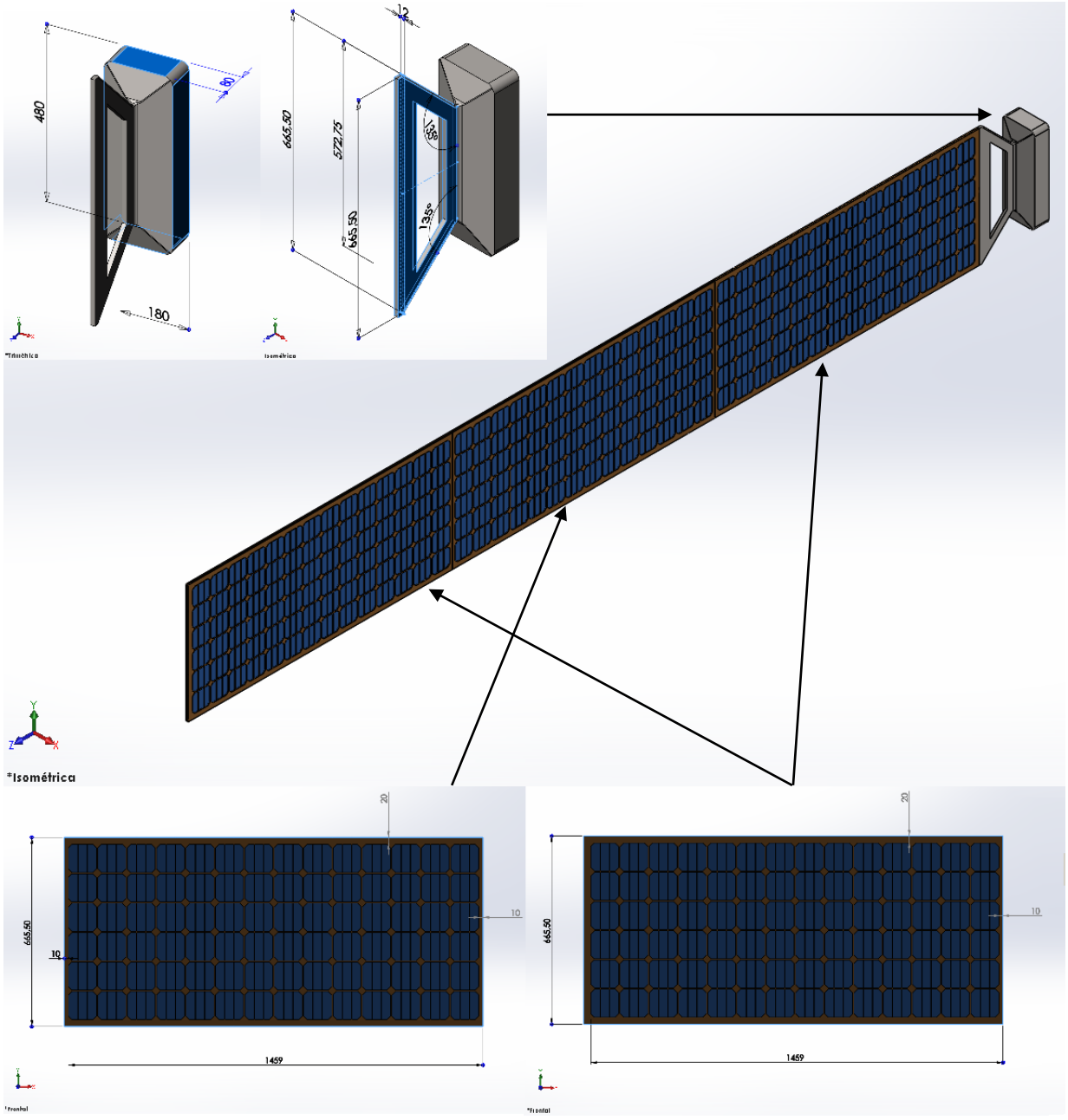


Ilustración 52. Ensamblaje de los paneles solares.

3.2.2. Antenas.

Este elemento ensamblado arriba y abajo del satélite cliente se ha realizado mediante el ensam-

blaje de la misma antena más un soporte, teniendo unas dimensiones en V-REP, tras su escalado, de 0,3 m de diámetro y 0,23 m de altura con ayuda de [18] además de las referencias bibliográficas señaladas en el apartado 3.2.

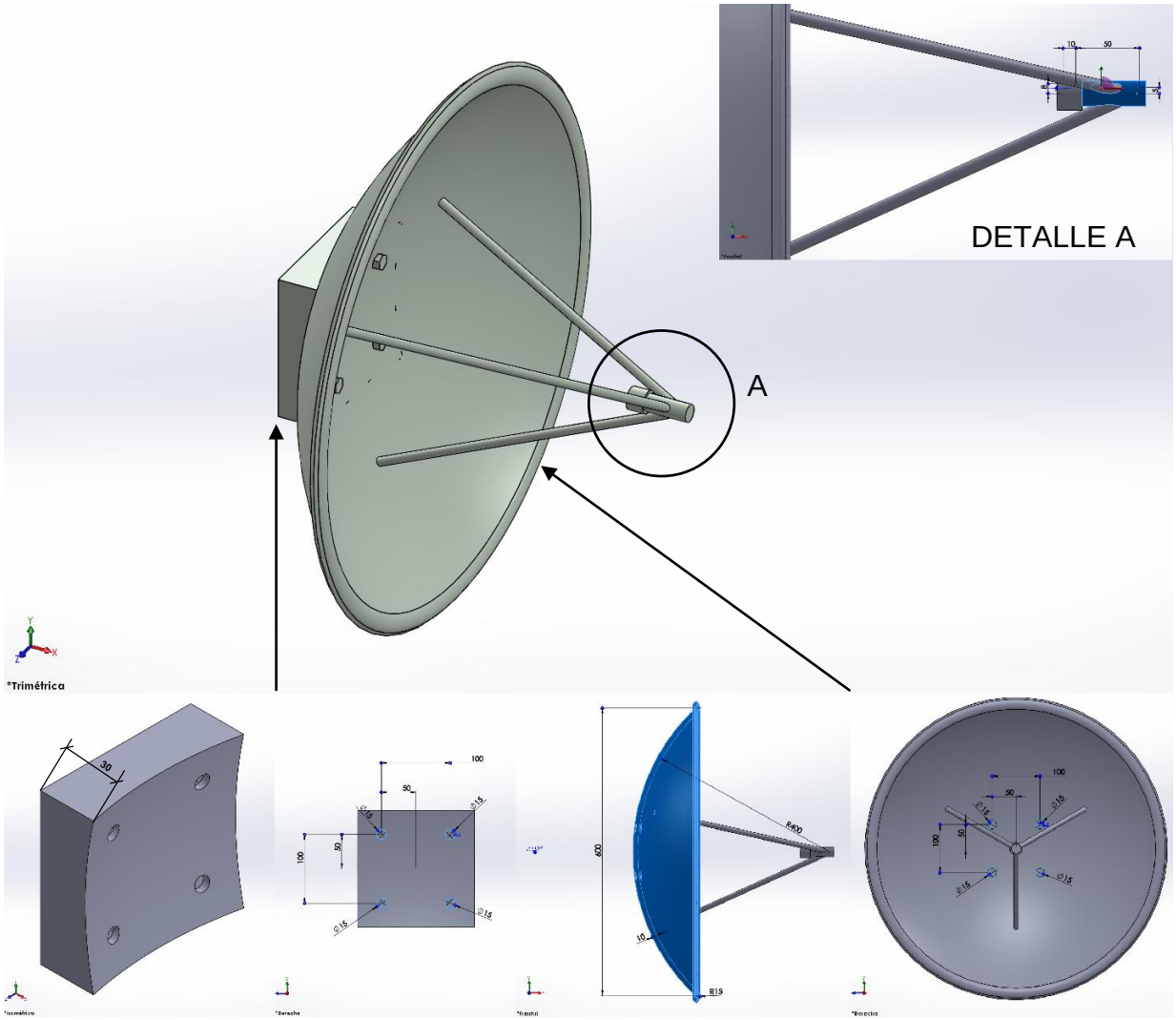


Ilustración 53. Ensamblaje de la antena.

3.2.3. Panel de control.

Este elemento se ha realizado mediante el ensamblaje de las diferentes piezas que lo componen tratando de ser lo más fiel posible al panel de mando presentado en la web de ESA Telerobotics [19], que en ausencia de sus medidas, se han supuesto todas tratando de parecerse en apariencia a dicho panel. Para ello se ha servido de la ayuda de [20] además de las referencias señaladas.

El panel de ESA Telerobotics se define como una célula de trabajo que contiene muchas características susceptibles de ser configuradas a libertad para analizar diversas tareas tales como la de enclavamiento en el agujero, la activación de la llave de torsión, cajones que pueden ser extraidos, puertas que se pueden abrir (girar), pestillos que se pueden bloquear, interruptores que

pueden ser pulsados y una configuración de tarea de acoplamiento/desacoplamiento de conector [19].

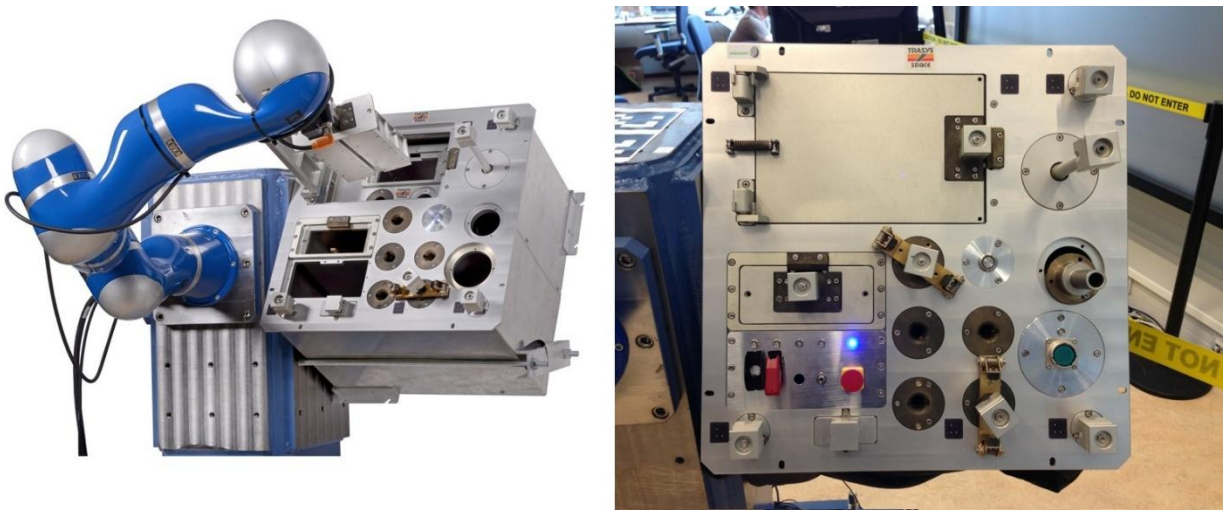


Ilustración 54. Panel de control de ESA Telerobotics (adaptado de [19]).

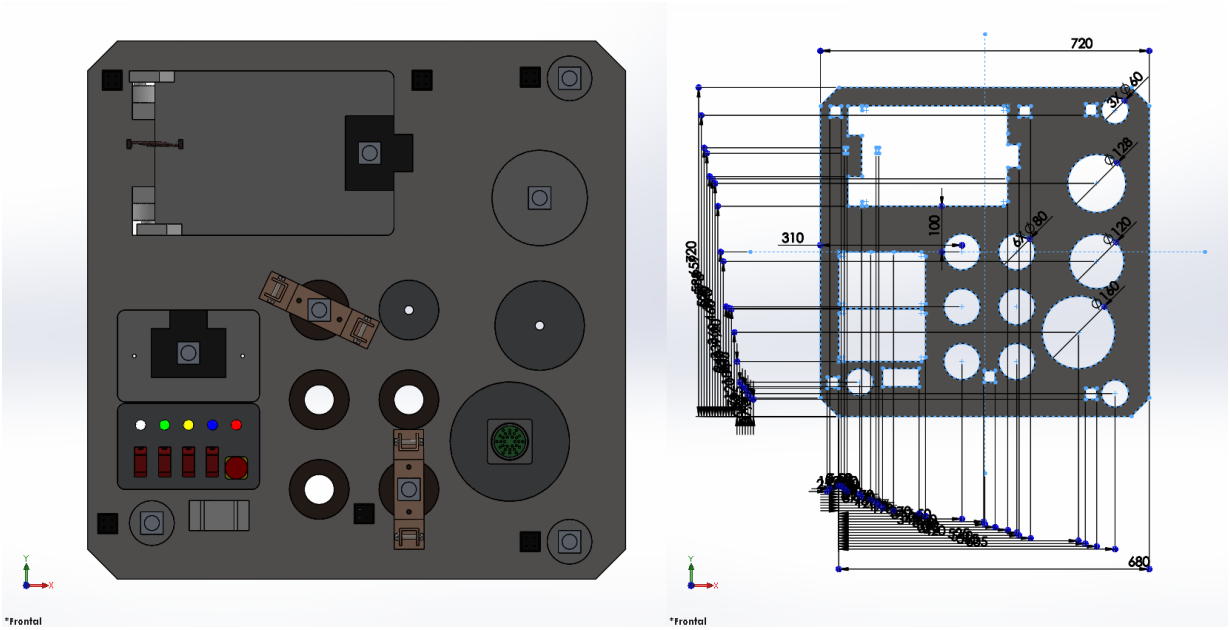


Ilustración 55. Alzado del panel de control en SolidWorks.

El panel de control se compone de 51 piezas, siendo el elemento que más tiempo lleva realizar en consecuencia. No obstante, han sido construidas muchas de ellas a partir de varias instancias derivadas de un mismo archivo de pieza gracias a la herramienta *ConfigurationManager*.

La herramienta *ConfigurationManager* de SolidWorks es una forma de crear, seleccionar y ver múltiples configuraciones de piezas y de ensamblajes [10]; por ello tan sólo han sido falta construir 19 archivos de pieza solamente, y mediante la generación de las diferentes configuraciones necesarias de éstas piezas guardadas en un solo archivo de pieza gracias a dicha

herramienta, se han podido realizar las 51 de las que consta el panel de forma práctica y sencilla.

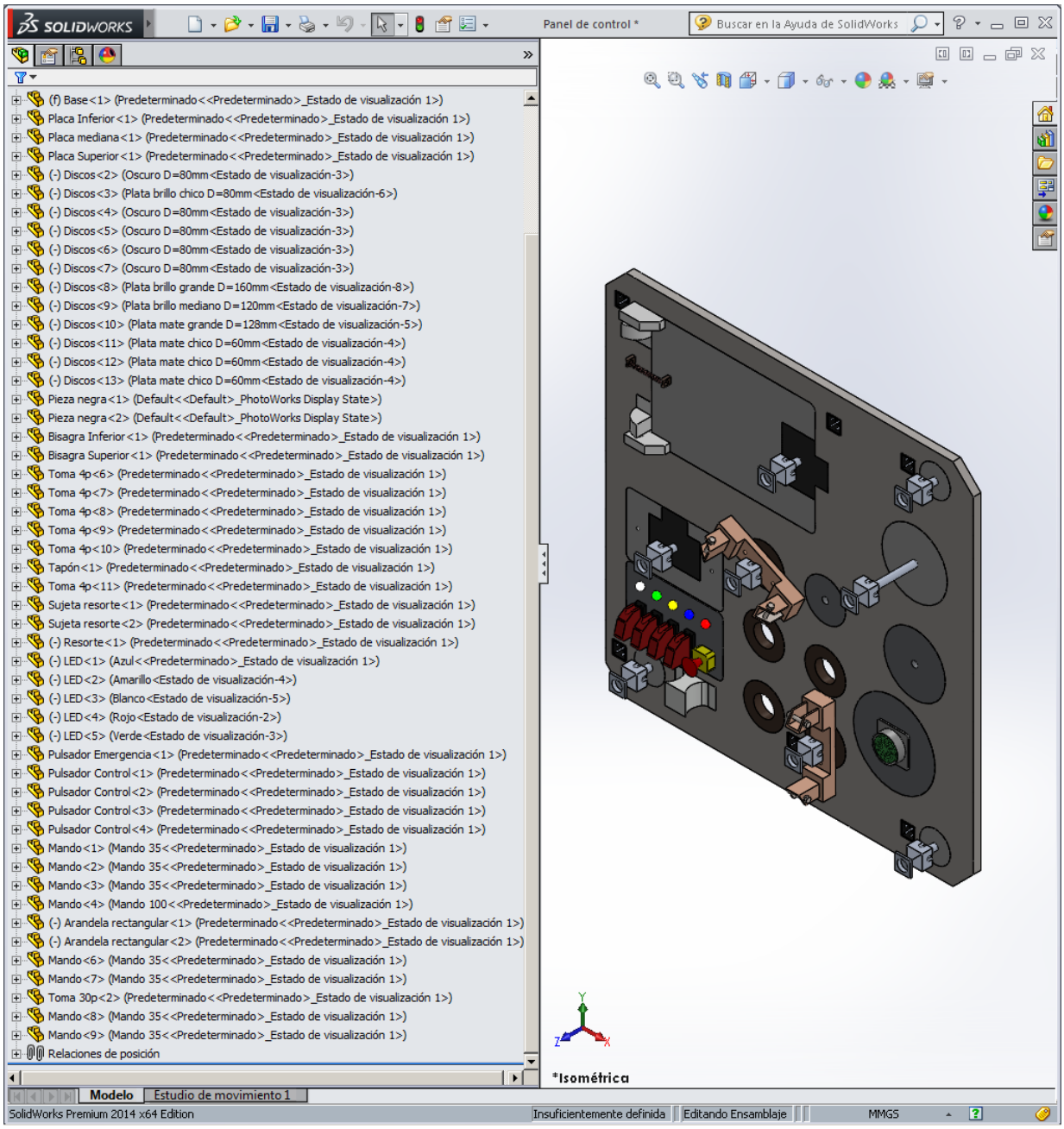


Ilustración 56. Interfaz de usuario de SolidWorks: gestor de objetos de ensamblaje (izquierda) y reproducción 3D del panel de control (derecha).

3.2.4. Tronco.

Este elemento, a diferencia de los anteriores, no se ha constituido a partir de un ensamblaje pues se ha construido como una pieza a la que se le ha realizado los huecos necesarios para que encajen los paneles solares, las antenas y el panel de control. En su parte delantera se ha realizado una cúpula tratando de parecerse a una lente de un instrumento óptico incorporado que fotografía y/o graba el espacio circundante.

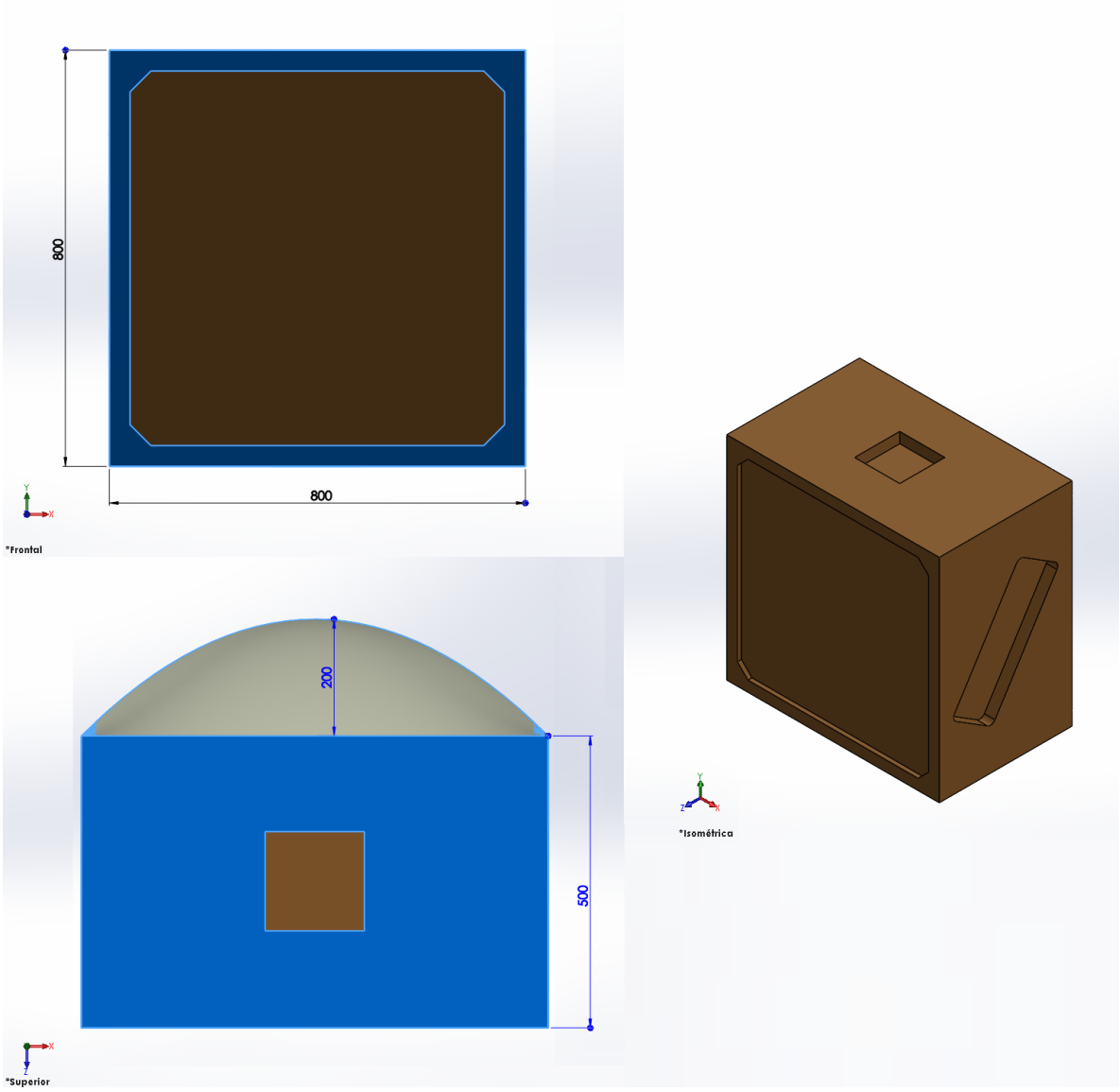


Ilustración 57. Vistas del tronco del satélite cliente.

3.2.5. Jerarquía de objetos.

Una vez importado el modelo del satélite al entorno de simulación, se ha agrupado todos los elementos que lo constituían y han sido destinados a la representación visual. V-REP no ha conservado los colores originarios de SolidWorks, sino que ha asignado a cada objeto colores aleatorios diferentes siendo una tarea ardua volver a reestablecerlos, por lo que se han mantenido intactos ya que no son relevantes de cara a la simulación.

Posteriormente, se ha creado un objeto de forma convexa simple como forma reaccionable y dinámica del satélite de las mismas dimensiones seleccionando el modelo y haciendo clic con el botón derecho sobre su icono representativo en la jerarquía de escena, y se ha pulsado en [*Add > Convex hull of selection*] para su obtención. Ello, como se dijo en el apartado 2.6.2 ahorra tiempo de computación y agiliza la simulación.

Finalmente, se ha añadido como hijo dos gráficas de posición y orientación respectivamente accesibles desde el menú "añadir" (*Add*) a fin de registrar el movimiento del satélite tras la interacción con el satélite de servicio.

A la entidad resultante de añadir todos los elementos citados anteriormente se ha guardado como modelo y nombrado como «Satellite», que representa en su conjunto al satélite cliente a simular.

Ilustración 58. Vista del modelo «Satellite» en el navegador de modelos.

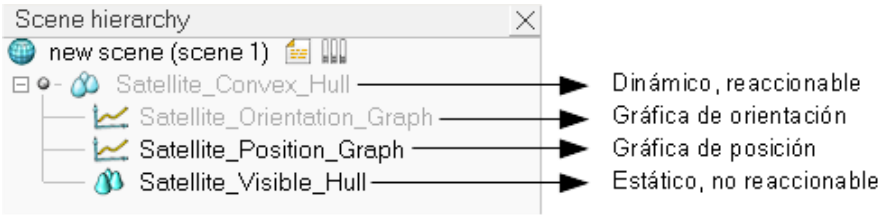


Ilustración 59. Jerarquía de objetos del modelo «Satellite».

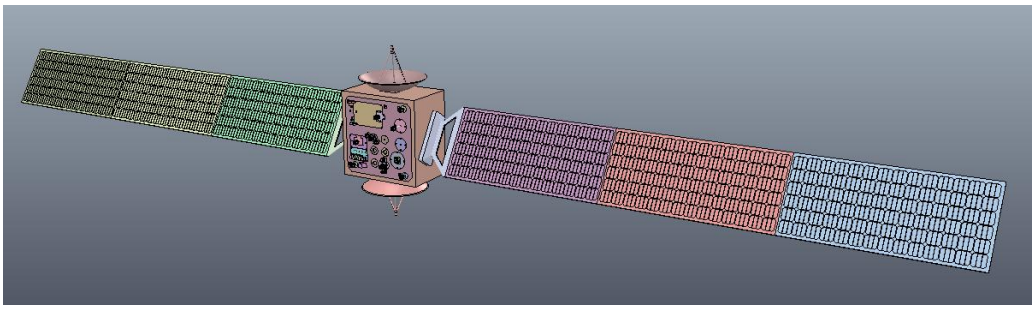


Ilustración 60. Visualización del contenido visual del modelo «Satellite».

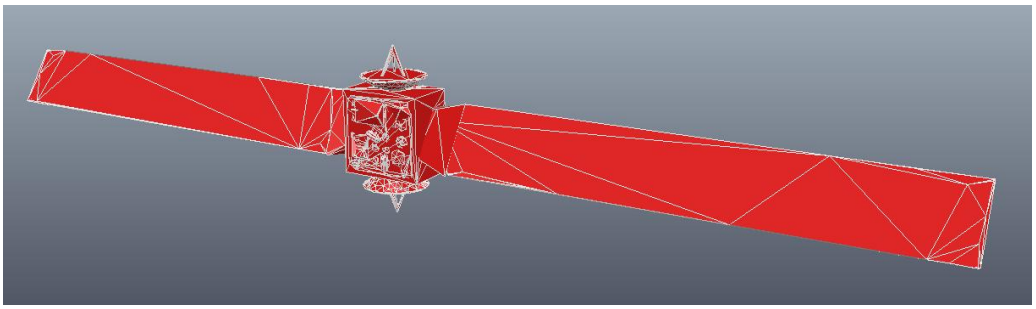


Ilustración 61. Visualización del contenido dinámico del satélite del modelo «Satellite».

3.3. Satélite de servicio.

Representa el satélite que debe capturar al satélite muerto mediante un sistema de visión, suponiendo que funciona autónomamente mientras que el satélite cliente vaga con velocidad relativa nula y dinámica incierta.

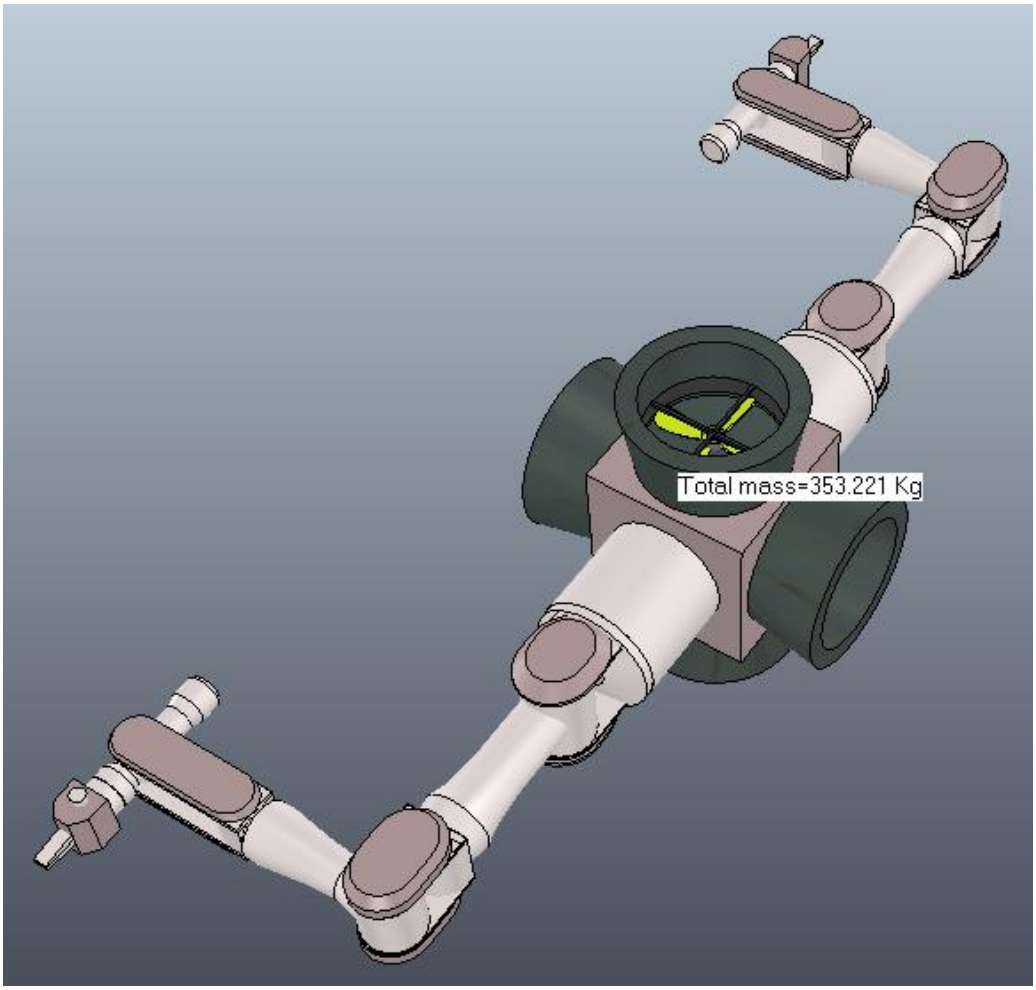


Ilustración 62. Modelo del satélite de servicio «Manipulator».

El modelo de satélite de servicio consta de un cubo como base del modelo al cual se adjuntan 2 brazos robóticos que le permiten interactuar con su entorno y 4 propulsores que dirigen su movimiento a lo largo del espacio. A la entidad resultante de añadir todos estos elementos se ha guardado como modelo y nombrado como «Manipulator», que representa en su conjunto al satélite de servicio a simular que "manipulará" el satélite cliente en la escena.

Las características del satélite son resultado de la definición de los subcomponentes, de los cuales se han hallado las referencias adecuadas para suponer unas dimensiones, pesos y características generales acorde a la realidad como veremos en los siguientes apartados. De este modo, se ha definido el modelo con un peso de 353,221 kg y unas dimensiones de 2,15 x 0,38 x 0,38 m.

Para alcanzar la estabilidad dinámica deseada del modelo, se ha considerado los momentos de inercia, según el ajuste experimental establecido en V-REP para hallar los parámetros de simulación más estables, siguientes: I_x = 2658 m^2, I_y = 1801 m^2 y I_z = 1796 m^2.

3.3.1. Manipulador robótico.

Se trata del modelo de brazo robot usado en la simulación y que viene a reproducir el manipulador robótico de la literatura [1] definido con unas dimensiones de 0,975 x 0,2 x 0,2 m y una masa de 65,01 kg similares al del manipulador "K-1207i" [21].

El manipulador robótico usado en la simulación proviene de uno de los modelos por defecto proporcionados por V-REP denominado «7 DoF Manipulator», con la salvedad de que se ha añadido 1 GDL para las pinzas; se han adaptado sus parámetros dinámicos para poder ser simulado en ingravidez ensamblado a un cuerpo central; se han añadido limitaciones a sus articulaciones; y se han modificado sus parámetros de control convenientemente para ser manejado a través de una interfaz de usuario personalizada.

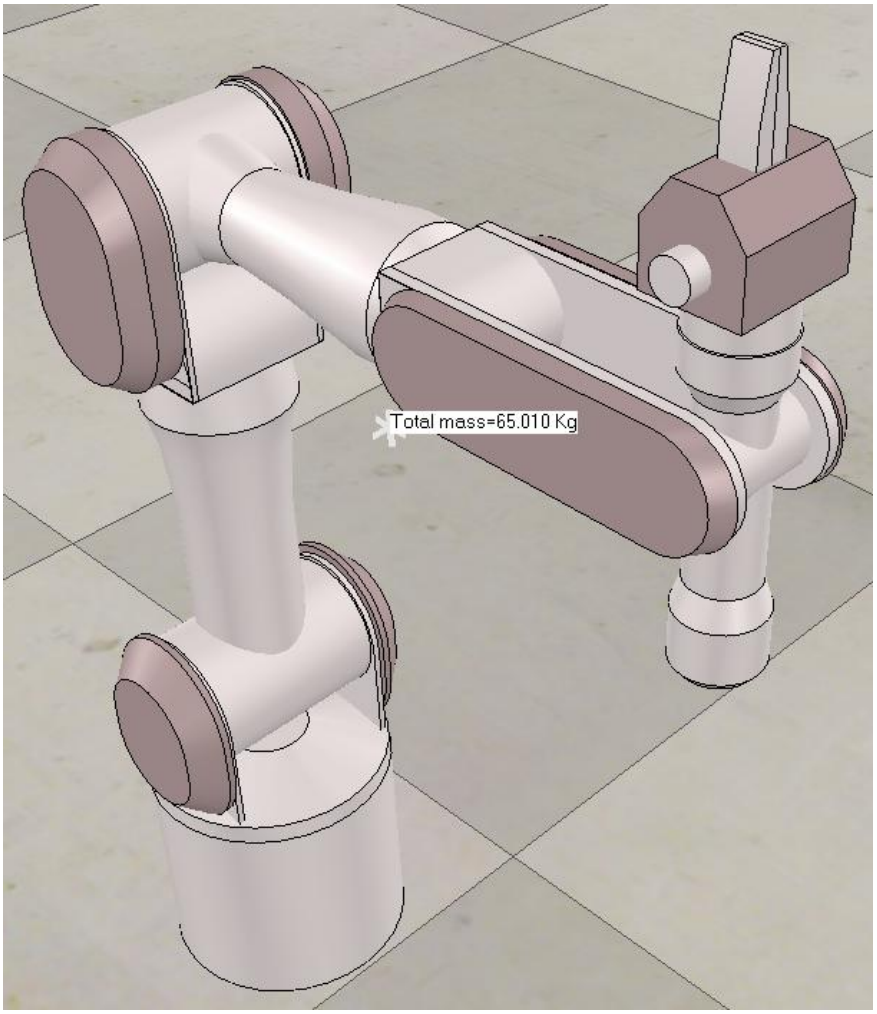


Ilustración 63. Modelo de brazo robótico «8 DoF Manipulator».

3.3.1.1. Descripción.

El manipulador robótico está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos [22]. La constitución física del robot guarda cierta similitud con la anatomía del brazo humano, por lo que en ocasiones, para hacer referencia a los distintos elementos que lo componen, se usan términos como cuerpo, brazo, codo, muñeca y dedos.

El robot es una cadena cinemática abierta con articulaciones de tipo rotación (con un solo GDL cada una), siendo sencillo encontrar el número de grados de libertad que posee, pues coincide con el número de articulaciones de que se compone; en consecuencia, debido a la configuración que tiene, se trata de un robot angular o antropomórfico, cuyo esquema se representa en la ilustración siguiente:

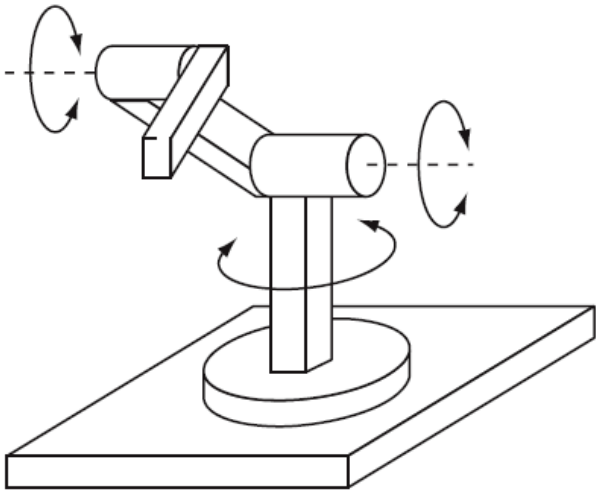


Ilustración 64. Robot angular o antropomórfico [22].

Puesto que para posicionar y orientar un cuerpo de cualquier manera en el espacio son necesarios seis parámetros, tres para definir la posición y tres para la orientación, si se pretende que un robot posicione y oriente su extremo (y con él la pieza o herramienta manipulada, que en nuestro caso es una pinza) de cualquier modo en el espacio, se precisarán de al menos seis GDL.

En la práctica, a pesar de ser necesarios los seis GDL comentados para tener total libertad en el posicionado y orientación del extremo del robot, nuestro manipulador precisa más de seis según [1] para que pueda tener acceso a todos los puntos de su entorno. Así, cuando trabaje en el espacio con obstáculos, el dotar al robot de grados de libertad adicionales le permitirá acceder a posiciones y orientaciones de su extremo a las que, como consecuencia de los obstáculos, no hubiera llegado con seis GDL. Cuando el número de grados de libertad del robot es mayor que los necesarios para realizar una determinada tarea se dice que el robot es redundante, como se caracteriza el modelo tratado.

En la figura siguiente, con el fin de dar a entender lo desarrollado, se representa el caso de un robot planar al que le bastaría con 2 GDL para posicionar su extremo en cualquier punto del plano.

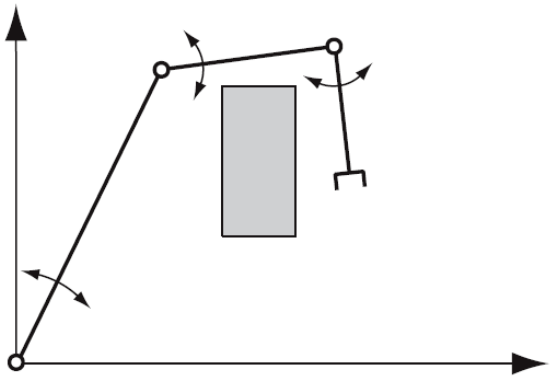


Ilustración 65. Robot plano redundante con 3 GDL para aumentar su maniobrabilidad ante un obstáculo.

Por todo lo expuesto, los robots redundantes son la opción habitual en las aplicaciones espaciales, predominando los brazos de 7 GDL y a parte las pinzas -como es el caso del modelo tratado–. Las ventajas de este diseño son [23]:

- Capacidad para controlar toda la configuración del brazo (incluyendo el codo).
- Capacidad para trabajar en un espacio desordenado evitando colisiones con obstáculos en el área de trabajo.
- Capacidad de alcanzar y trabajar dentro de un espacio de trabajo estrecho.
- Optimización de diferentes medidas de rendimiento.
- Espacio de trabajo alcanzable mejorado a través de la gestión de singularidades y límites de articulación.

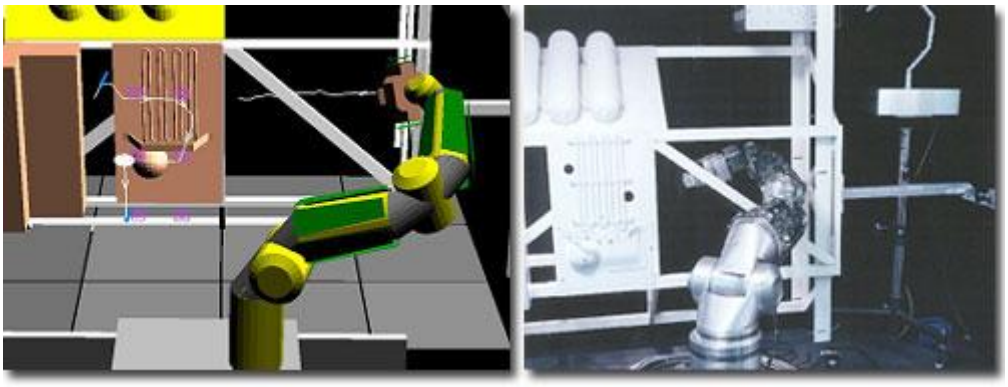


Ilustración 66. Manipulación simulada y real en un ambiente restringido mediante un modelo de brazo robot redundante [23].

Los robots presentan en general limitaciones en sus articulaciones y nuestro manipulador robótico no se queda atrás a fin de acercarnos a la realidad por la alusión realizada en [1] y [23]; luego cada articulación presenta limitaciones en su movilidad, las cuales se recogen en la

siguiente tabla, y cuyos valores han sido establecidos en las propiedades específicas de cada articulación a excepción de la 9 porque es dependiente de la 8. Además, se especifica las magnitudes de longitud y diámetro que se han establecido para poder ser reconocidos con facilidad en la visualización del contenido dinámico subyacente.

Articulación	Longitud [m]	Diámetro [m]	Pos. Mín. [º]	Rango [º]
1	0.030	0.160	-90	180
2	0.2	0.02	-90	180
3	0.03	0.1	-70	140
4	0.2	0.02	-45	90
5	0.03	0.08	-90	180
6	0.2	0.02	-90	180
7	0.2	0.03	-180	360
8	0.08	0.002	0	25
9	0.08	0.002	-	-

Tabla 2. Propiedades específicas de las articulaciones.

3.3.1.2. Jerarquía de objetos.

La jerarquía de objetos del modelo «8 DoF Manipulator» se compone de objetos estáticos en general caracterizados por ser de formas aleatorias simples/compuestas que están destinados a la representación visual; y de objetos dinámicos caracterizados por ser articulaciones y formas puras simples/compuestas en mayoría, a excepción de los dedos de la pinza que son objetos de forma convexa simple destinados a la representación visual también.

Ilustración 67. Vista del modelo «8 DoF Manipulator» en el navegador de modelos.

Mediante el acceso a las propiedades específicas y comunes de los objetos, siguiendo la metodología descrita en apartados anteriores, se han configurado adecuadamente cada uno de los objetos para poder realizar la simulación dinámica, mostrándose indicadas las características fundamentales en la ilustración 68.

Además, se puede apreciar en dicha ilustración una línea roja punteada que describe el vínculo

dummy-dummy establecido en la configuración para el cálculo correspondiente de la cadena cinemática del brazo robot. Para dicho cálculo ha sido necesario usar el módulo de cálculo de cinemática inversa, el cual se haya en el apartado de cinemática inversa (*Inverse kinematics*) dentro del cuadro de diálogo de propiedades del módulo de cálculo (*Calculation modules*) accesible a través de la barra de herramientas nº 2 y la barra de menú.

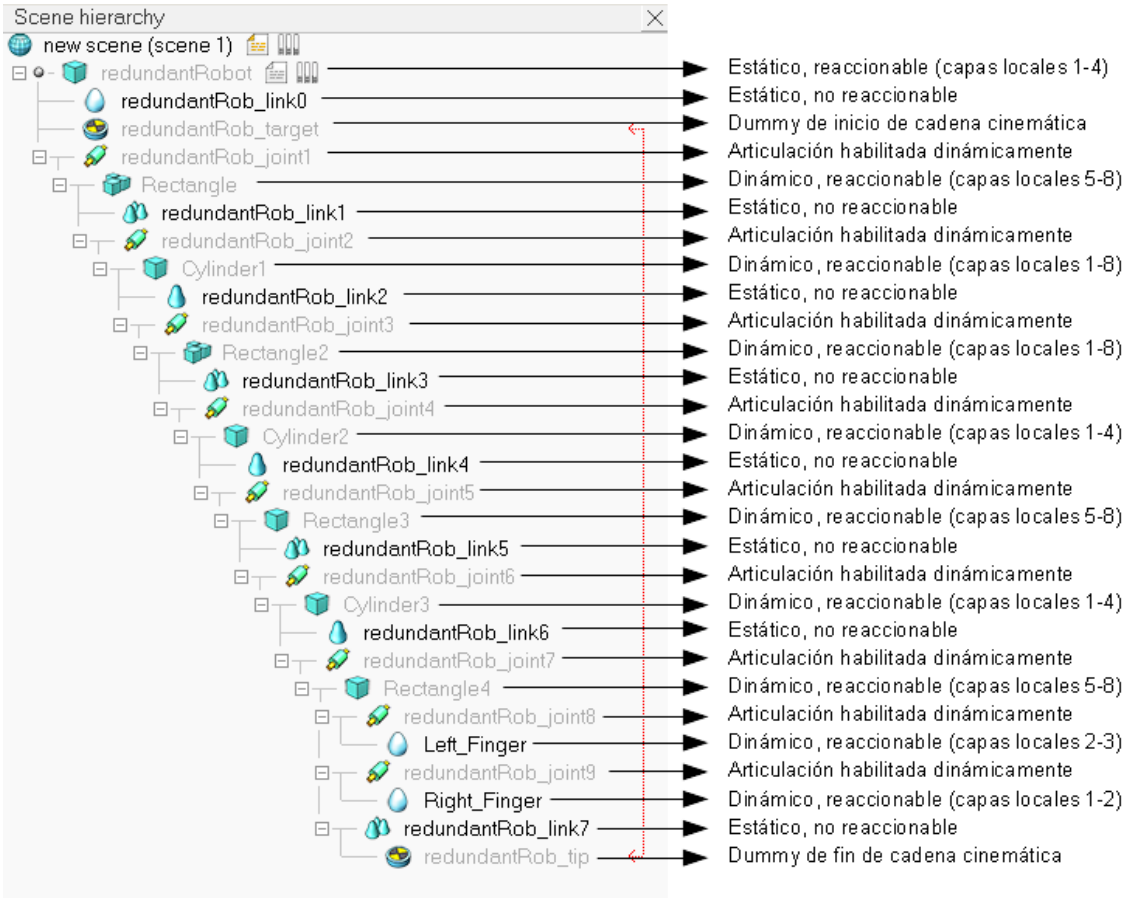


Ilustración 68. Jerarquía de objetos del modelo «8 DoF Manipulator».

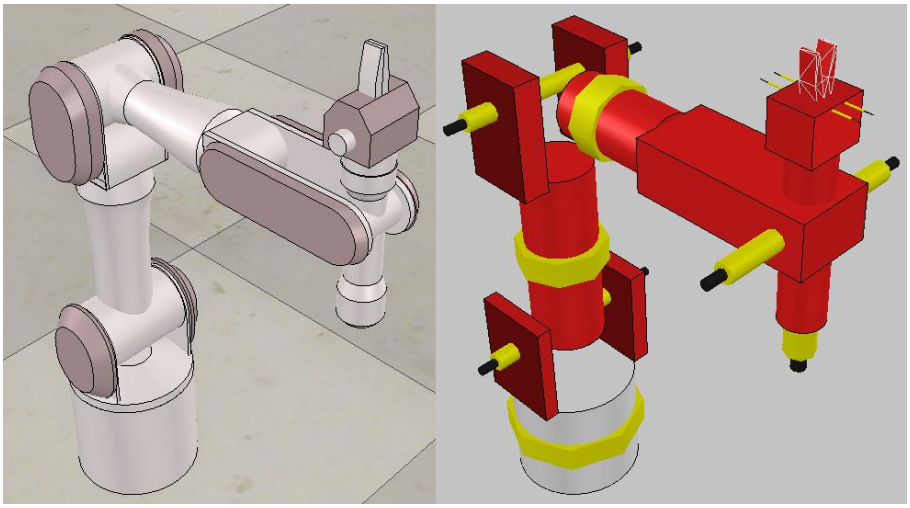


Ilustración 69. Visualización del contenido visual (izquierda) y dinámico (derecha) del modelo «8 DoF Manipulator».

Ante todo, lo primero ha sido establecer, para las articulaciones, el modo de cinemática inversa, que indica a V-REP que sea él quien calcule el valor de la articulación [24]. En este sentido, se ha modificado los modos de funcionamiento de las articulaciones de la 1 a la 8 al modo de cinemática inversa (*Inverse kinematics mode* o *IK mode*) marcando la casilla "operación hibrida" (*Hybrid operation*) explicada anteriormente. La articulación 9 es una articulación dependiente de la articulación 8, luego se ha seleccionado el modo dependiente (*Dependent mode*); y a continuación, se ha pulsado el botón de ajuste de la ecuación dependiente (*Adjust dependency equation*) para poder introducir la ecuación tal y como se muestra en la ilustración siguiente:

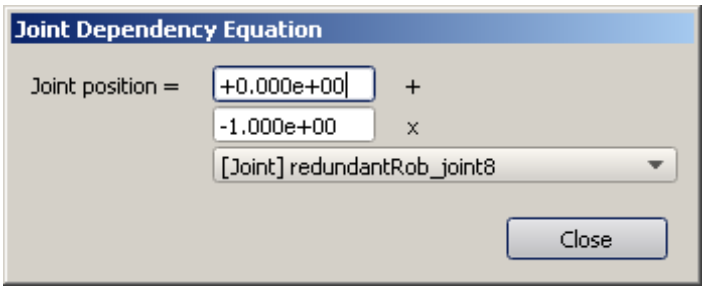


Ilustración 70. Ecuación de dependencia de la articulación 9.

Una vez configuradas las articulaciones, V-REP necesita para poder realizar los cálculos, por un lado, un dummy posicionado en el efector final del robot con la posición y orientación correspondiente, y ubicado en el último lugar de la jerarquía de modelo como se representa en la ilustración 68. Este dummy representa el final de la cadena cinemática y en el argot del programa es conocido como "punta" (*tip*).

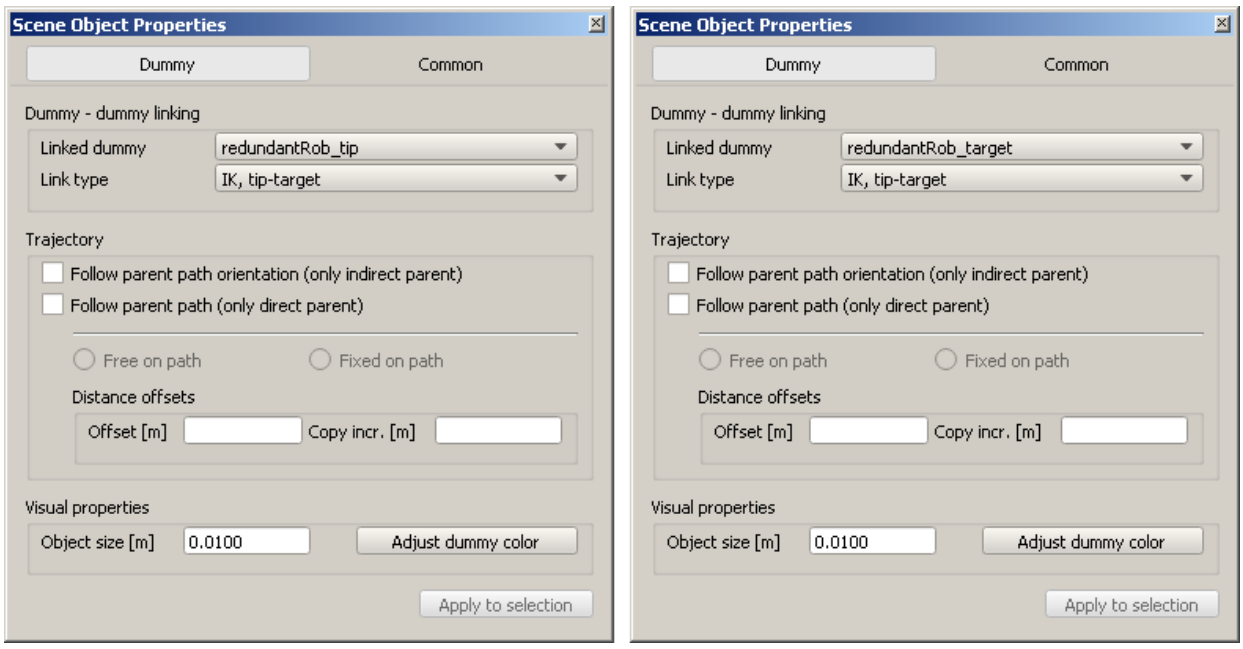


Ilustración 71. Configuraciones específicas de los dummy de fin e inicio de cadena cinemática.

Por otro lado, V-REP necesita un dummy que indique el inicio de la cadena cinemática posicio-

nado en el efector final también, pero a diferencia del anterior, debe ir ubicado como hijo de la base del modelo. Este dummy se conoce en el argot del programa como "objetivo" (*target*). Por tanto, entrando en las propiedades específicas de cada dummy para establecer dicha relación, descritas en el apartado 2.6.3.1.4, se obtiene el resultado expuesto en la ilustración 71.

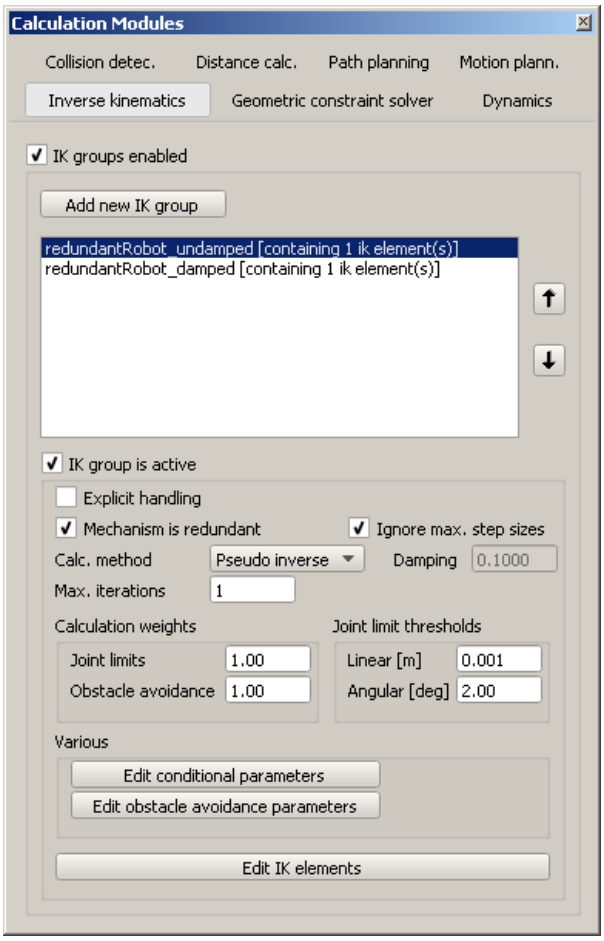


Ilustración 72.Propiedades del módulo de cálculo de cinemática inversa para el método de cálculo no amortiguado (undamped).

En esta etapa, todos los elementos para la definición de la tarea cinemática inversa están listos, y sólo se necesita registrar la cadena cinemática a través de lo que se denomina "grupos IK" (*IK group*). Además, de este cuadro de diálogo es necesario configurar 3 características u opciones que se describen a continuación [5]:

- El mecanismo es redundante (*Mechanism is redundant*)

  Este ajuste es necesario establecerse cuando se fijan limitaciones en las articulaciones, como en nuestro caso, o parámetros de evitación de obstáculos, que no lo es.

- Método de cálculo (*Calc. Method*)

  El software ofrece dos métodos de cálculo: pseudoinversa y DLS. El primero es un méto-

do rápido y no amortiguado, por lo que resulta ineficaz ante configuraciones singulares y puede provocar inestabilidades en la simulación; el segundo es un método lento, pues necesita más coste computacional, pero es amortiguado, por lo que resuelve las configuraciones singulares en función del coeficiente de amortiguación (*Damping*), es decir, cuando el valor de amortiguamiento crece, la resolución se vuelve más estable y más lenta, y viceversa.

- Máximas iteraciones (*Max. Iterations*)

Establece el número de iteraciones máximas de cálculo que debe realizarse para el grupo cinemático inverso o IK.

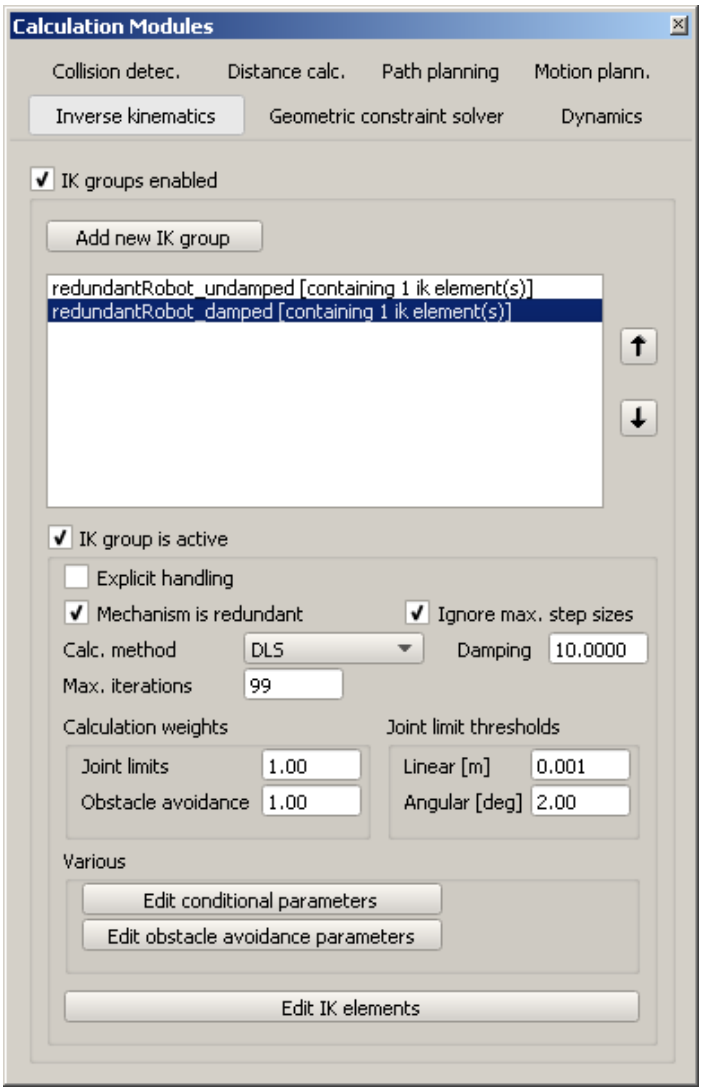


Ilustración 73. Propiedades del módulo de cálculo de cinemática inversa para el método de cálculo amortiguado (damped).

En consecuencia, para tratar de aprovechar las ventajas de ambos métodos de cálculo, se ha pulsado sobre el botón correspondiente (*Add new IK group*) para añadir dos grupos de cálculo no amortiguado (*undamped*) y amortiguado (*damped*) respectivamente, para los cuales se ha

establecido un máximo de iteraciones igual 1 y 99 respectivamente, y un coeficiente de amortiguamiento de 10 para el cálculo amortiguado. El resultado de dichas configuraciones se muestra en las ilustraciones 72 y 73.

Por último, se necesita establecer una cláusula en el grupo de cálculo amortiguado que permita ser usado cuando el método de cálculo no amortiguado falle, para lo cual se debe pulsar sobre el botón de edición de parámetros condicionales (*Edit conditional parameters*) y en el cuadro de diálogo emergente establecer dicha cláusula del modo en que se presenta en la ilustración 74.

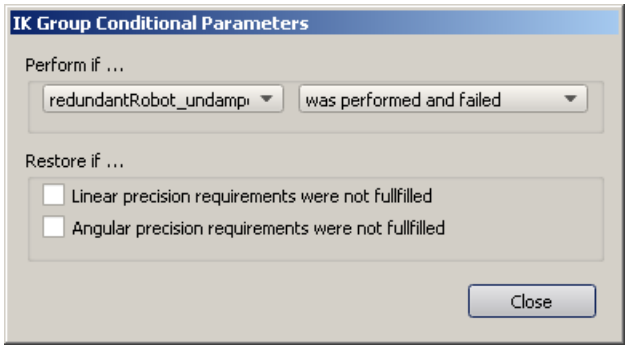


Ilustración 74. Cláusula establecida para el cálculo amortiguado.

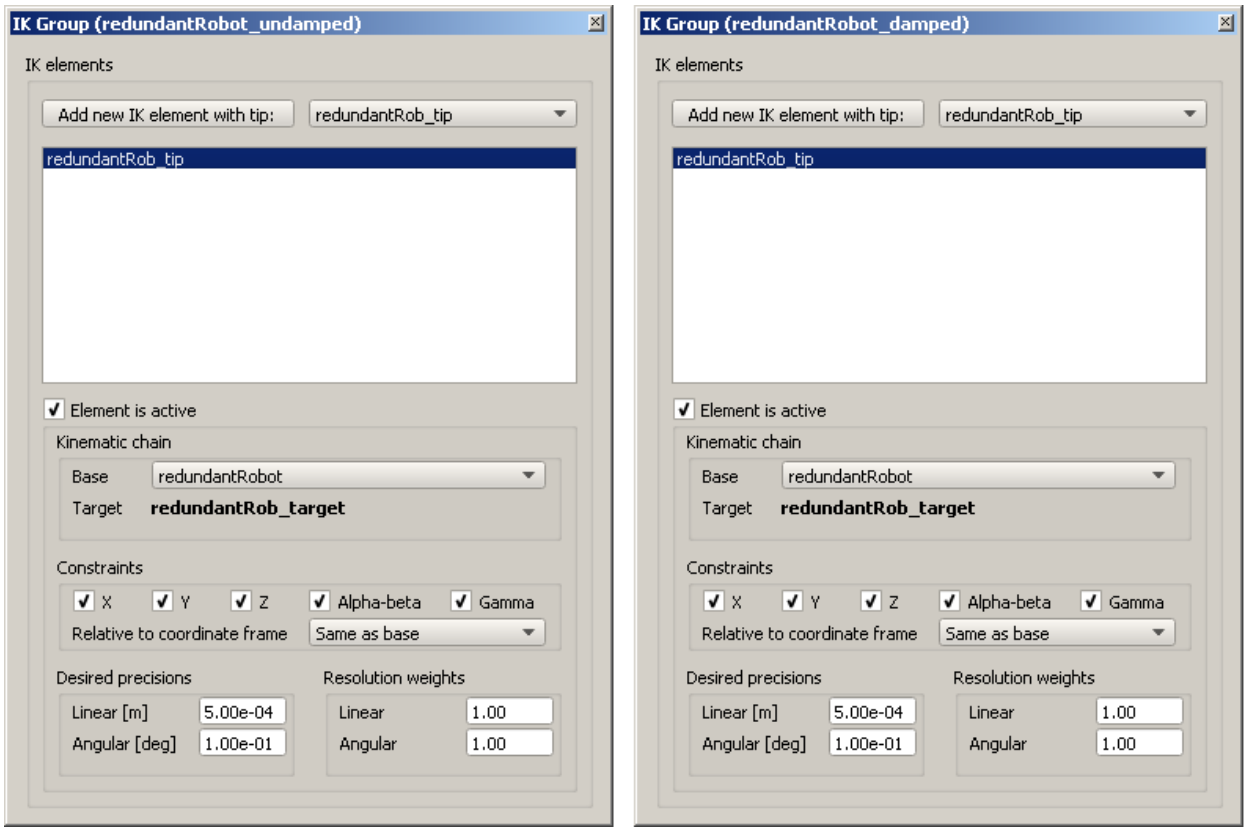


Ilustración 75. Cuadros de diálogo de edición de elementos IK del grupo de cálculo no amortiguado y amortiguado respectivamente.

Llegados a este punto, ya es posible introducir los elementos que participan en la cadena cine-

mática de cada grupo de cálculo, para lo cuales se procede del mismo modo: se pulsa el botón de "editar elementos IK" (*Edit IK elements*), y en el nuevo cuadro de diálogo que se abre se añade el dummy punta (*Tip*), nombrado en la jerarquía de modelo como "redundantRob_tip" –véase en la ilustración 75–, en el cuadro desplegable; y, a continuación, haciendo clic en "agregar nuevo elemento IK" (*Add new IK element with tip*). Además, en la parte inferior de cada diálogo de edición de elementos IK, se ha necesitado indicar cuál es la base del modelo, que en nuestro caso se denomina "redundantRobot"; y exigir que se cumpla las condiciones de cálculo en todas sus posibles coordenadas, ya que se pretende que nuestro dummy "punta" siga a nuestro dummy "objetivo" en posición y orientación. El resultado de dichas configuraciones se expone en la ilustración 75.

3.3.1.3. Controles de usuario.

V-REP permite crear interfaces de usuario (*UI*) de forma bastante sencilla que nos permiten, en tiempo de simulación, establecer determinadas opciones o modificar valores a determinadas variables o parámetros [24]. En nuestro caso, utilizaremos barras de deslizamiento para poder establecer la posición de las articulaciones.

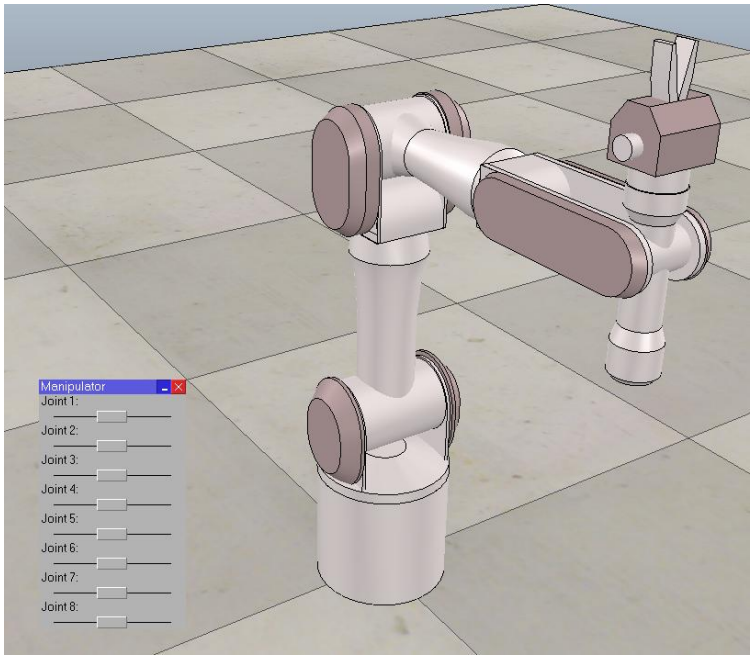


Ilustración 76. Interfaz de usuario personalizada para el control del manipulador robótico.

Para la creación del control de usuario, se debe acceder al editor de interfaces de usuario personalizado (*Custom UI editor*) accesible a través de la barra de herramientas nº2. A continuación, si se pulsa "Añadir nuevo interfaz de usario" (*Add new user interface*) se creará una nueva interfaz, de la que debemos establecer el tamaño del área de trabajo X e Y (*client size-x* y *client size-y*). Lo recomendable es utilizar un tamaño de X = 10 y dos por cada elemento de con-trol que queramos incorporar en Y. El objetivo del presente TFG es introducir 8 barras de

deslizamiento para la posición deseada de cada articulación, por tanto se necesita al menos un tamaño de Y = 16. Sobre el nuevo control creado es posible introducir un título, para ello se pincha en la barra de título y en el campo "etiqueta de botón superior" (*Button label (up state)*) se escribe; en nuestro caso, se ha elegido como título *Manipulator* que viene a decir en español "Manipulador".

Para la creación de los controles y sus etiquetas identificativas, se han seguido los siguientes pasos:

1) Se selecciona toda la primera fila haciendo uso de la tecla "control".
2) Se hace clic sobre el botón "insertar botón combinado" (*Insert merged button*)
3) Se selecciona el tipo "etiqueta" (*Label*) y se desmarca la opción "centrado horizontalmente" (Horizontally centered).
4) Introducimos el nombre de la etiqueta, en nuestro caso es *Joint 1* que viene a decir en español "Articulación 1".
5) Se repite los pasos 1 y 2 para la segunda fila y se selecciona el tipo "deslizador" (*Slider*).

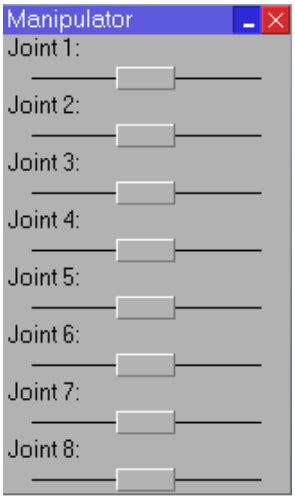


Ilustración 77. Control de usuario del modelo «8 DoF Manipulator».

Cada elemento que se ha introducido en un control de usuario tiene un maneja-dor (*handle*) al cual le corresponde un número, el cual es importante puesto que nos permite referirnos al él para conocer su estado o el valor; en nuestro caso, serán necesarios dichos números para conocer la posición del deslizador así como asociar cada control a su articulación correspondiente.

3.3.1.4. Script asociado.

El script usado para el control del manipulador robótico es un script secundario no secuencial, cuyo código se ha escrito en el presente trabajo con la finalidad adicional de mostrar por la ventana de consola información sobre los límites de cada articulación y la posición en la que se encuentran los deslizadores de control de cada articulación al inicio de la simulación.

La forma de asociar un script a un objeto es haciendo clic con el botón derecho sobre él en la jerarquía de escena, y en el menú emergente se debe pulsar: [*Add >Associated child script > Non threaded*]. Al agregarse al objeto, el primer aspecto que presenta al hacer doble clic sobre el script en la escena de jerarquía es el siguiente:

```
Non-threaded child script (XYZCameraProxy)
1  if (sim_call_type==sim_childscriptcall_initialization) then
2
3      -- Put some initialization code here
4
5
6  end
7
8
9  if (sim_call_type==sim_childscriptcall_actuation) then
10
11     -- Put your main ACTUATION code here
12
13 end
14
15
16 if (sim_call_type==sim_childscriptcall_sensing) then
17
18     -- Put your main SENSING code here
19
20 end
21
22
23 if (sim_call_type==sim_childscriptcall_cleanup) then
24
25     -- Put some restoration code here
26
27 end
```

Los scripts normalmente vienen estructurados por bloques facilitando al usuario la labor de escribir el código acorde a las propias normas que establece el programa. En este caso, se diferencian cuatro bloques de ejecución: inicialización (*initialization*), actuación (*actuation*), detección (*sensing*) y restauración (*restoration*); de los cuales sólo serán necesarios el de inicialización, para la declaración de variables y obtención de parámetros; y el de actuación, para la ejecución de las órdenes de orientación enviadas a las articulaciones.

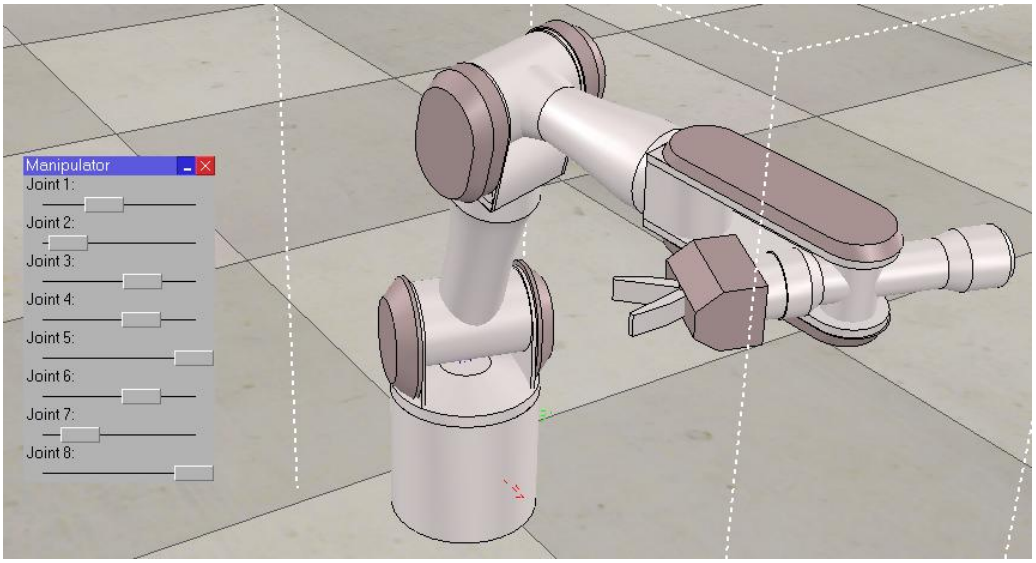


Ilustración 78. Ejemplo de control ejercido sobre el manipulador usando el panel de control asociado.

En definitiva, se ha creado un script que sirve como intermediario entre el panel de control de

usuario y el manipulador; de modo que, cuando se mueve un deslizador, el script captura la nueva posición de éste y lo traduce en una nueva posición de la articulación determinada. El código escrito para el modelo se halla adjunto en el anexo A, para el cual se ha usado las funciones API descritas a continuación [8], [24]:

- simGetUIHandle.

  Obtiene el manejador de la interfaz de usuario.

- simGetObjectHandle.

  Obtiene el manejador de objetos de la simulación.

- simGetJointInterval.

  Devuelve dos valores, primero la variable booleana indicando si la articulación es cíclica[3] (true) o no (false); y un segundo en forma de vector o tabla con los intervalos, compuesto por dos elementos a su vez: el primero devuelve el valor del ángulo mínimo de la articulación, y el segundo devuelve su rango.

- simGetUISlider.

  Devuelve un valor de 0 a 1000 con la posición relativa del deslizador.

- simSetJointPosition.

  Permite especificar como primer argumento el manejador de la articulación que queremos posicionar y como segundo argumento el valor en radiantes de la articulación.

La impresión de la información capturada por las funciones «simGetJointInterval» y «simGetUISlider» se ha realizado con ayuda de la función de Lua denominada "print", cuyo efecto en el código escrito tiene como resultado el lanzamiento de información presentado en la ilustración 79.

Una vez finalizado la configuración del script, queda adaptarlo cuando se sume junto a otro manipulador robótico en la escena puesto que el modelo «Manipulator» que se simulará se compone de dos brazos robóticos y en la información lanzada por consola no se podrá distinguir de qué manipulador es.

Por defecto, cuando se agrega un segundo modelo que ya existe en la escena, V-REP renombra a todos los objetos que componen el modelo con el sufijo "-#0" como puede verse en la

[3] La posición cíclica de una articulación se define como aquella con una posición mínima igual a -180º y un rango igual a 360º.

ilustración 13 junto a una tercera copia con el sufijo "-#1", una cuarta con el sufijo "#2", y así sucesivamente. Esto no afecta en el código en absoluto, pero se debe añadir en el script asociado de la copia adjuntos a los títulos de dicha información el mismo sufijo para así poder distinguir ésta; es decir, si los títulos son "Límites articulaciones" y "Sliders" como se refleja en la ilustración 79, en el script de la copia se debe renombrar como "Límites articulaciones #0" y "Sliders #0".

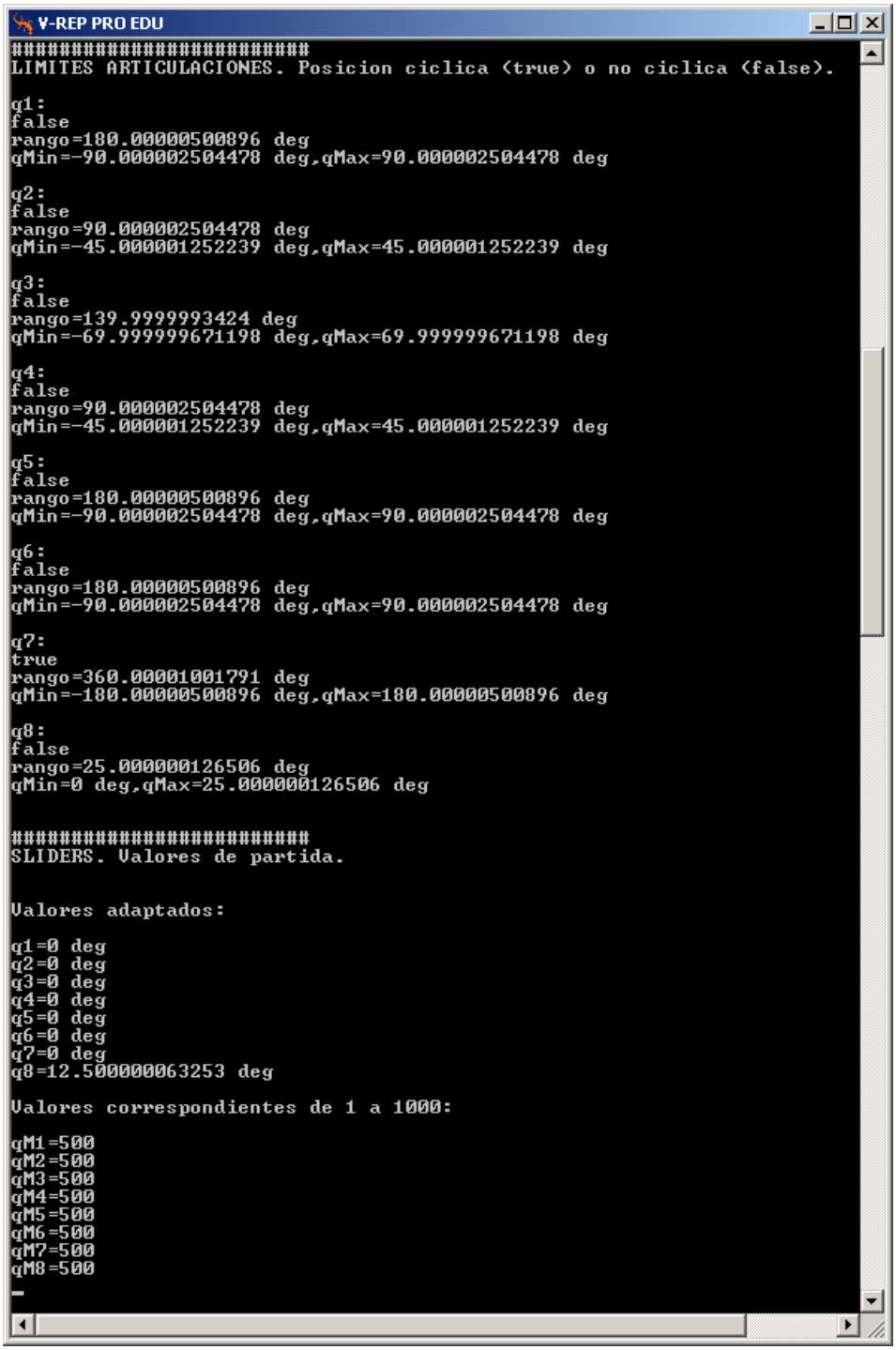


Ilustración 79. Información sobre los límites de cada articulación y la posición relativa de los sliders mostrada en la ventana de la consola.

Para indicar al usuario esto que se acaba de expresar, se ha usado el cuadro reservado a este fin en el diálogo de propiedades de modelo donde se ha escrito en inglés:

If you include more than one model, try to adapt the scripts of each one in order

to correctly identify both joints and user controls.

Que viene a decir en español:

Si incluyes más de un modelo, procura adaptar los scripts de cada uno con el fin de identificar correctamente tanto las articulaciones como los controles de usuario.

De este modo, quedaría definido el modelo y cuando se cargue en la escena se mostrará esta información automáticamente. El resultado de este ajuste se muestra en la ilustración 80.

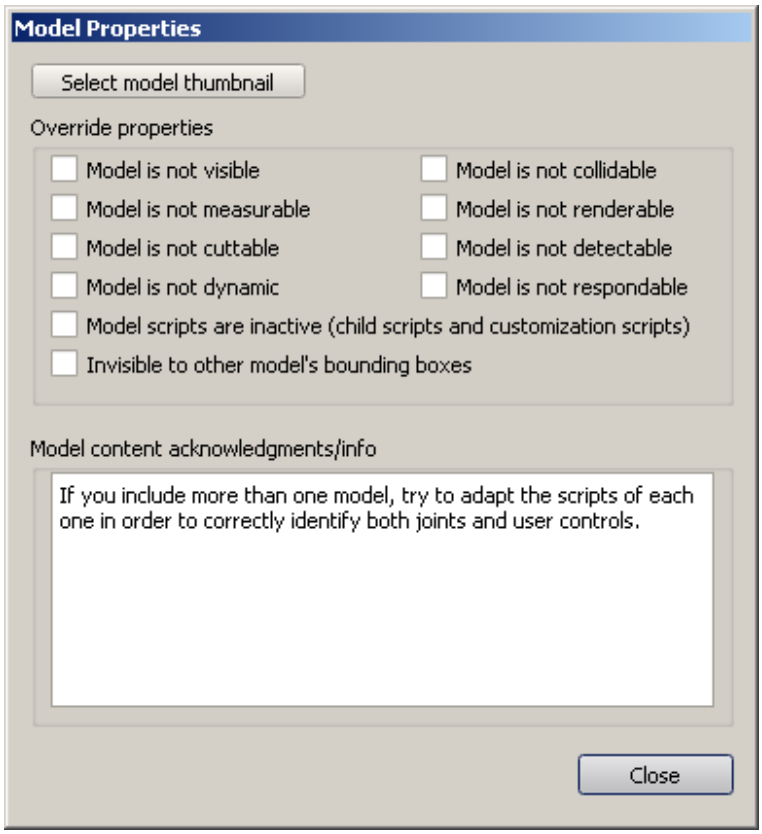


Ilustración 80. Propiedades de modelo de «8 DoF Manipulator»

3.3.2. Propulsor.

Se trata del modelo propulsor usado en el modelo «Manipulator» que permitirá el desplazamiento de éste una distancia desde una posición de partida hasta aproximarse frente al satélite, donde se frena y queda en reposo. Para su composición se ha servido del software SolidWorks para la construcción del cuerpo y de un modelo de hélice proporcionado por V-REP, guardándose como

modelo y nombrándolo como «Thruster» (“Propulsor”).

Las características del modelo se han tratado de reproducir conforme a la realidad, por lo que se ha definido el modelo con unas dimensiones de 90 mm de altura, un diámetro superior de 175 mm y uno inferior de 208 mm que oscilan dentro de los márgenes reales tomando como referencia las dimensiones del monopropulsor “BGT-X5” [25] como orientación; y una masa de 0,7 kg y una potencia de 22 N tomando como referencia el monopropulsor “MONARC-22” [26].

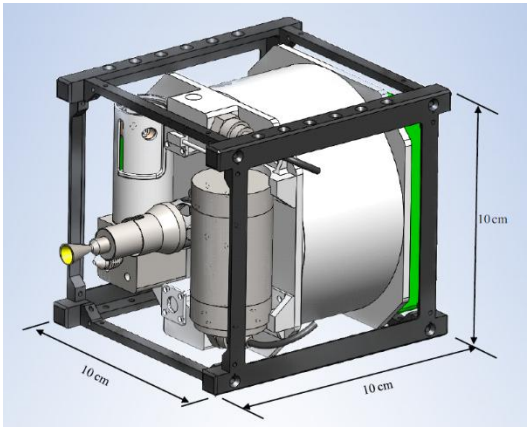


Ilustración 81. Dimensiones del monopropulsor BGT-X5 [25].

Performance Characteristics

Engine	MONARC-1	MONARC-5	MONARC-22-6	MONARC-22-12	MONARC-90LT	MONARC-90HT	MONARC-445
Steady State Thrust	0.22 lbf (1N) @275 psia	1.0 lbf (4.5 N) @325 psia	5 lbf (22N) @275 psia	5 lbf (22N) @190 psia	20 lbf (90 N) @ 235 psia	26 lbf (116 N) @ 235 psia	100 lbf (445N) @ 275 psia
Feed Pressure	70 – 400 psia (4.8 – 27.6 bar)	80 – 420 psia (5.5 – 29.0 bar)	70 – 400 psia (4.8 – 27.6 bar)	70 – 400 psia (4.8 – 27.6 bar)	80 – 400 psia (5.5 – 27.6 bar)	80 – 370 psia (5.5 -25.5 bar)	70 – 400 psia (4.8-27.6 bar)
Nozzle Expansion	57:1	135:1	60:1	40:1	40:1	50:1	50:1
Valve Power	18 watts	18 watts	30 watts	30 watts	72 watts	72 watts	58 watts
Mass	0.83 lbm (0.38 kg)	1.08 lbm (0.49 kg)	1.58 lbm (0.72 kg)	1.51 lbm (0.69 kg)	2.47 lbm (1.12 kg)	2.47 lbm (1.12 kg)	3.5 lbm (1.6 kg)
Engine Length/Exit Diam	5.2 in (13.3 cm) / .2 in (0.5 cm)	9.4 in (41.8 cm) /1 in (2.5 cm)	8 in (20.3 cm) / 1.5 in (3.8 cm)	9 in (22.9 cm) / 1.2 in (5.3 cm)	12 in (30 cm) / 3.3 in (8.4 cm)	12 in (30 cm) / 3.3 in (8.4 cm)	16 in (41 cm) / 5.8 in (14.8 cm)
Specific Impulse	227.5 sec	226.1 secs	229.5 secs	228.1 secs	232.1 secs	234.0 secs	234.0 secs
Minimum Impulse Bit	0.0006 lbf-sec (2.6 mN-sec)	0.0007 lbf-sec (3.1 mN-sec)	0.07 lbf-sec (312m N-sec)	0.12 lbf-sec (526m N-sec)	0.04 lbf-sec (1.8 N-sec)	0.26 lbf-sec (1.16 N-sec)	2.59 lbf-sec (11.52 N-sec)
Total Impulse	25,000 lbf-sec (111,250 N-sec)	138,000 lbf-sec (613,852 N-sec)	120,000 lbf-sec (533,784 N-sec)	263,720 lbf-sec (1,173,085 N-sec)	786,000 (3,500,000 N-sec)	459,100 lbf-sec (2,042,178 N-sec)	1,250,000 lbf-sec (5,600,000 N-sec)
Pulses	375,000	205,000	230,000	160,000	50,000	70,000	12,000

Ilustración 82. Tabla de características de los diferentes monopropulsores comerciados por MOOG [26].

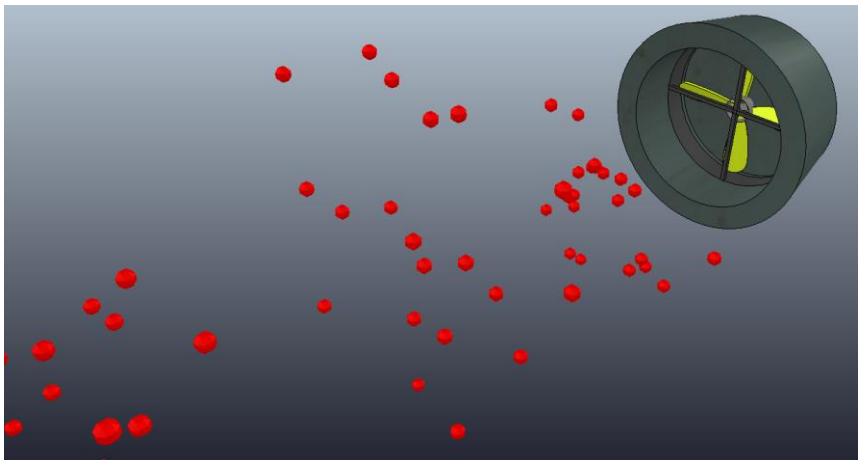


Ilustración 83. Modelo propulsor «Thruster».

3.3.2.1. Descripción.

El propulsor del robot consta de un de modelo proporcionado por V-REP denominado «Propeller», cuyos parámetros dinámicos y script asociado han sido adaptados para poder ser simulado en el entorno de simulación creado, y que representa unas hélices que expulsan partículas; éste va ensamblado al cuerpo del propulsor, el cual se ha importado a V-REP; y éste a su vez, se halla adjunto a un dummy usado como apoyo para la ubicación del modelo propulsor.

Ilustración 84. Vista de los modelos «Propeller» y «Thruster» en el navegador de modelos.

El modelo se ha diseñado para trabajar a una potencia del 75 %, cuya fuerza de empuje por partícula se calcula mediante la siguiente ecuación:

$$F = \rho \cdot v \cdot \pi \cdot \frac{D^3}{6} \cdot \frac{1}{t_s} \left[\frac{N}{partículas}\right] \quad (3.1)$$

Donde «ts» representa el tiempo de paso de simulación, «ρ» representa la densidad de las partículas expresada en la unidad kg/m^3, «v» su velocidad, «D» es su diámetro. El tiempo de paso de simulación es por defecto 50 ms en el entorno de simulación –véase la barra de herramientas nº 1–, que viene a decir la frecuencia con la que se ejecuta los cálculos en V-REP; es decir, 20 veces por segundo. Las demás variables se establecerán en el apartado 3.3.2.3 cuando

tratemos con la programación del propulsor a fin de cumplir con la especificación de potencia del 75 %.

Debido a la alternancia de funcionamiento necesario para el impulso y frenada del satélite de servicio para acercarse al satélite cliente y detenerse, existen dos subtipos de propulsores: los que funcionan sólo una vez durante la simulación, y los que funcionan dos veces. A estos propulsores se les ha llamado respectivamente "propulsor de una etapa" (One-Stage Thruster) y "propulsor de dos etapas" (Two-Stage Thruster).

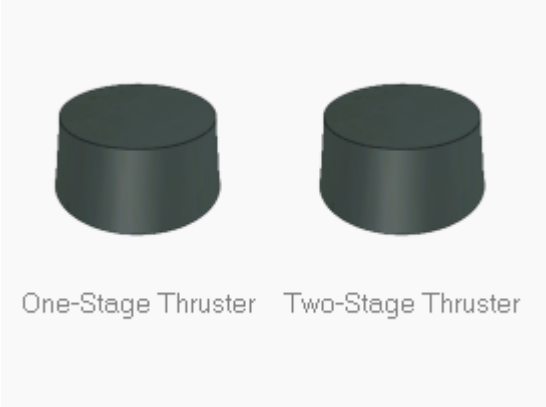


Ilustración 85. Vista de los modelos «One-Stage Thruster» y «Two-Stage Thruster» en el navegador de modelos.

3.3.2.2. Jerarquía de objetos.

La jerarquía de objetos del modelo «Thruster» se compone de un dummy como base del modelo con el fin de posibilitar su posicionamiento, cuatro objetos estáticos de formas aleatorias simples/compuestas, y tres objetos dinámicos. La diferencia entre el modelo de una etapa y el de dos reside en el número de scripts presentes en su jerarquía de modelo, siendo de uno y dos respectivamente.

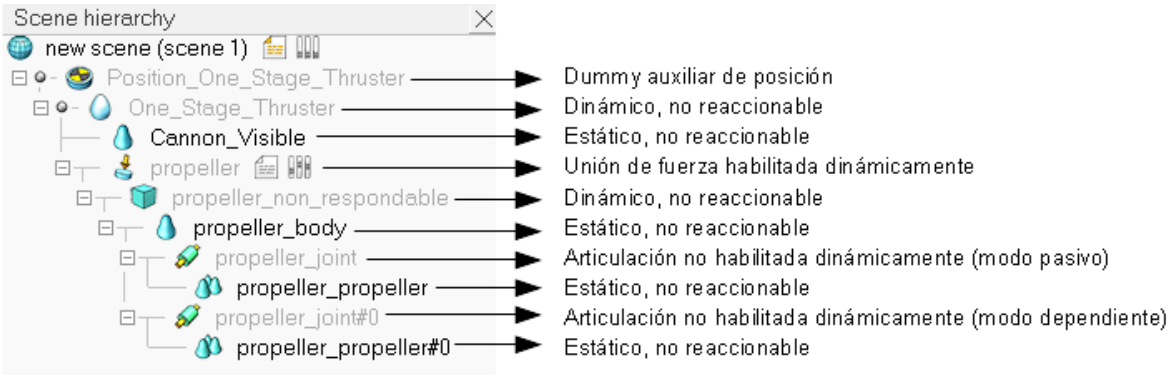


Ilustración 86. Jerarquía de objetos del modelo «One-Stage Thruster».

Mediante el acceso a las propiedades específicas y comunes de los objetos, siguiendo la metodología descrita en apartados anteriores, se han configurado adecuadamente cada uno de

los objetos para poder realizar la simulación dinámica, mostrándose indicadas las características fundamentales en la ilustración anterior y posterior.

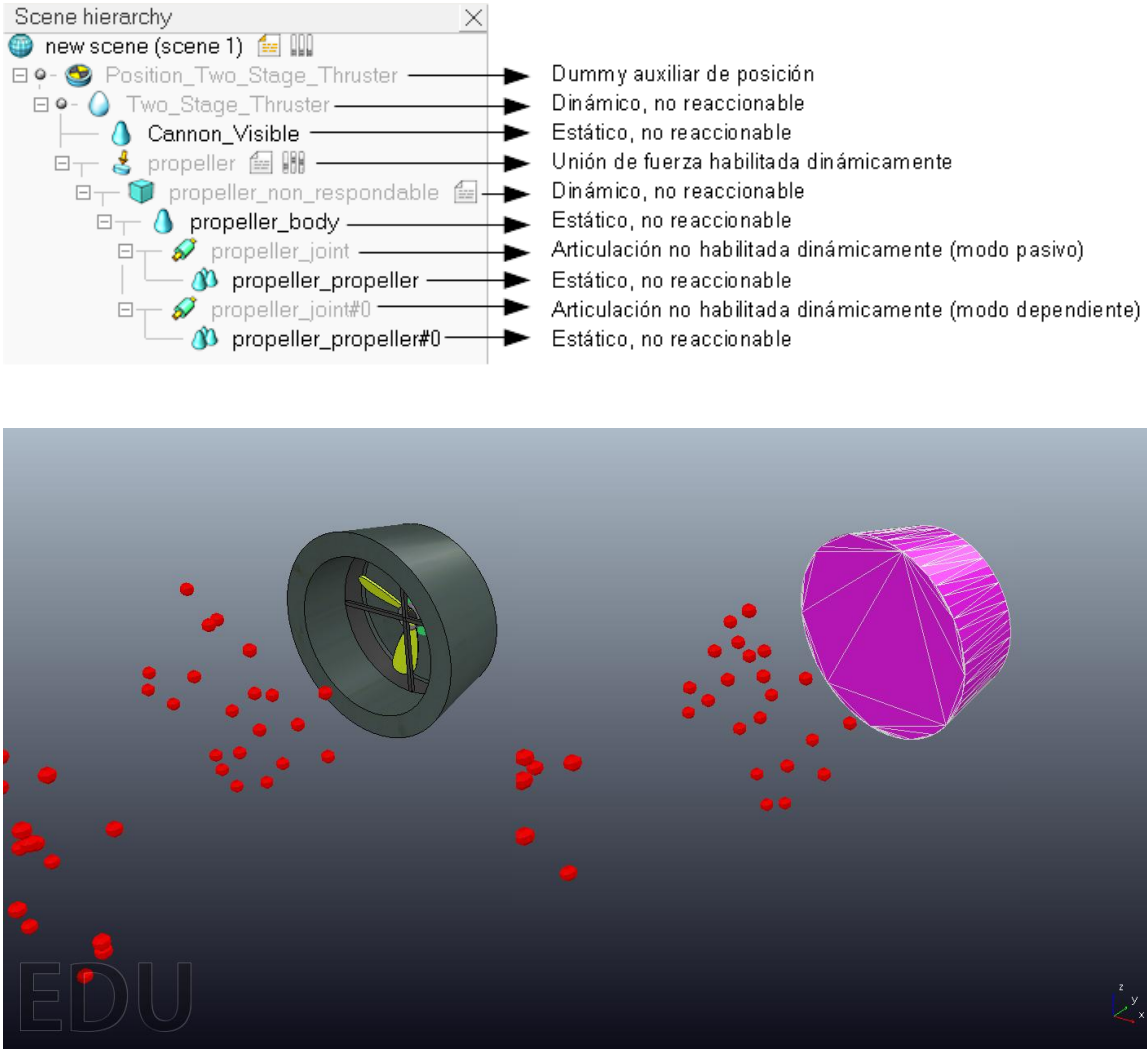


Ilustración 87. Visualización del contenido visual (izquierda) y dinámico (derecha) del modelo «Thruster».

Existen dos modos de funcionamiento establecidos en las articulaciones de sendos modelos de propulsores como puede apreciarse en las ilustraciones anteriores, donde la articulación en modo pasivo es accionado por el script asociado y la articulación en modo dependiente tiene ajustada su ecuación tal y como se muestra a continuación:

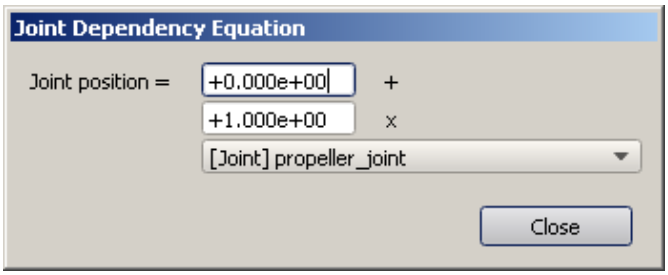


Ilustración 88. Ecuación de dependencia de la articulación «propeller_joint#0».

El modelo está destinado a ser ensamblado al modelo «Manipulator» mediante una unión de fuerza habilitada dinámicamente, para lo cual se usa el dummy auxiliar a fin de lograr su posicionamiento. Esto requiere que el dummy sea eliminado tras el acoplamiento, pues la unión de fuerza debe tener como hijo un objeto forma dinámico, el cual se ha preparado y nombrado en la jerarquía de modelo como *Thruster* –véase la ilustración 65–.

Luego para indicar al usuario esto que se acaba de expresar, se ha usado el cuadro reservado a este fin en el diálogo de propiedades de modelo del dummy donde se ha escrito en inglés:

Use the dummy "Position" to assembly the model and delete it later to obtain dynamically enabled joints or forcé sensors.

Que viene a decir en español:

Utilice el dummy "Posición" para ensamblar el modelo y elimínelo depués para obtener las articulaciones o uniones de fuerza habilitadas dinámicamente.

De este modo, quedaría definido el modelo y cuando se cargue en la escena se mostrará esta información automáticamente. El resultado de este ajuste se muestra en la siguiente ilustración:

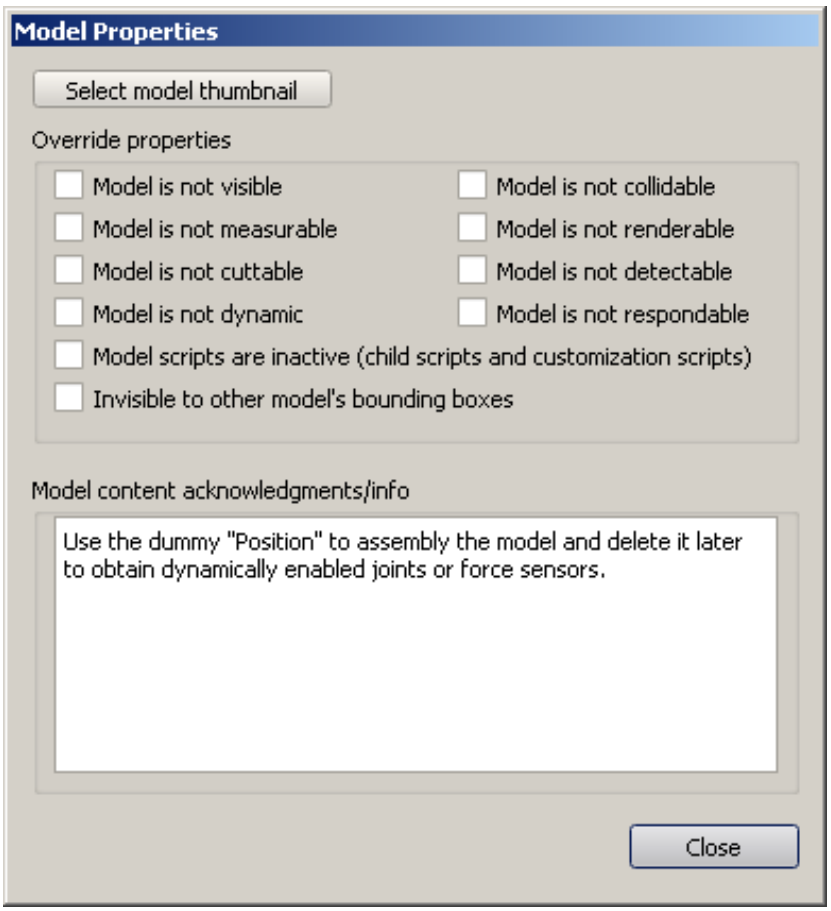


Ilustración 89. Propiedades de modelo del dummy auxiliar.

3.3.2.3. Script asociado.

El script usado para la simular tanto el empuje como la simulación de partículas del propulsor es un script secundario no secuencial al igual que el usado en el modelo del manipulador robótico, el cual ya viene asociado y escrito pues proviene del modelo ofrecido por defecto en el software como se comentó previa-mente. Para el entorno de simulación configurado y el objetivo que debe cumplir, se debe editar considerando para ello principalmente la densidad y color de las partículas, la distancia a recorrer y el empuje necesario para ello. El código escrito para el modelo se halla adjunto en el anexo B, para el cual se ha hecho uso en la edición del script principalmente de las funciones API descritas a continuación [8]:

- simGetScriptSimulationParameter

  Recupera el parámetro de una secuencia de comandos principal o secundaria de su lista de parámetros de simulación. Útil para la simple interacción con el usuario, o para el intercambio de parámetros simples con otros scripts.

- simSetJointPosition.

  Permite especificar como primer argumento el manejador de la articulación que queremos posicionar y como segundo argumento el tiempo que va a actuar, pudiendo omitir el argumento de ángulo como se ha hecho ya que se pretende la hélice gire de forma ininterrumpida hasta que se sitúe frente al satélite, donde debe detenerse en el tiempo establecido.

- simAddParticleObject.

  Agrega un objeto de partículas que se simulará y mostrará en escena. Permite especificar como argumentos el tipo, el tamaño, la densidad, el color, el número de partículas por segundo, etc.

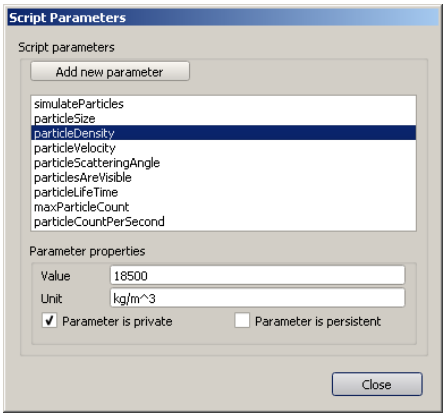


Ilustración 90. Lista de parámetros del objeto partículas asociados al script

- simAddParticleObjectItem.

  Permite especificar como primer argumento el objeto de partículas previamente definido y como segundo argumento la posición y orientación de éstas.

- simRemoveParticleObject.

  Elimina las partículas añadidas previamente.

- simAddForceAndTorque.

  Añade una fuerza y/o par a un objeto de forma dinámicamente habilitado, como es nuestra unión de fuerza.

La lista de parámetros de simulación que definen al propulsor se recoge en la tabla siguiente, donde se ha tenido que definir dos nuevas variables respecto a la lista de parámetros original y variar el tamaño de las partículas establecida por defecto.

Parámetros	Significado	Valor	Unidad
simulateParticles	Simular	True	Boolean
particleSize	Tamaño	2	Relative to model size*0.005
particleDensity	Densidad	18500	kg/m^3
particleVelocity	Velocidad	4	m/s
particleScatteringAngle	Ángulo	30	Degrees
particlesAreVisible	Visibilidad	True	Boolean
particleLifeTime	Tiempo de vida	0,5	s
maxParticleCount	Nº máximo	400	-
particleCountPerSecond	Partículas/seg	430	s^{-1}
StartTime	Tiempo de inicio	Depende	Depende
FinalTime	Tiempo final	Depende	Depende

Tabla 3. Lista de parámetros conectados al script del modelo «Thruster».

De los parámetros establecidos, se puede obtener la fuerza de empuje resultante por partícula usando la expresión (3.1). Cabe señalar que el parámetro de tamaño establecido se traduce en que el tamaño de la partícula es igual al 1 % del tamaño del modelo.

$$F = 18500\left[\frac{kg}{m^3}\right] \cdot 4\left[\frac{m}{s}\right] \cdot \pi \cdot \frac{(2 \cdot 0.005)^3}{6} \cdot \frac{1}{0.05} = 0{,}7749\ \left[\frac{N}{partículas}\right] \quad (3.2)$$

La uniformidad de las partículas se ha configurado en el código mediante un bucle while-do

–véase el anexo B–, cuyo funcionamiento es el siguiente: se ejecuta el bloque de sentencias y luego evalúa la condición; si ésta es cierta se vuelve al principio del bucle, pero si es falsa se termina. La condición se comprueba, cada vez, después de la ejecución del cuerpo del bucle; por tanto, el bloque de sentencias siempre se ejecuta como mínimo una vez. Todo ello nos lleva a dos posibles valores de partículas en función de las partículas establecidas por segundo (particleCountPerSecond) y la fuerza de empuje por partícula. Por ello, en dicho código se usa "particleCnt" en lugar de "requiredparticleCnt", pues se pretende reproducir una señal triangular que simula la reacción de propulsión uniforme. En resumen, el código evalúa los dos siguientes valores de partículas: 21 porque la condición es verdadera; y 22, porque aun siendo falsa la condición se ejecuta como mínimo una vez. Luego:

$$F = \begin{cases} 0{,}7749 \left[\frac{N}{partículas}\right] \cdot 21[partículas] = 16{,}25\ N \\ 0{,}7749 \left[\frac{N}{partículas}\right] \cdot 22[partículas] = 17{,}04\ N \end{cases} \quad (3.3)$$

Si comprobamos el 75 % de la fuerza de empuje del monopropulsor MONARC-22 se obtiene 16,5 N, valor muy próximo al valor medio de la fuerza de empuje obtenida. Todo lo anterior se puede ver adjuntando una gráfica de fuerza en V-REP que mida dicha fuerza a través de la unión de fuerza que une el propulsor al satélite como se muestra en la ilustración siguiente:

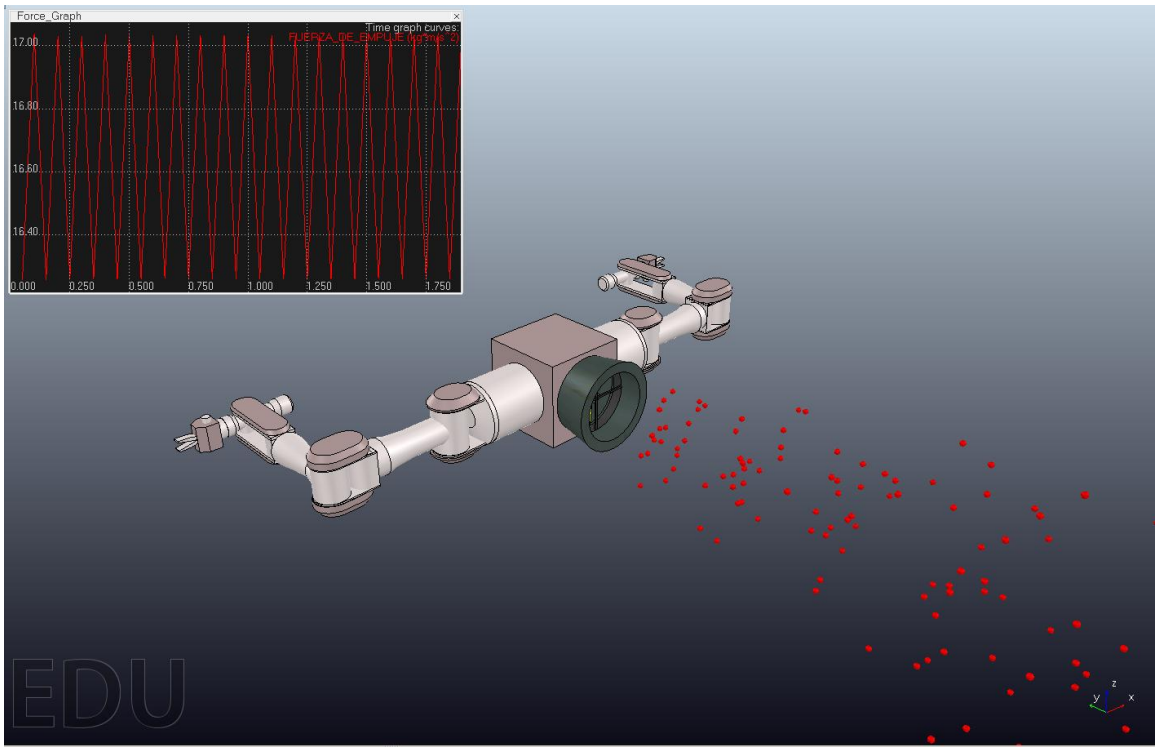


Ilustración 91. Fuerza de empuje del modelo «Thruster».

Los parámetros "StartTime" y "FinalTime" dependen del ciclo de funcionamiento de cada pro-

pulsor. Teniendo en cuenta que los propulsores superior e inferior son de dos etapas y el frontal y el posterior son de una etapa, se presenta a continuación el diagrama de funcionamiento de los distintos propulsores:

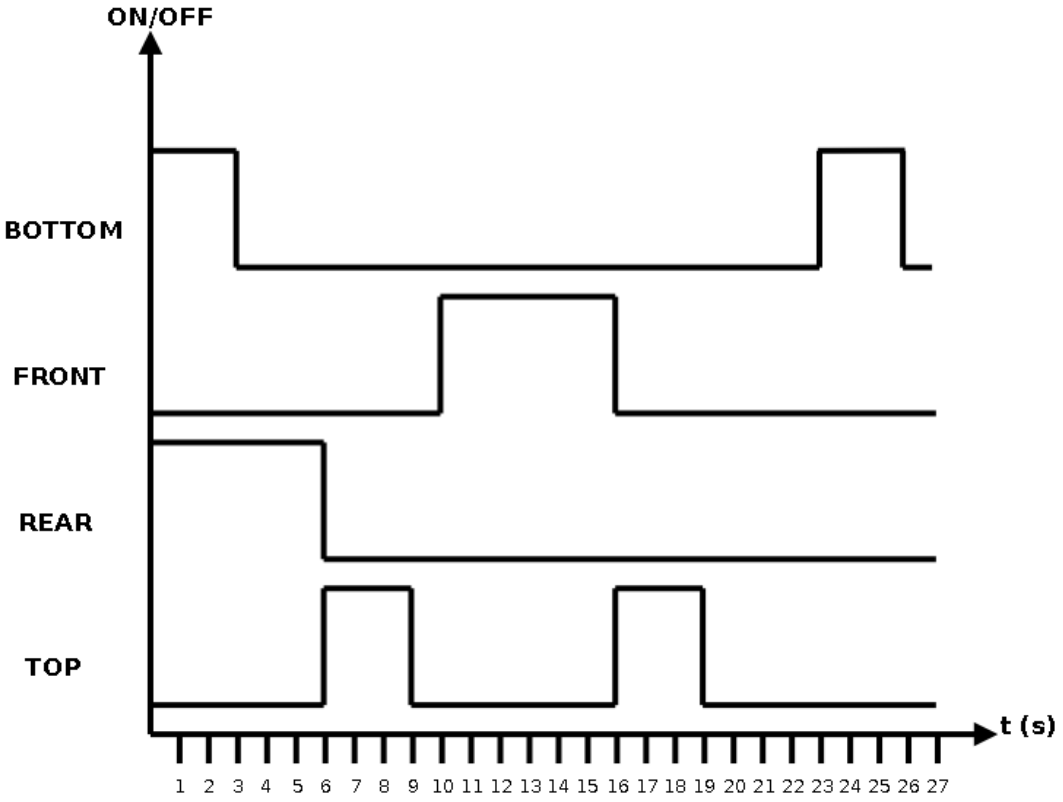


Ilustración 92. Diagrama de funcionamiento de los propulsores del satélite de servicio.

3.3.3. Tronco.

El tronco del satélite cliente se ha identificado como la base del modelo, y a través de él se ha compensado el peso y la inercia necesarios, tras ensamblar los brazos robots y los propulsores, a fin de lograr la estabilidad dinámica del modelo. Por ello, mediante el ajuste experimental, se ha llegado a la conclusión de que con una masa de 200 kg y los siguientes módulos de inercia de : I_x = 2658 m^2, I_y = 1801 m^2 y I_z = 1796 m^2, se alcanza la estabilidad dinámica.

Ilustración 93. Vista del modelo «Manipulator» en el navegador de modelos.

3.3.3.1. Descripción.

El modelo consta de una forma convexa simple como objeto dinámico y reaccionable que abarca el volumen de los propulsores y de sí mismo; de un objeto de forma pura, osea el cubo central visible, destinado a la reproducción visual; y por último, 5 uniones de fuerza destinados a ensamblar los manipuladores robóticos así como los propulsores.

Se presenta inicialemente con las uniones de fuerza visible para dar idea al usuario de donde ensamblar los elementos, tras lo cual debe ocultarse a la capa de visibilidad correspondiente.

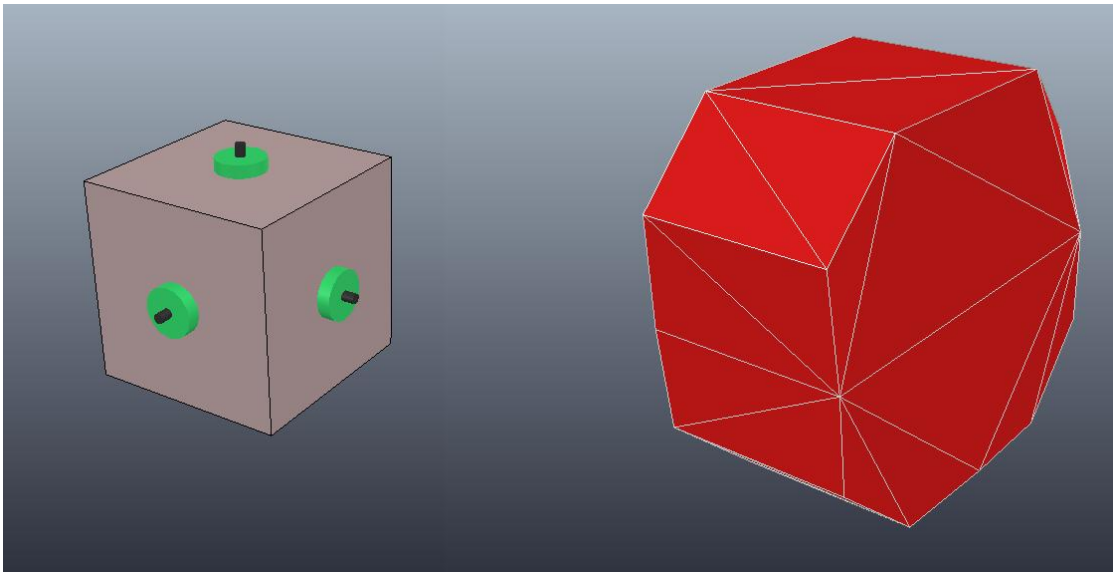


Ilustración 94. Visualización del contenido visual (izquierda) y dinámico (derecha) del modelo «Manipulator».

3.3.3.2. Jerarquía de objetos.

La jerarquía de objetos del modelo «Manipulator» se presenta incompleta debido a que "espera" a ser habilitado dinámicamente, pues sus uniones como se pueden ver no lo están. En las uniones laterales deben ensamblarse los brazos robóticos y en las restantes los propulsores.

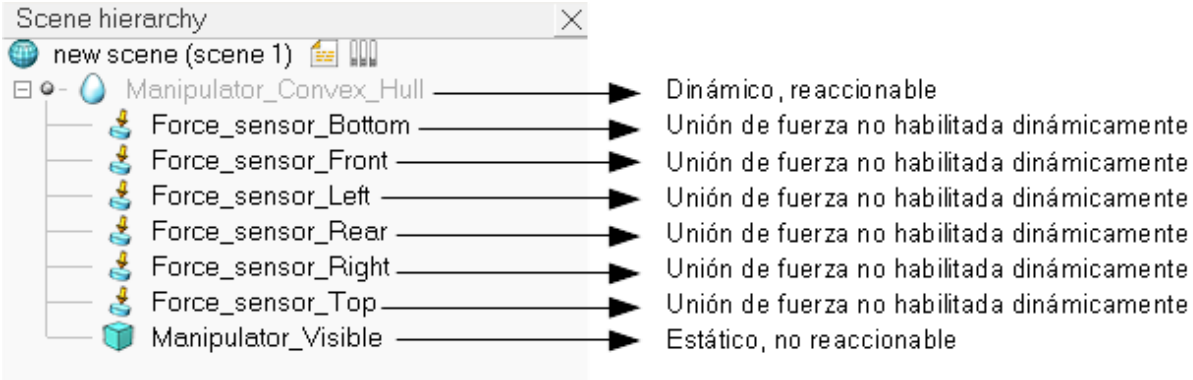


Ilustración 95. Jerarquía de objetos del modelo «Manipulator».

3.4. Otros modelos.

En este último apartado se presentan modelos auxiliares destinados a suplir funciones secundarias del entorno de simulaicón, como son el registro del tiempo de simulación, la visualización del centro de masa de los satélites durante la simulación, y los mensajes emergentes necesarios para comunicar información al usuario durante la simulación.

3.2.1. Reloj.

Es un modelo que viene por defecto en el programa y se halla presente en el navegador de modelos, en la sección *other*. Sirve para medir el tiempo de simulación a partir de su ejecución.

Ilustración 96. Modelo de reloj de tiempo de simulación.

3.4.2. Herramienta de visualización del centro de masa.

Este modelo va ensamblado como hijo de ambos satélites y se halla presente en el navegador de modelos, concretamente en la sección *tools*.

Ilustración 97. Modelo de la herramienta de visualización del centro de masa en el navegador de modelos.

3.4.3. Pop-up.

Este modelo, que usa como base un dummy, sirve tan solo para notificar al usuario que deshabilite la opción que oculta la ventana de consola en los ajustes de usuario, y anuncia por la barra de estado el título del proyecto. El código encargado de dichas operaciones se adjunta en el anexo C.

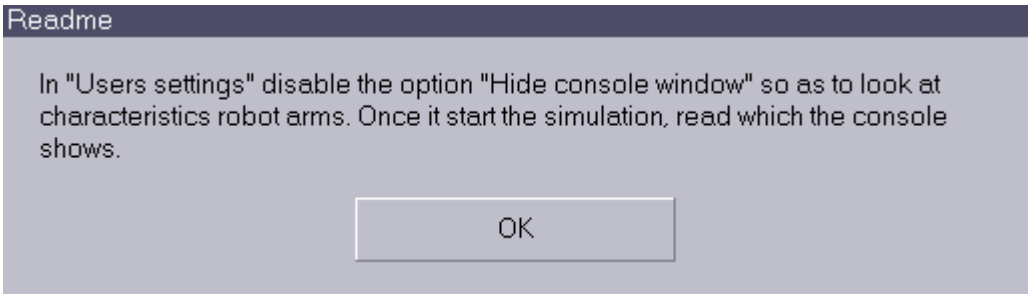


Ilustración 98. Ventana emergente lanzado al inicio de la simulación.

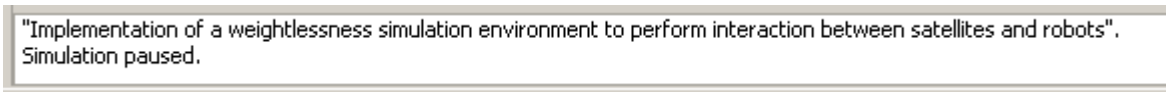


Ilustración 99. Anuncio del título del TFG por medio de la barra de estado.

4. VALIDACIÓN.

En este último capítulo, se lleva a cabo la validación y verificación del contenido dinámico de la escena comprobando que todo lo desarrollado hasta ahora ha sido configurado correctamente de cara a los propósitos dispuestos al principio: acercamiento del satélite servicio frente al satélite mediante la actuación de los propulsores y la interacción producida a través de los controles de usuario debe producir un lanzamiento del satélite cliente a la deriva a fin de comprobar que la escena ha sido configurada correctamente para reproducir el espacio.

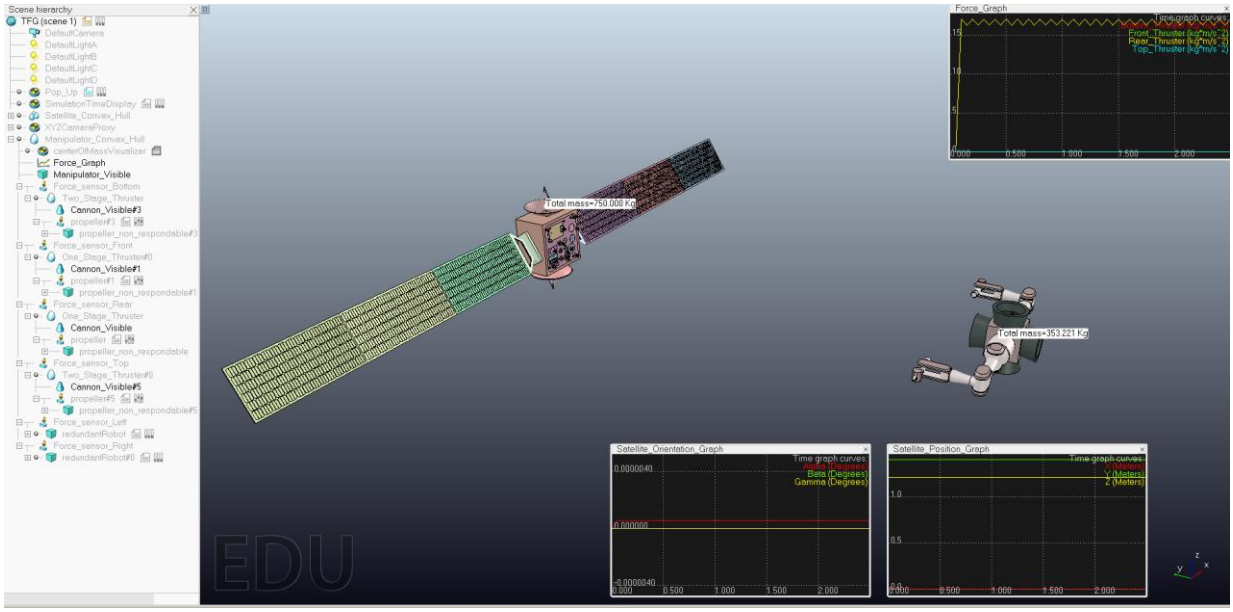


Ilustración 100. Entorno de simulación del TFG junto a la jerarquía de escena resultante.

4.1. Introducción.

Sentados los conceptos, el desarrollo a seguir y la configuración que debe establecerse para el entorno de simulación enfocado a la órbita espacial, nos dirigimos a ejecutar la simulación mediante el botón correspondiente de la barra nº 1 de la barra de herramientas y alternando el botón de visualización de contenido dinámico a fin de comprobar que la configuraciones de la jerarquía de escena es la correcta.

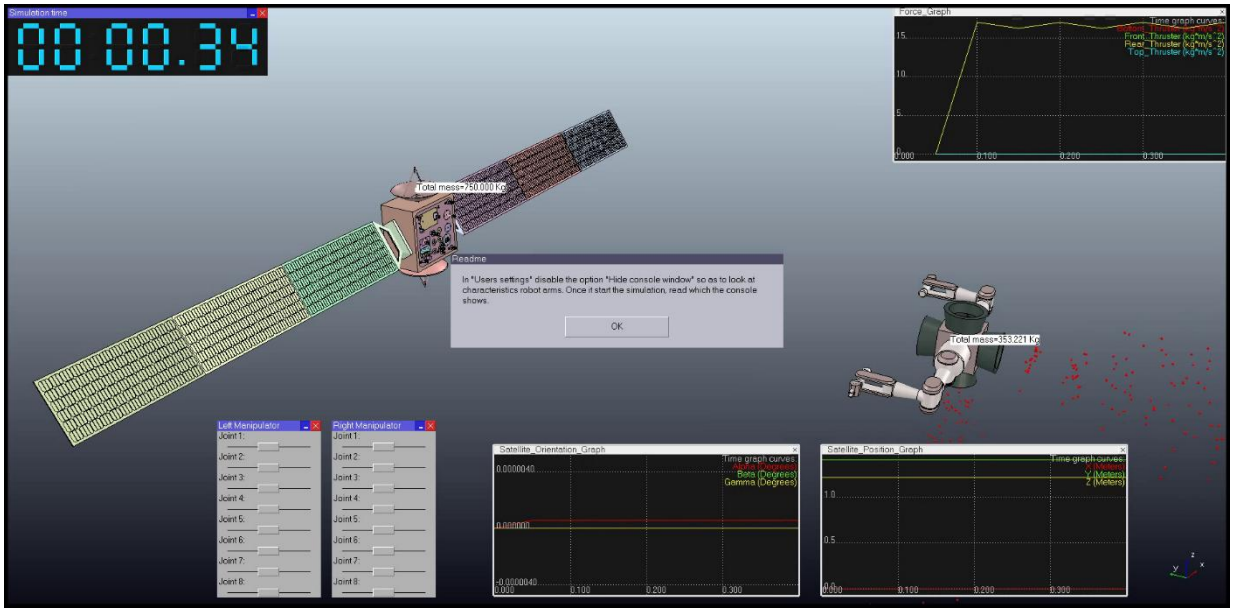
4.2. Simulación.

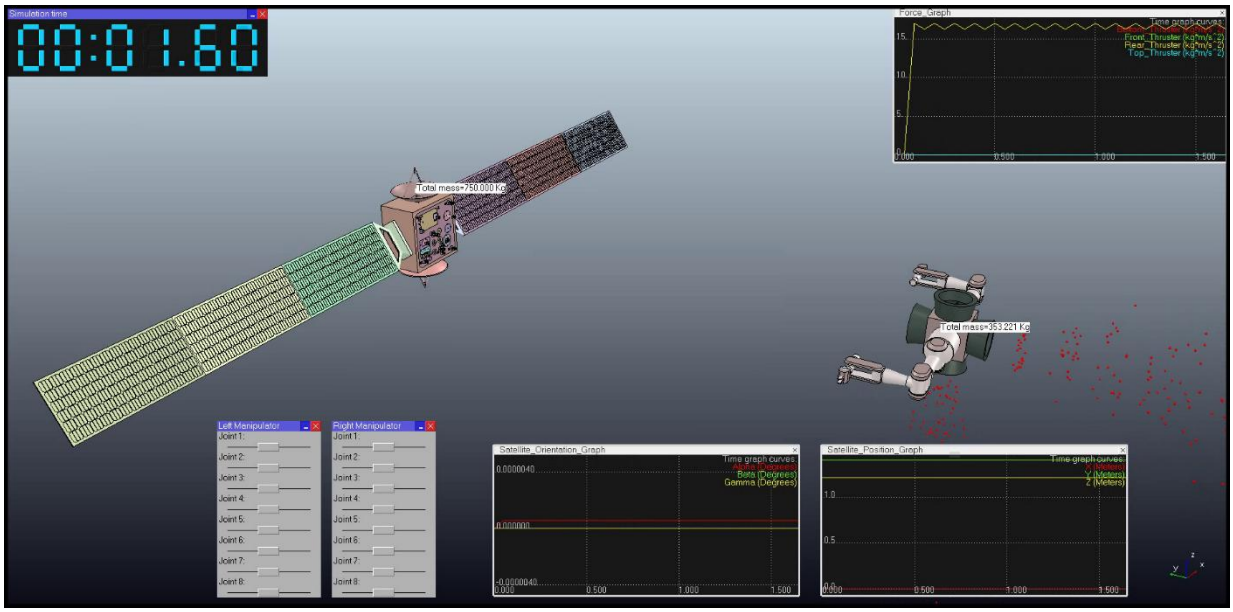
En este apartado se presentarán los resultados de la simulación. Por un lado se mostrarán secuencias sucesivas de la reproducción visual, y a posteriori la reproducción dinámica. De este modo corroboraremos que tanto los objetos visuales como los dinámicos son válidos para la simulación. Además de ello, se podrán ver los registros de movimiento del satélite y el funcionamiento secuencial de los propulsores mediante las gráficas configuradas a tal efecto.

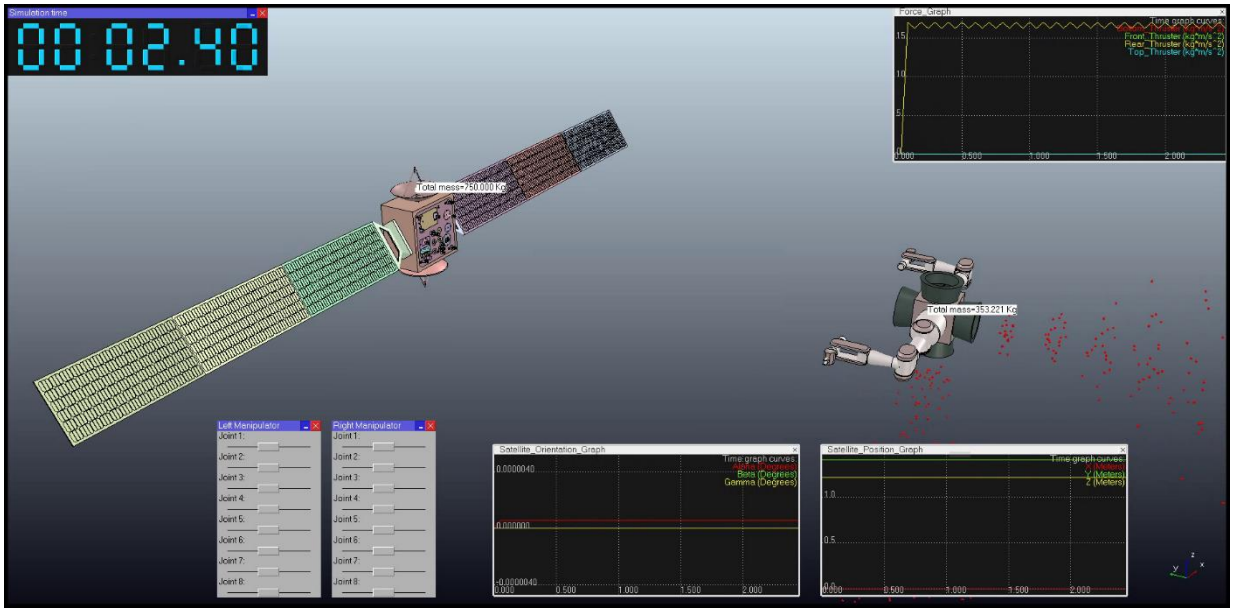
4.2.1. Reproducción visual.

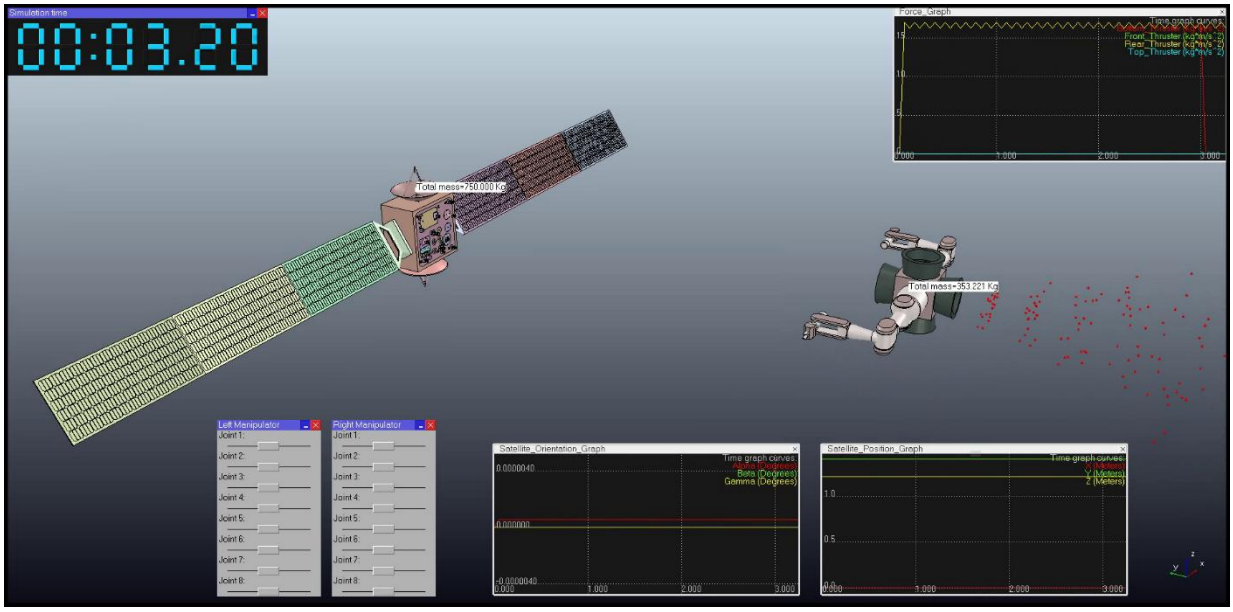
En este apartado se mostrarán de forma concatenada las secuencias de simulación resultantes de la reproducción visual. No se harán referencia alguna en el índice de referencias pues no tiene relevancia enumerar y nombrar cada imagen en vista de la función que cumple dicha concatenación.

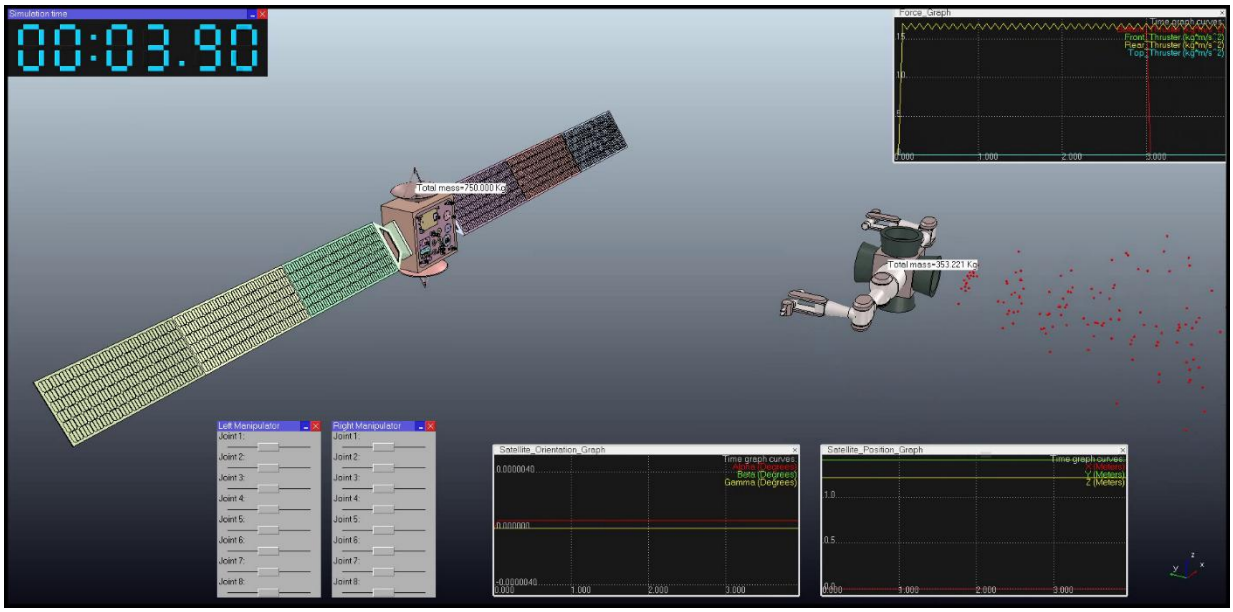
La simulación se inicia con la ventana emergente indicando habilitar la consola para ver la información relacionada con las articulaciones, que tras pulsar “ok” sigue ejecutándose.

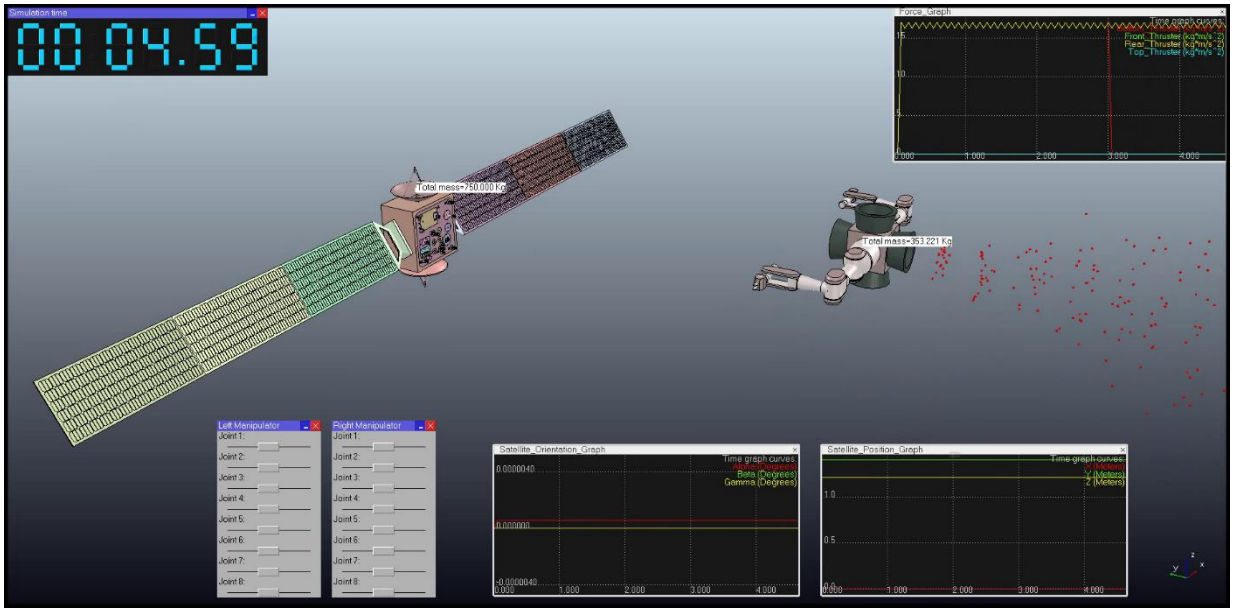


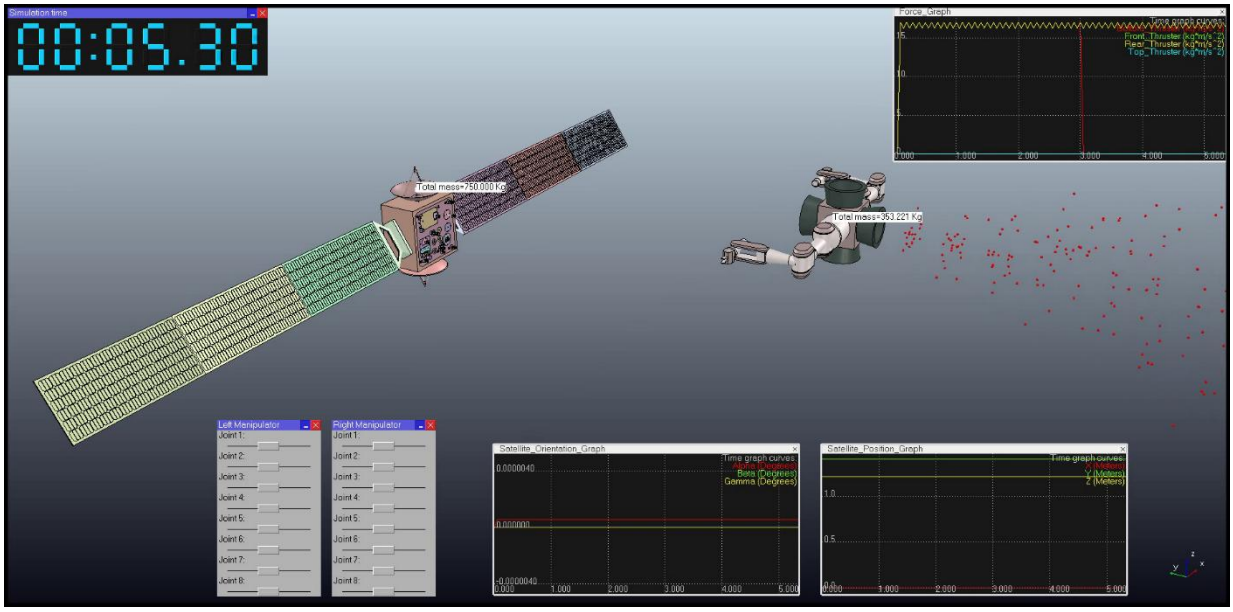

Simulation time
00:01.60
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

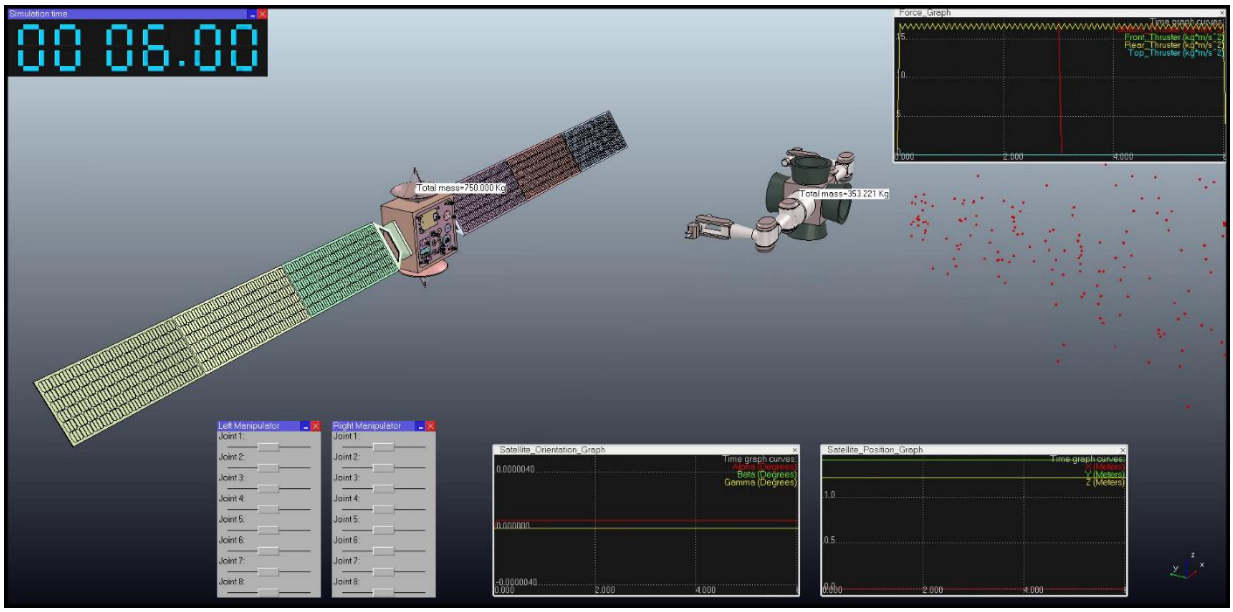

Simulation time
00 02.40
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

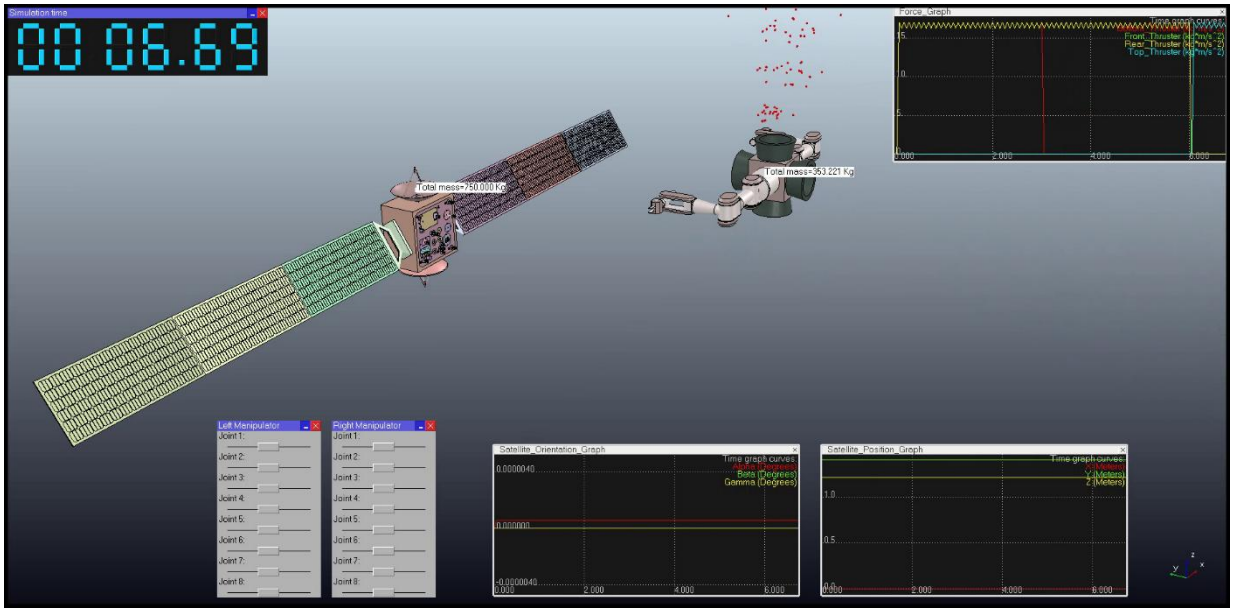

Simulation time
00:03.20
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

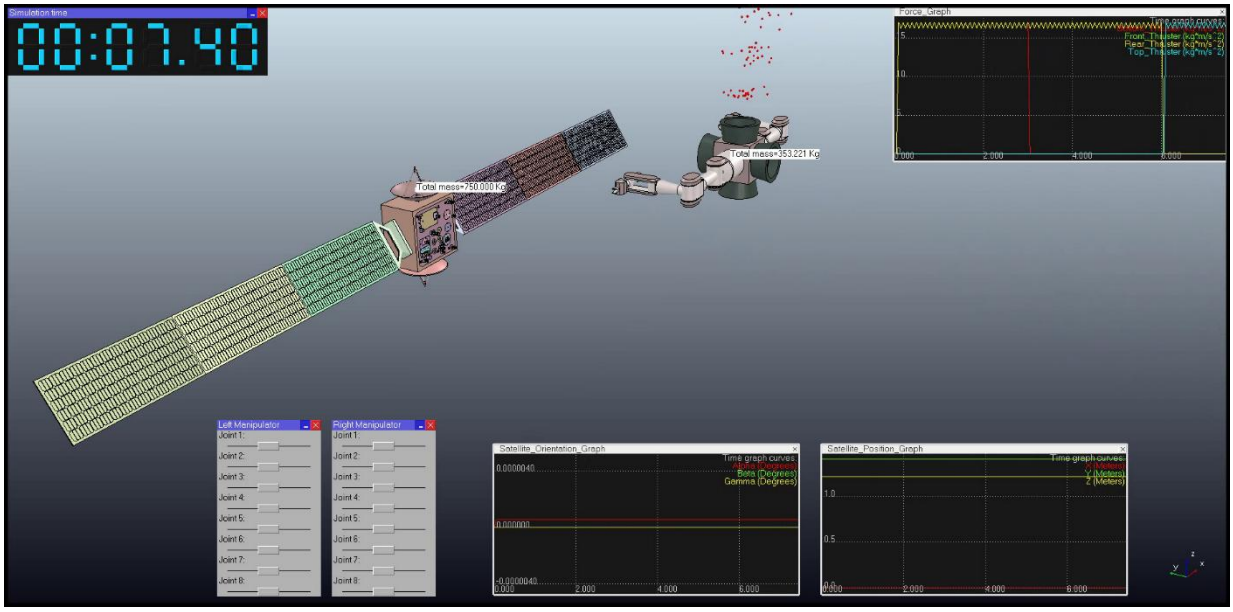

Simulation time
00:03.90
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

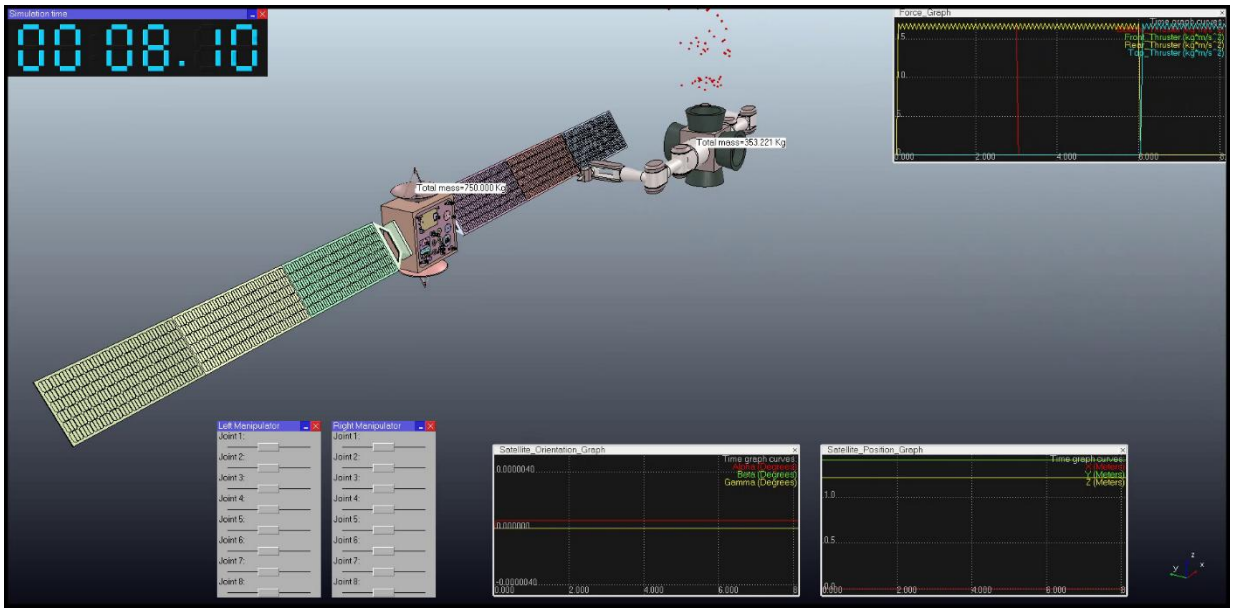

Simulation time
00 04.59
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

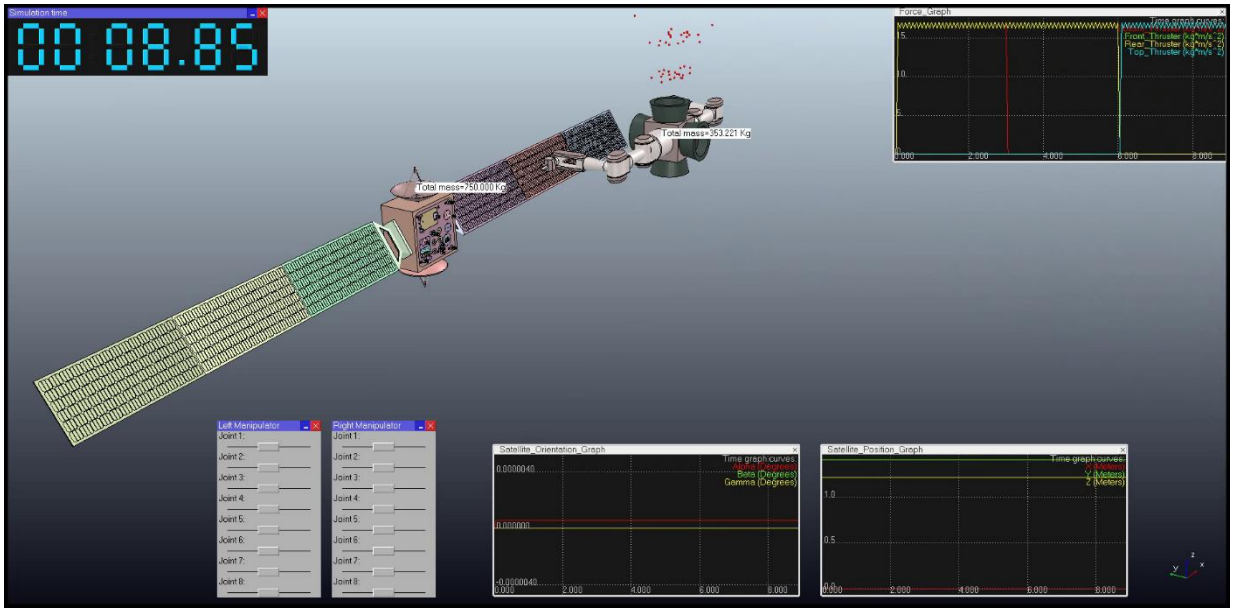

Simulation time
00:05.30
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

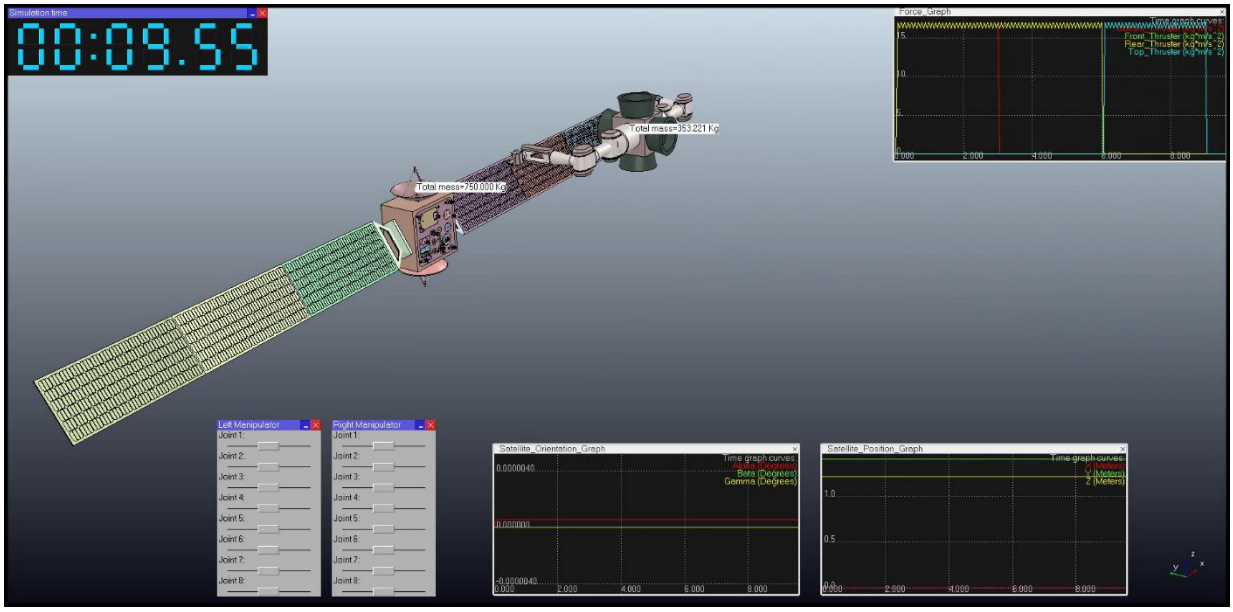

Simulation time
00 06.00
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

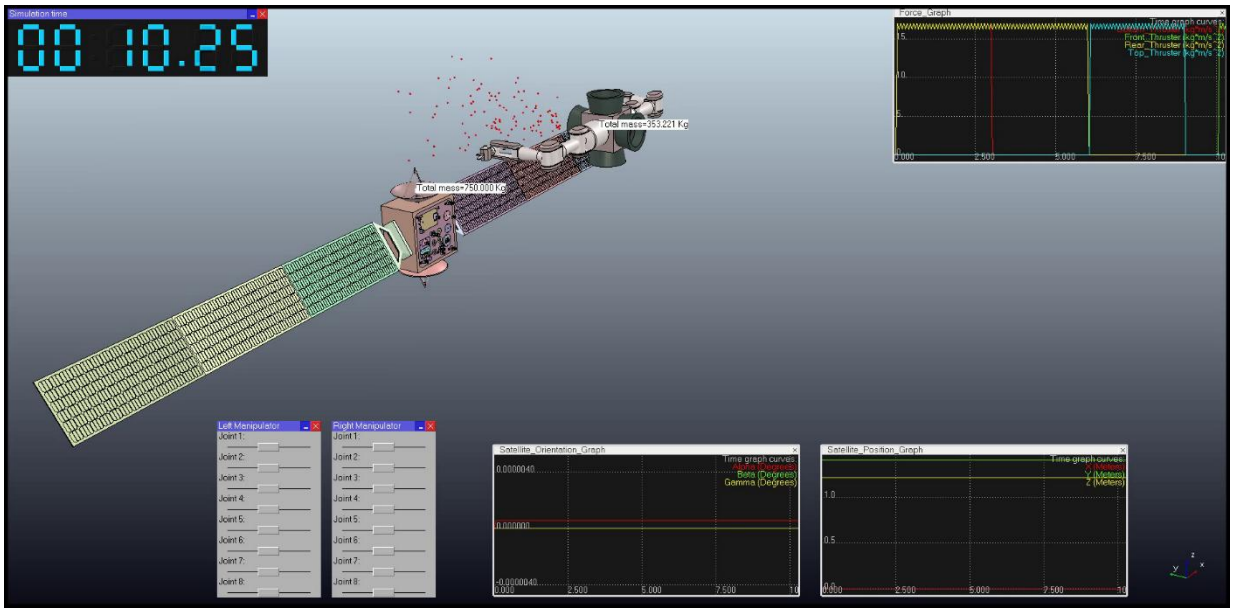

Simulation time
00 06.69
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

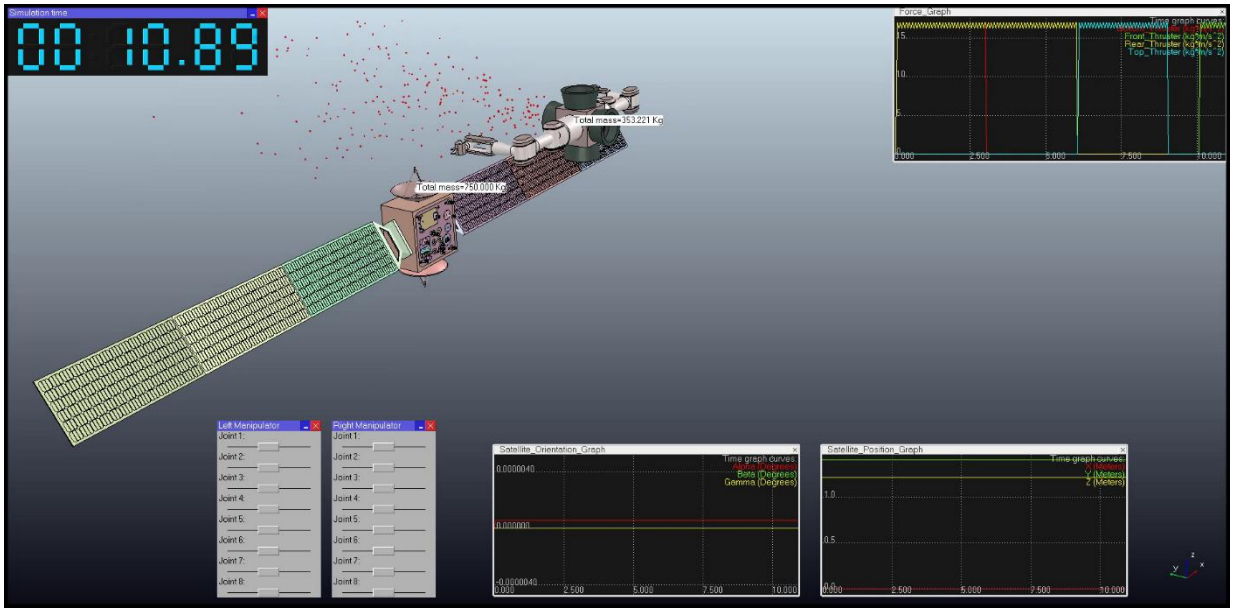

Simulation time
00:07.40
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

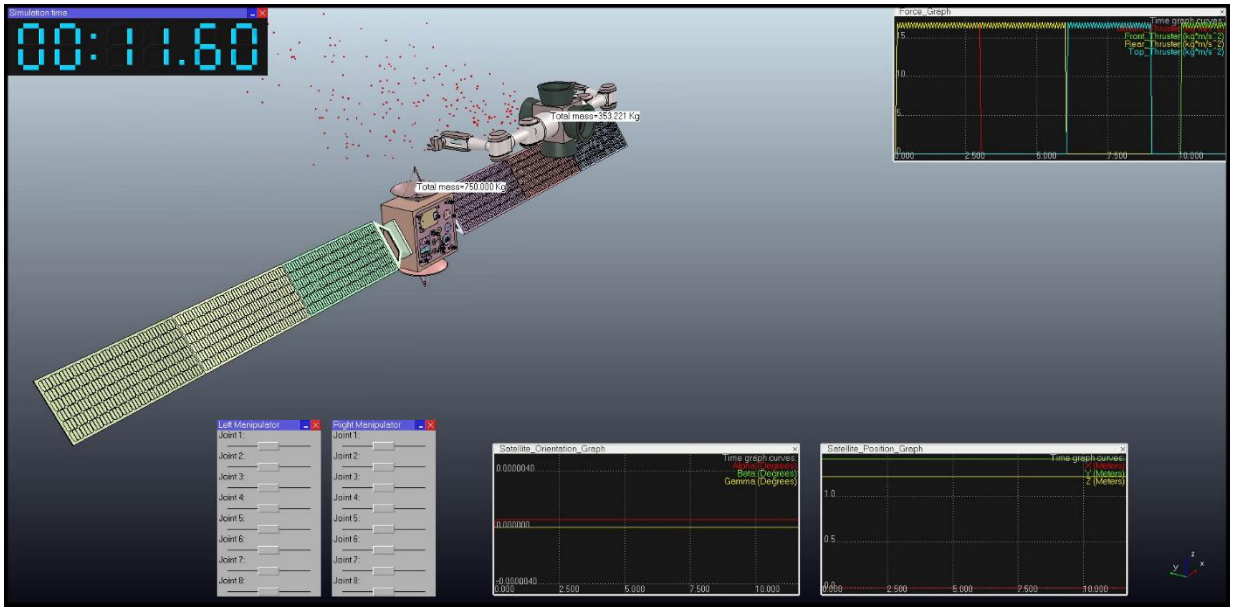

Simulation time
00 08.10
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

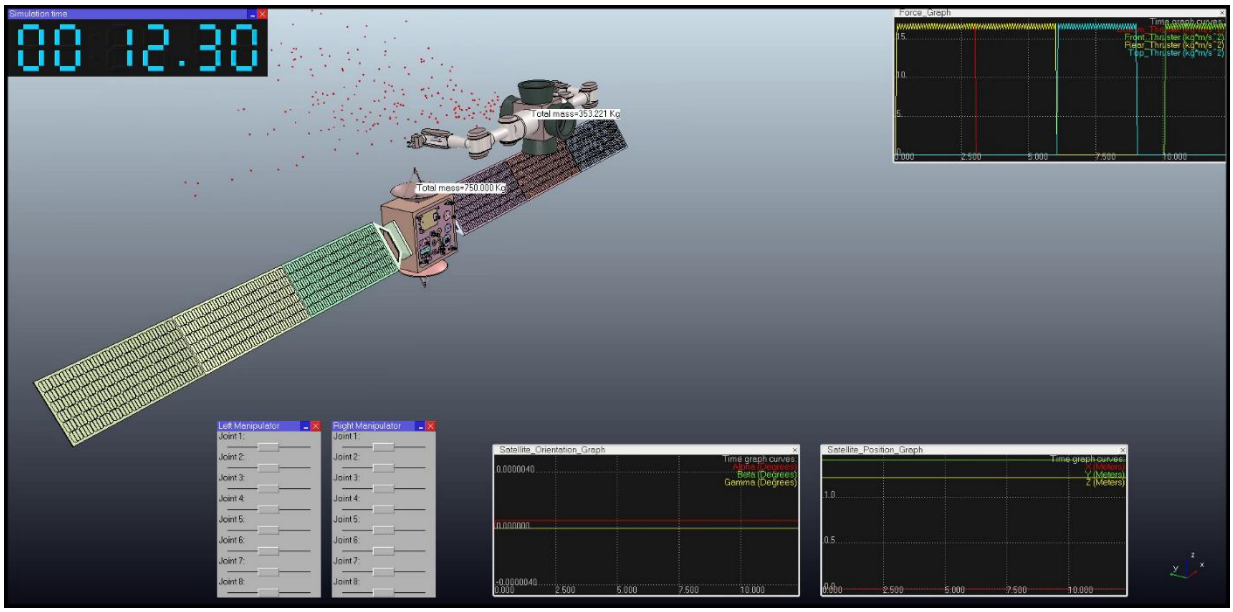

Simulation time
00 08.85
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

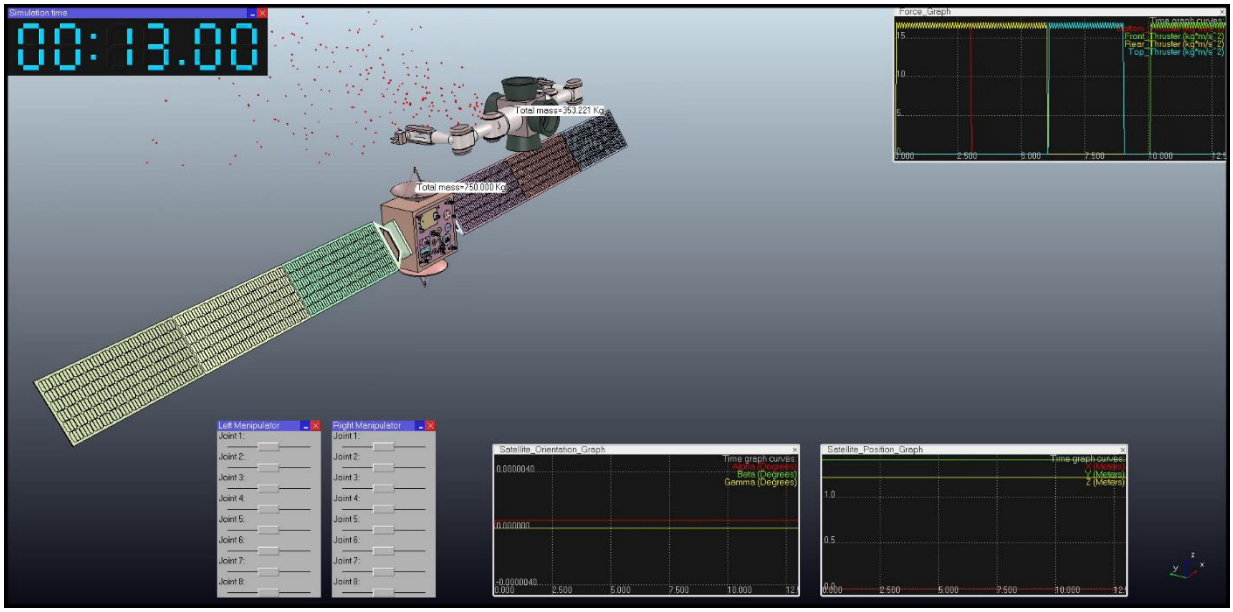

Simulation time
00:09.55
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

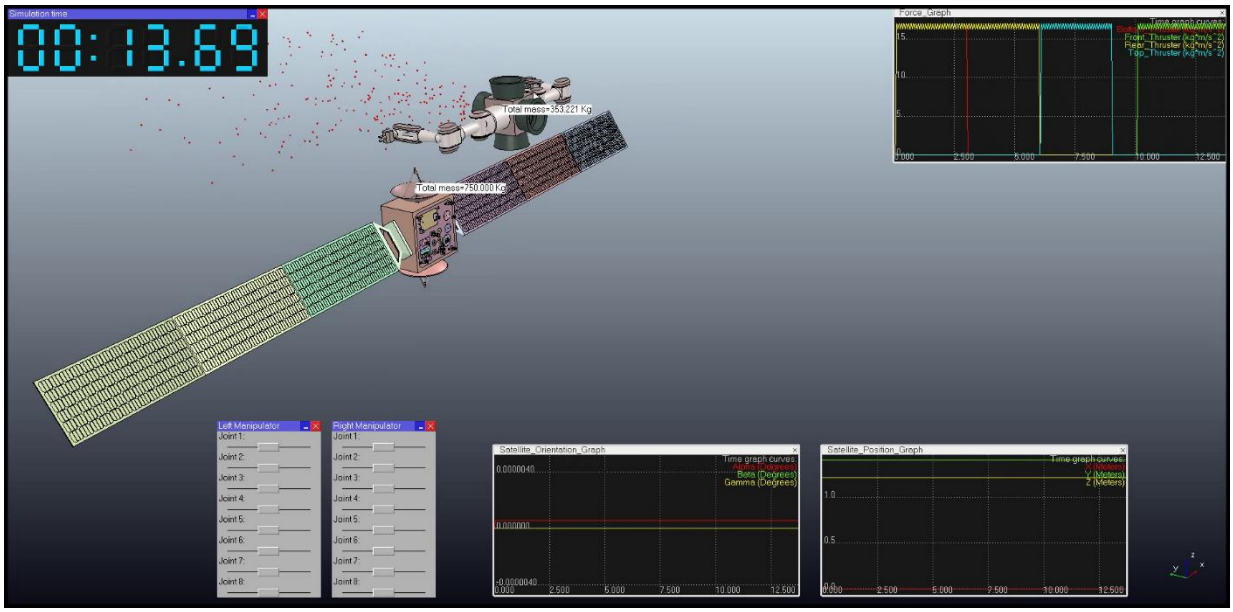

Simulation time
00 10.25
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

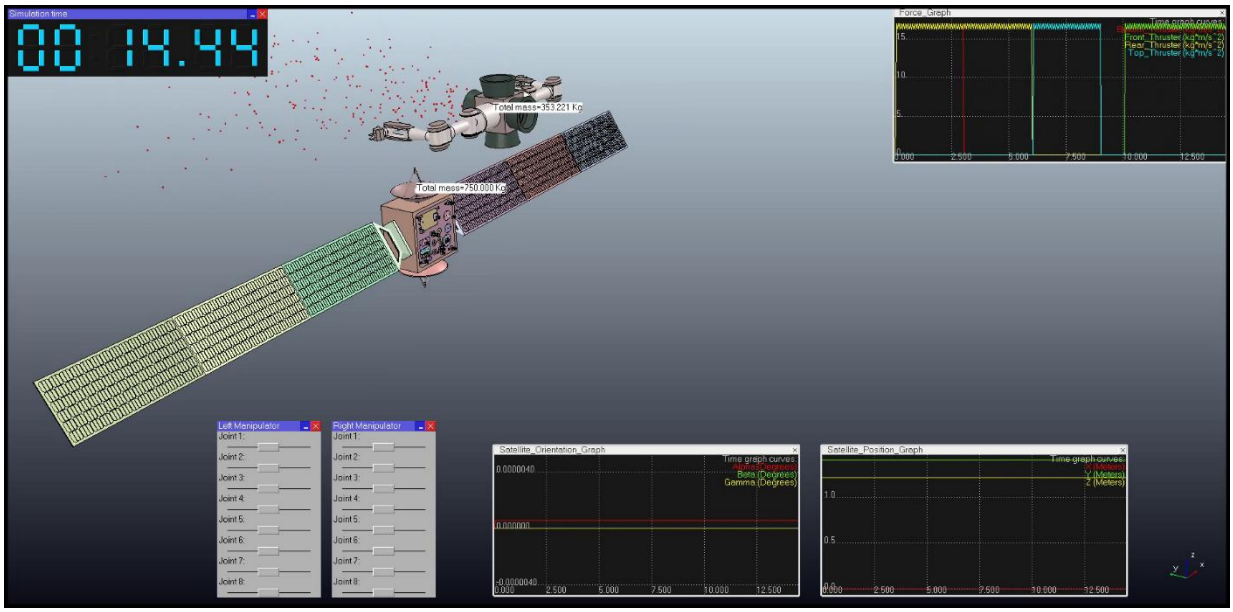

Simulation time
00 10.89
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

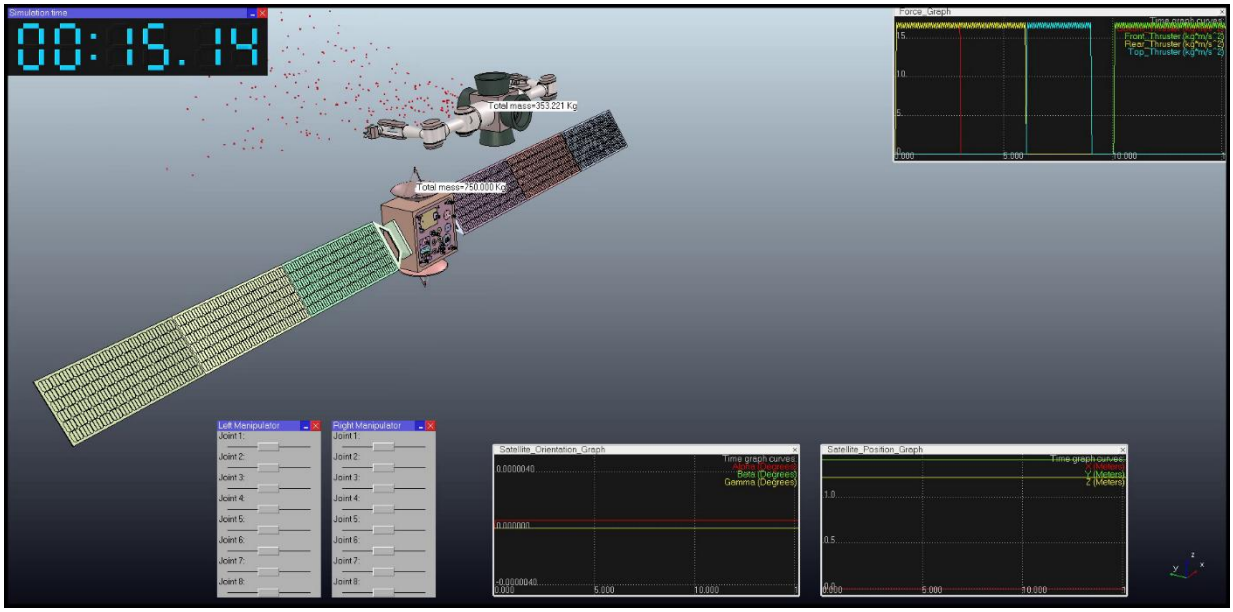

Simulation time
00: 11.60
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

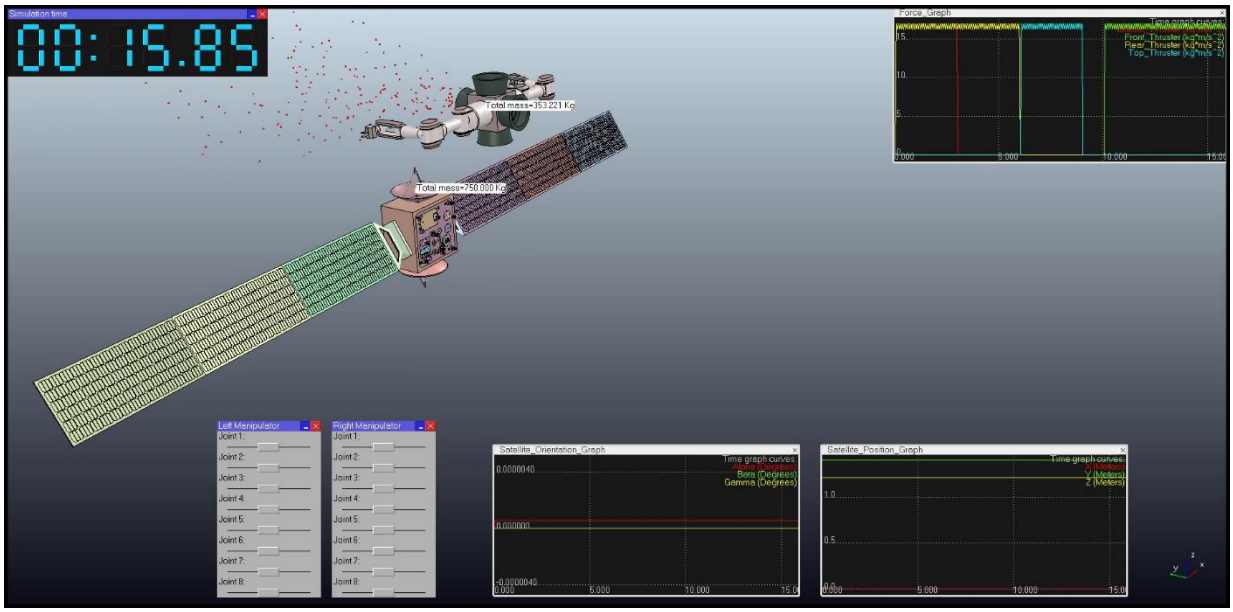

Simulation time
00 12.30
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

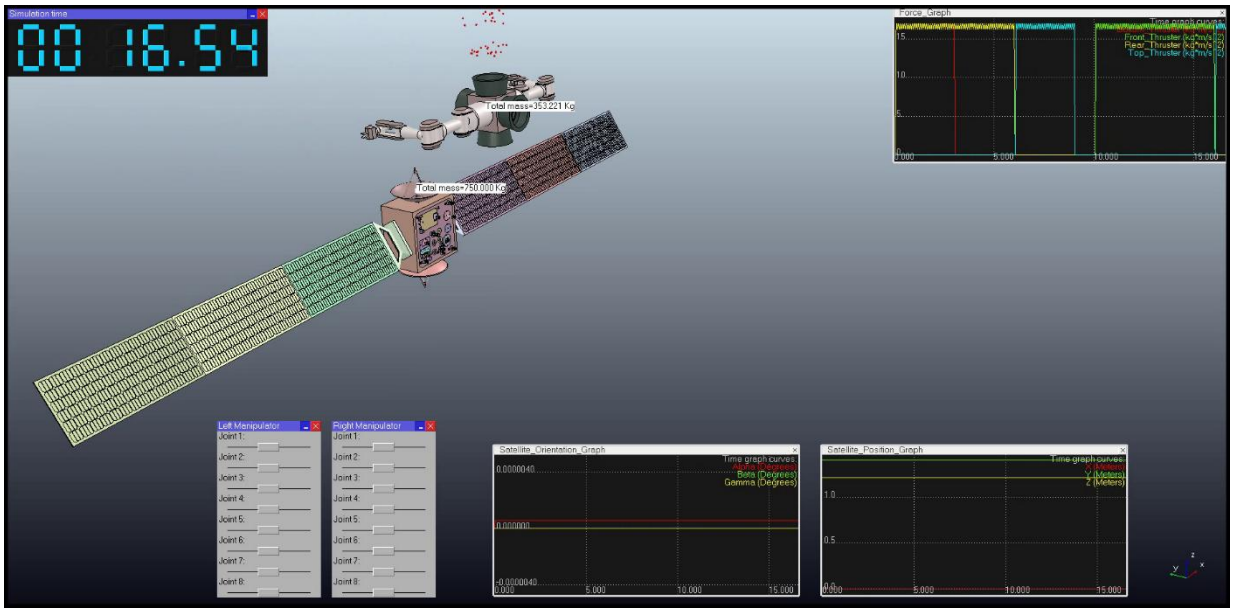

Simulation time
00: 13.00
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

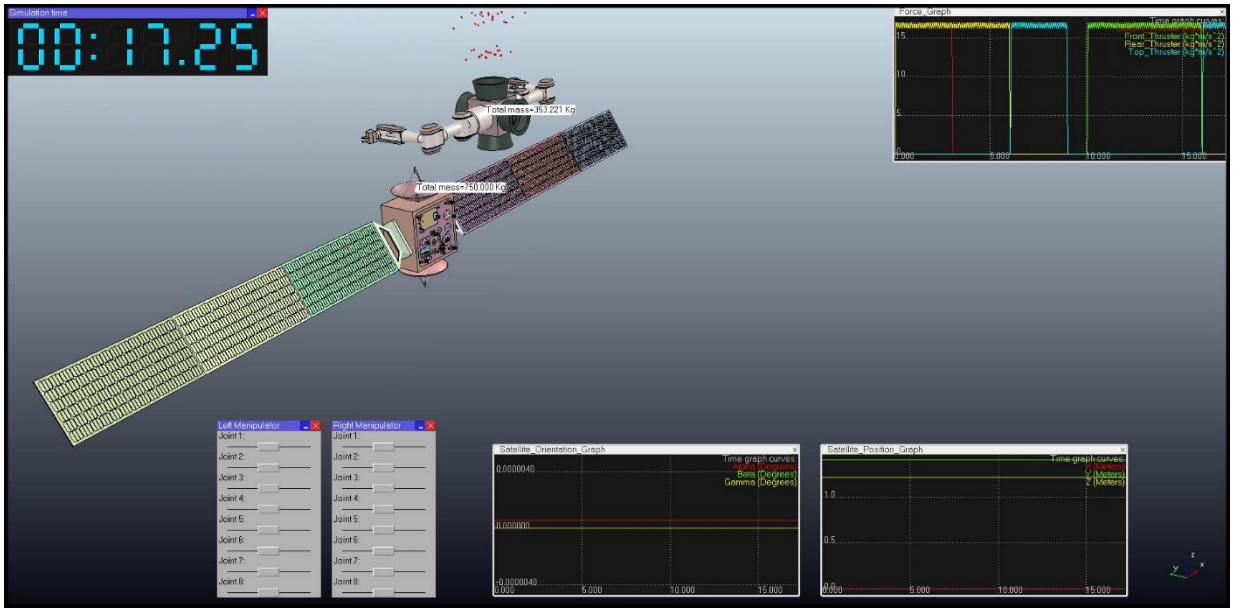

Simulation time
00: 13.69
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

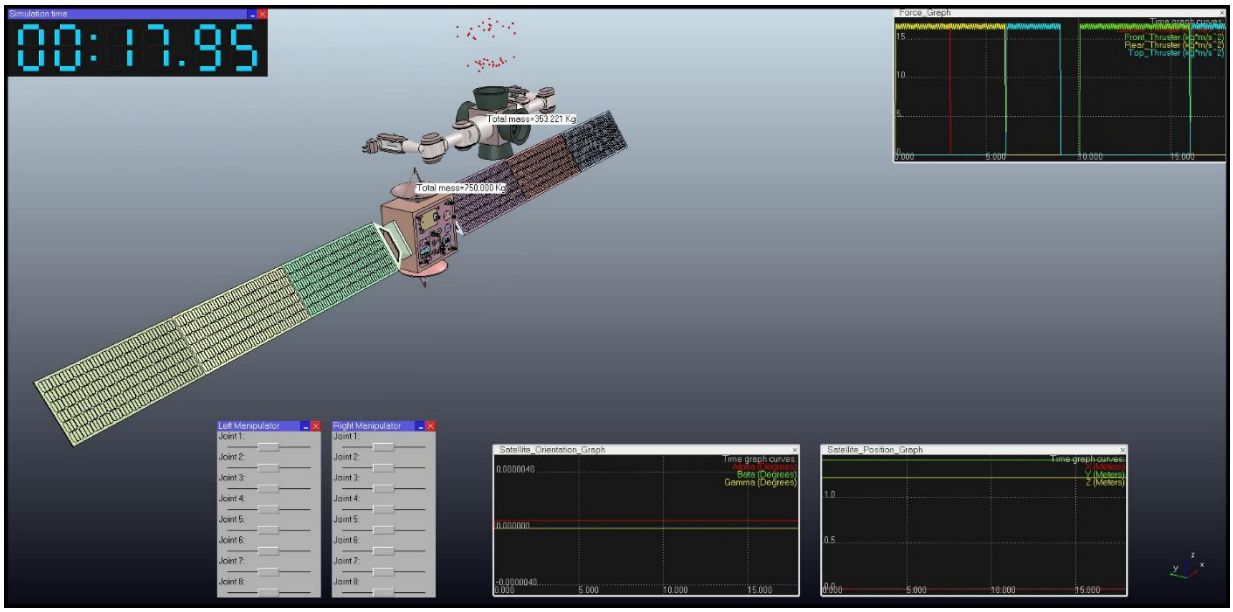

Simulation time
00 14.44
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

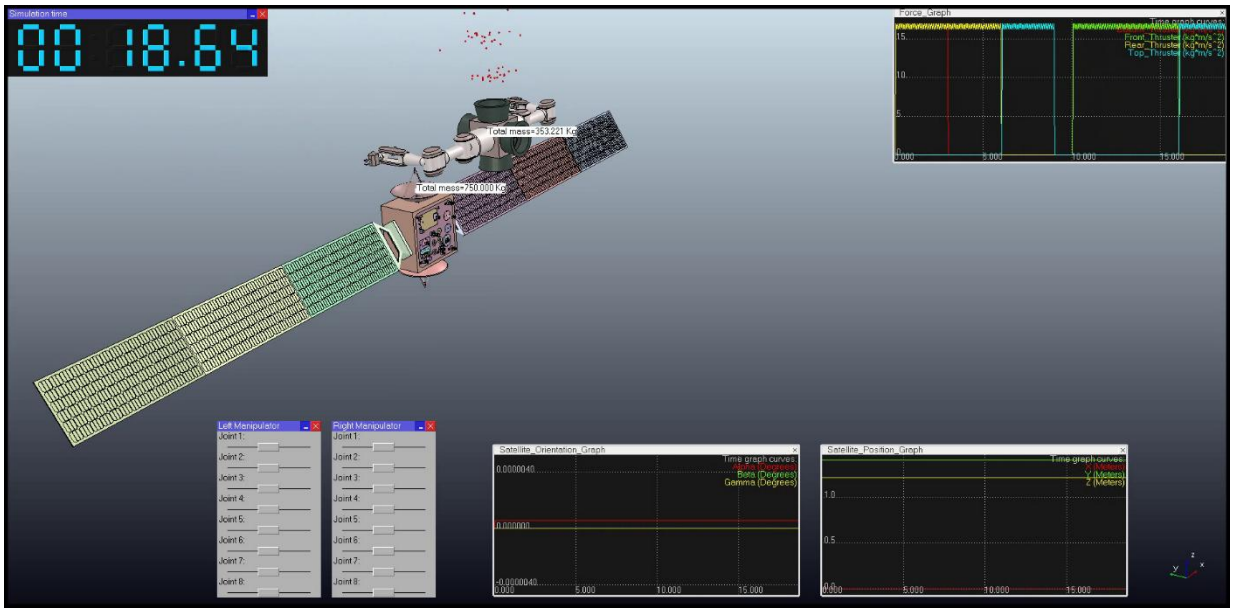

Simulation time
00: 15. 14
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

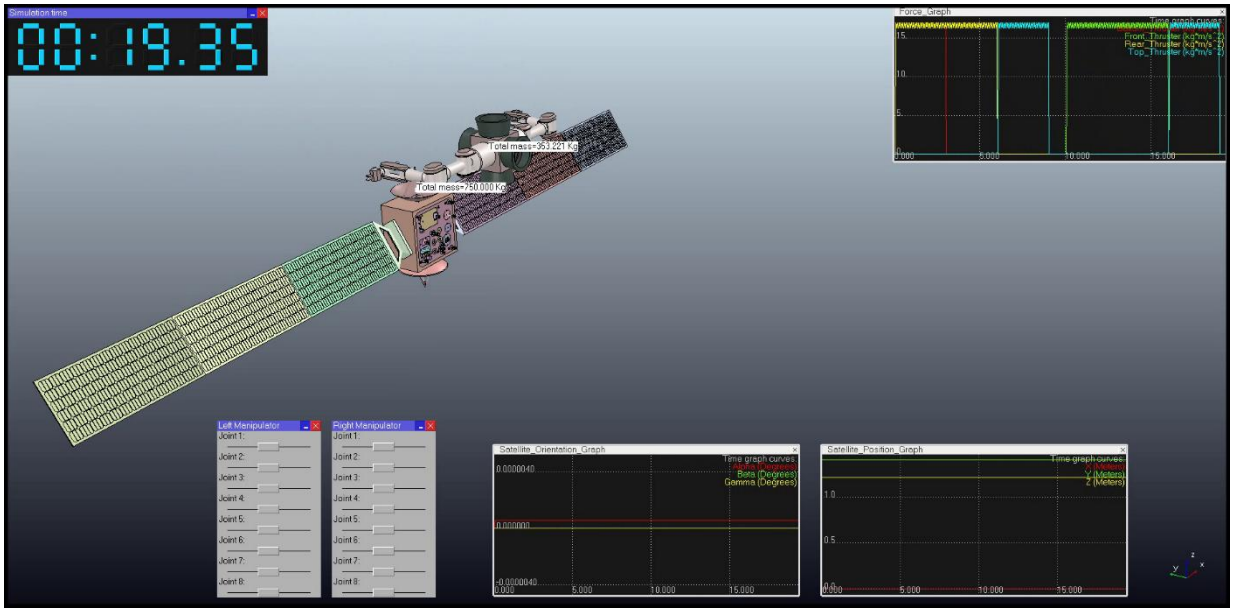

Simulation time
00: 15.85
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

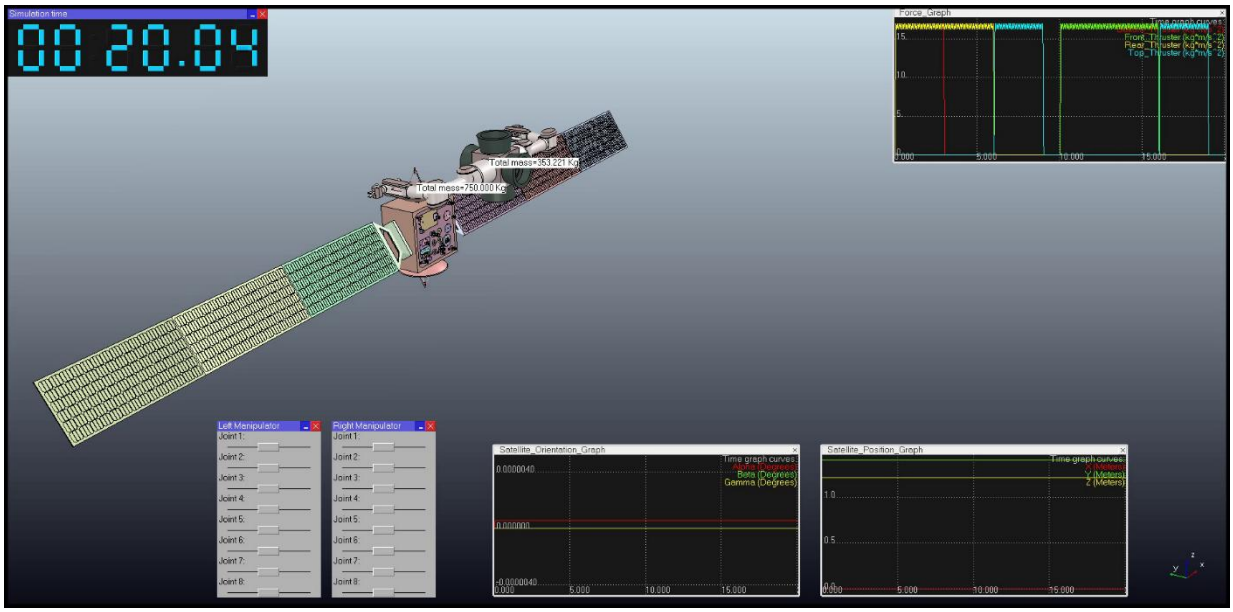

Simulation time
00 16.54
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

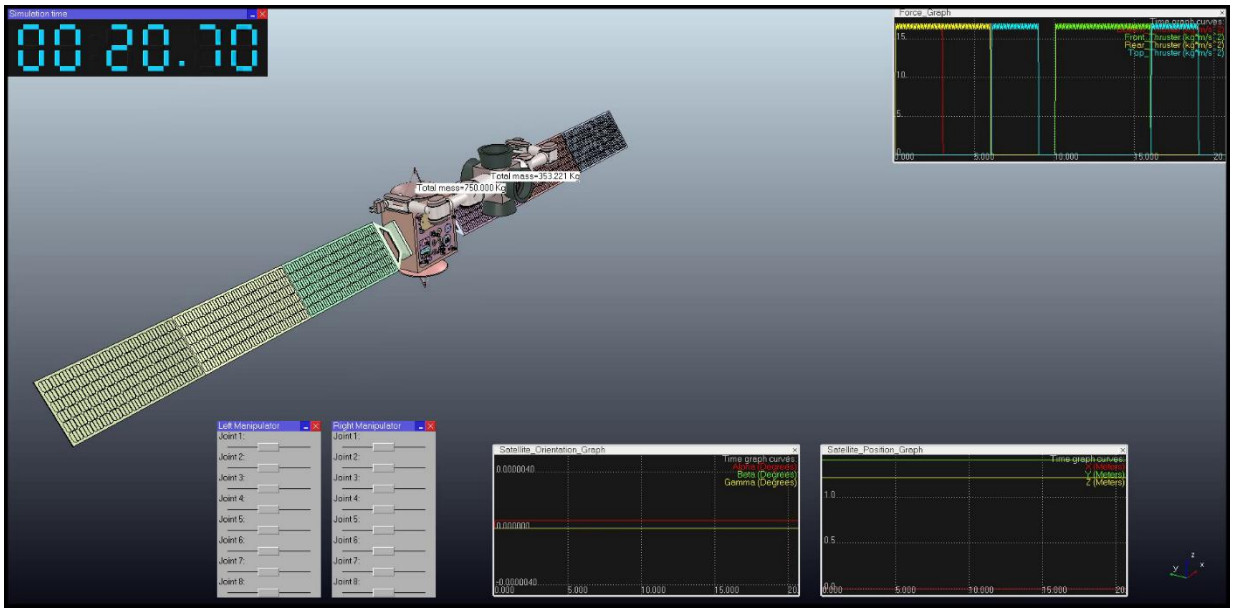

Simulation time
00: 17.25
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

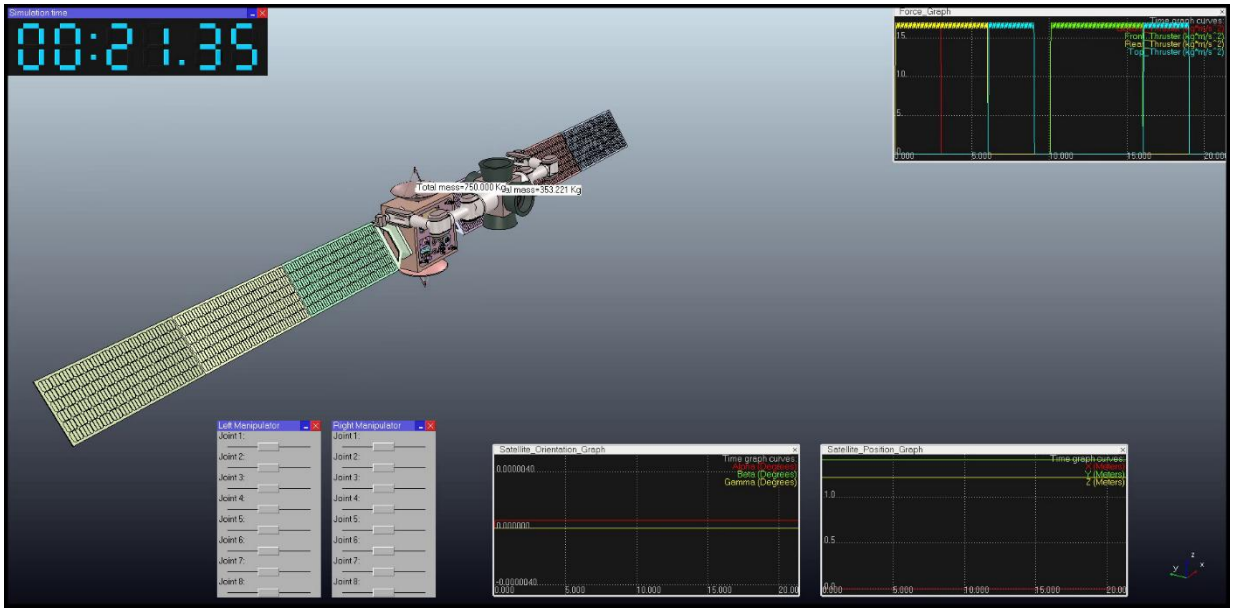

Simulation time
00: 17.95
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

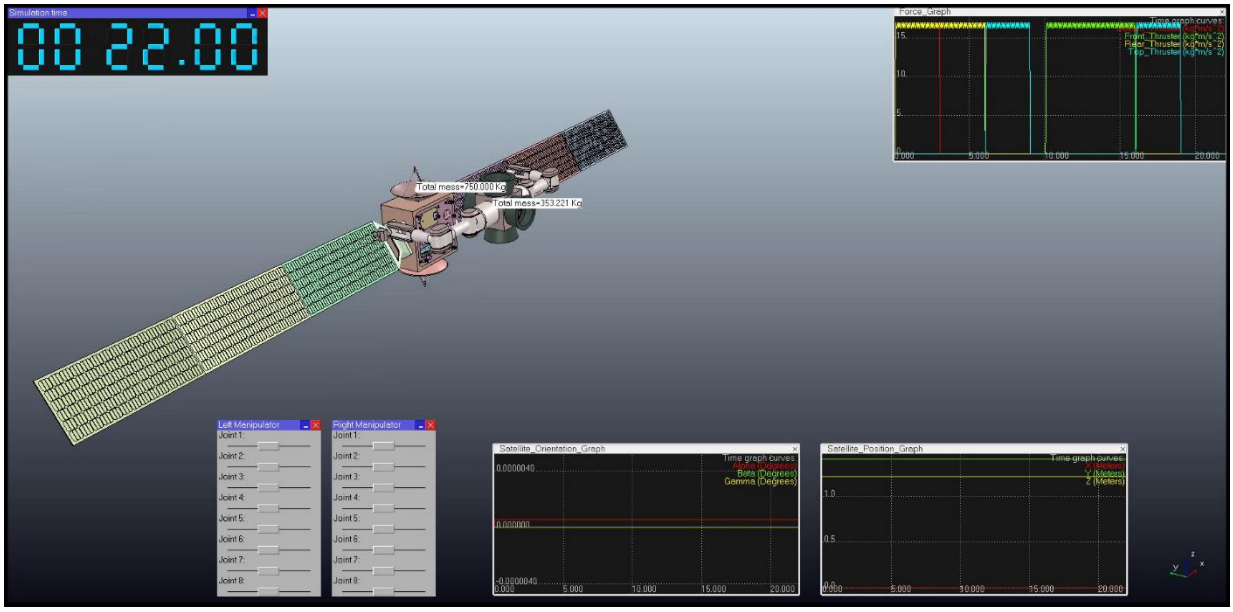

Simulation time
00 18.64
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

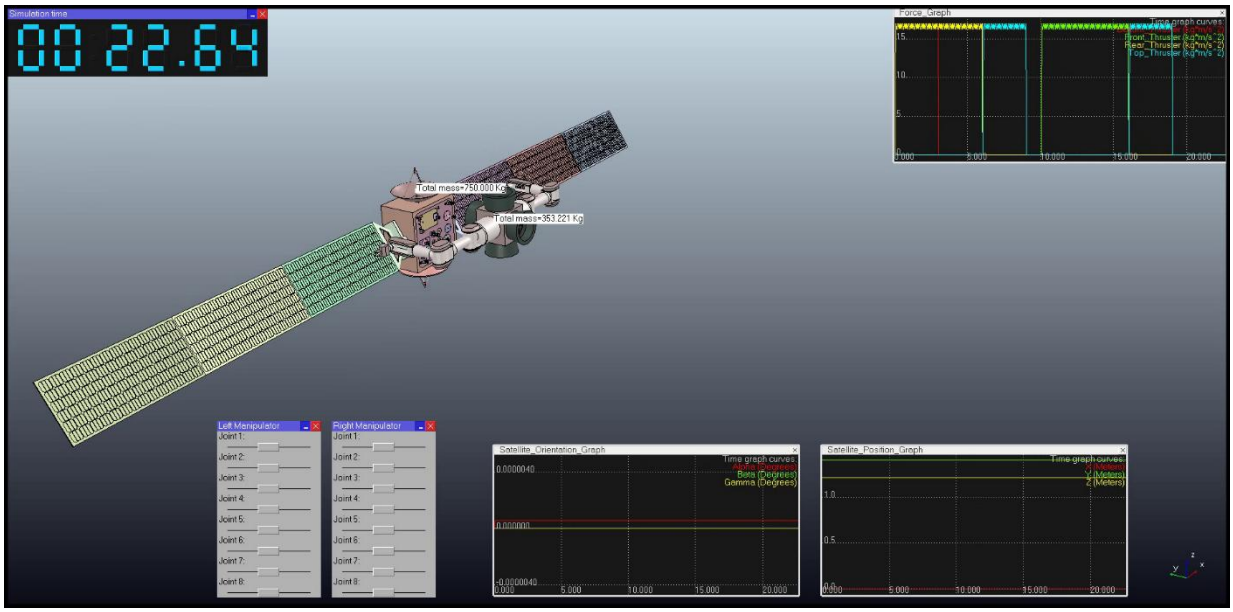

Simulation time
00: 19.35
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

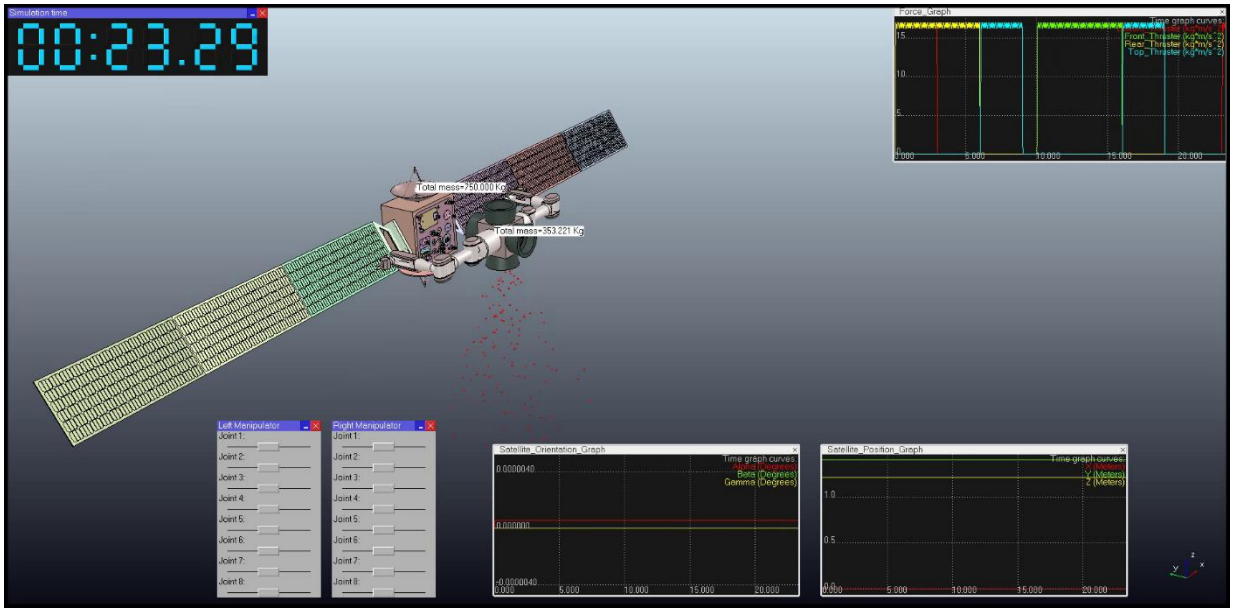

Simulation time
00 20.04
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

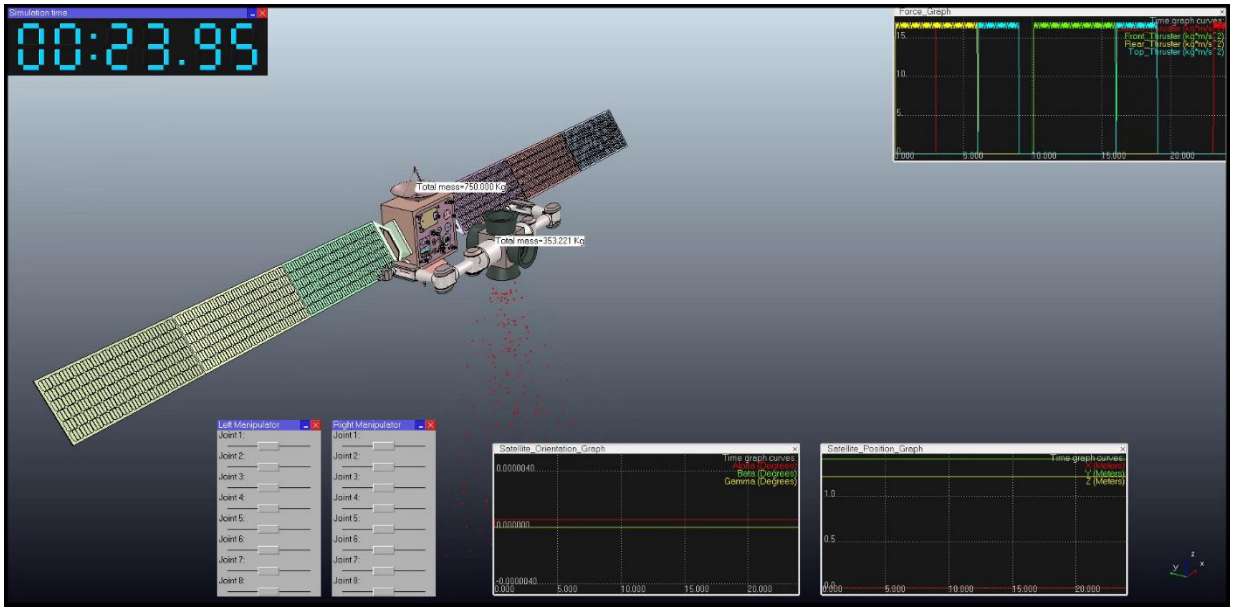

Simulation time
00 20.70
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

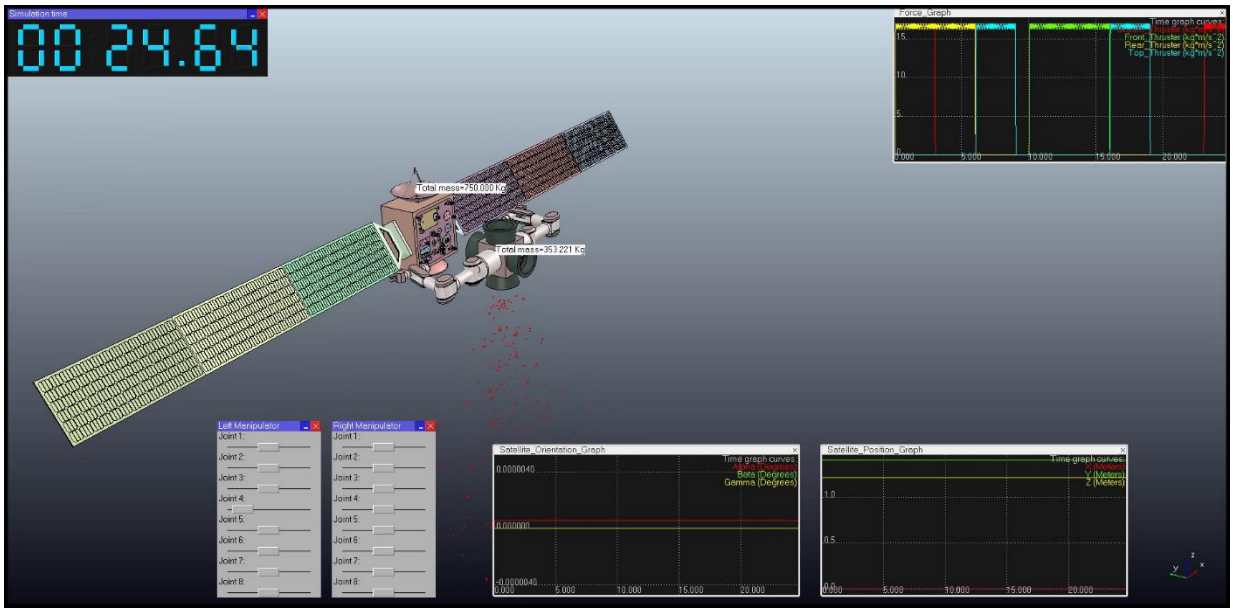

Simulation time
00:21.35
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

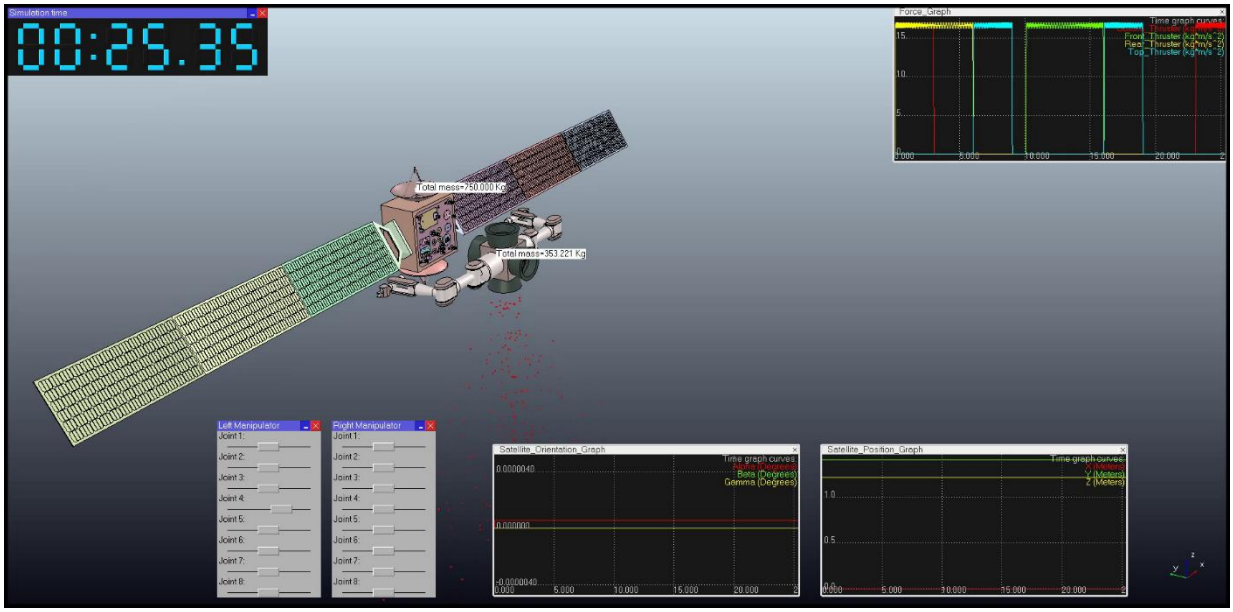

Simulation time
00 22.00
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

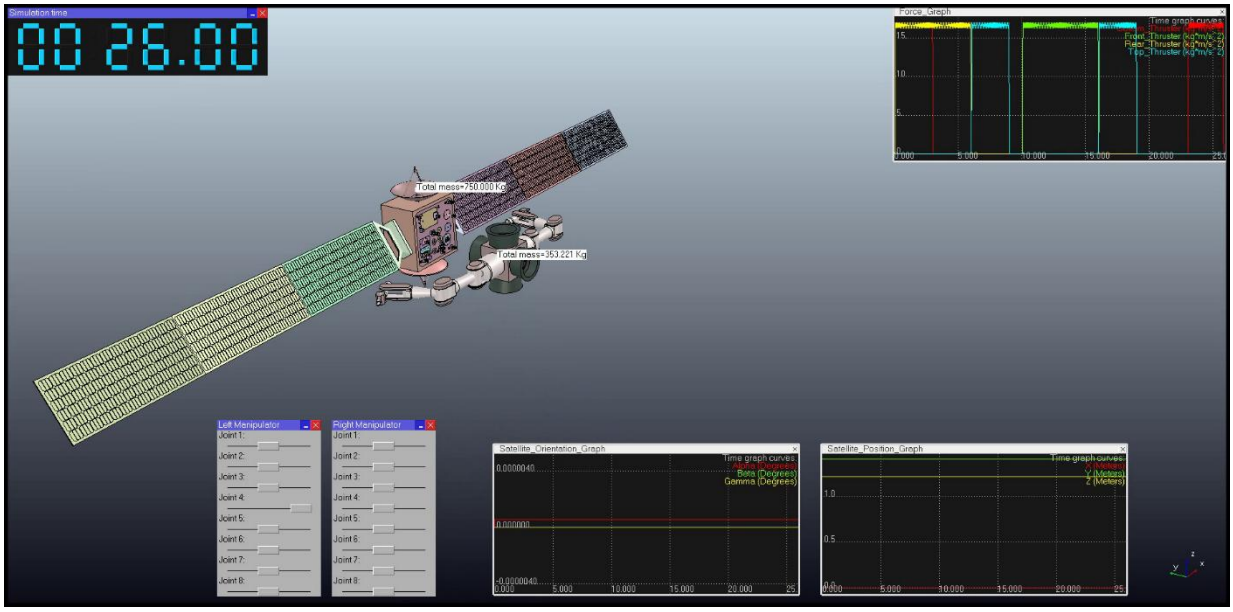

Simulation time
00 22.64
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

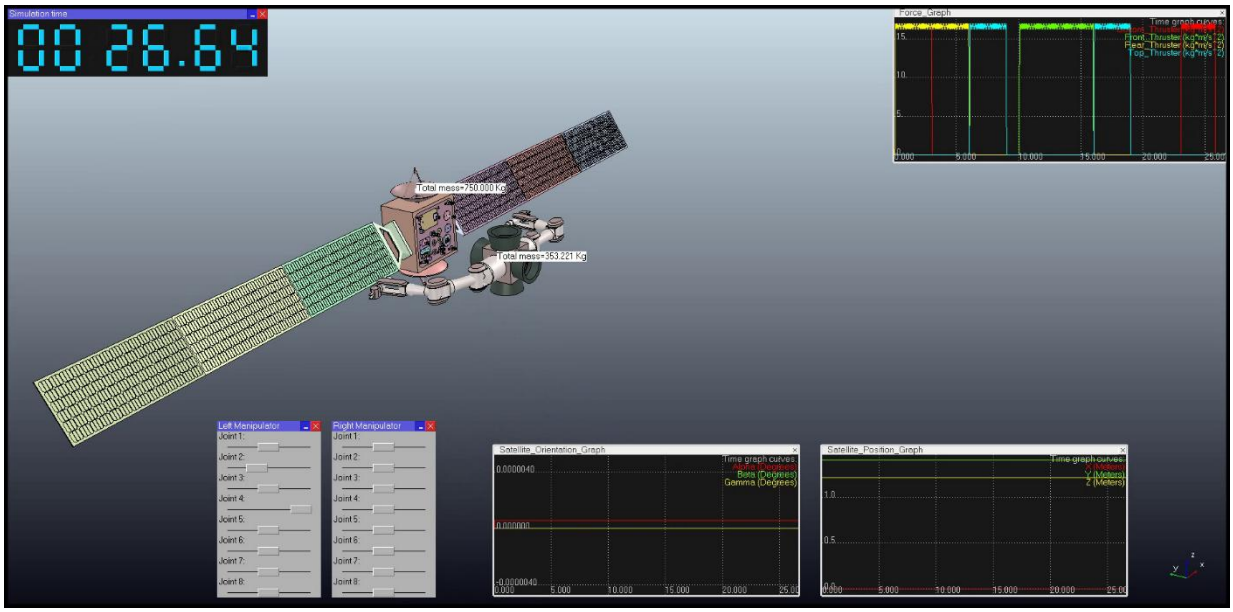

Simulation time
00:23.29
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

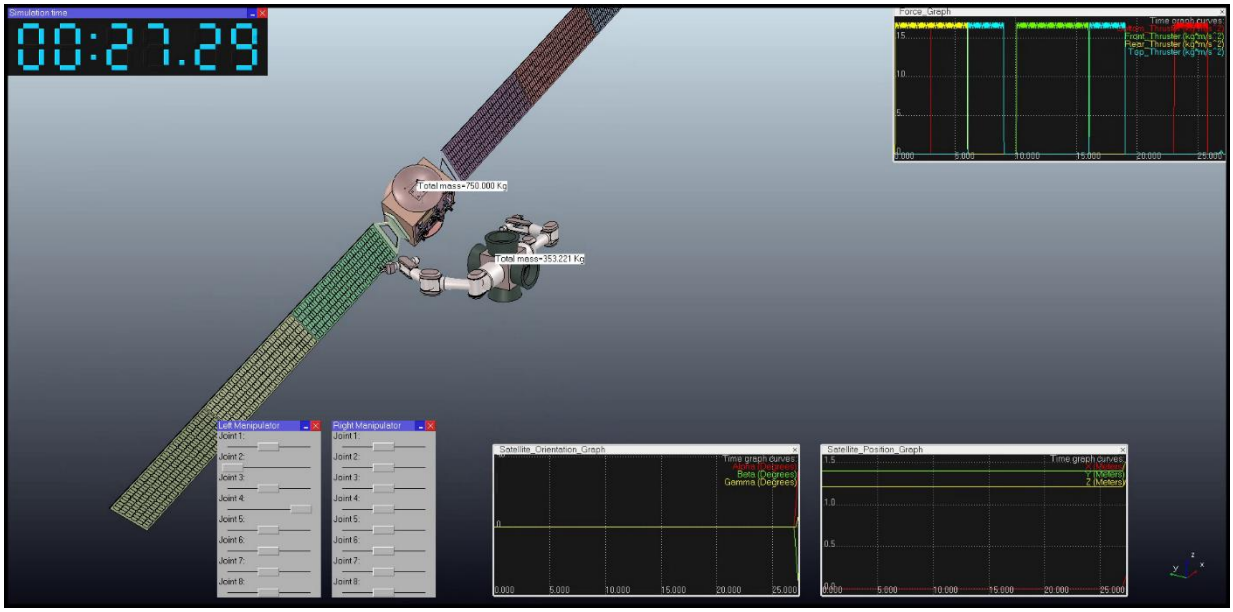

Simulation time
00:23.95
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

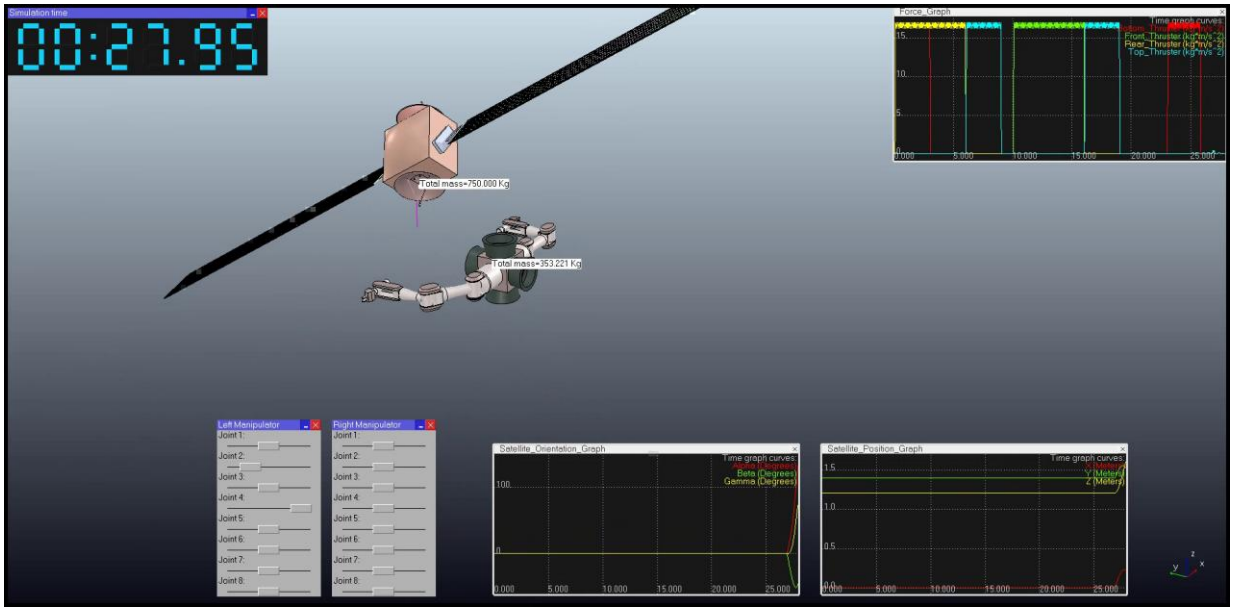

Simulation time
00 24.64
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
Y (Meters)
Z (Meters)

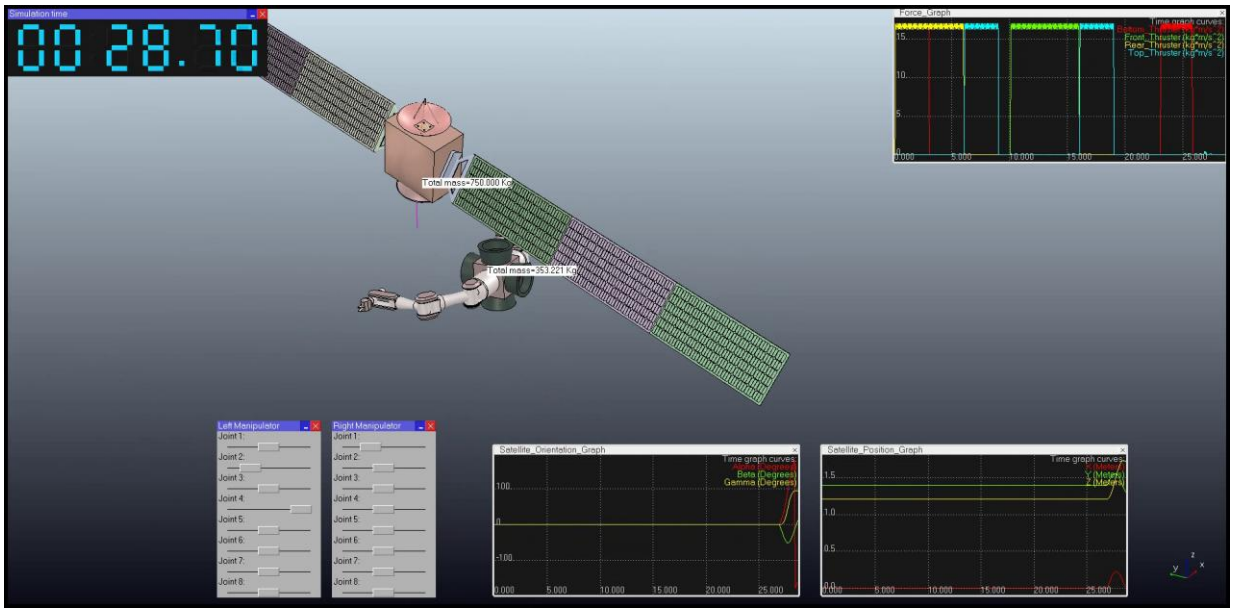

Simulation time
00:25.35
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
Y (Meters)
Z (Meters)

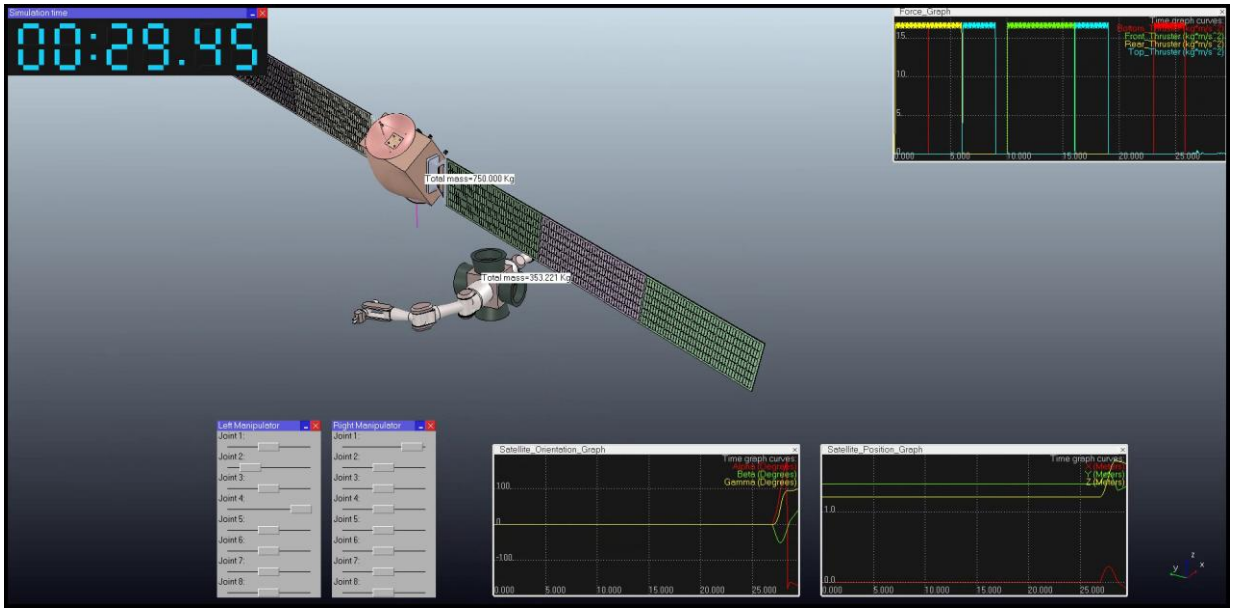

Simulation time
00 26.00
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
Y (Meters)
Z (Meters)

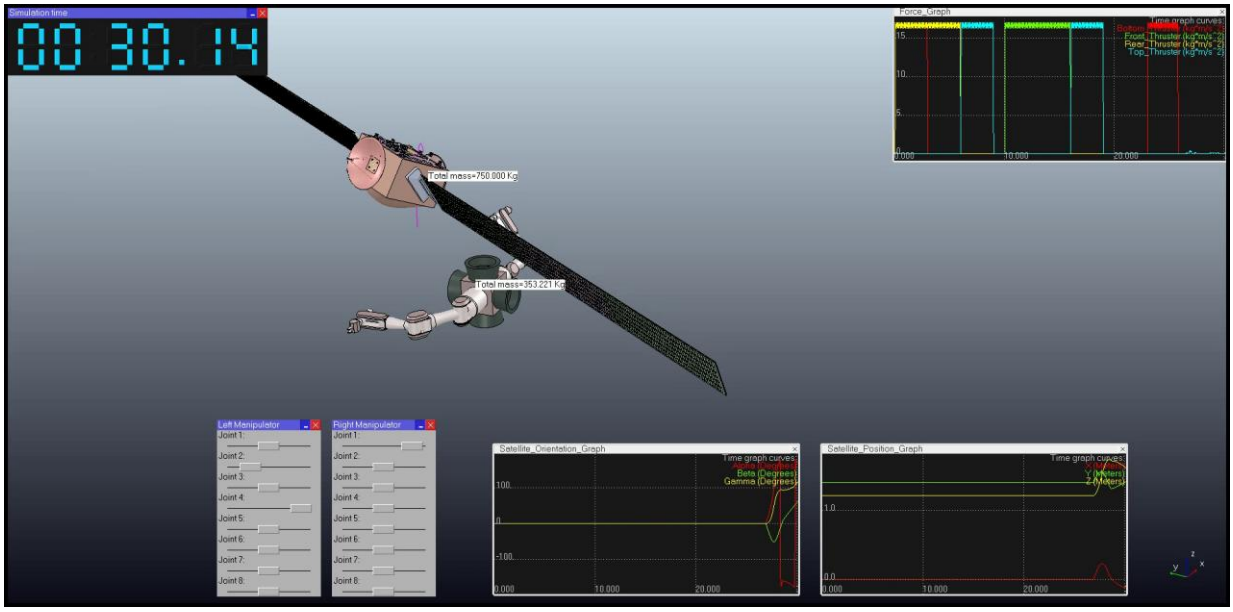

Simulation time
00 26.64
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

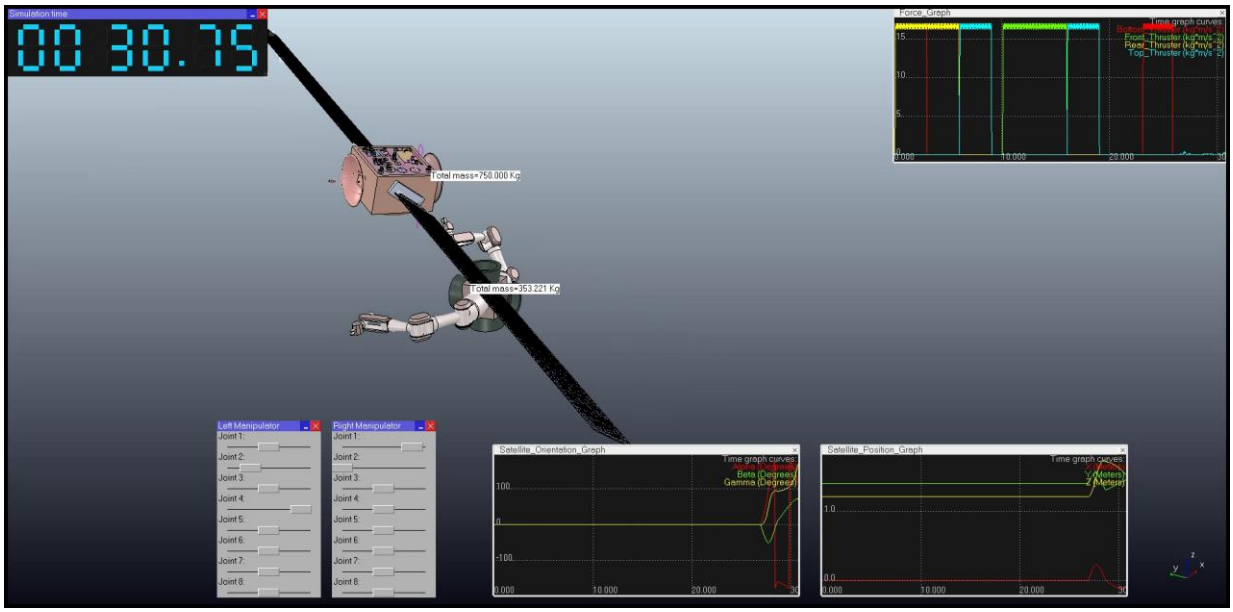

Simulation time
00:27.29
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

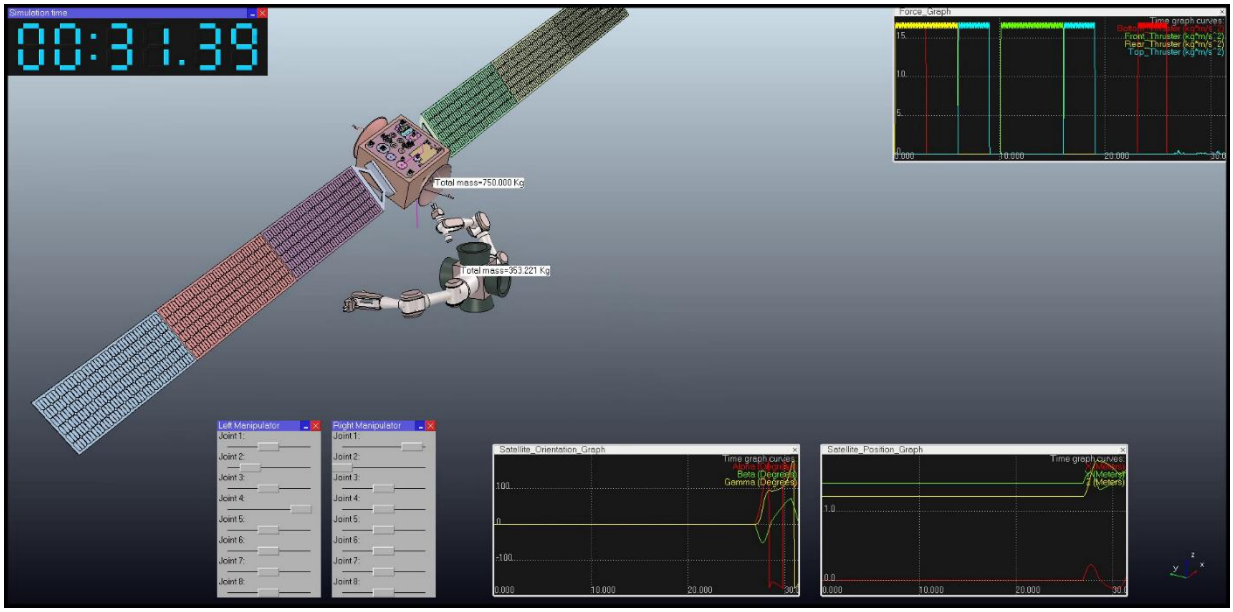

Simulation time
00:27.95
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

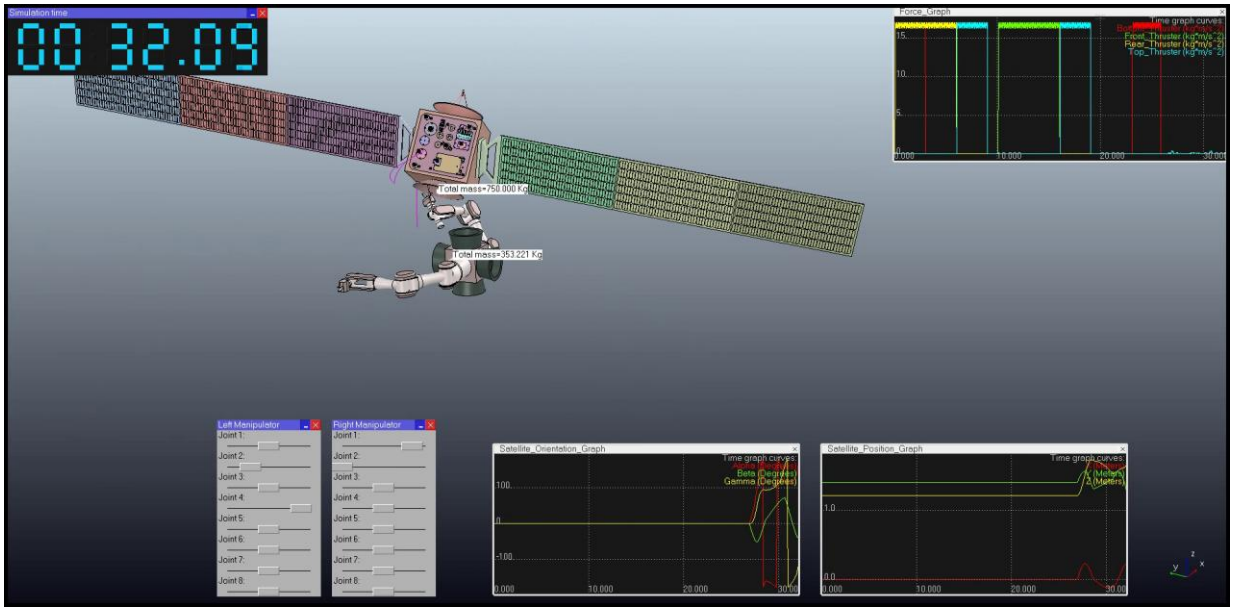

Simulation time
00 28.70
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

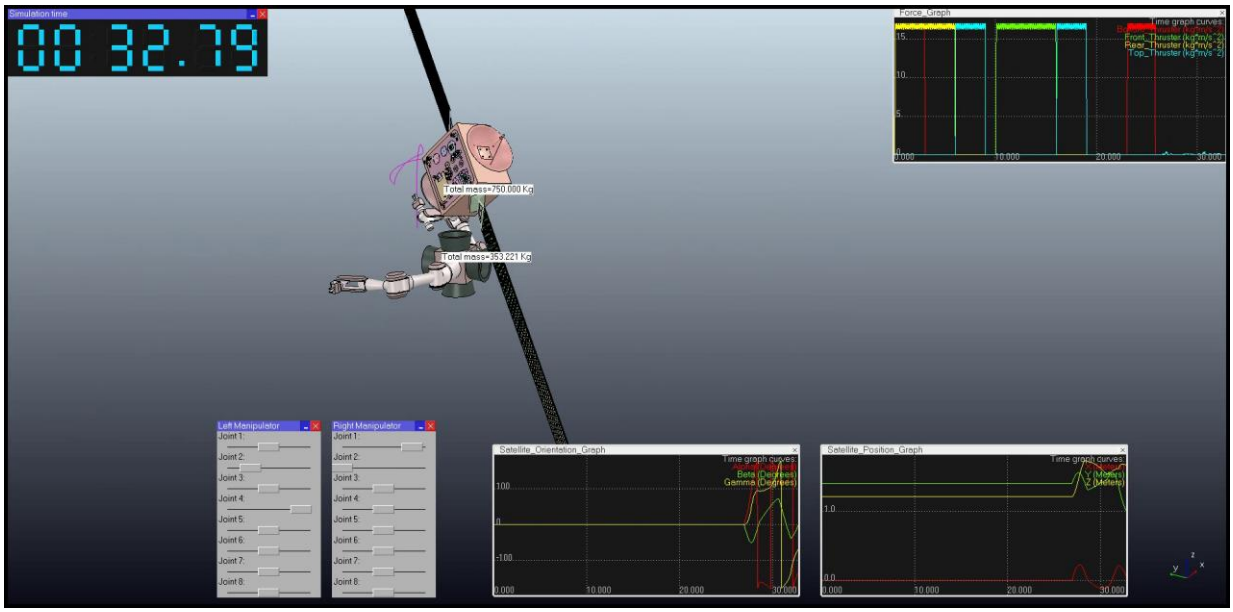

Simulation time
00:29.45
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

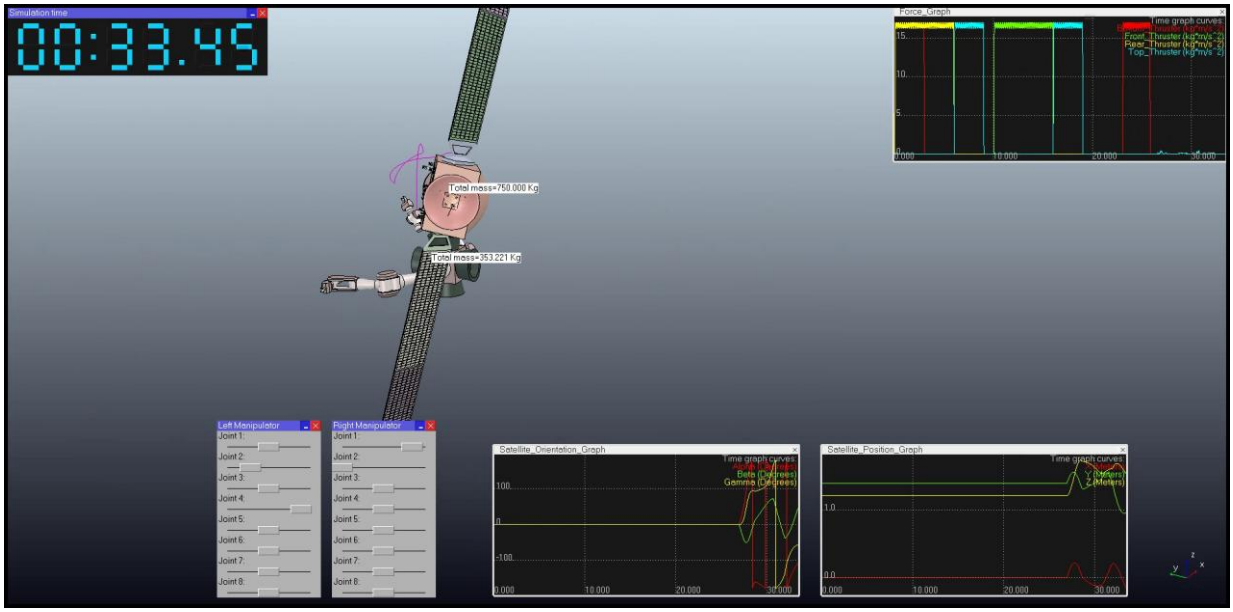

Simulation time
00 30.14
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

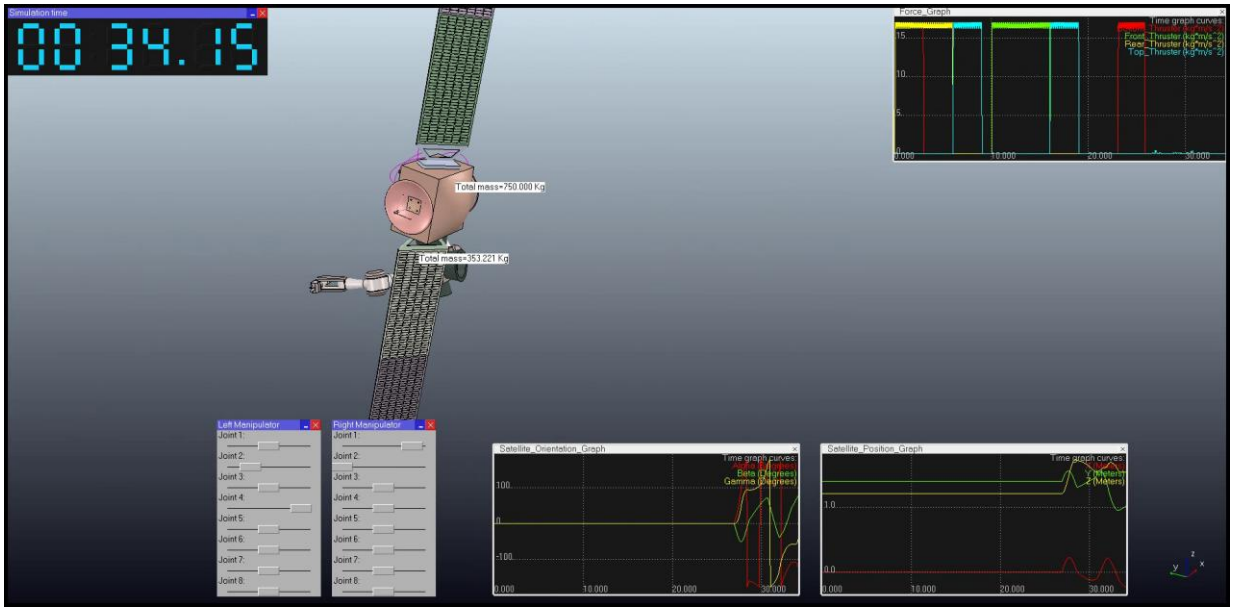

Simulation time
00 30.75
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

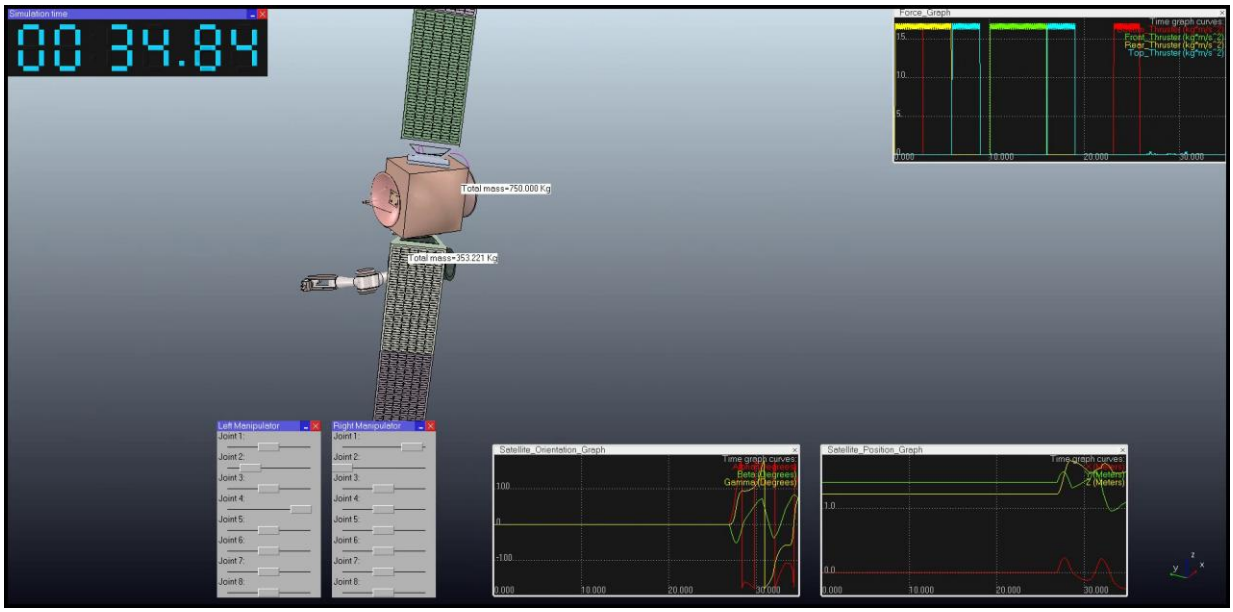

Simulation time
00:31.39
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

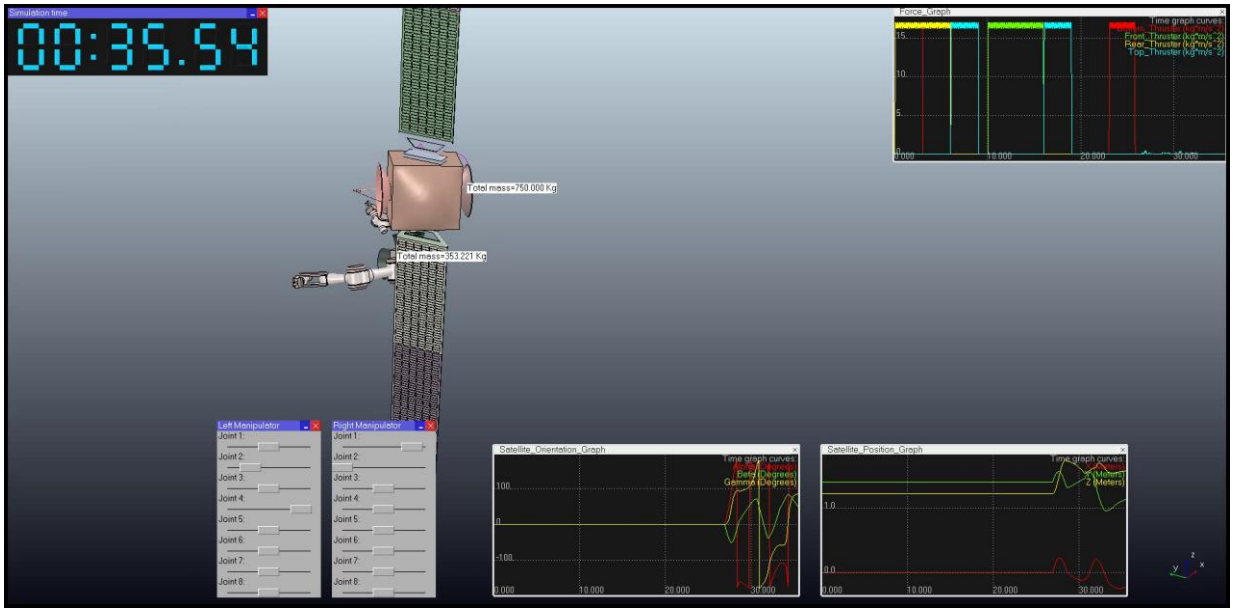

Simulation time
00 32.09
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

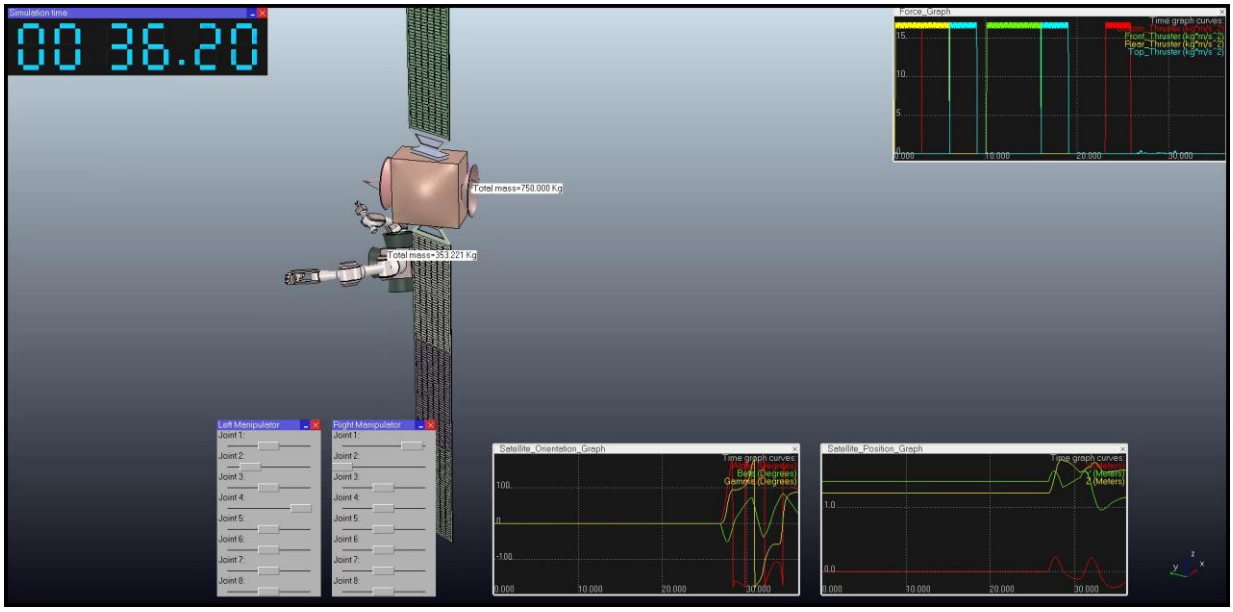

Simulation time
00 32.79
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

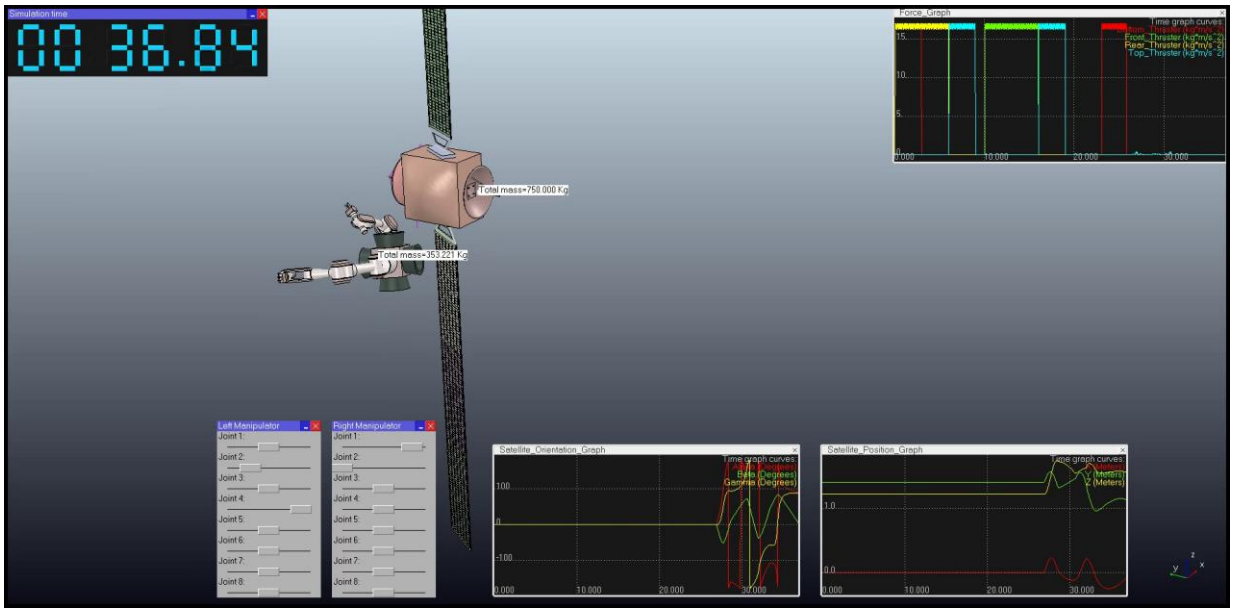

Simulation time
00:33.45
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

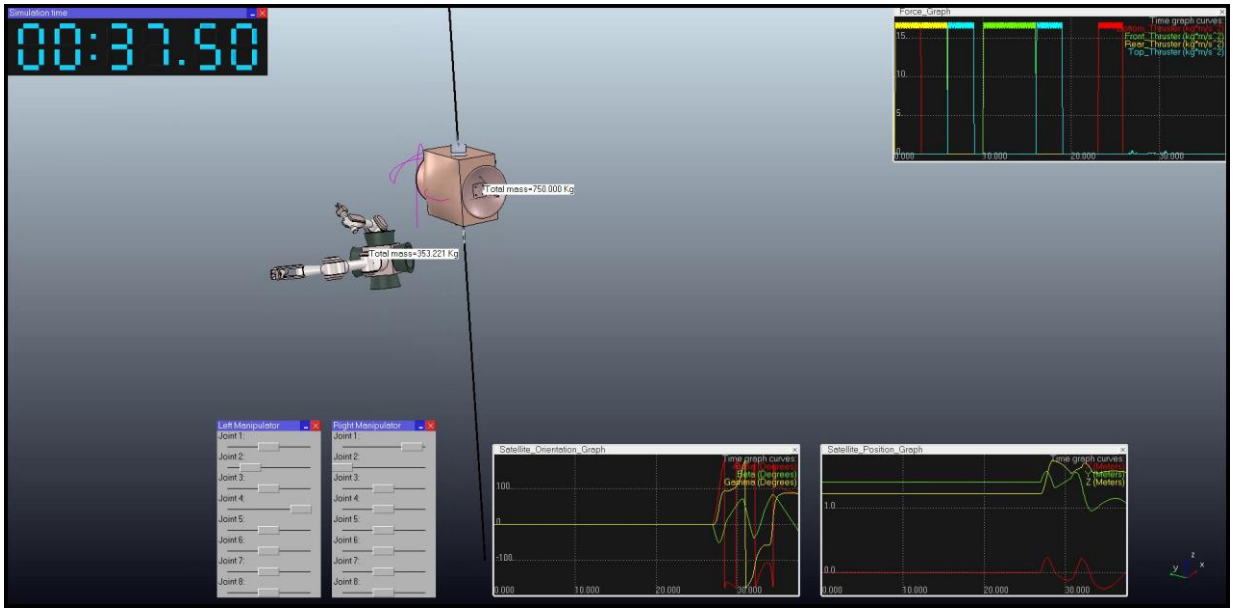

Simulation time
00 34.15
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

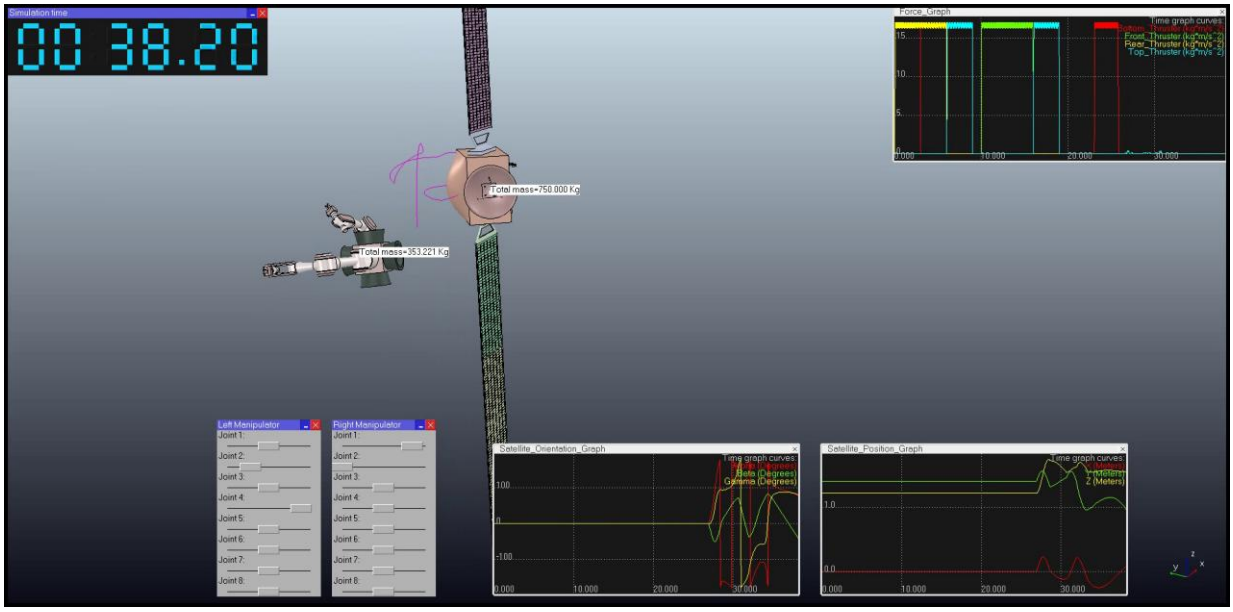

Simulation time
00 34.84
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

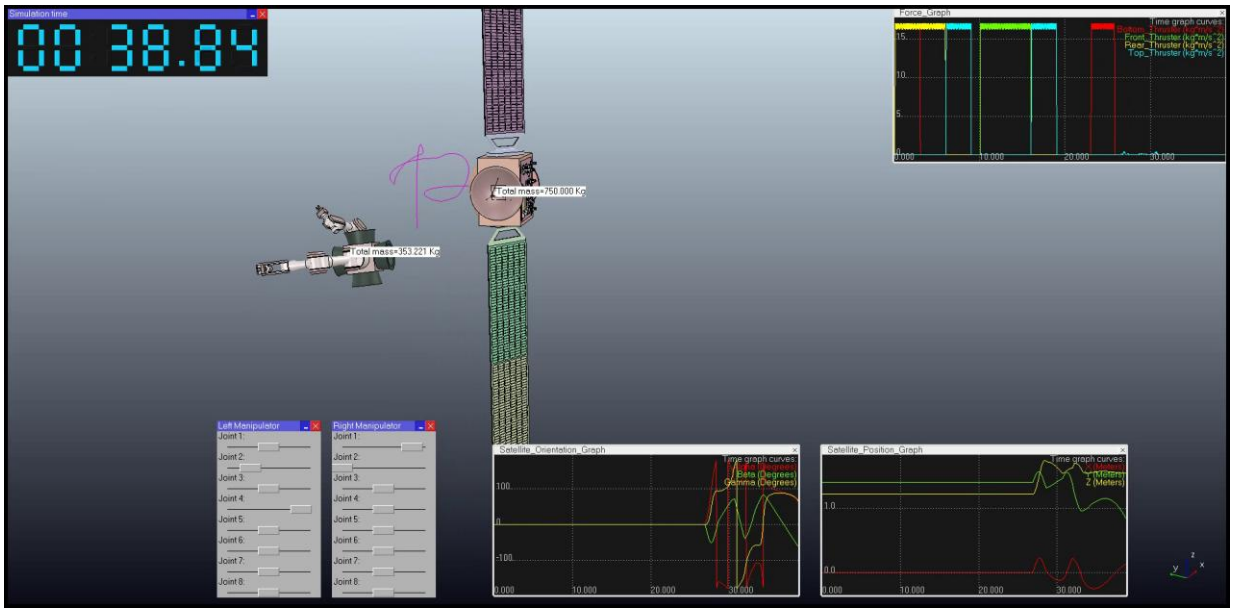

Simulation time
00:35.54
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

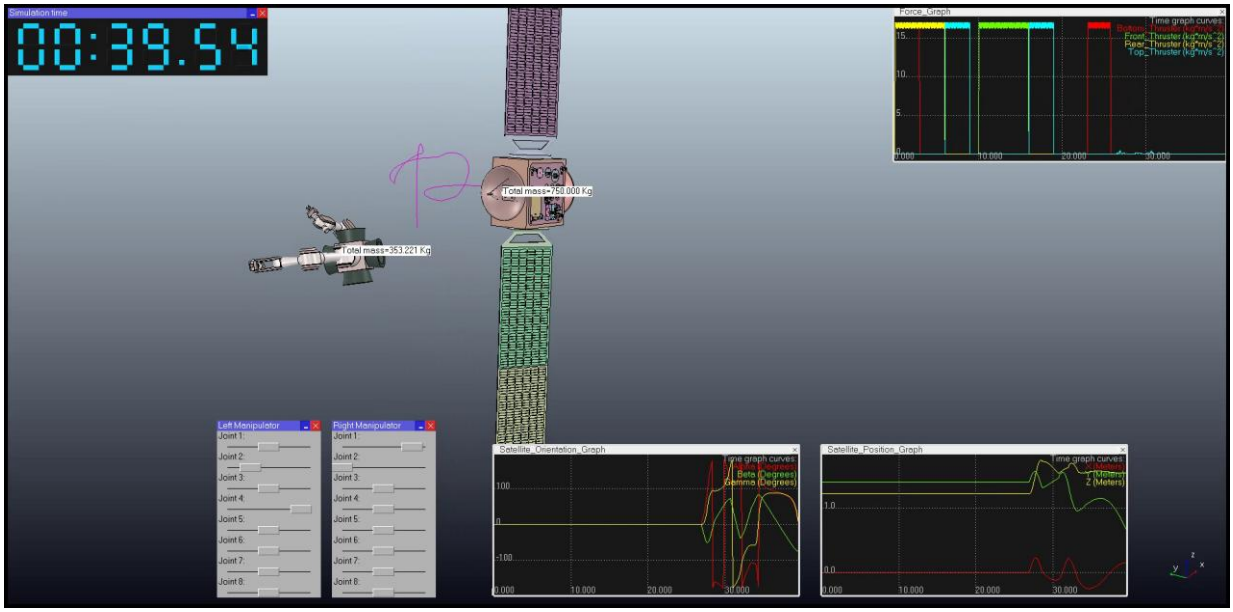

Simulation time
00 36.20
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

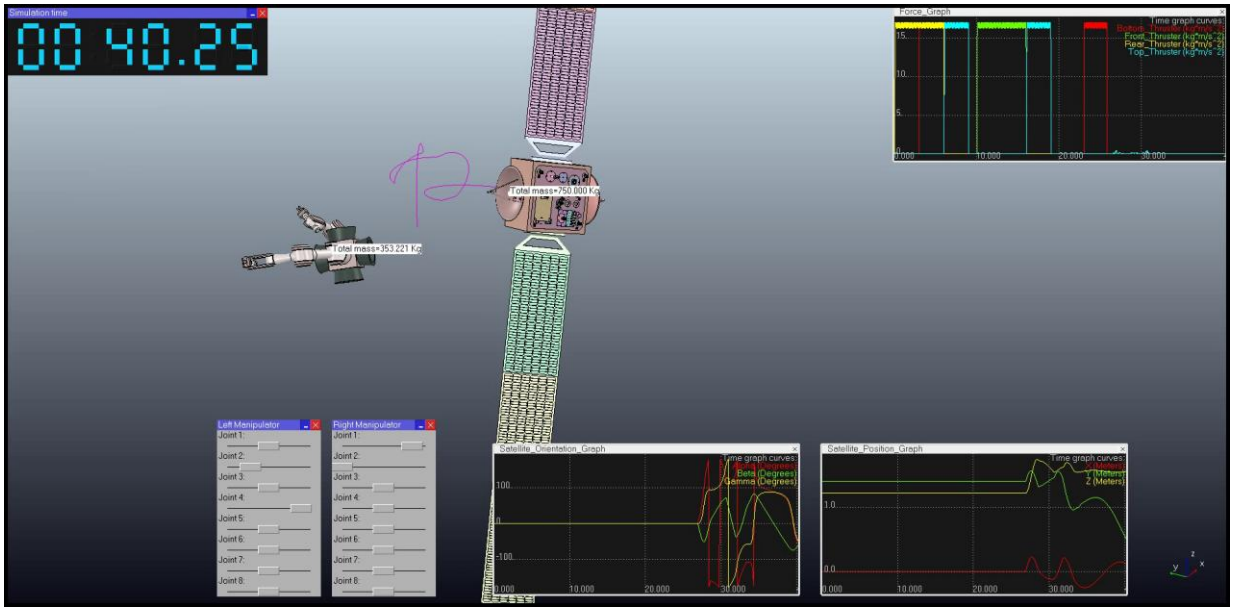

Simulation time
00 36.84
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

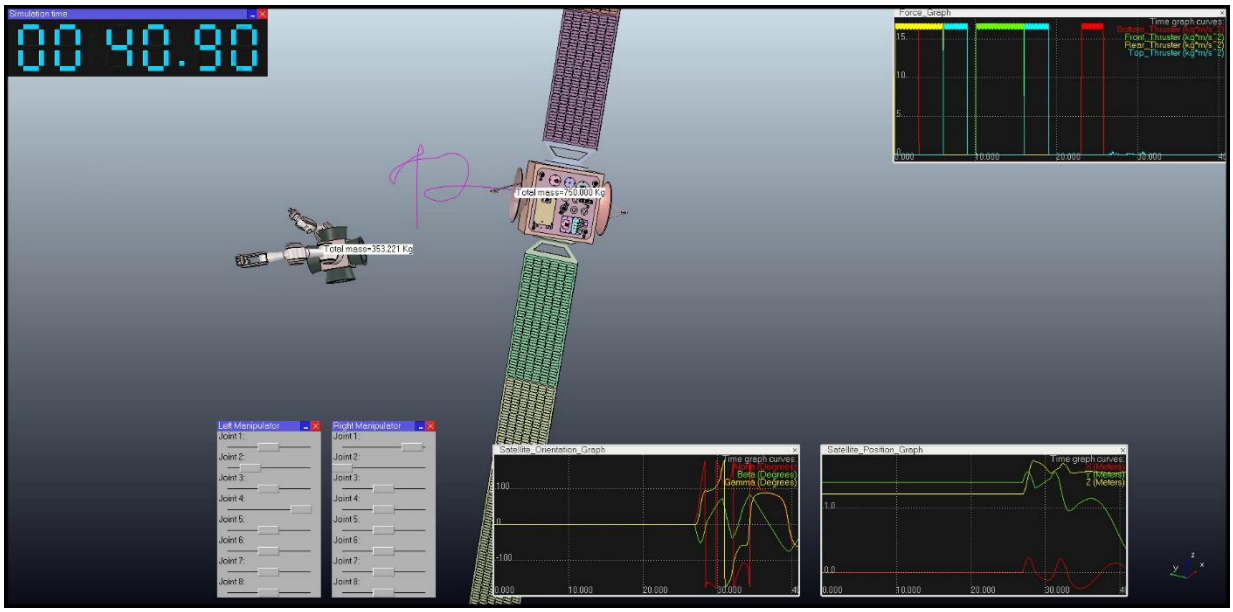

Simulation time
00:37.50
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

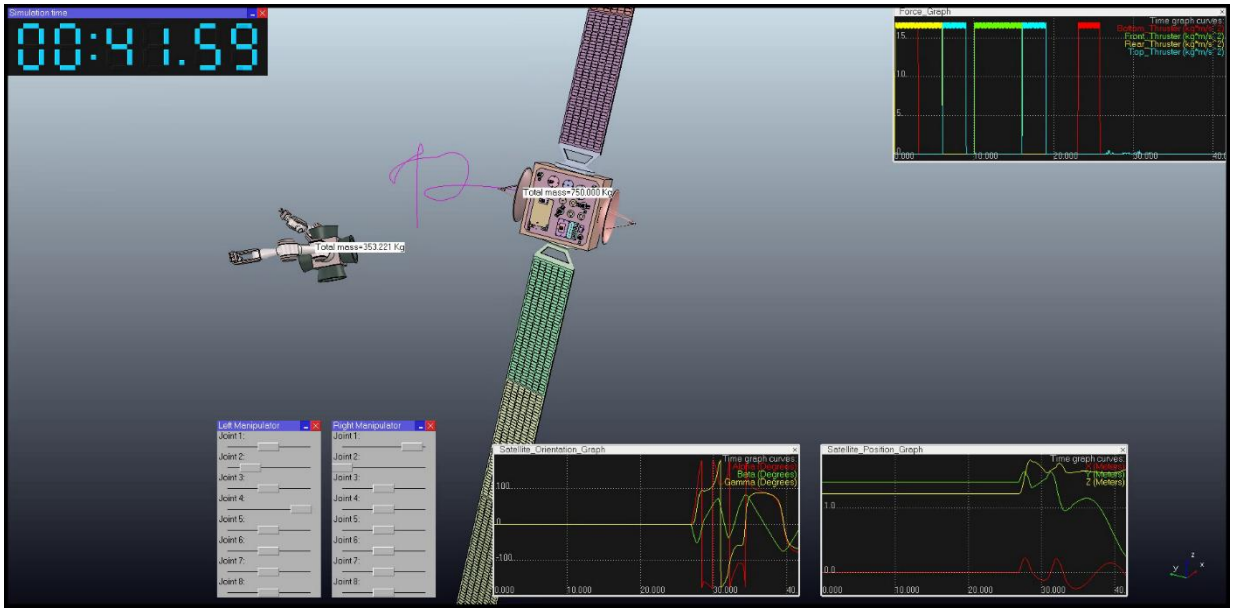

Simulation time
00 38.20
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

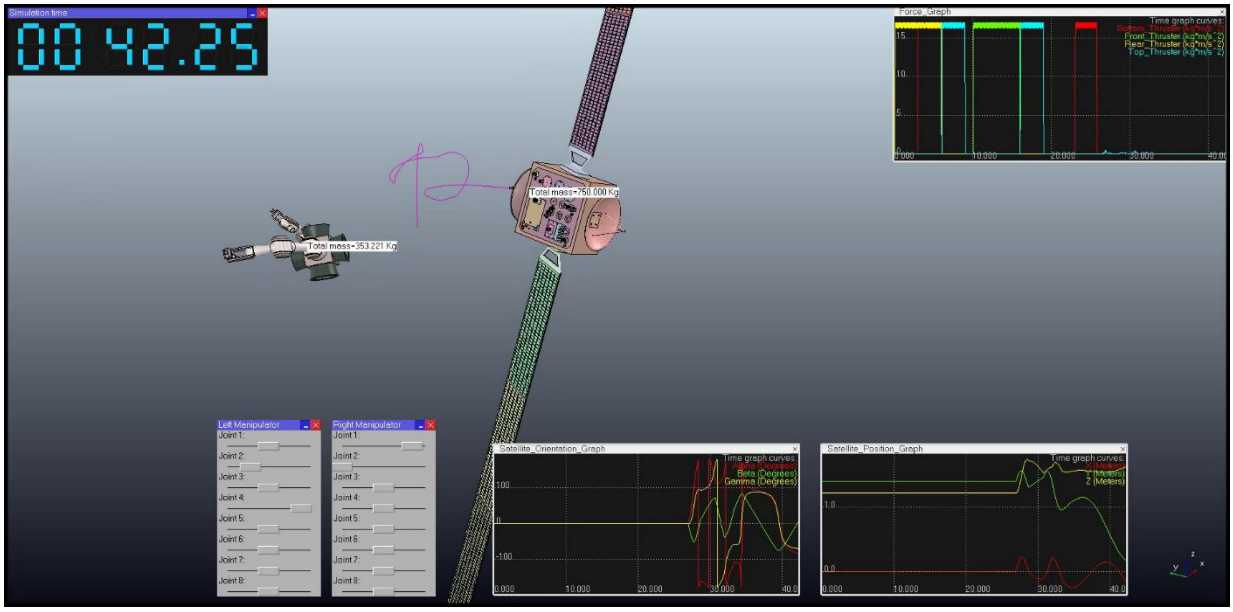

Simulation time
00 38.84
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

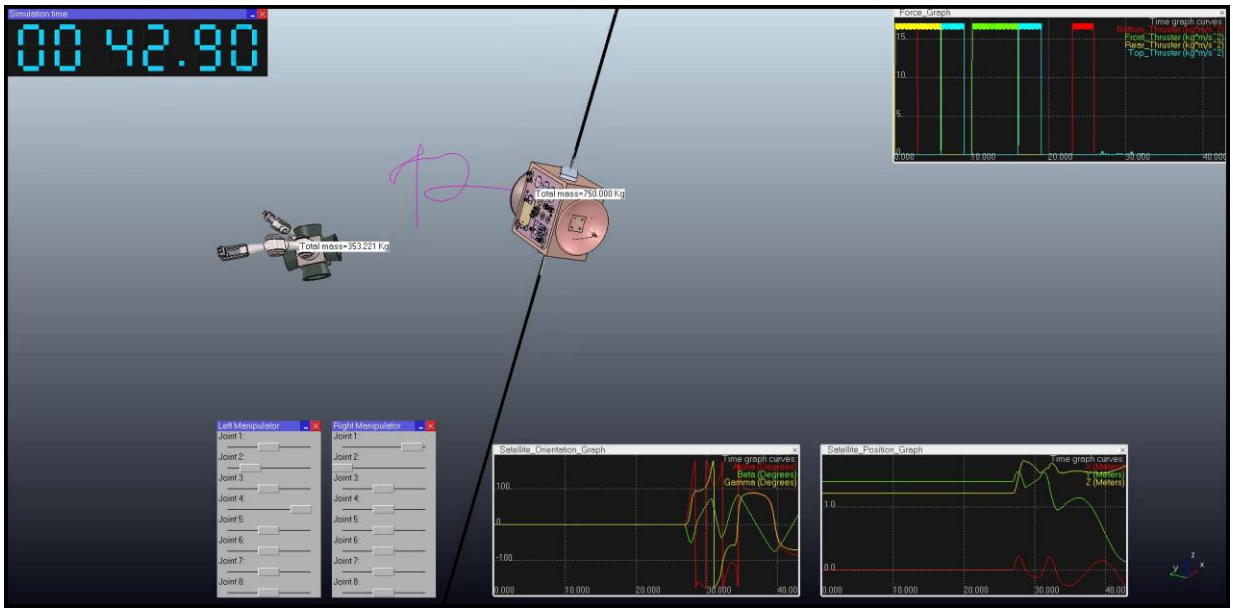

Simulation time
00:39.54
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

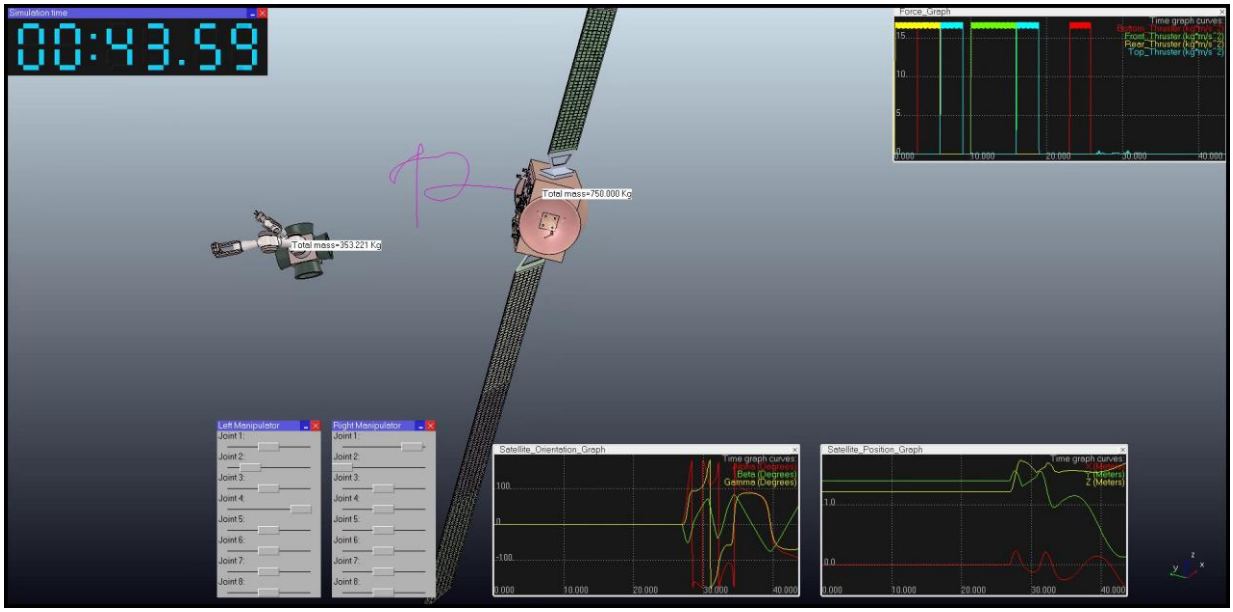

Simulation time
00 40.25
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

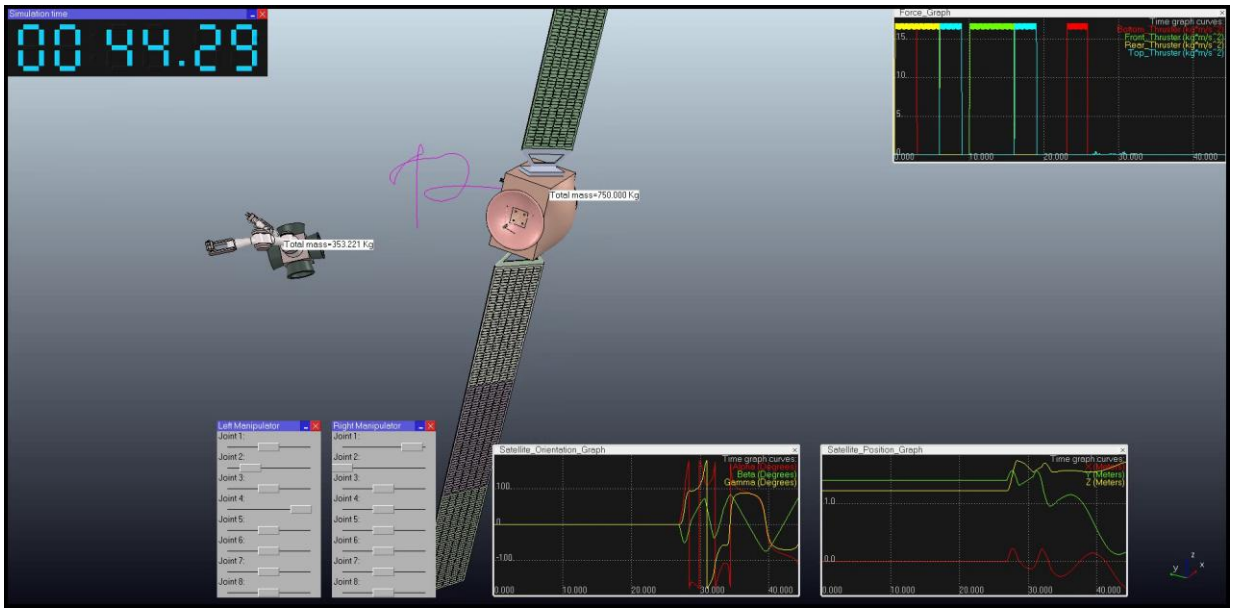

Simulation time
00 40.90
Force_Graph
Satellite_Orientation_Graph
Satellite_Position_Graph
Left Manipulator
Right Manipulator

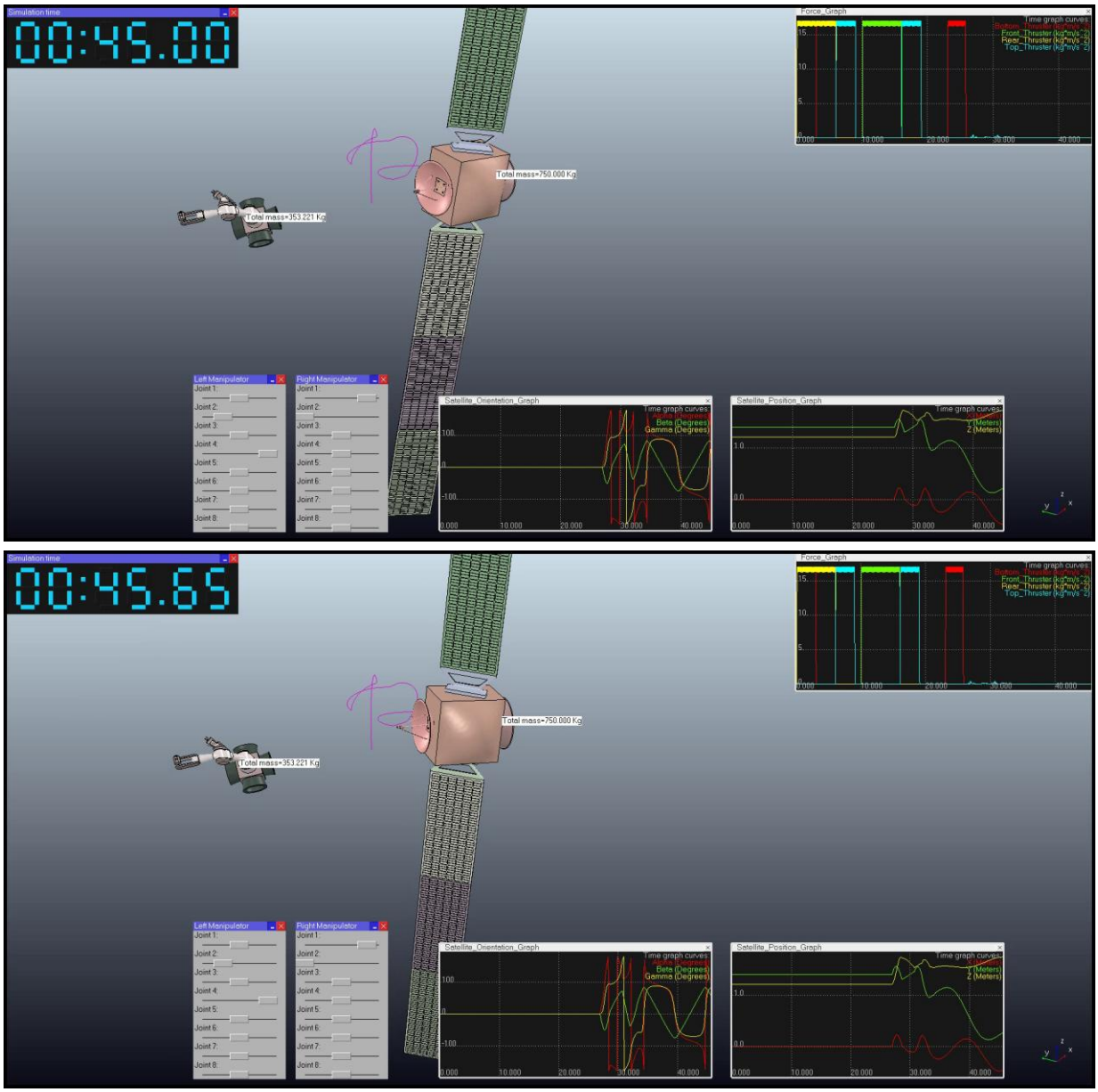

Simulation time
00:41.59
Force_Graph
Satellite_Orientation_Graph
Satellite_Position_Graph
Left Manipulator
Right Manipulator


Simulation time
00 42.25
Force_Graph
Satellite_Orientation_Graph
Satellite_Position_Graph
Left Manipulator
Right Manipulator


Simulation time
00 42.90
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Alpha (Degrees)
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Y (Meters)
Z (Meters)


Simulation time
00:43.59
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Alpha (Degrees)
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Y (Meters)
Z (Meters)


Simulation time
00 44.29
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Alpha (Degrees)
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Y (Meters)
Z (Meters)

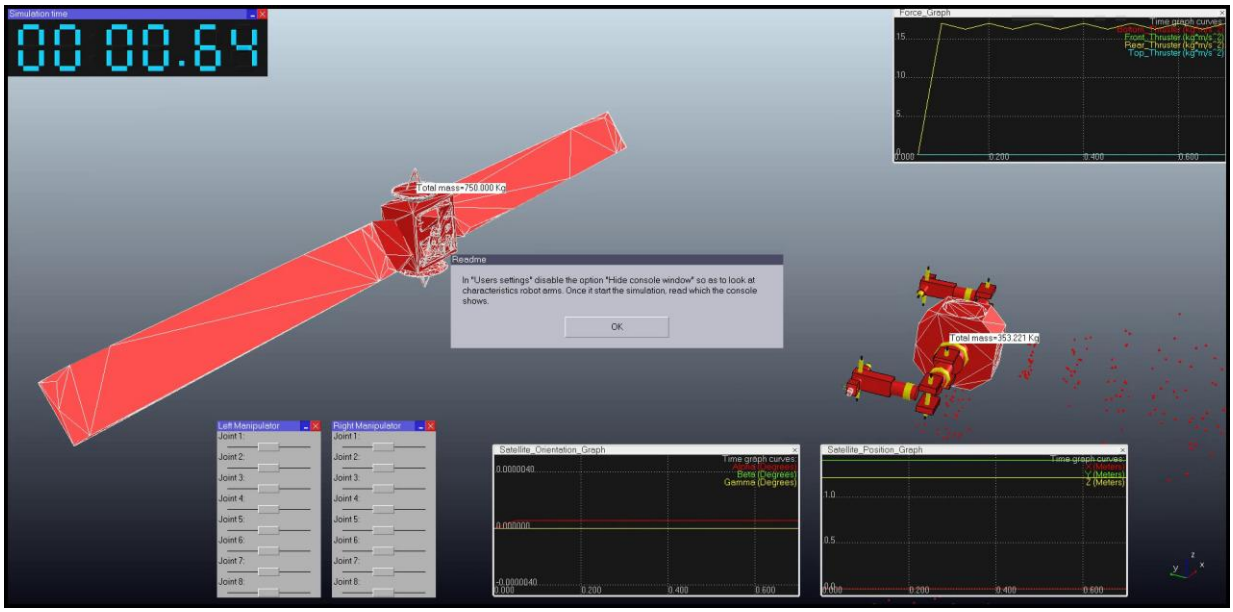


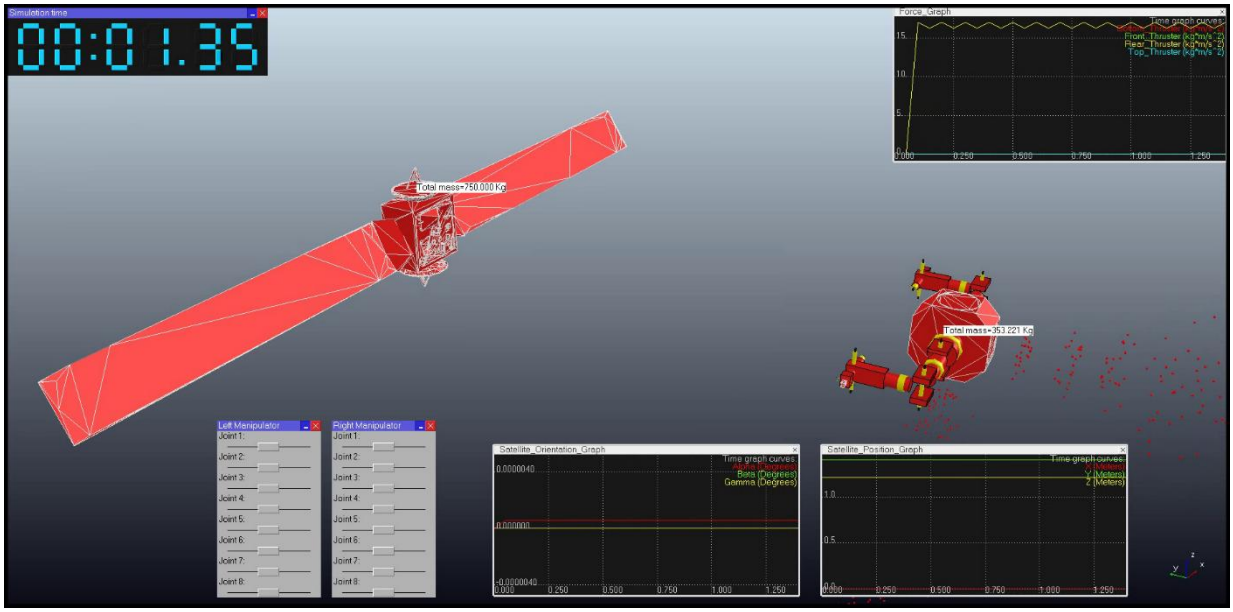
4.2.2. Reproducción dinámica.

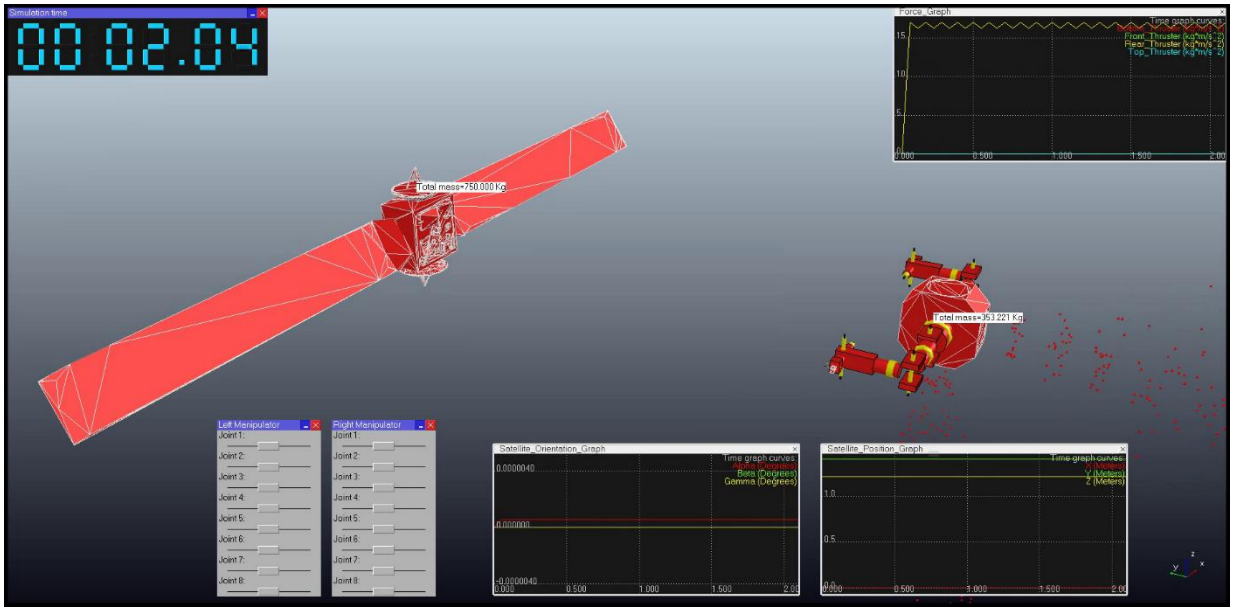
En esta ocasión, se mostrarán de forma concatenada las secuencias de simulación resultantes de la reproducción dinámica. No se harán referencia alguna en el índice de referencias pues no tiene relevancia enumerar y nombrar cada imagen en vista de la función que cumple dicha concatenación.

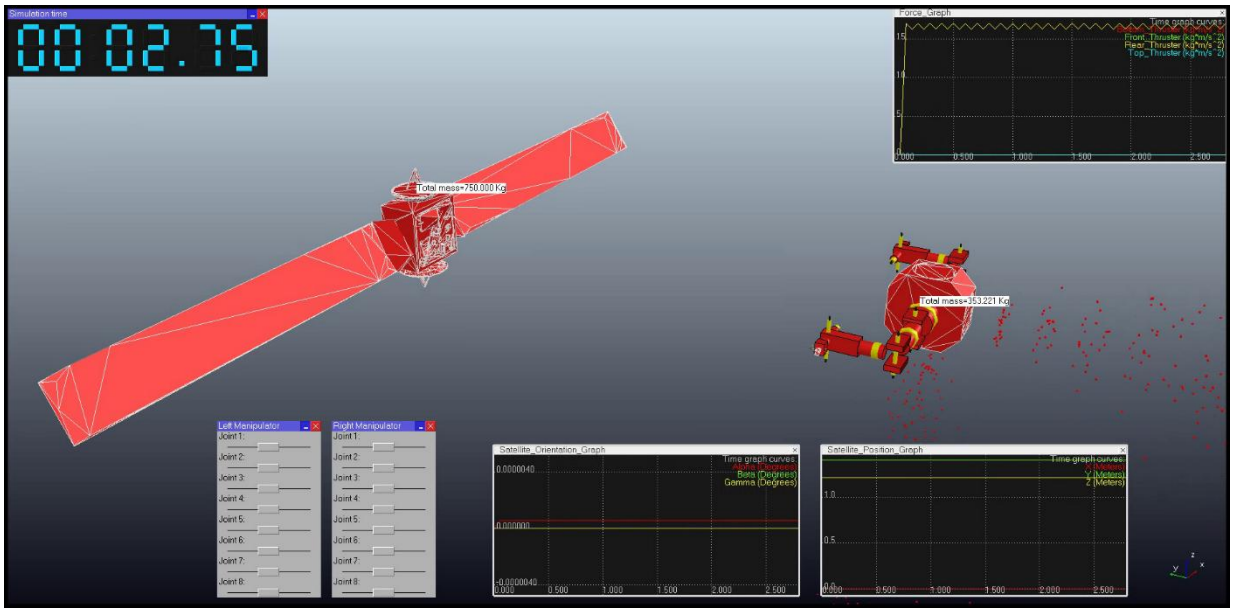
La simulación se inicia con la ventana emergente indicando habilitar la consola para ver la información relacionada con las articulaciones, que tras pulsar “ok” sigue ejecutándose.

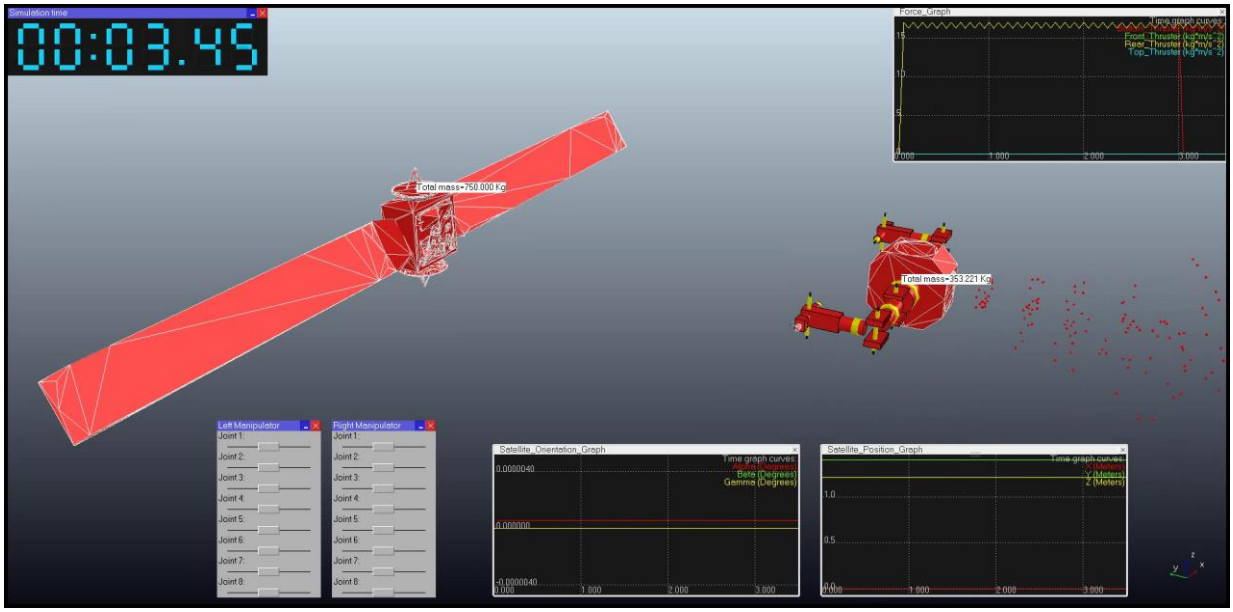
Se recomienda consultar la sección de “Inspección rigurosa del contenido dinámico de la escena” del apartado 2.6.2 a fin de entender la leyenda de colores correspondiente al modo de visualización del contenido dinámico subyacente.

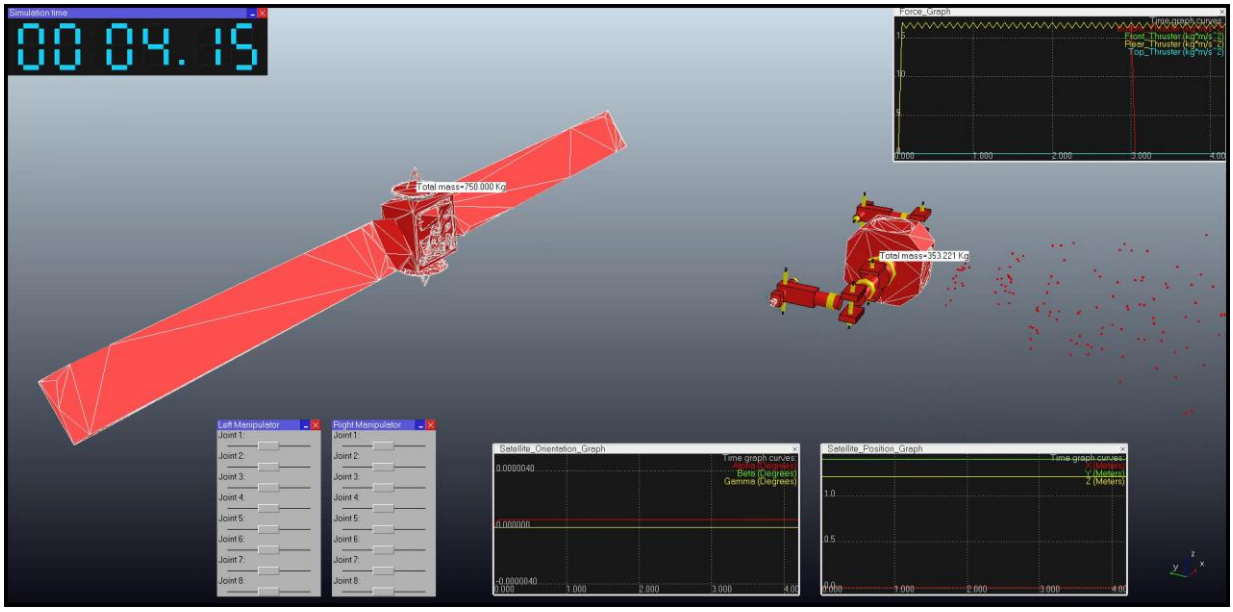

Simulation time
00 00.64
Force_Graph
Total mass=750.000 Kg
Readme
In "Users settings" disable the option "Hide console window" so as to look at characteristics robot arms. Once it start the simulation, read which the console shows.
OK
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

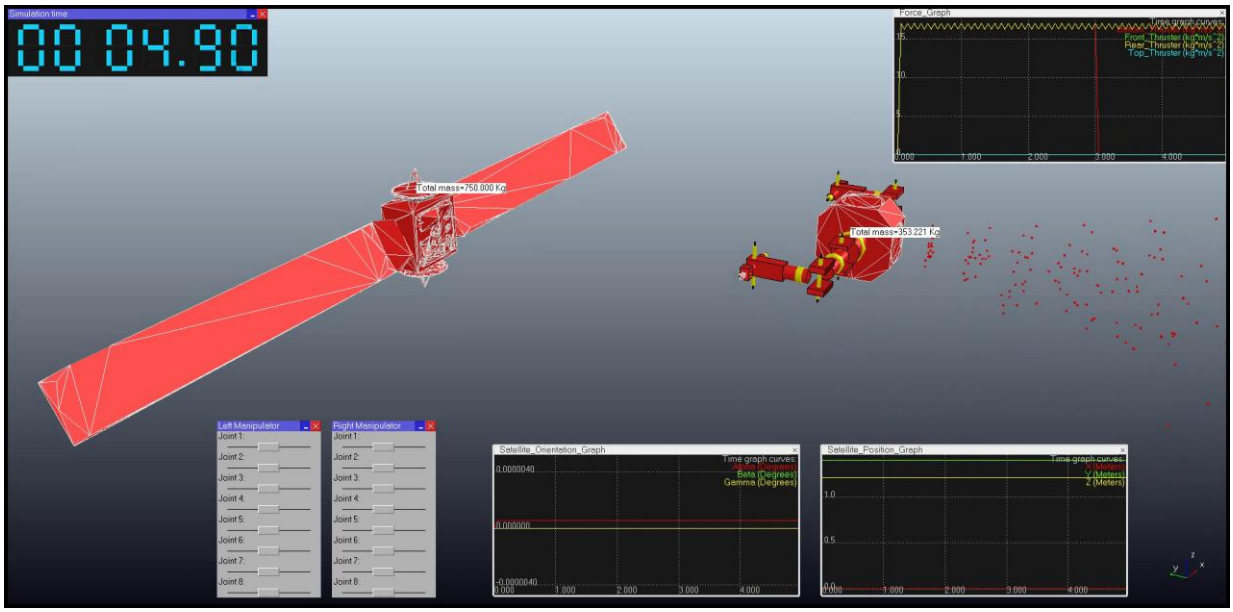

Simulation time
00:01.35
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

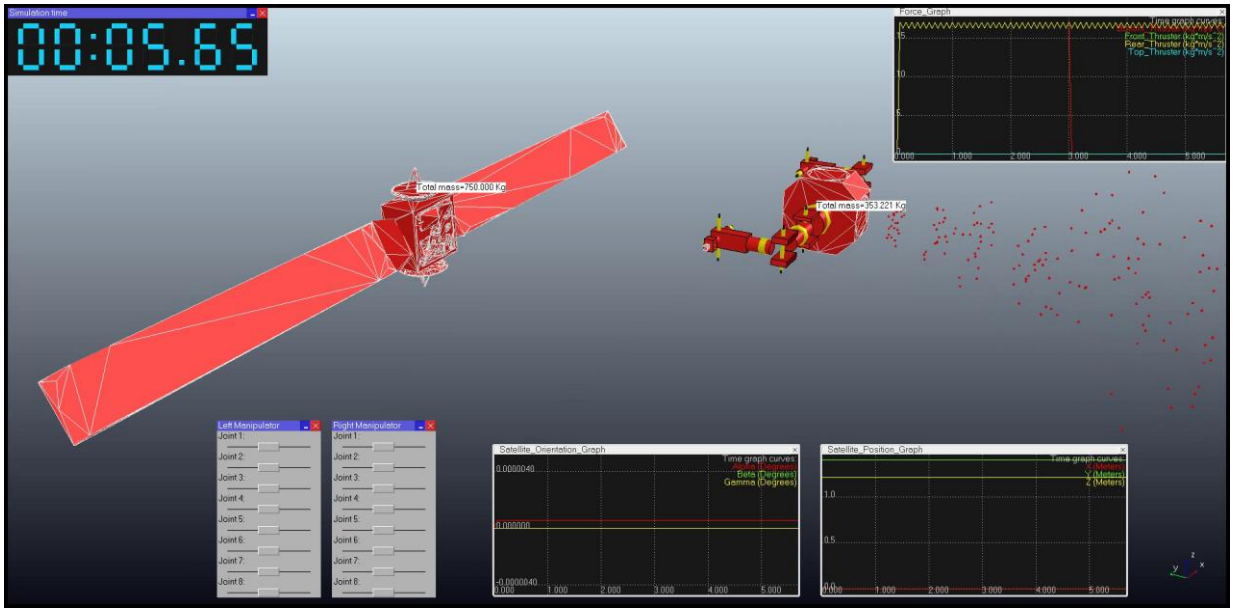

Simulation time
00 02.04
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

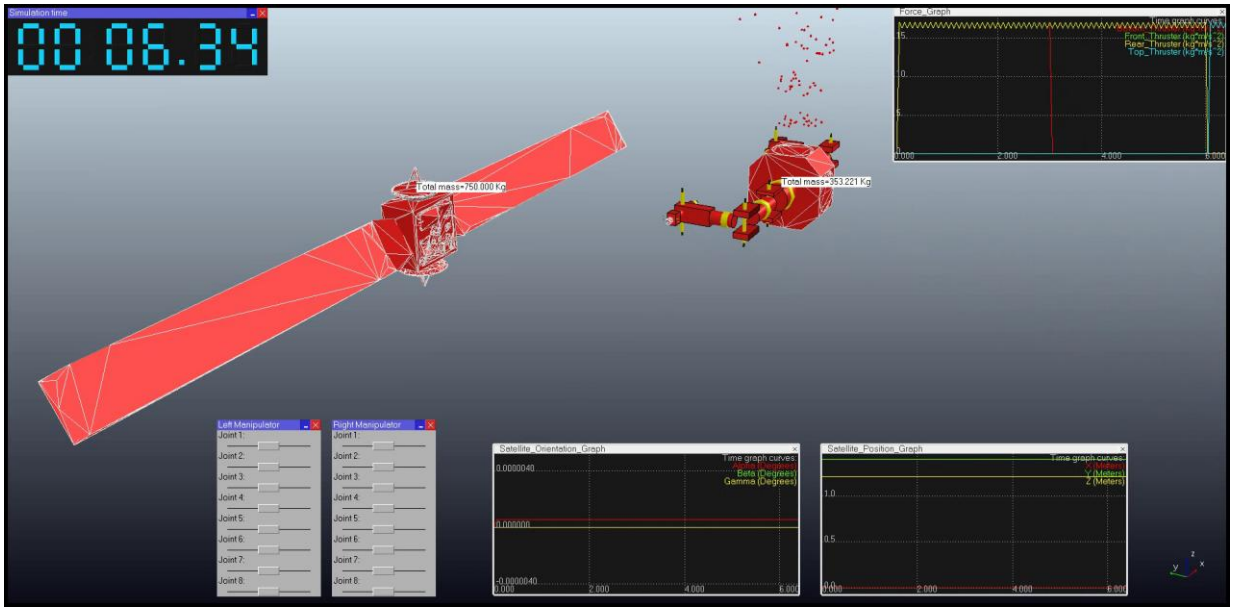

Simulation time
00 02.75
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

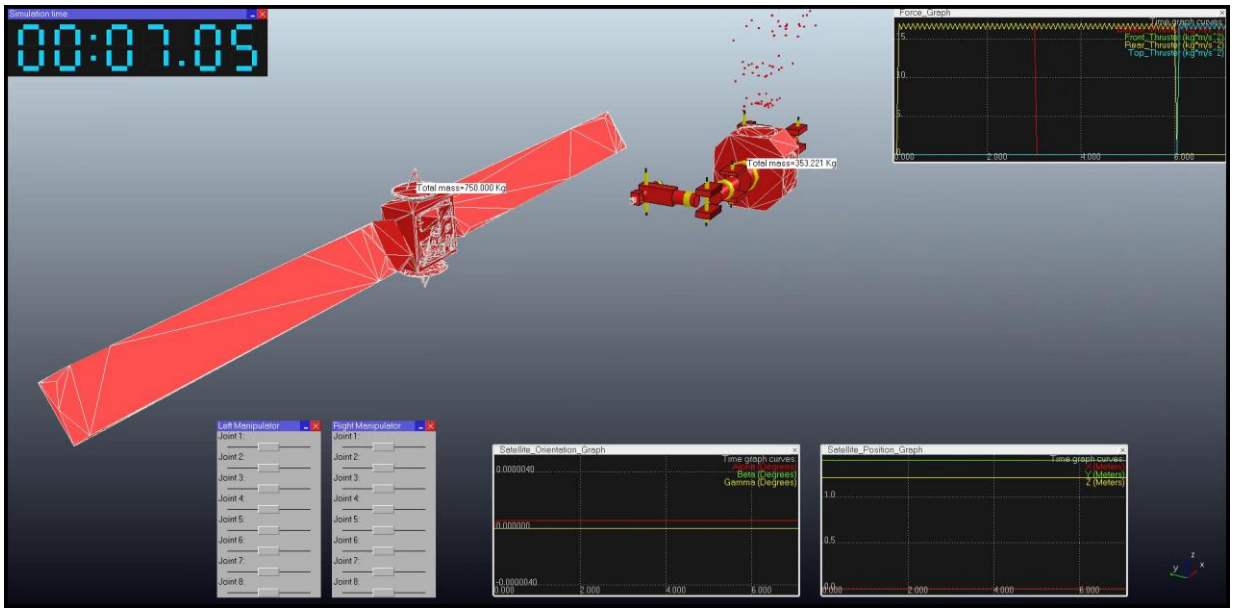

Simulation time
00:03.45
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

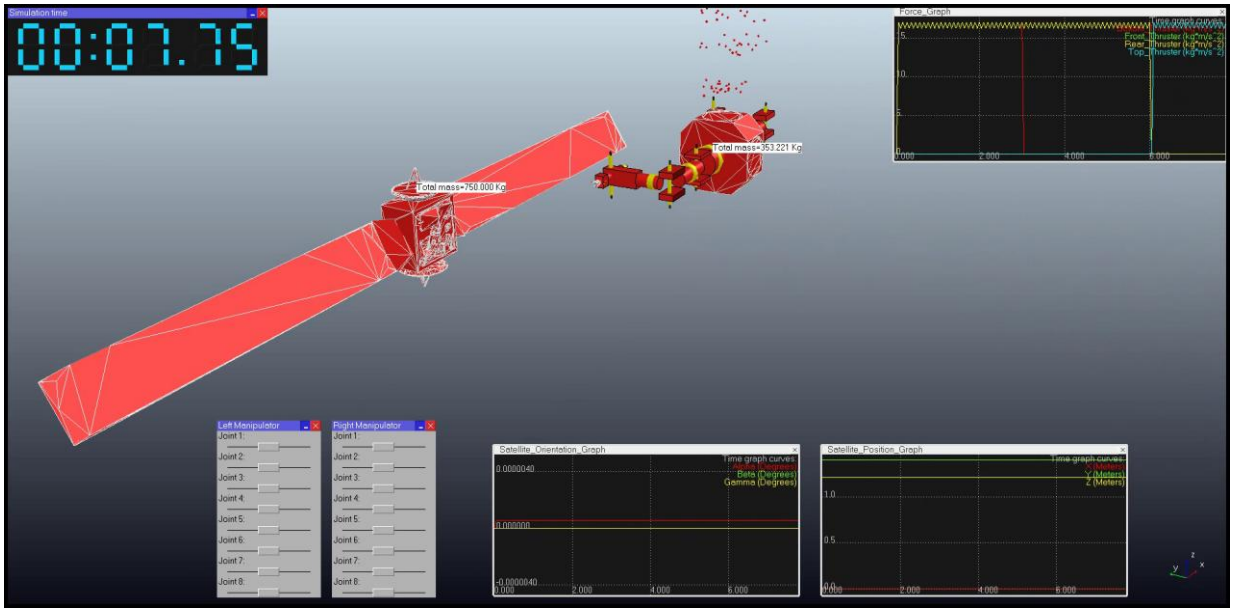

Simulation time
00 04.15
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

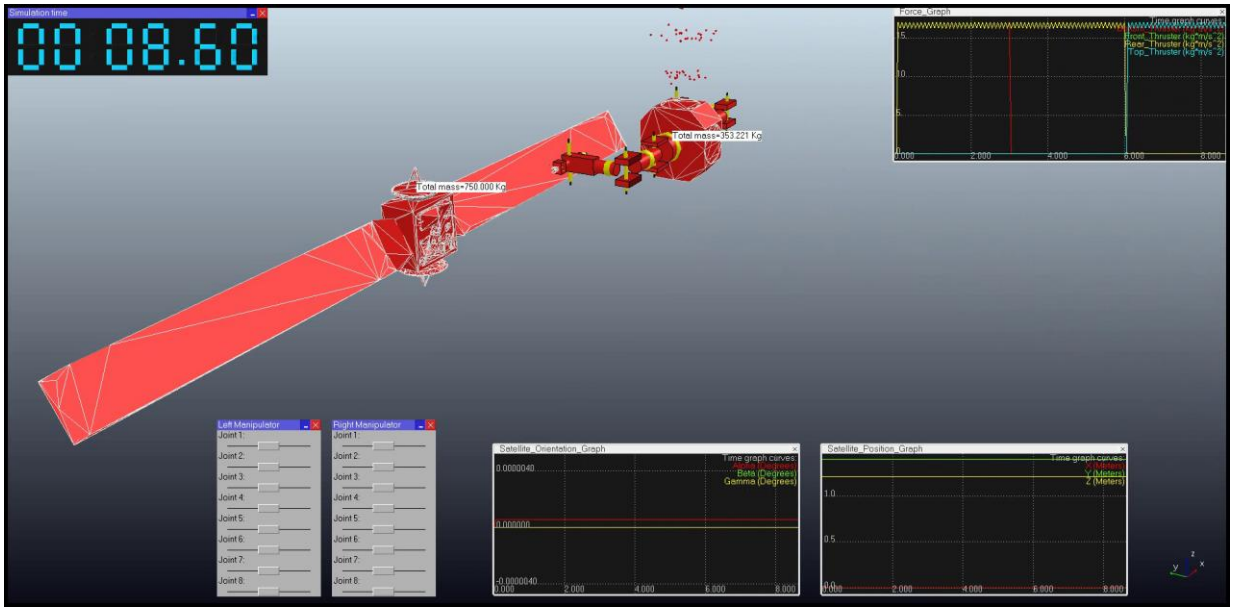

Simulation time
00 04.90
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

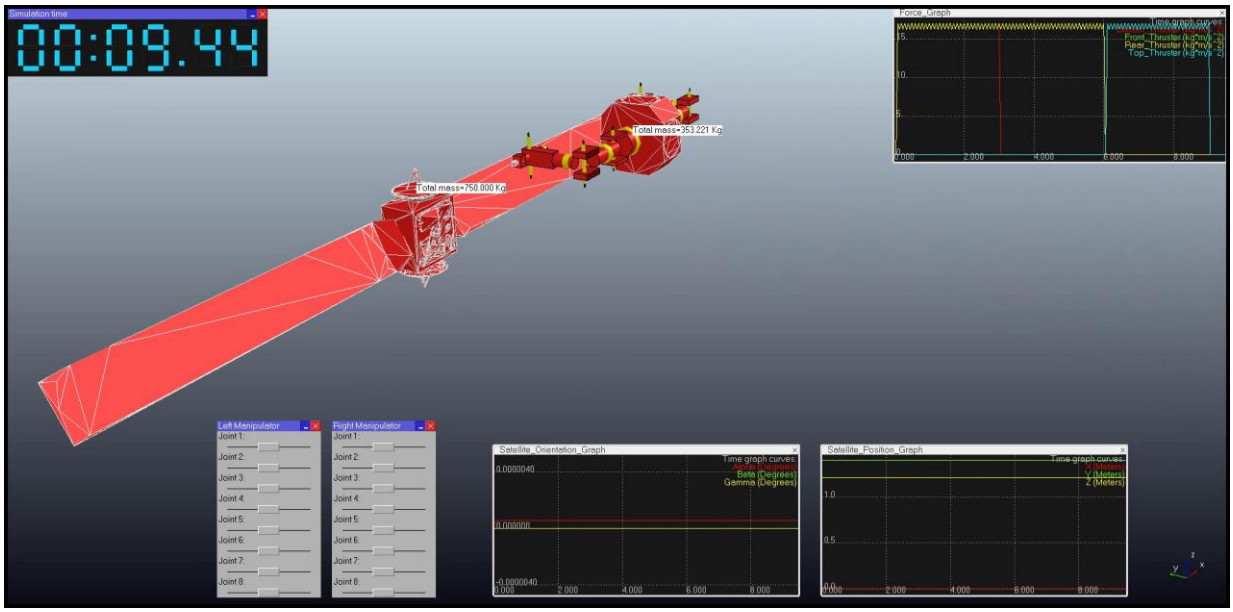

Simulation time
00:05.65
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

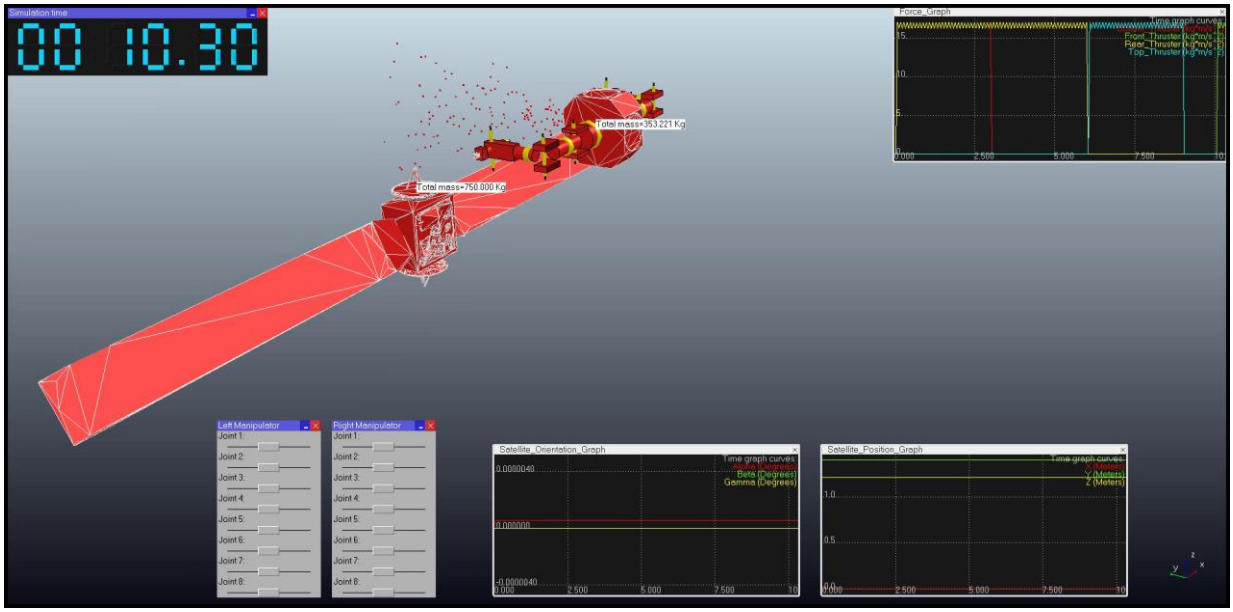

Simulation time
00 06.34
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

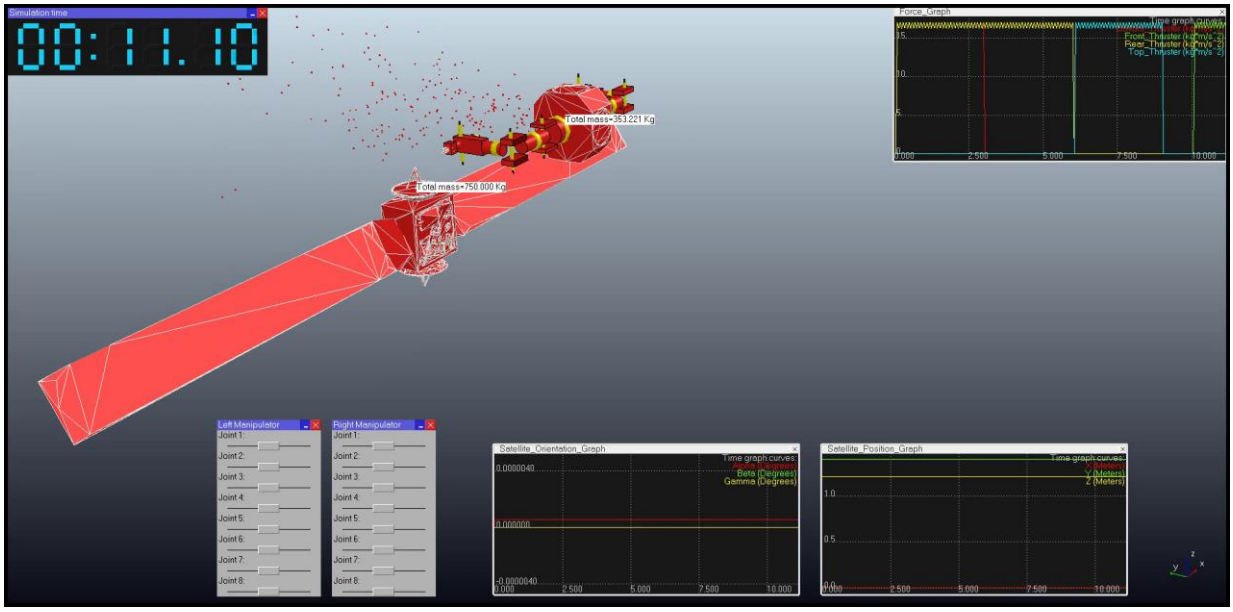

Simulation time
00:07.05
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

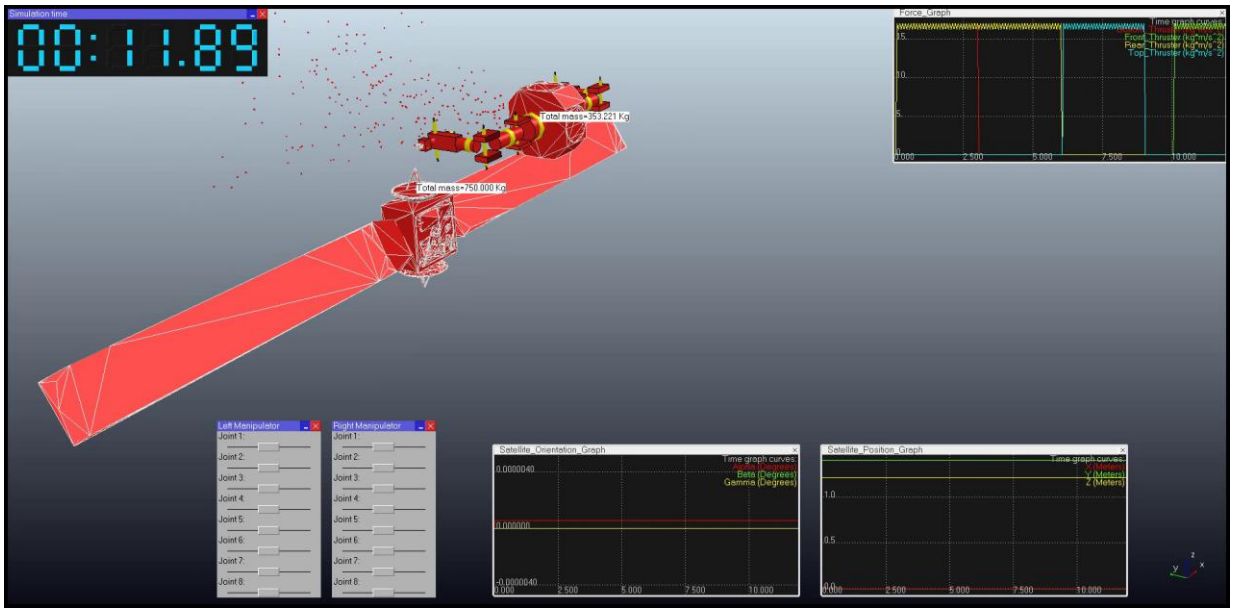

Simulation time
00:07.75
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

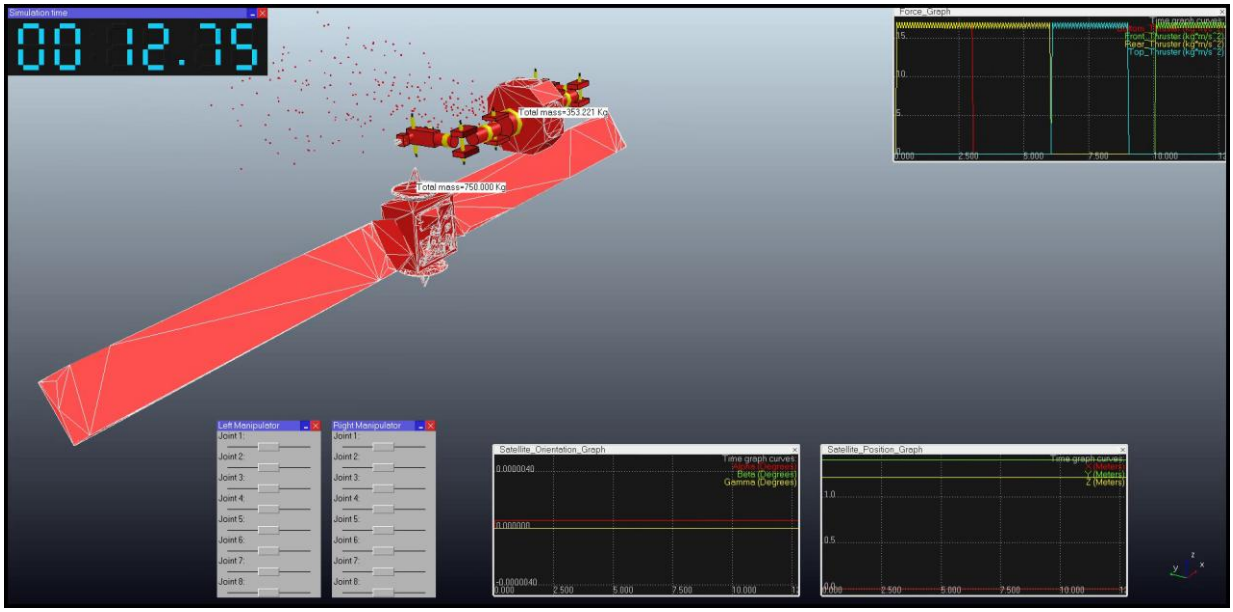

Simulation time
00 08.60
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

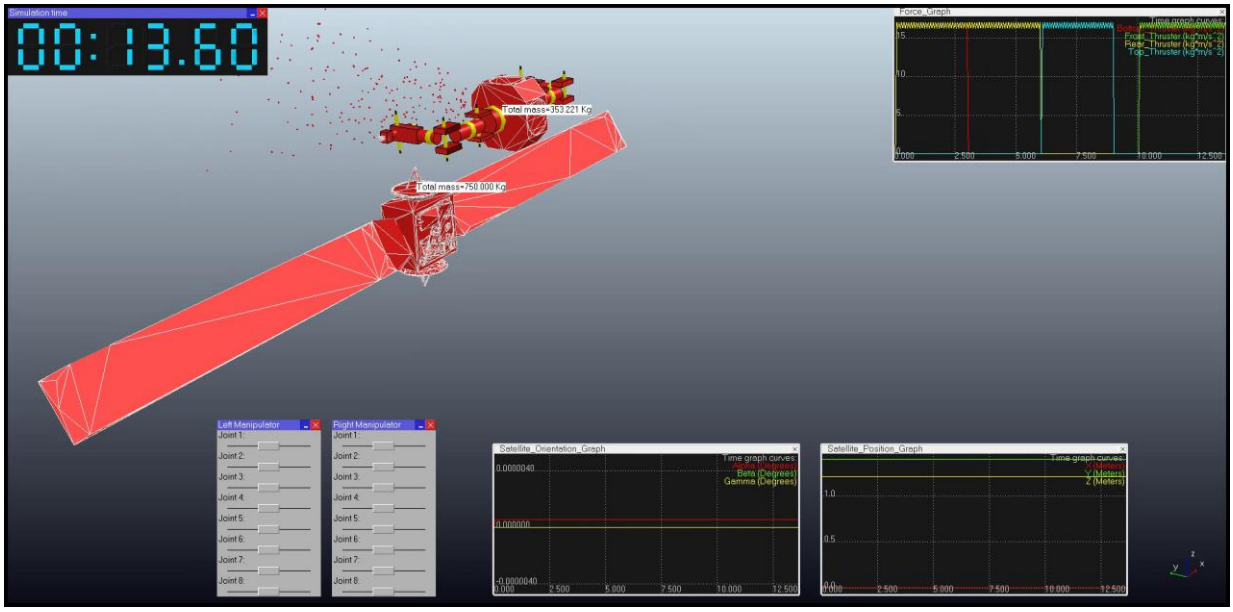

Simulation time
00:09.44
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

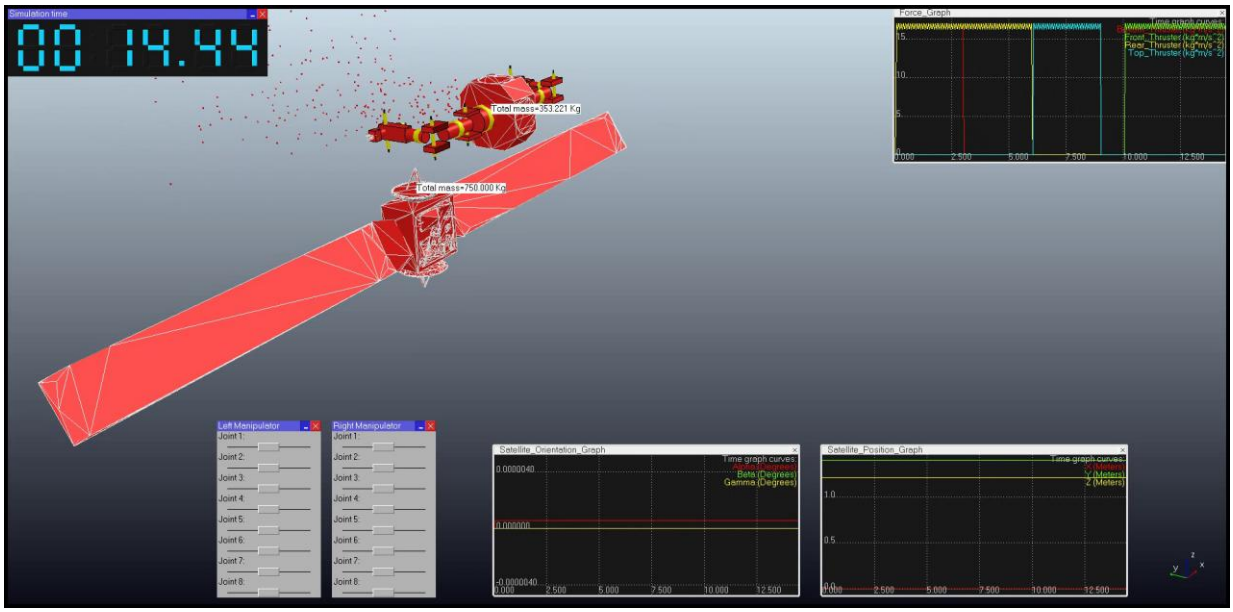

Simulation time
00:10.30
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

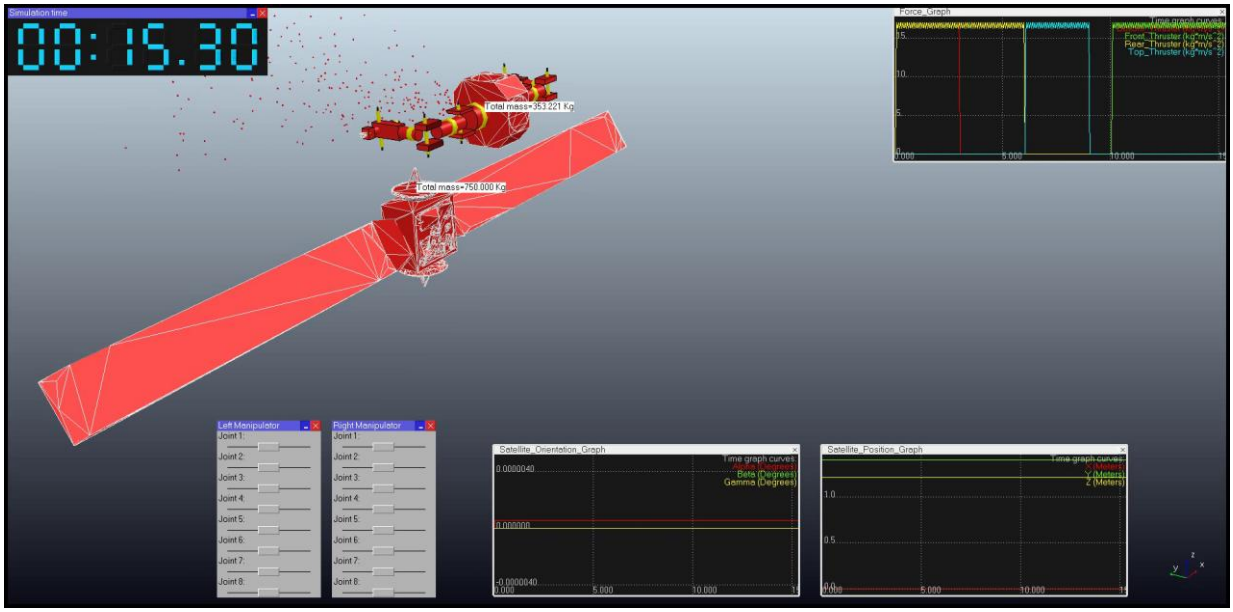

Simulation time
00:11.10
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

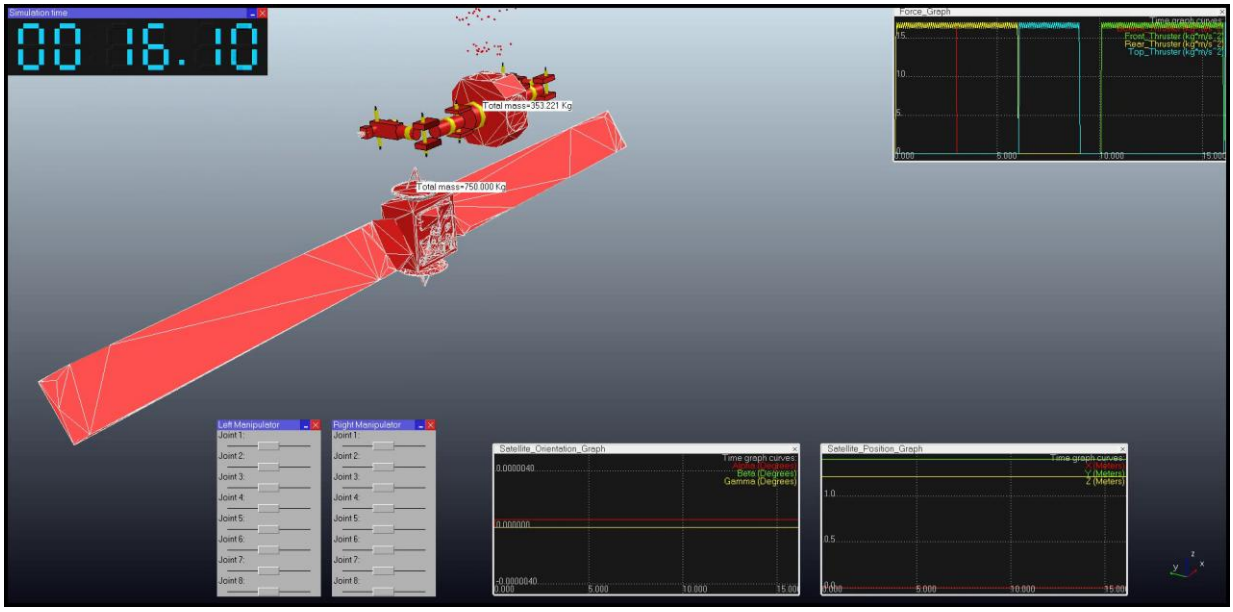

Simulation time
00: 11.89
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

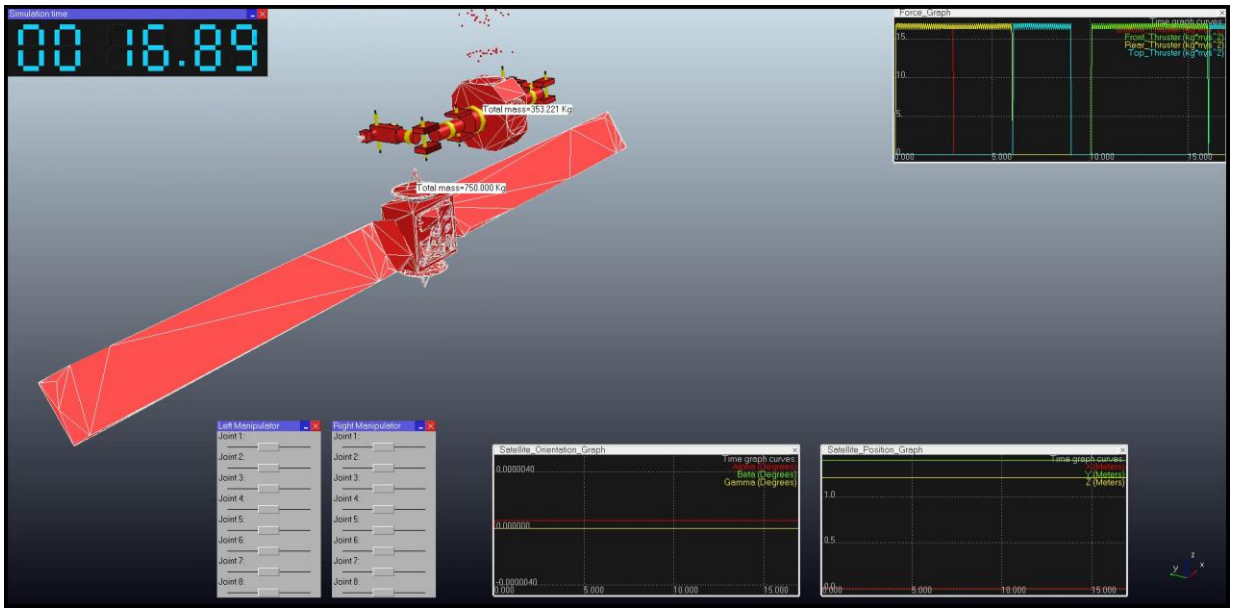

Simulation time
00 12.75
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

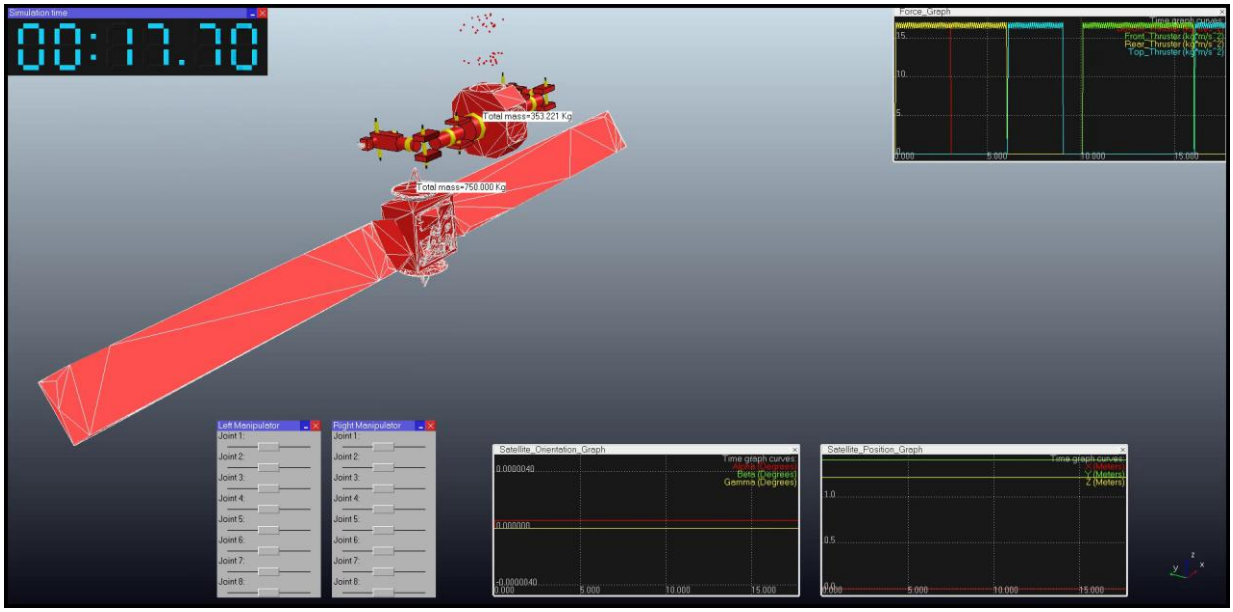

Simulation time
00: 13.60
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

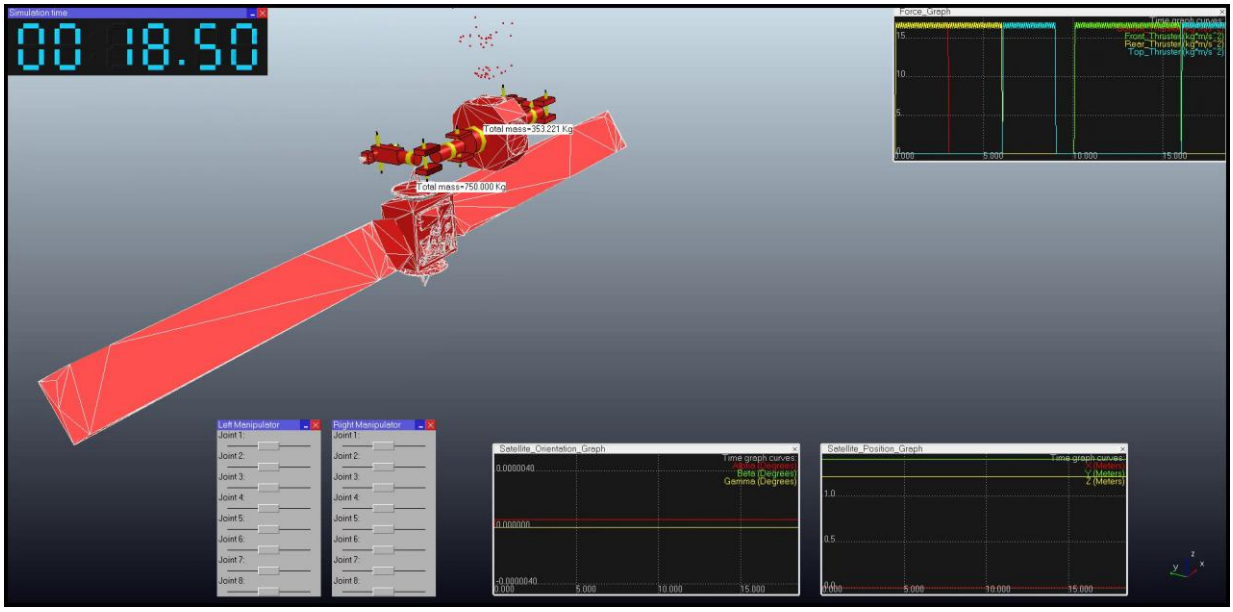

Simulation time
00 14.44
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

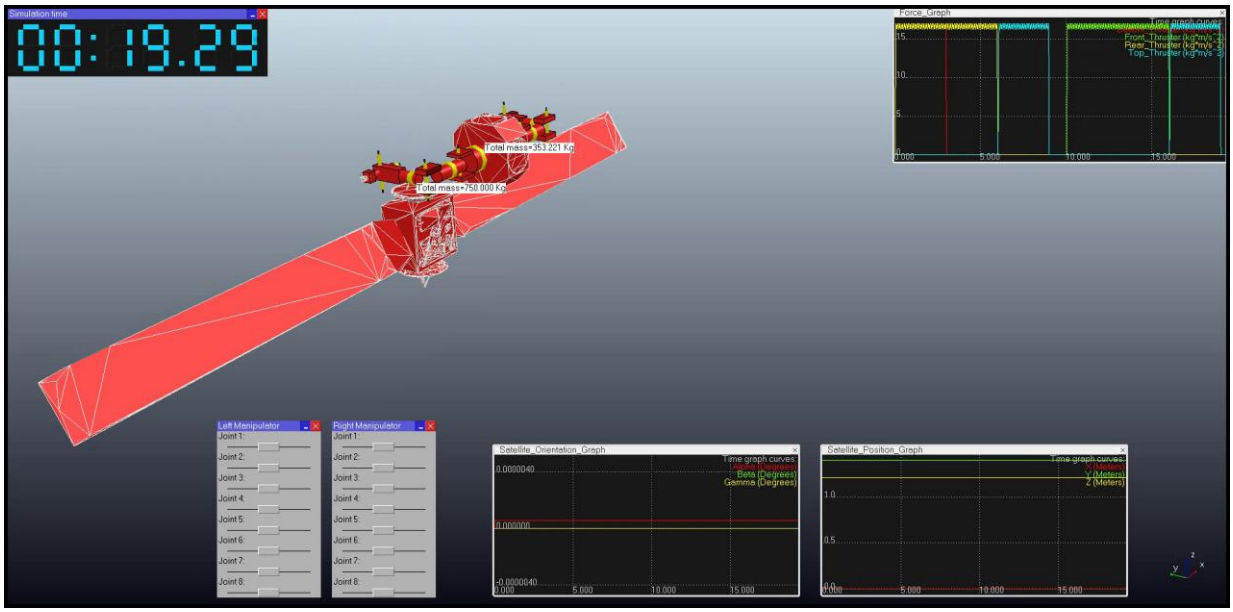

Simulation time
00: 15.30
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

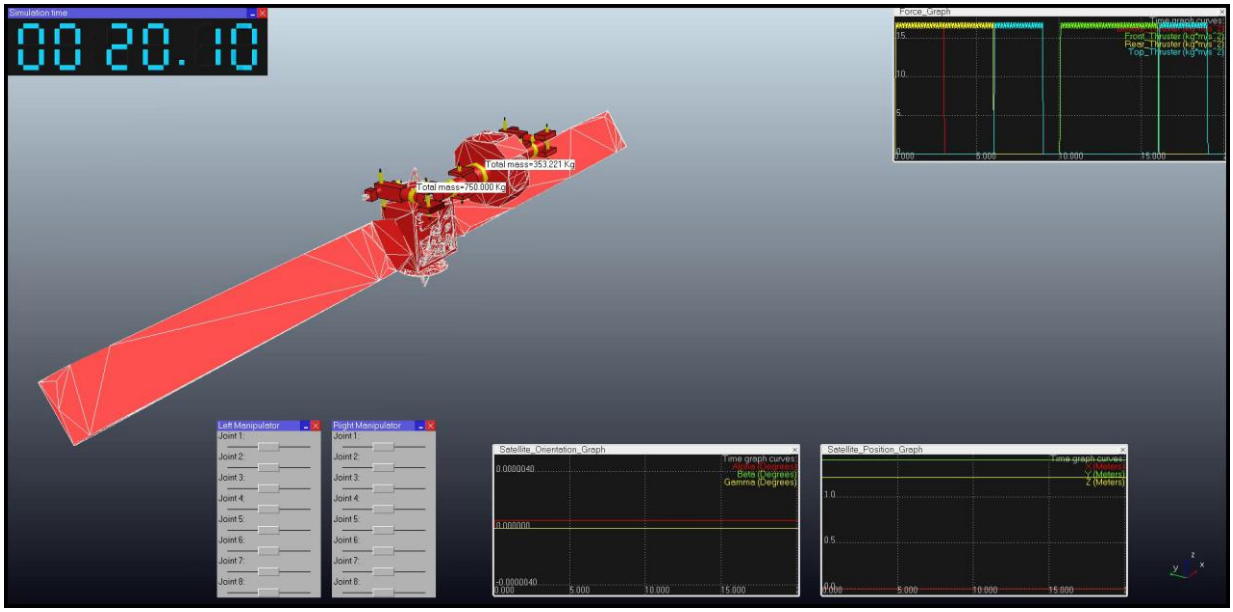

Simulation time
00 16.10
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

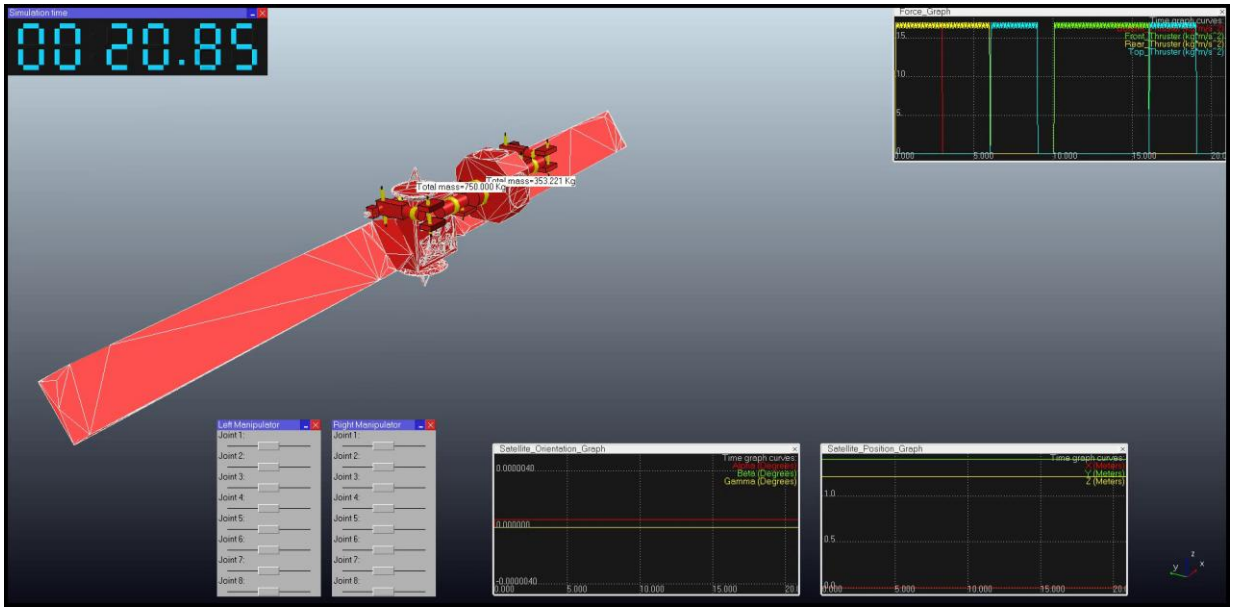

Simulation time
00 16.89
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

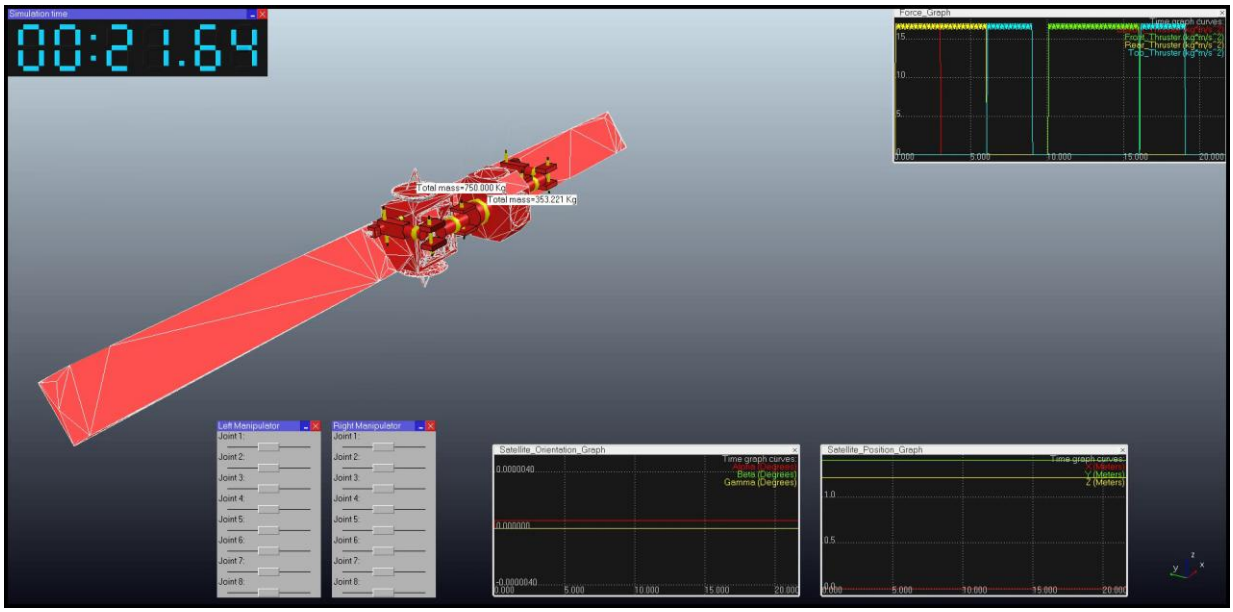

Simulation time
00: 17.70
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

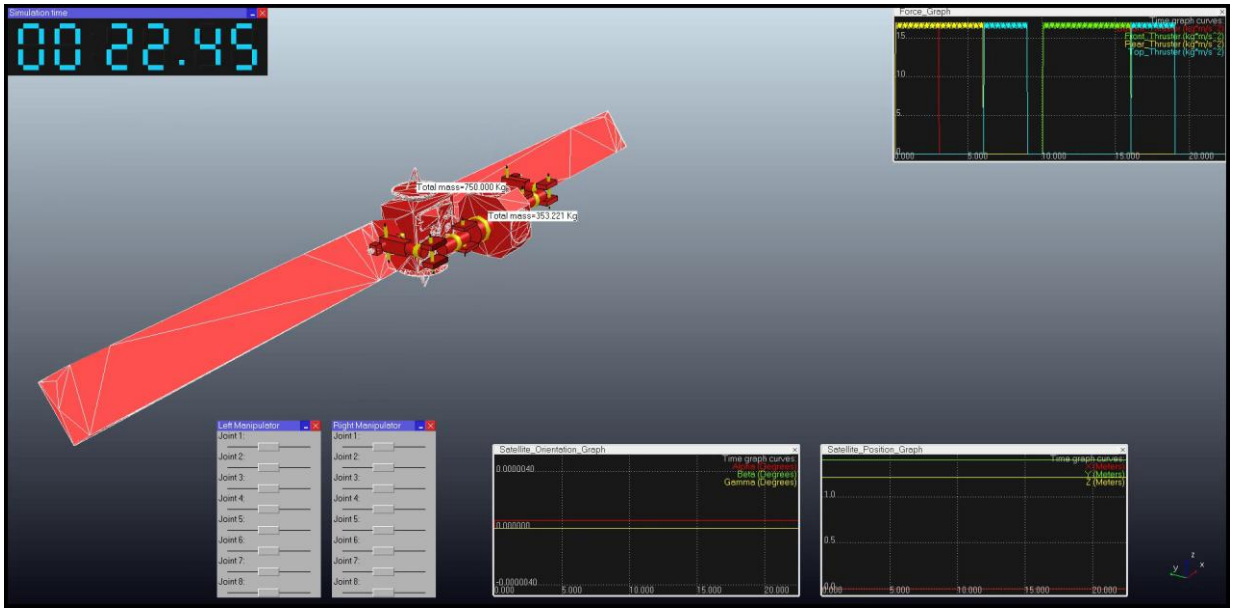

Simulation time
00 18.50
Total mass=353.221 Kg
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

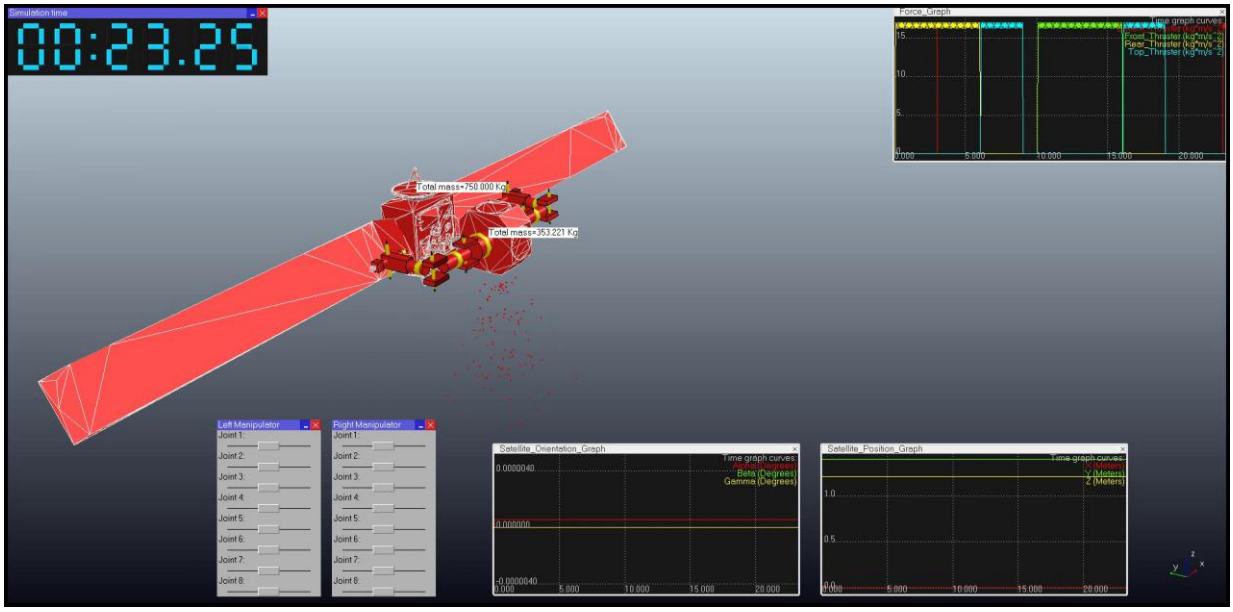

Simulation time
00:19.29
Force_Graph
Total mass=353.221 Kg
Total mass=750.000 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

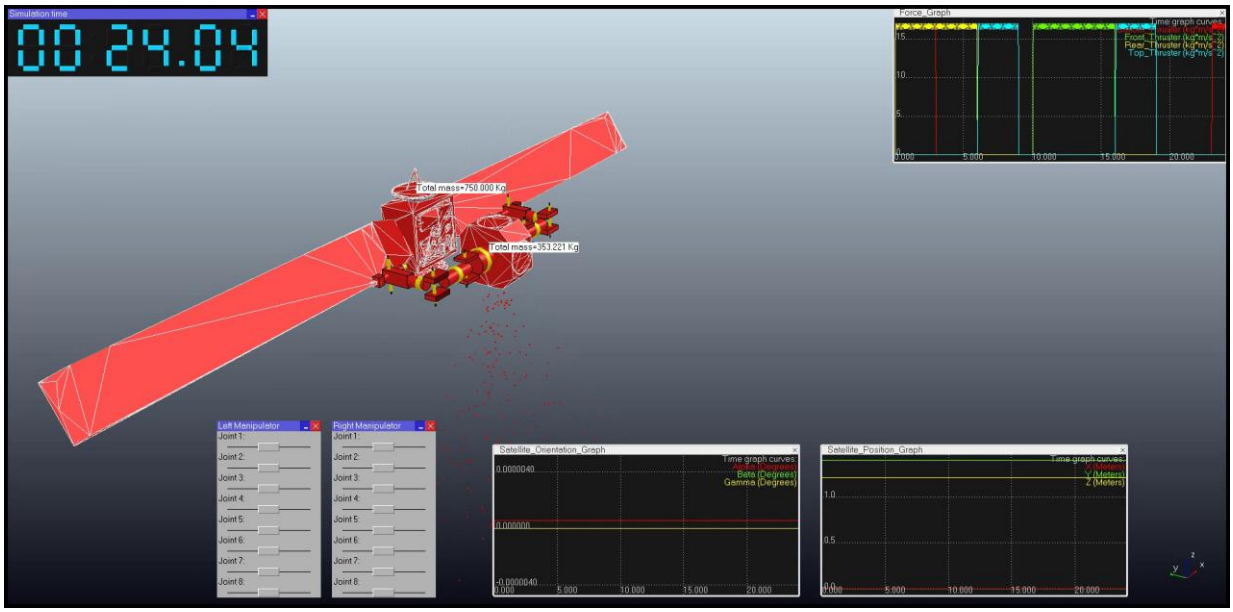

Simulation time
00 20.10
Force_Graph
Total mass=353.221 Kg
Total mass=750.000 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

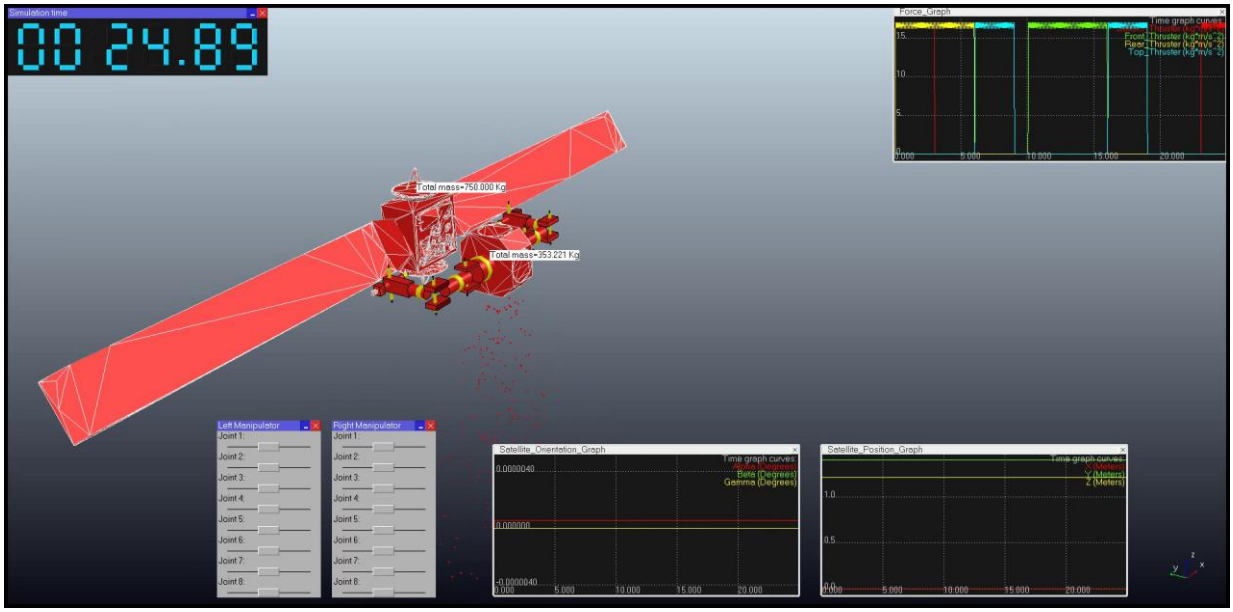

Simulation time
00 20.85
Force_Graph
Total mass=353.221 Kg
Total mass=750.000 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

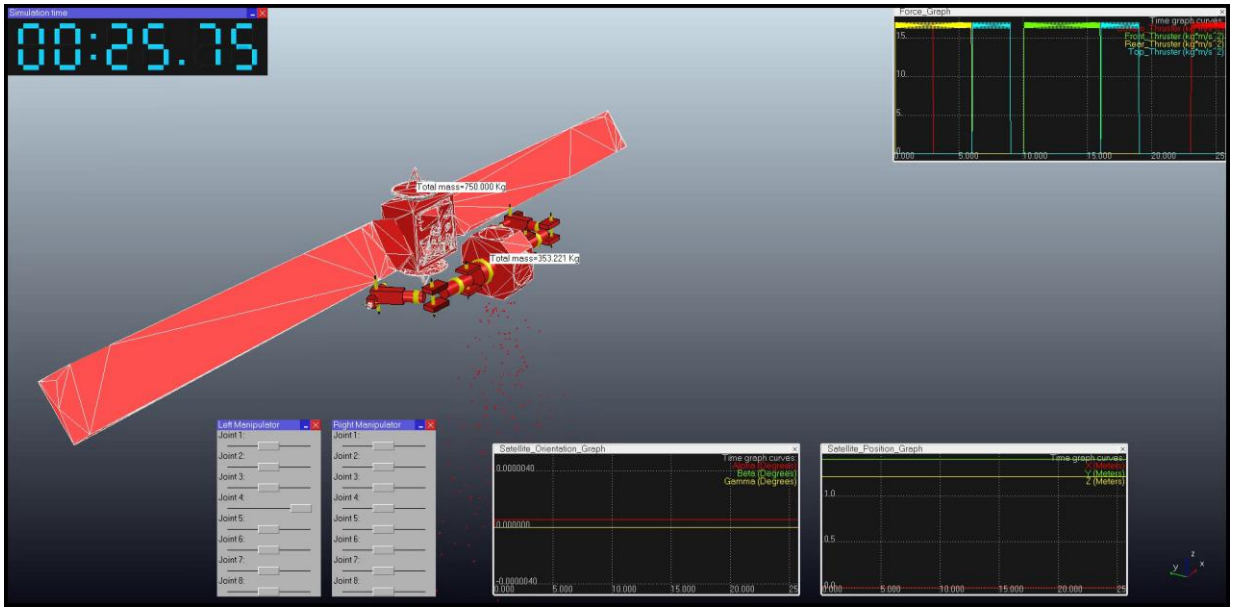

Simulation time
00:21.64
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

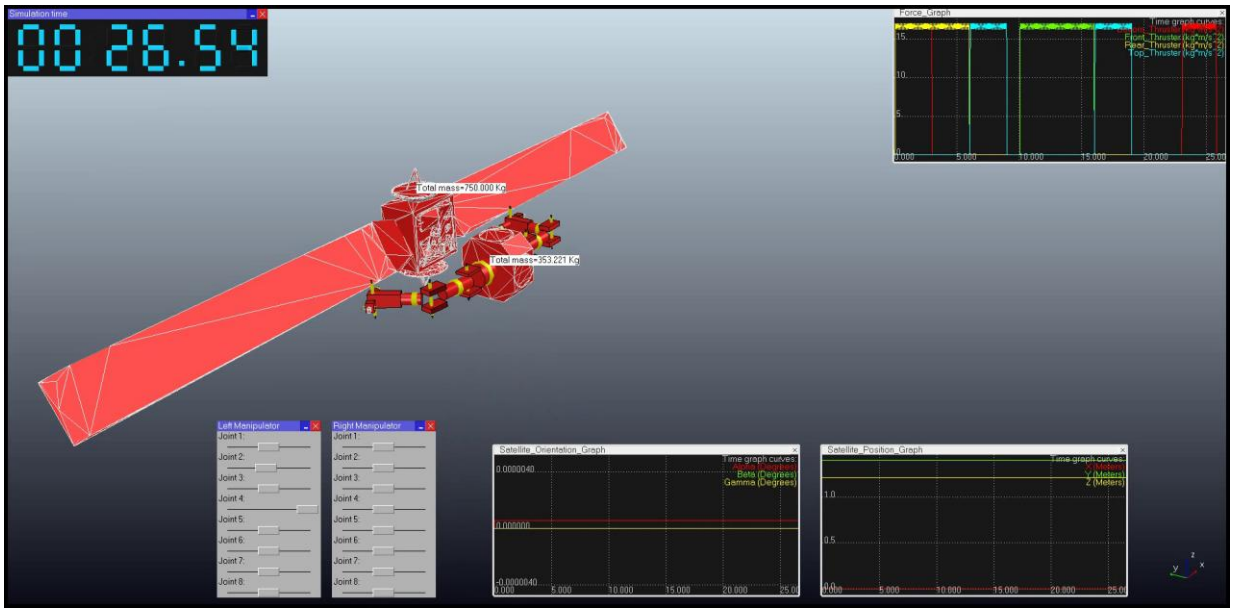

Simulation time
00 22.45
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

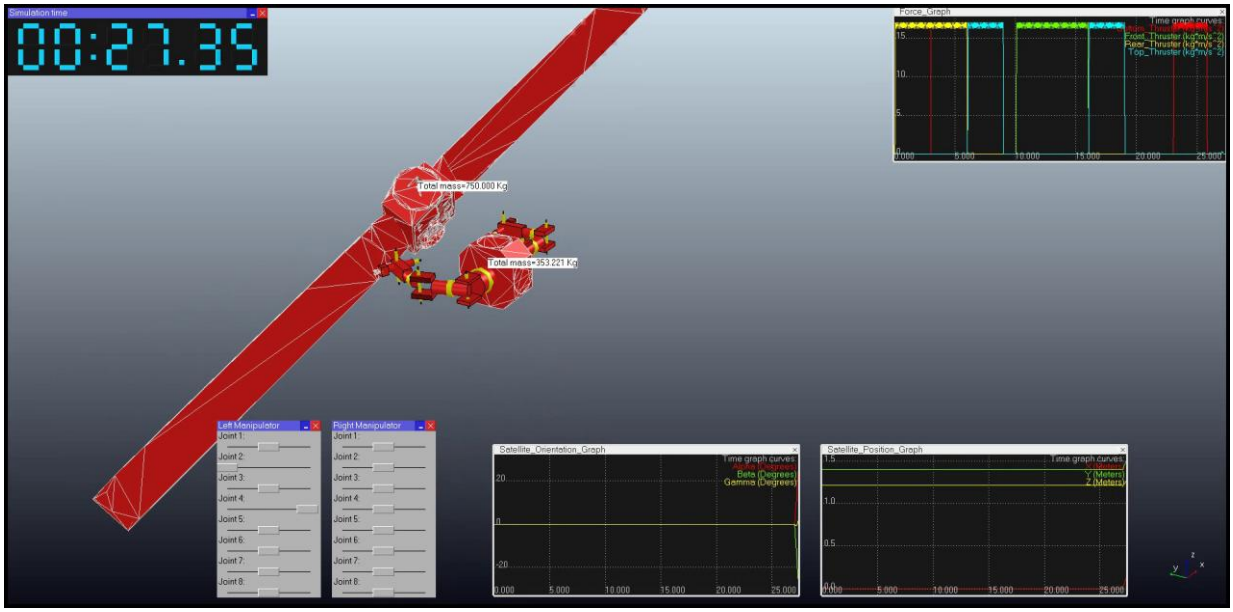

Simulation time
00:23.25
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

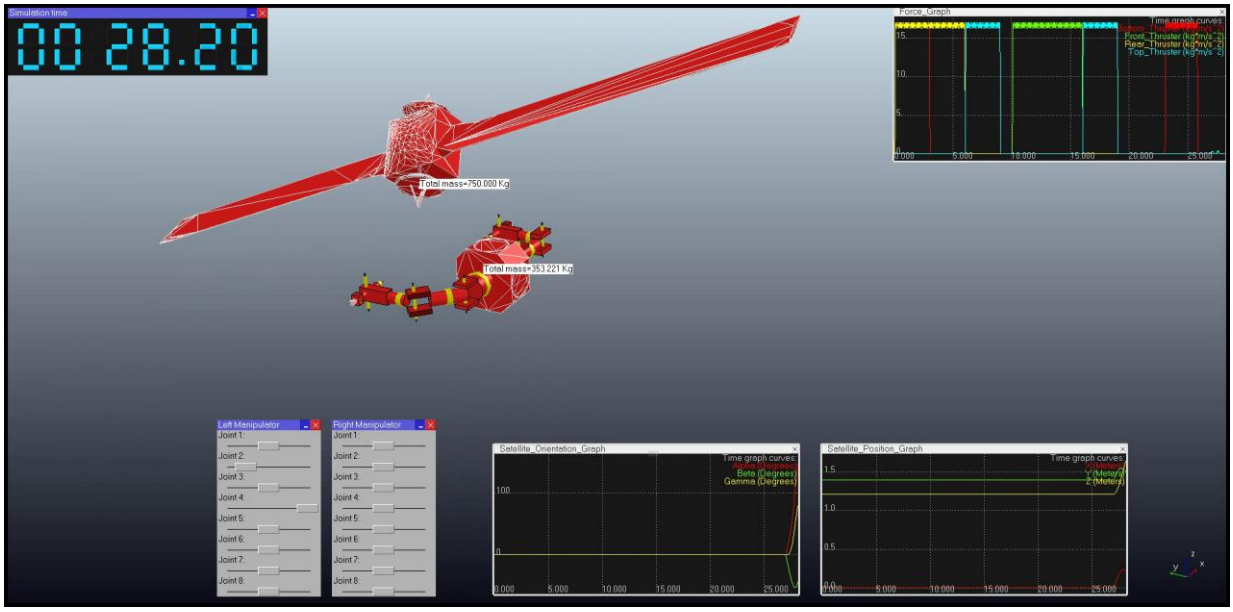

Simulation time
00 24.04
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

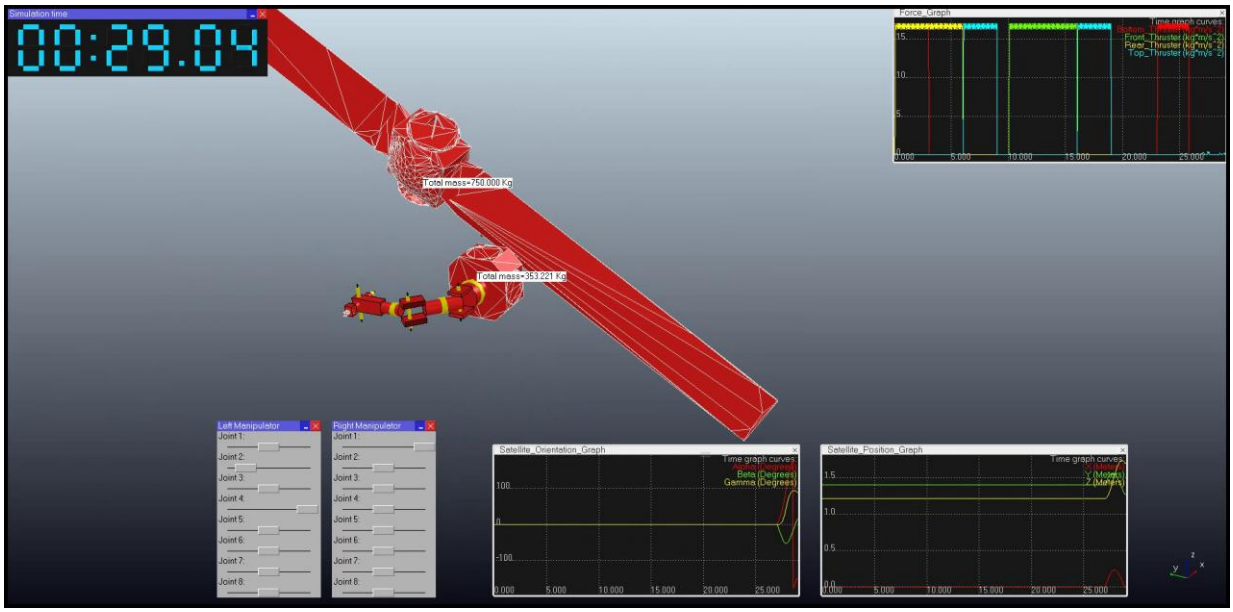

Simulation time
00 24.89
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

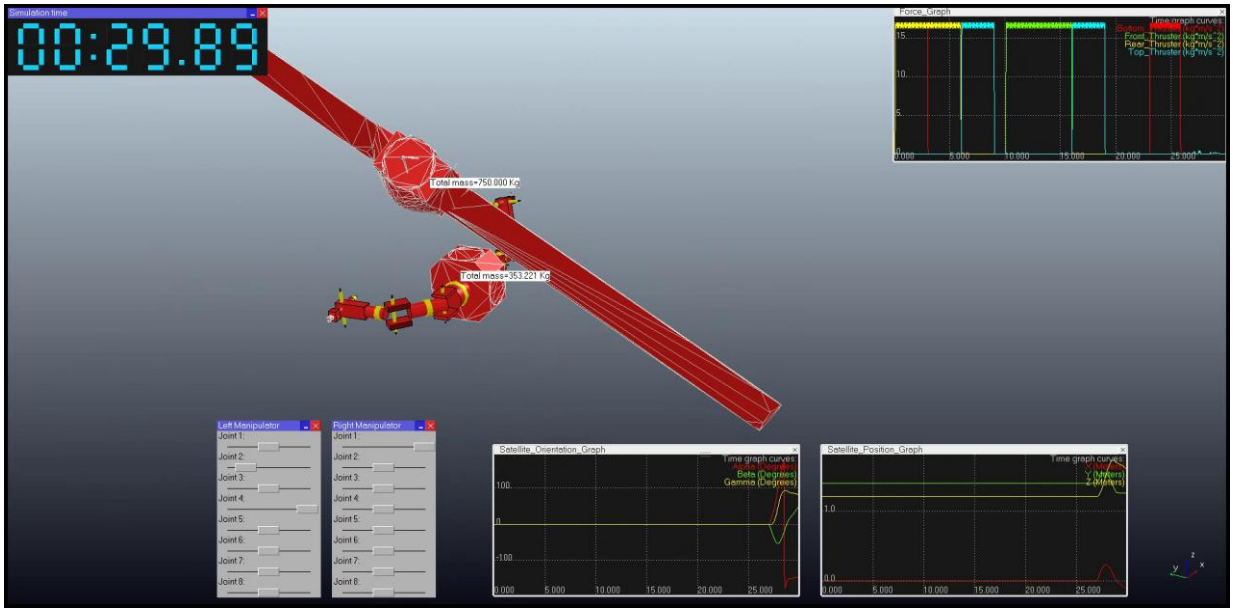

Simulation time
00:25.75
Force_Graph
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

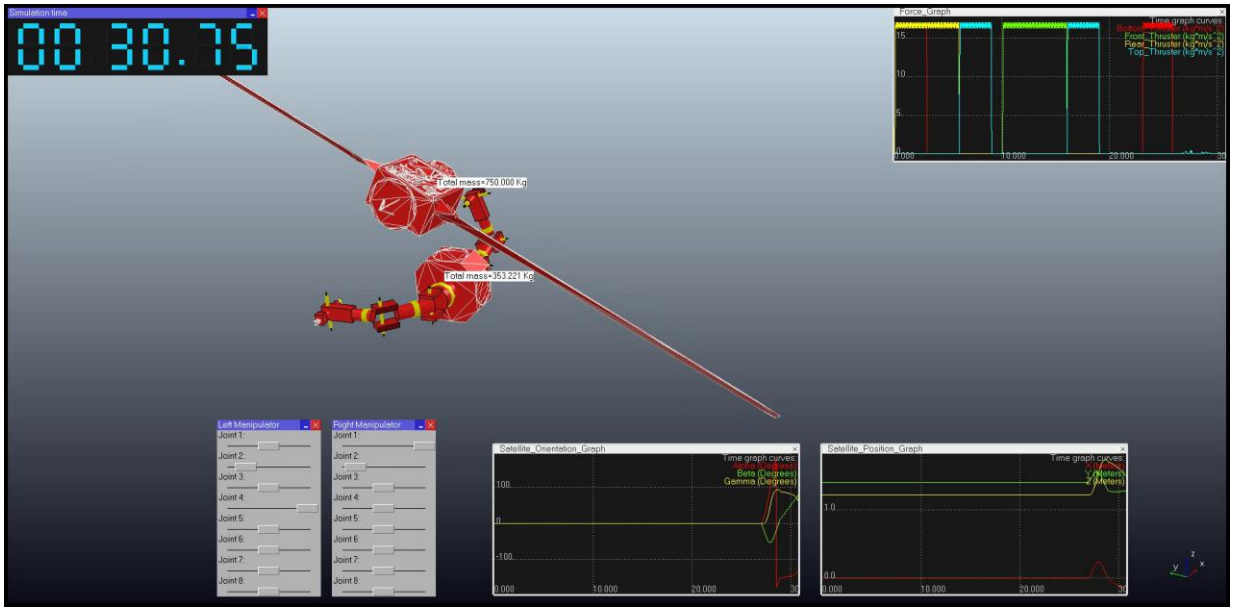

Simulation time
00 26.54
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Alpha (Degrees)
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Y (Meters)
Z (Meters)

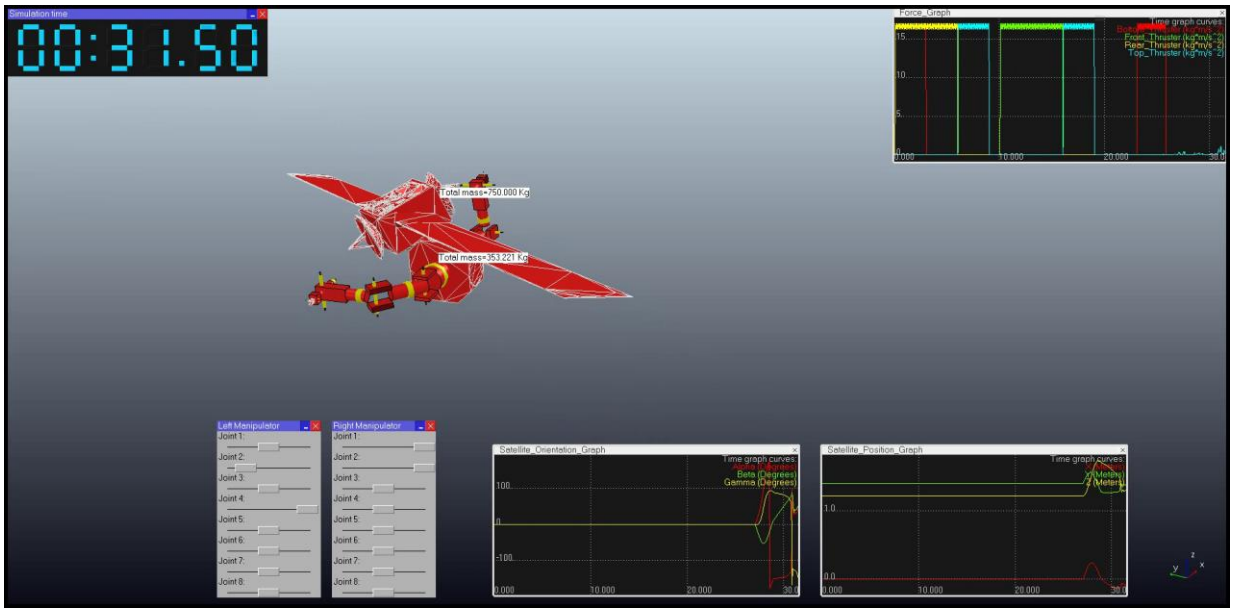

Simulation time
00:27.35
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Alpha (Degrees)
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Y (Meters)
Z (Meters)

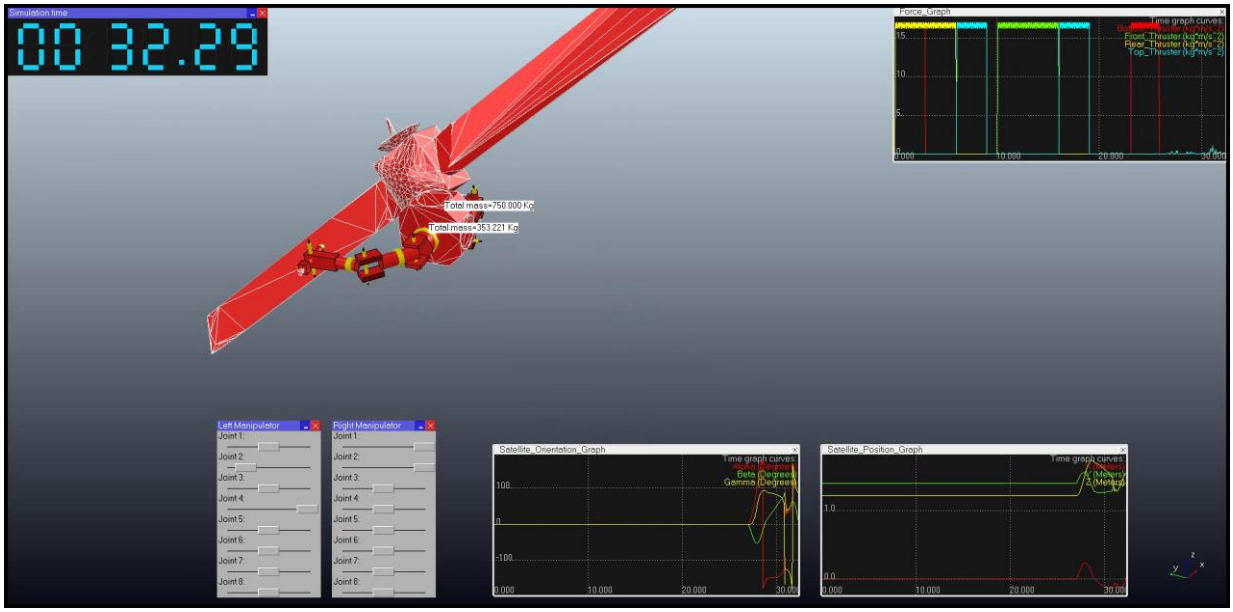

Simulation time
00 28.20
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Alpha (Degrees)
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Y (Meters)
Z (Meters)

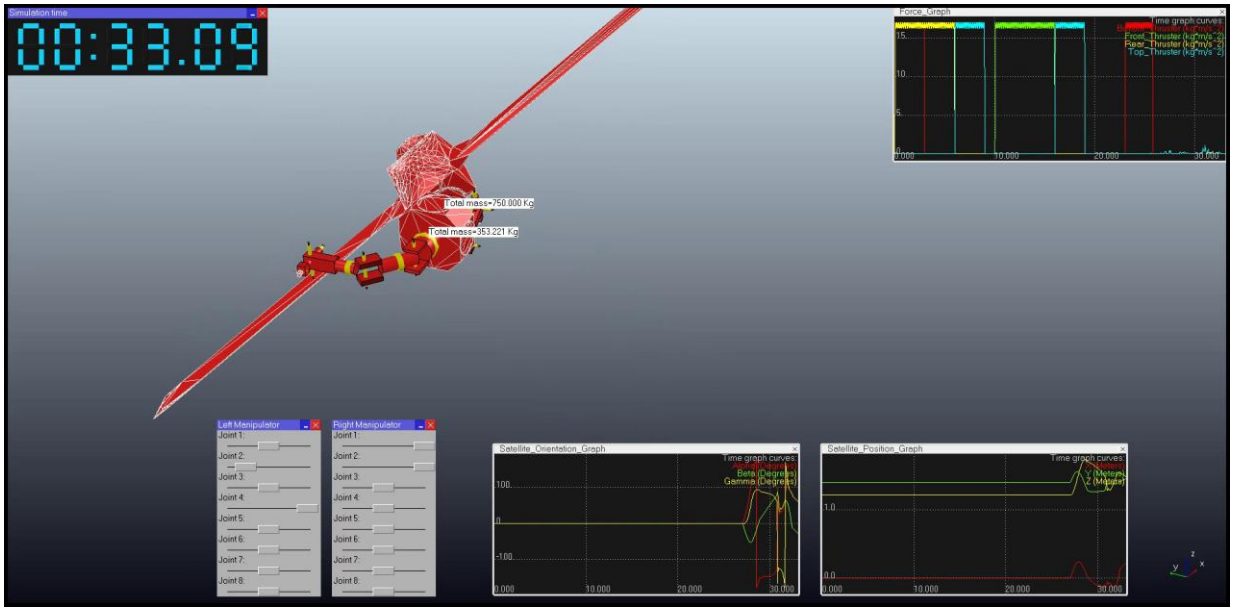

Simulation time
00:29.04
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

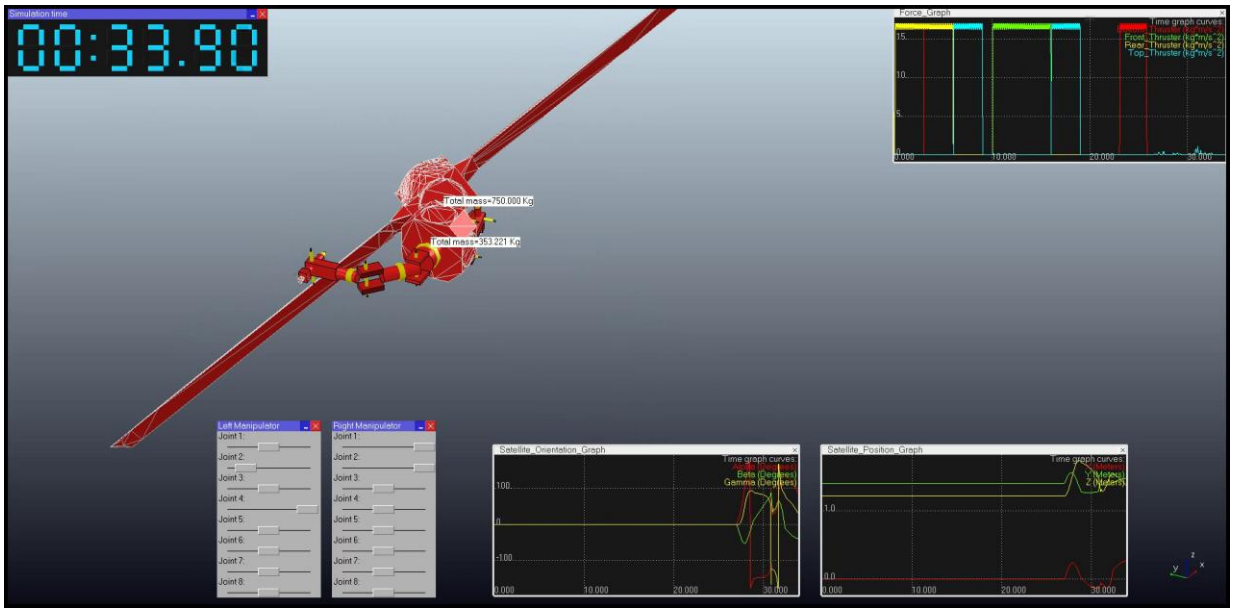

Simulation time
00:29.89
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

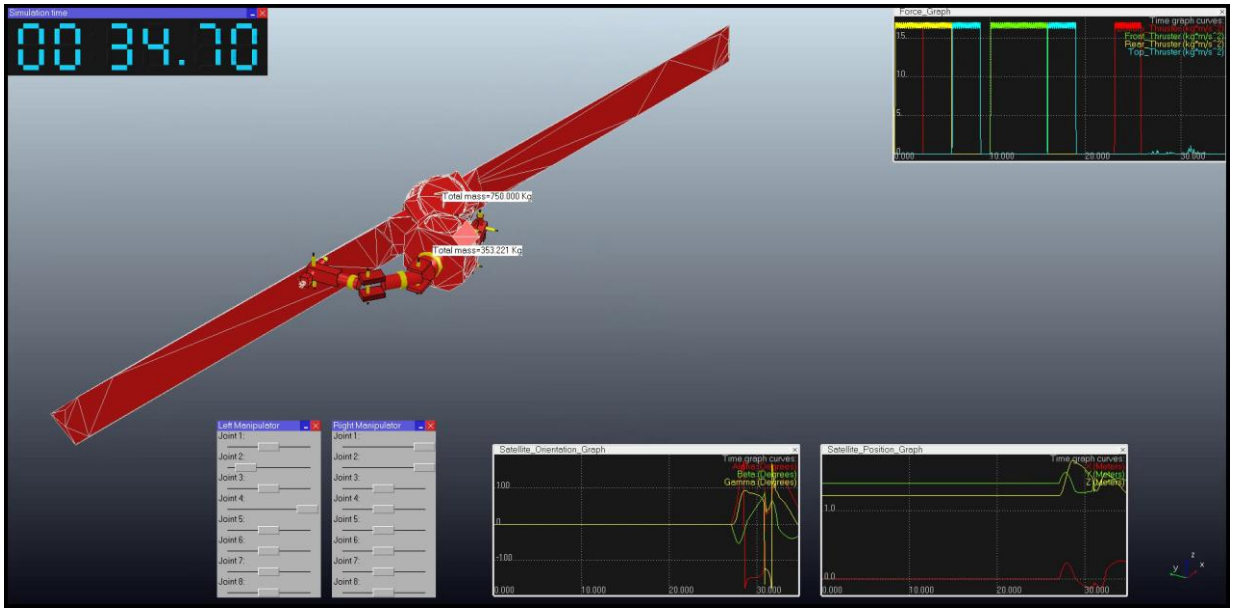

Simulation time
00 30.75
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

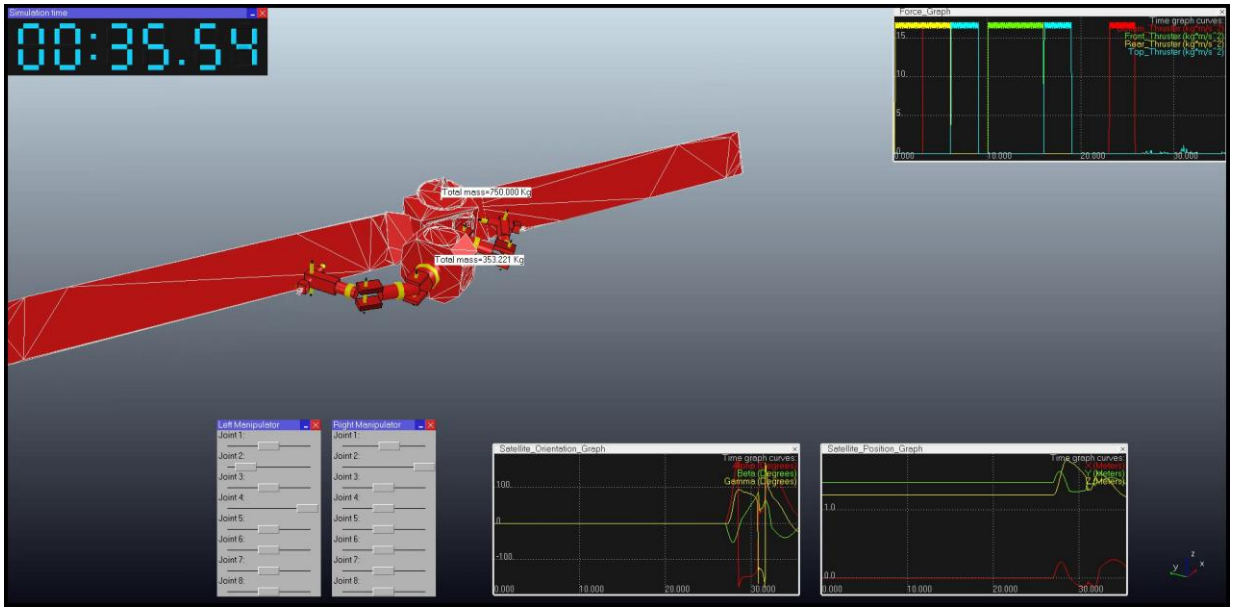

Simulation time
00:31.50
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Z (Meters)


Simulation time
00 32.29
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Z (Meters)

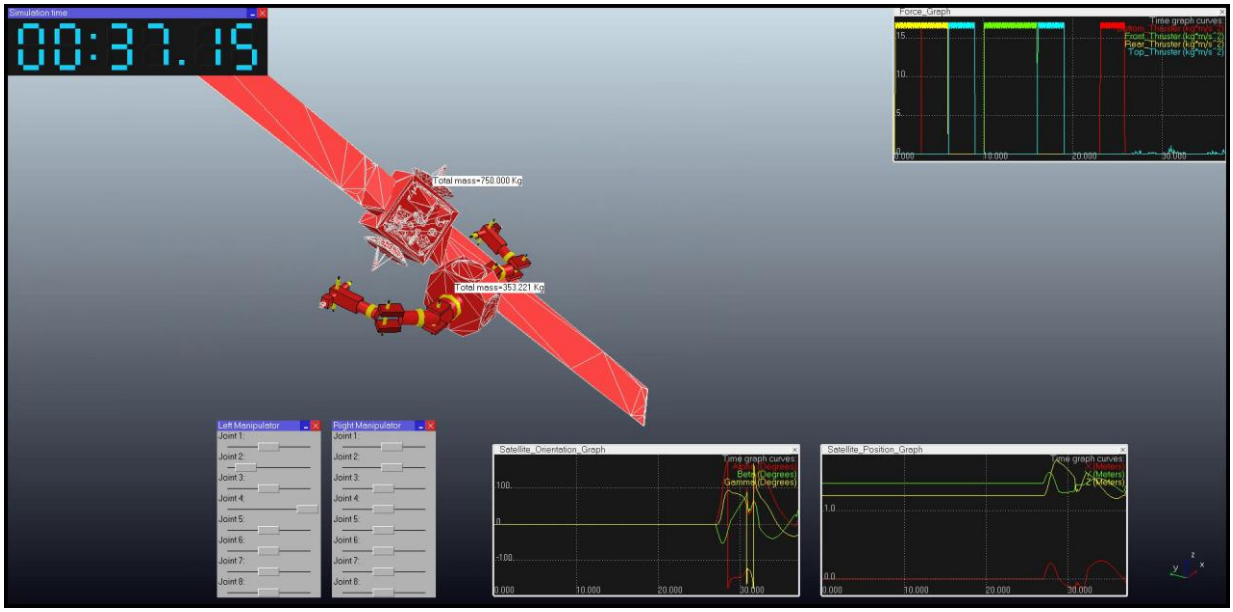

Simulation time
00:33.09
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Z (Meters)

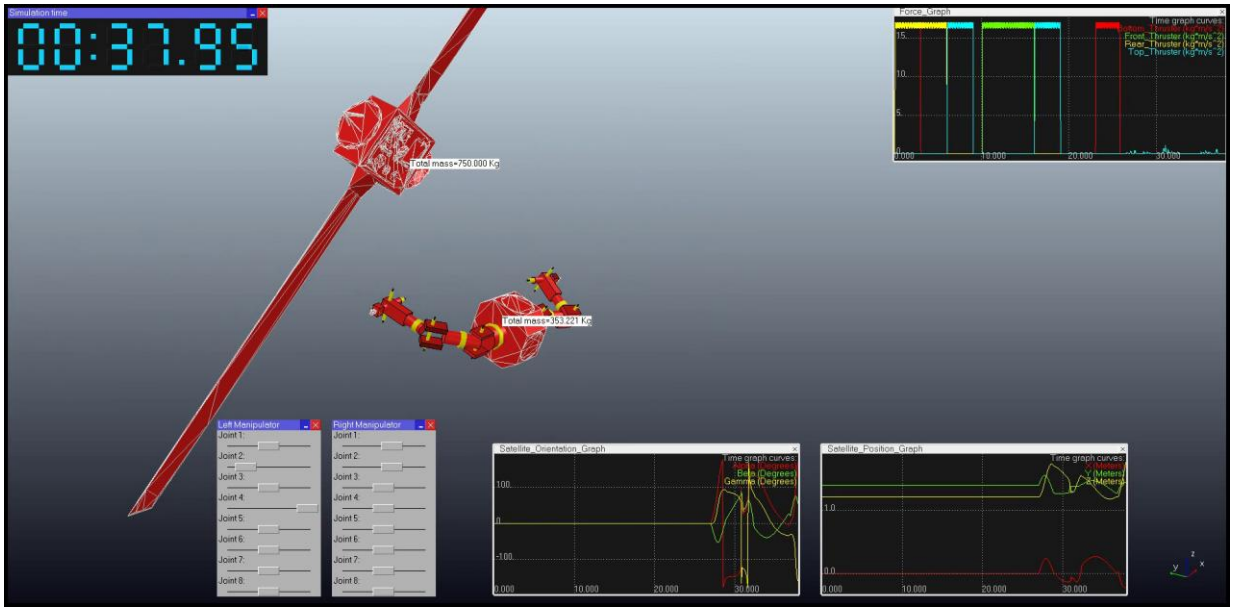

Simulation time
00:33.90
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Z (Meters)

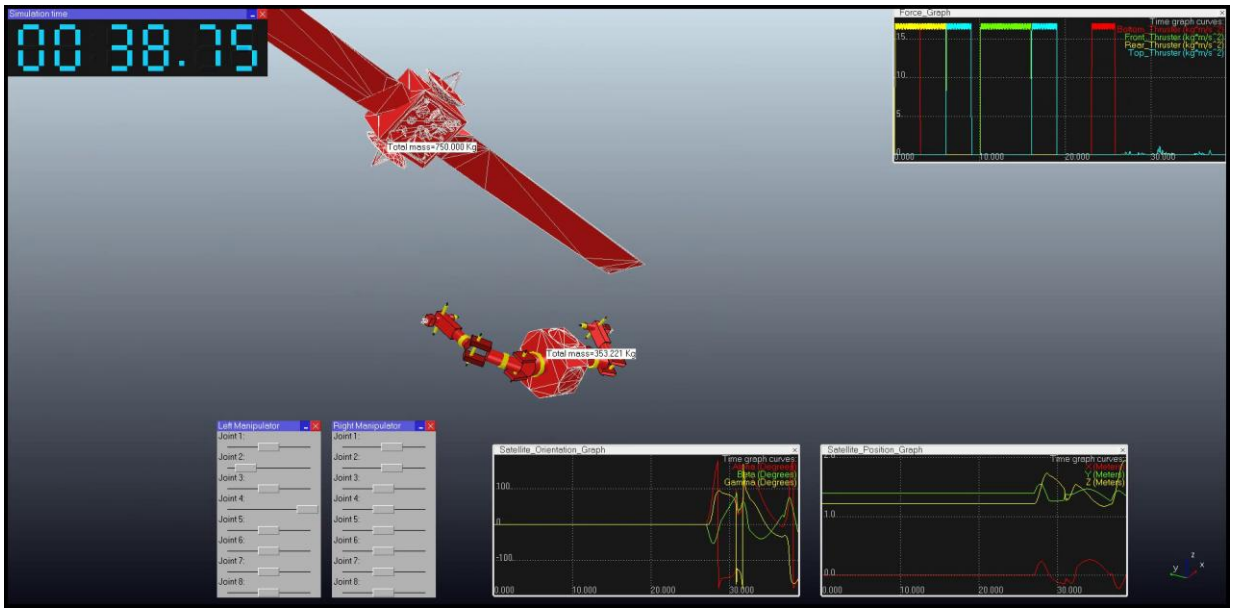

Simulation time
00 34.70
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Z (Meters)

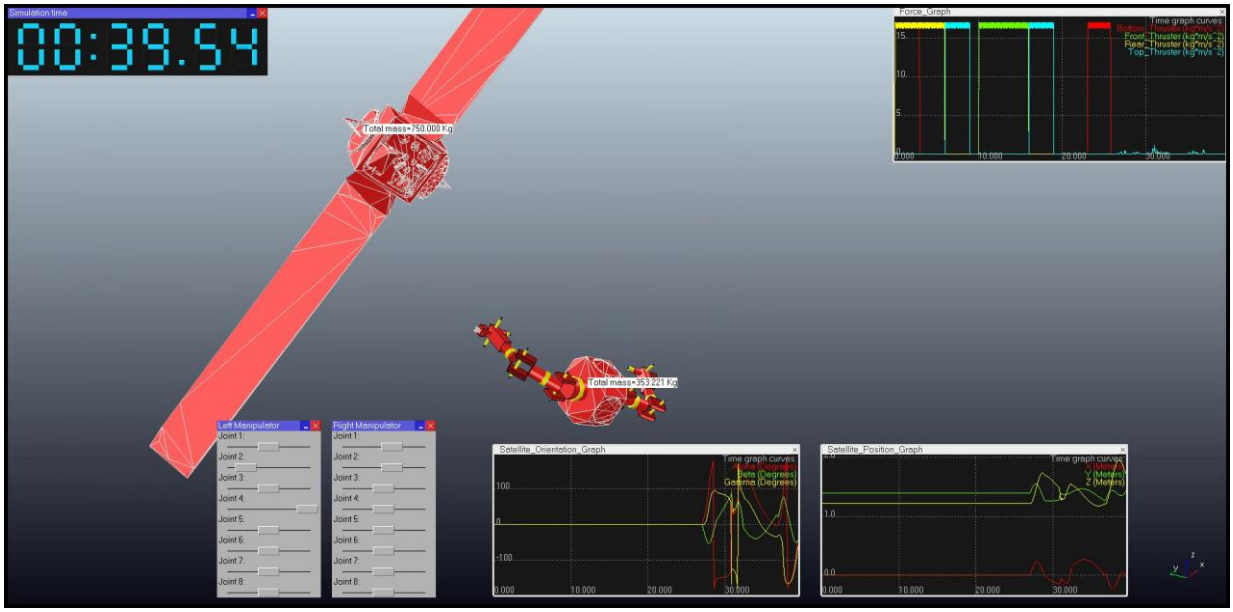

Simulation time
00:35.54
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
X (Meters)
Z (Meters)

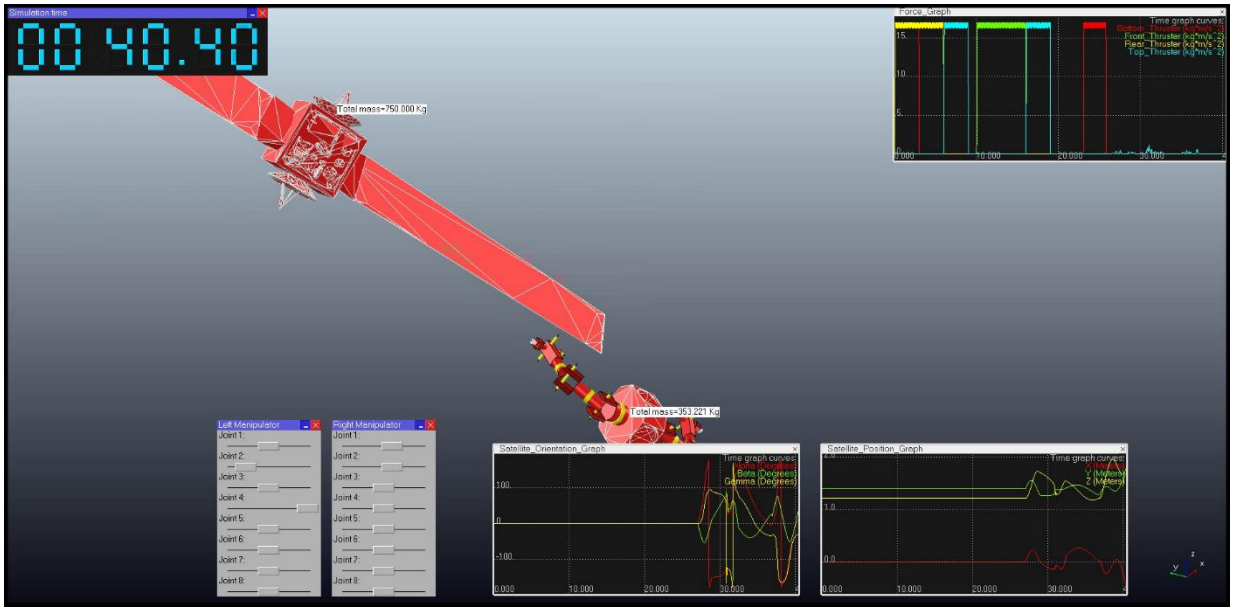

Simulation time
00 36.34
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
Y (Meters)
Z (Meters)

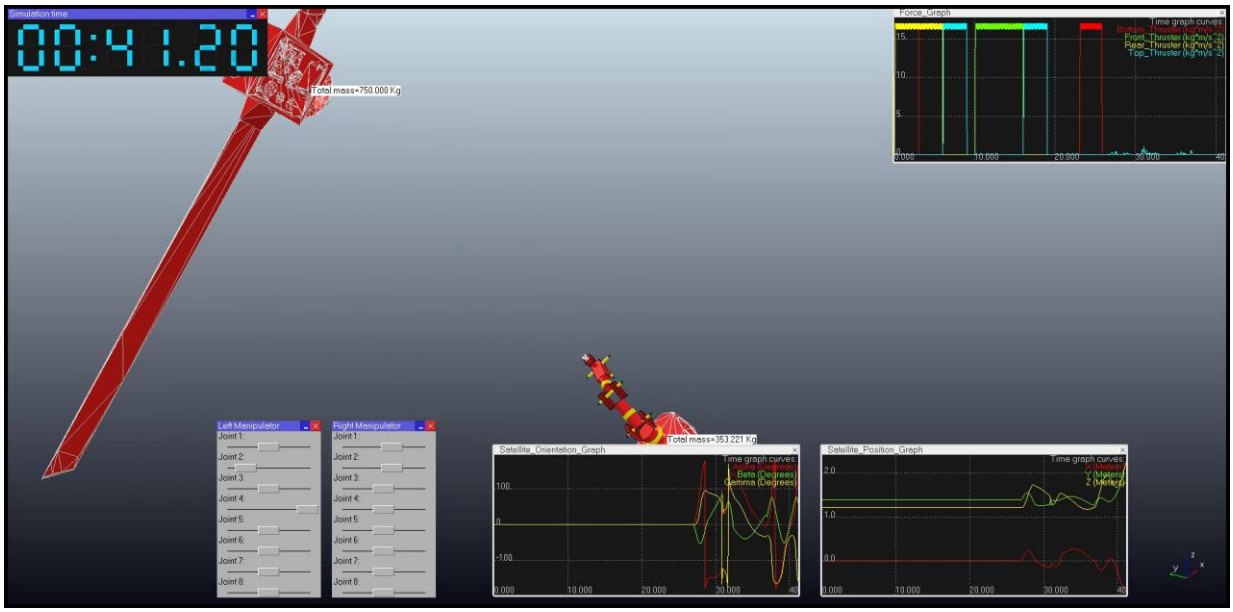

Simulation time
00:37.15
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
Y (Meters)
Z (Meters)

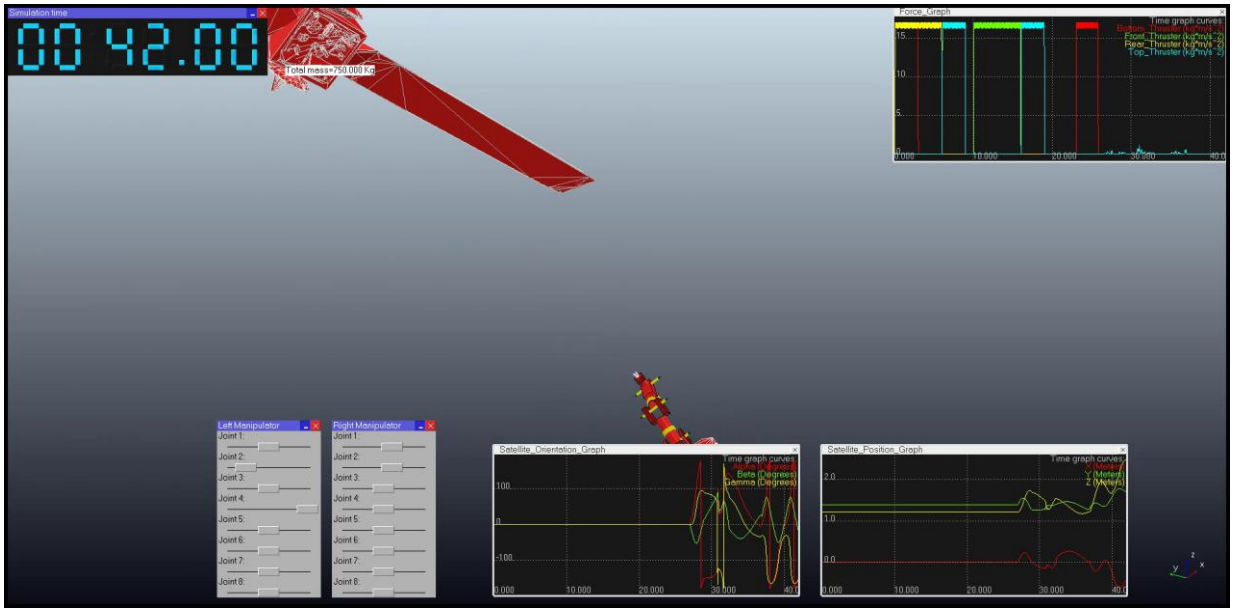

Simulation time
00:37.95
Force_Graph
Time graph curves
Front_Thruster (kg*m/s^2)
Rear_Thruster (kg*m/s^2)
Top_Thruster (kg*m/s^2)
Total mass=750.000 Kg
Total mass=353.221 Kg
Left Manipulator
Right Manipulator
Joint 1:
Joint 2:
Joint 3:
Joint 4:
Joint 5:
Joint 6:
Joint 7:
Joint 8:
Satellite_Orientation_Graph
Time graph curves
Beta (Degrees)
Gamma (Degrees)
Satellite_Position_Graph
Time graph curves
Y (Meters)
Z (Meters)

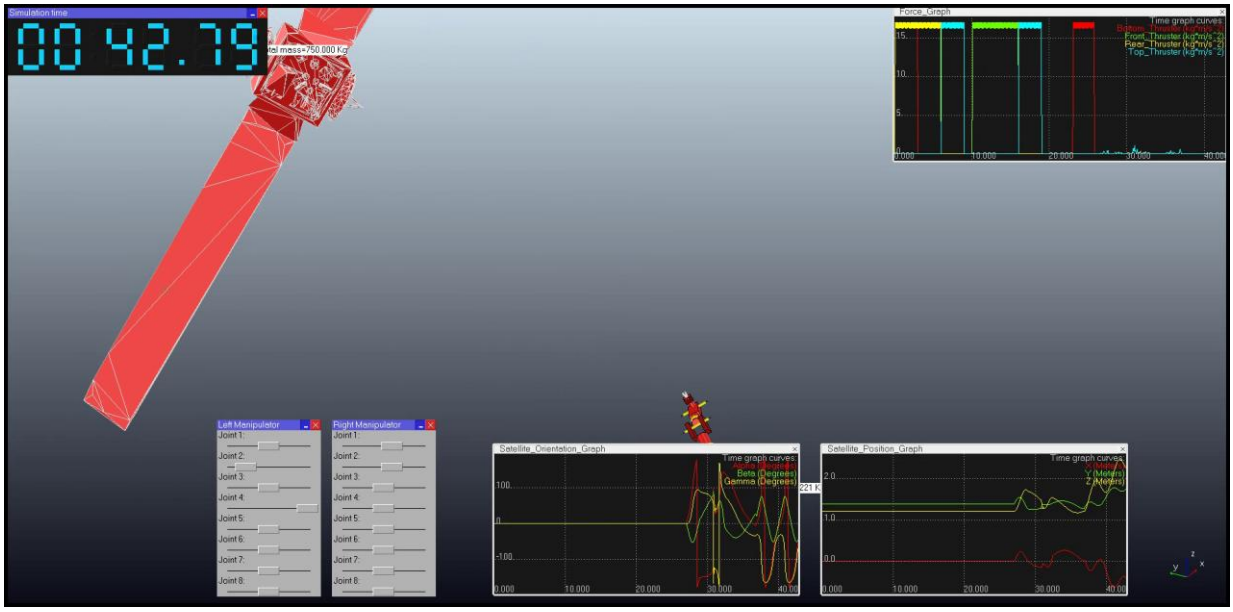

Simulation time
00 38.75
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph

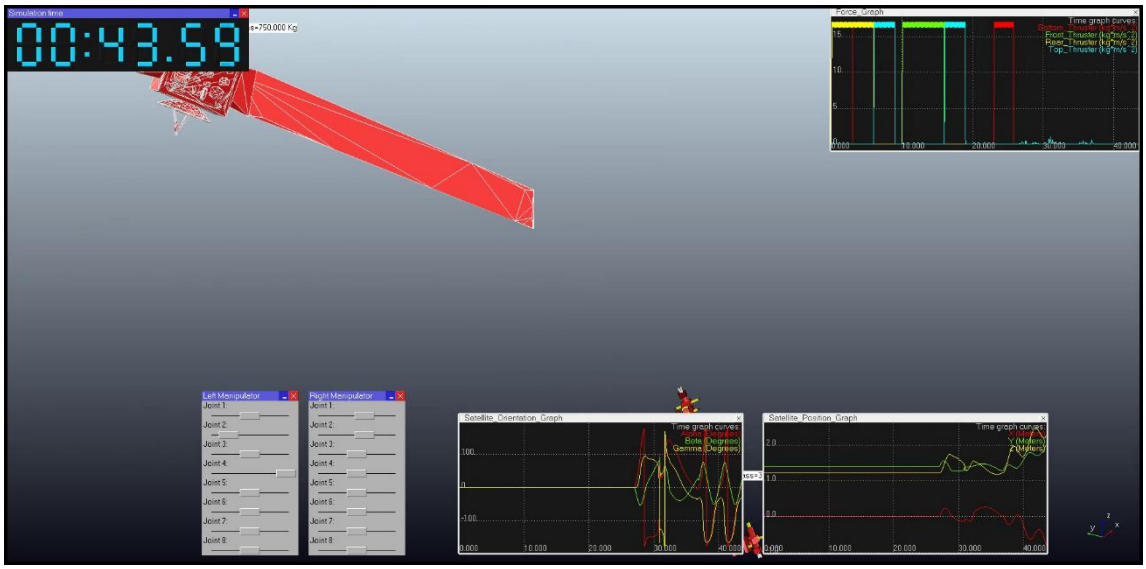

Simulation time
00:39.54
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph


Simulation time
00 40.40
Total mass=750.000 Kg
Total mass=353.221 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph


Simulation time
00:41.20
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Total mass=353.221 Kg
Satellite_Orientation_Graph
Satellite_Position_Graph


Simulation time
00 42.00
Total mass=750.000 Kg
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph


Simulation time
00 42.79
Force_Graph
Left Manipulator
Right Manipulator
Satellite_Orientation_Graph
Satellite_Position_Graph



5. ANÁLISIS ECONÓMICO DEL TFG.

Haciendo un análisis económico teniendo en cuenta las horas de trabajo empleadas, en el caso de estar usando licencias no educativas, y contando con horas de formación en cursos dedicados a V-REP y SolidWorks, se plantea la siguiente tabla que podría representar un posible balance económico del TFG realizado:

DESCRIPCIÓN	UDS	PRECIO	% DTO	TOTAL
Licencia SolidWorks Premium **[27]**	1	7500 €	20	6.000 €
Licencia V-REP PRO **[28]**	1	3000 €	10	2.700 €
Curso de formación SolidWorks	1	900 €	30	630 €
Curso de formación V-REP	1	500 €	25	375 €
Mano de obra (1 operario)	10€ x 8h x 26 días x6 meses	12480€	5	11.856 €
		TOTAL BRUTO		21.561€
	IVA		21%	4.527,81 €
TOTAL PRESUPUESTO……				**26.088,81 €**

Tabla 4. Análisis económico del TFG.

6. CONCLUSIONES.

La validación de la simulación se ha hecho efectiva sin inconveniente alguno, pues la configuración de la jerarquía de escena así como los parámetros que rigen el entorno de simulación en V-REP no ha notificado error alguno ya que se nos ha indicado el icono de validación dinámica (ilustración 21) junto a cada objeto habilitado dinámicamente.

Es cierto que ha habido falta de datos para poder realizar una simulación real, puesta que nos hemos basado en fuentes diversas y ninguna en concreto, es decir, en ninguna situación expresa con ambos satélites y datos dinámicos específicos de cada uno y las circunstancias del entorno.

No obstante, se puede afirmar que los datos, aunque dispersos, son reales, dispersos, pero de fuentes originales y fiables; por tanto, es posible haber tenido una aproximación acertada en la simulación ejecutada.

Cabe decir que los momentos de inercia son los únicos datos que no se han podido corroborar, puesto que se han obtenido de forma experimental en función de las necesidades acaecidas para poder responder ante la simulación con una estabilidad dinámica aceptable.

La inercia del satélite cliente dista mucho del satélite de servicio, en la simulación vuela muy ligero ante su interacción, y es posible que no se haya tomado dichos datos correctamente.

En conclusión, se ha obtenido una simulación coherente respecto a los datos aportados, pero quizás diste de lo que realmente sería puesto que se ha desarrollado en ausencia clara de ellos.

7. BIBLIOGRAFÍA.

[1] A. Farhad, «A Prediction and Motion-Planning Scheme for Visually Guided Robotic Capturing of Free-Floating Tumbling Objects With Uncertain Dynamics.,» *IEEE Transactions on Robotics*, vol. 28, nº 3, p. 16, 2012.

[2] E. Marín, «Blogthinkbig.com,» Telefónica, 12 Julio 2014. [En línea]. Available: http://blogthinkbig.com/vida-util-satelites/. [Último acceso: 20 Noviembre 2016].

[3] Ó. D. de Cózar Macías, «EPS - Trabajo fin de Grado,» 16 Noviembre 2015. [En línea]. Available: www.uma.es/media/tinyimages/file/Reglamento_TFG_-_EPS_-_Ed_2015-11-16.pdf. [Último acceso: 20 Noviembre 2016].

[4] C. t. A. 1. -. Proyectos, «UNE 157001:2014,» 18 Junio 2014. [En línea]. Available: http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0052985. [Último acceso: 20 Noviembre 2016].

[5] «V-REP User Manual,» Coppelia Robotics, [En línea]. Available: http://www.coppeliarobotics.com/helpFiles/index.html. [Último acceso: 7 Noviembre 2016].

[6] R. Ierusalimschy, L. H. de Figueiredo y W. Celes, «Manual de Referencia de Lua,» PUC-Rio, 2008. [En línea]. Available: https://www.lua.org/manual/5.1/es/manual.html. [Último acceso: 13 Noviembre 2016].

[7] R. Ierusalimschy, Programming in Lua Third Edition, Río de Janeiro: PUC-Rio, 2013.

[8] «Regular API function list (alphabetical order),» Coppelia Robotics, [En línea]. Available: http://www.coppeliarobotics.com/helpFiles/en/apiFunctionListAlphabetical.htm. [Último acceso: 13 Noviembre 2016].

[9] «SolidWorks,» Dassault Systèmes, [En línea]. Available: http://www.solidworks.es/sw/products/3d-cad/solidworks-premium.htm. [Último acceso: 7 Noviembre 2016].

[10] «2014 Ayuda de SOLIDWORKS,» Dassault Systèmes, [En línea]. Available: http://help.solidworks.com/2014/spanish/SolidWorks/sldworks/c_introduction_toplevel_topic.htm. [Último acceso: 20 Noviembre 2016].

[11] J. D. Bethune, Engineering Design and Graphics with SolidWorks, Columbus: Pearson, 2009.

[12] M. Lombard, SolidWorks 2013 Bible, Indianápolis: John Wiley & Sons, Inc., 2013.

[13] «NSSDCA/COSPAR ID: 2008-049A,» National Aeronautics and Space Administration, [En línea]. Available: http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=2008-049A. [Último acceso: 29 Enero 2016].

[14] «NSSDCA/COSPAR ID: 2004-018A,» National Aeronautics and Space Administration, [En línea]. Available: http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=2004-018A. [Último acceso: 29 Enero 2016].

[15] «VENESAT-1,» Agencia Bolivariana para Actividades Espaciales, [En línea]. Available: http://www.abae.gob.ve/web/VENESAT-1.php. [Último acceso: 29 Enero 2016].

[16] «TDRS Characteristics | NASA,» NASA, 10 Septiembre 2014. [En línea]. Available: https://www.nasa.gov/directorates/heo/scan/services/networks/txt_tdrs_characteristics.html. [Último acceso: 29 Enero 2016].

[17] S. Zen, «Solidworks Solar Panel Tutorial,» YouTube, 5 Febrero 2014. [En línea]. Available: https://www.youtube.com/watch?v=gtM0WlXK2FE. [Último acceso: 1 Febrero 2016].

[18] S. Official, «Solidworks Dish Drawing,» Youtube, 10 Enero 2015. [En línea]. Available: https://www.youtube.com/watch?v=wipEagRydgM. [Último acceso: 1 Febrero 2016].

[19] «LWR & Taskboard Workcell,» ESA Telerobotics, [En línea]. Available: http://esa-telerobotics.net/gallery/Research/Teleoperation-workcells/LWR-Taskboard-Workcell/16. [Último acceso: 1 Febrero 2016].

[20] J. R. C. Ruiz, «Curso de SolidWorks 2014 - Tutorial de Solidworks 2014 en español | DVC-Resorte de paso variable,» Youtube, 26 Octubre 2013. [En línea]. Available: https://www.youtube.com/watch?v=k4LHT82deiw. [Último acceso: 1 Febrero 2016].

[21] «Dexterous Manipulators and Advanced Control Systems,» Robotics Research Corporation, [En línea]. Available: http://www.robotics-research.com/. [Último acceso: 20 Noviembre 2016].

[22] A. Barrientos, L. F. Peñín, C. Balaguer y R. Aracil, Fundamentos de Robótica. 2ª Edición., Madrid: McGraw-Hill, 2007.

[23] «The Telerobotics Laboratory,» Jet Propulsion Laboratory California Institute of Technology, [En línea]. Available: http://www-robotics.jpl.nasa.gov/facilities/facility.cfm?Facility=3. [Último acceso: 20 Noviembre 2016].

[24] Á. L. Armesto, Prácticas de Robótica Articulada V-REP, Valencia: Tirant Lo Blanch, 2015.

[25] B. S. P. a. Systems, «Busek Green Monoprop Thrusters,» 2016. [En línea]. Available: http://www.busek.com/index_htm_files/70008517E.pdf. [Último acceso: 20 Noviembre 2016].

[26] M. Inc., «Monopropellant Thrusters,» 2013. [En línea]. Available: http://www.moog.com/literature/Space_Defense/Spacecraft/Propulsion/Monopropellant_Thrusters_Rev_0613.pdf. [Último acceso: 20 Noviembre 2016].

[27] «SolidWorks Price Comparison,» Computer Aided Technology, 2016. [En línea]. Available: http://www.catistore.com/solidworks.html. [Último acceso: 21 Noviembre 2016].

[28] «Licensing,» Coppelia Robotics GmbH, 2016. [En línea]. Available: http://www.coppeliarobotics.com/helpFiles/index.html. [Último acceso: 21 Noviembre 2016].

ANEXO A. CÓDIGO LUA DEL BRAZO ROBÓTICO.

```
1. if (sim_call_type==sim_childscriptcall_initialization) then
2. Put some INITIALIZATION code here
3. AQUÍ SE DEFINE Y ASOCIA LOS MANEJADORES
4. ------------------------------------------------------------------------
5. --CONTROL DE USUARIO
6. if (simGetScriptExecutionCount()==0) then
7. ctrlID=simGetUIHandle("UI")
8. sliderID={4,6,8,10,12,14,16,18}
9. end
10. --ARTICULACIONES
11. artID={}
12. for i=1,8,1 do
13. artID[i]=simGetObjectHandle('redundantRob_joint' .. i)
14. end
15. qMin={}
16. qMax={}
17. rango={}
18. --Definimos la tabla de "rango" ya que en la fórmula de más adelante: rango[i]=qMax[i]-
    qMin[i].
19. print('\n#########################\nLIMITES ARTICULACIONES #0. Posicion
    ciclica (true) o no ciclica (false).' .. '\n')
20. --Los valores devueltos se expresan en radianes.
21. for i=1,#artID,1 do
22. cyclic,interval=simGetJointInterval(artID[i])
23. qMin[i]=interval[1]
24. qMax[i]=interval[1]+interval[2]
25. rango[i]=interval[2]
26. print('q' .. i ..':')
27. print(cyclic)
28. print('rango=' .. math.deg(rango[i]) .. ' deg')
29. print('qMin=' .. math.deg(qMin[i]) .. ' deg' .. ',' .. 'qMax=' .. math.deg(qMax[i]) .. ' deg' .. '\n')
30. end
31. --VALORES DE PARTIDA DE LOS SLIDERS
32. print('\n#########################\nSLIDERS #0. Valores de partida.' .. '\n')
33. qM={}
34. q={}
35. for i=1,#sliderID,1 do
36. slider_i = simGetUISlider(ctrlID,sliderID[i])
37. qM[i]=slider_i
38. q[i]=((slider_i/1000)*(rango[i]))+qMin[i]
39. --print('q' .. i .. '=' .. q[i])
```

```
end
--Comprobación de los datos almacenados
print('\nValores adaptados:\n')
for i=1,#q,1 do
print( 'q' .. i .. '=' .. math.deg(q[i]) .. ' deg')
end
print('\nValores correspondientes de 1 a 1000:\n')
for i=1,#qM,1 do
print( 'qM' .. i .. '=' .. qM[i])
end
-----------------------------------------------------------------------
end
if (sim_call_type==sim_childscriptcall_actuation) then
Put your main ACTUATION code here
AQUÍ SE DEFINE LA POSICIÓN QUE TOMARÁ LOS MANEJADORES EN FUNCIÓN DE CIERTOS PARÁMETROS
-----------------------------------------------------------------------
--VINCULACIÓN DE LOS SLIDERS
for i=1,#sliderID,1 do
slider_i = simGetUISlider(ctrlID,sliderID[i])
q[i]=((slider_i/1000)*(rango[i]))+qMin[i]
simSetJointPosition(artID[i], q[i])
end
-----------------------------------------------------------------------
end
if (sim_call_type==sim_childscriptcall_sensing) then
Put your main SENSING code here
end
if (sim_call_type==sim_childscriptcall_cleanup) then
Put some restoration code here
end
```

ANEXO B. CÓDIGO LUA DEL PROPULSOR.

```
1.  if (sim_call_type==sim_childscriptcall_initialization) then
2.      -- Make sure we have version 2.4.13 or above (the particles are not supported otherwise)
3.      v=simGetInt32Parameter(sim_intparam_program_version)
4.      if (v<20413) then
5.          simDisplayDialog('Warning','The propeller model is only fully supported from V-REP version 2.4.13 and above.&&nThis simulation will not run as expected!',sim_dlgstyle_ok,false,'',nil,{0.8,0,0,0,0,0})
6.      end
7.
8.      propeller=simGetObjectHandle('propeller')
9.      propellernonRespondable=simGetObjectHandle('propeller_non_respondable')
10.     propellerJoint=simGetObjectHandle('propeller_joint')
11.
type=sim_particle_roughspheres+sim_particle_respondable1to4+sim_particle_respondable5to8+
12.         sim_particle_cyclic+sim_particle_ignoresgravity
13.     simulateParticles=simGetScriptSimulationParameter(sim_handle_self,'simulateParticles')
14.
particleCountPerSecond=simGetScriptSimulationParameter(sim_handle_self,'particleCountPerSecond')
15.     particleDensity=simGetScriptSimulationParameter(sim_handle_self,'particleDensity')
16.     particleVelocity=simGetScriptSimulationParameter(sim_handle_self,'particleVelocity')
17.
particleScatteringAngle=simGetScriptSimulationParameter(sim_handle_self,'particleScatteringAngle')
18.     particleLifeTime=simGetScriptSimulationParameter(sim_handle_self,'particleLifeTime')
19.
maxParticleCount=simGetScriptSimulationParameter(sim_handle_self,'maxParticleCount')
20.     StartTime=simGetScriptSimulationParameter(sim_handle_self,'StartTime')
21.     FinalTime=simGetScriptSimulationParameter(sim_handle_self,'FinalTime')
22.     if (simGetScriptSimulationParameter(sim_handle_self,'particlesAreVisible')==false) then
23.         type=type+sim_particle_invisible
24.     end
25.
maxParticleDeviation=math.tan(particleScatteringAngle*0.5*math.pi/180)*particleVelocity
26.     particleObject=nil
27.     is=0 -- previous size factor
28.     notFullParticles=0
29.     params={2,1,0.2,3,0.4}
30. end
```

```
31.
32. if (sim_call_type==sim_childscriptcall_cleanup) then
33.
34. end
35.
36.
37. if (sim_call_type==sim_childscriptcall_actuation) then
38.     local t=simGetSimulationTime()
39.
40.     s=simGetObjectSizeFactor(propeller) -- current size factor
41.     if (s~=is) then
42.         if (particleObject) then
43.             simRemoveParticleObject(particleObject)
44.             particleObject=nil
45.         end
46.
    particleSize=simGetScriptSimulationParameter(sim_handle_self,'particleSize')*0.005*s
47.         is=s
48.     end
49.
50.     ts=simGetSimulationTimeStep()
51.
52.     if (particleObject==nil)and(simulateParticles) then
53.
    particleObject=simAddParticleObject(type,particleSize,particleDensity,params,particleLife
    Time,maxParticleCount,{1,0,0})
54.     end
55.
56.     m=simGetObjectMatrix(propeller,-1)
57.     particleCnt=0
58.     pos={0,0,0}
59.     dir={0,0,1}
60.
61.     requiredParticleCnt=particleCountPerSecond*ts+notFullParticles
62.     notFullParticles=requiredParticleCnt % 1 --Returns the value of the variable minus its
    integer value.
63.     requiredParticleCnt=math.floor(requiredParticleCnt) --Returns the highest rendered
    integer value of the variable.
64.
65.     --Movement of propeller.
66.     if (FinalTime>t and t>StartTime) then
67.     simSetJointPosition(propellerJoint,t*10)
68.     end
69.
70.     --Particle object is added.
71.     if (FinalTime>t and t>StartTime) then
72.         while (particleCnt<requiredParticleCnt) do
73.             -- we want a uniform distribution:
74.             x=(math.random()-0.5)*2
75.             y=(math.random()-0.5)*2
```

```
76.          if (x*x+y*y<=1) then
77.              if (simulateParticles) then
78.                  pos[1]=x*0.08*s
79.                  pos[2]=y*0.08*s
80.                  pos[3]=-particleSize*0.6
81.                  dir[1]=pos[1]+(math.random()-0.5)*maxParticleDeviation*2
82.                  dir[2]=pos[2]+(math.random()-0.5)*maxParticleDeviation*2
83.                  dir[3]=pos[3]-particleVelocity*(1+0.2*(math.random()-0.5))
84.                  pos=simMultiplyVector(m,pos)
85.                  dir=simMultiplyVector(m,dir)
86.                  itemData={pos[1],pos[2],pos[3],dir[1],dir[2],dir[3]}
87.                  simAddParticleObjectItem(particleObject,itemData)
88.              end
89.              particleCnt=particleCnt+1
90.          end
91.      end
92.  end
93.  -- Apply a reactive force onto the body:
94.  if (FinalTime>t and t>StartTime) then
95.
totalExertedForce=particleCnt*particleDensity*particleVelocity*math.pi*particleSize*part
icleSize*particleSize/(6*ts)
96.  force={0,0,totalExertedForce}
97.  -- Make sure we take only the rotational part of the matrix into account:
98.  m[4]=0
99.  m[8]=0
100.     m[12]=0
101.     force=simMultiplyVector(m,force)
102.     torque={0,0,0.002*particleVelocity}
103.     torque=simMultiplyVector(m,torque)
104.     simAddForceAndTorque(propellernonRespondable,force,torque)
105.     end
106.
107.     -- Removes a previously added particle object.
108.     if (t>FinalTime) then
109.     simRemoveParticleObject(particleObject)
110.     particleObject=nil
111.     end
112.
113.  end
```

ANEXO C. CÓDIGO LUA DEL MODELO POP-UP.

```
1. threadFunction=function()
2. simDisplayDialog("Readme",txt,sim_dlgstyle_ok,true)
3. end
4.
5. -- Put some initialization code here:
6. simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching
7.
8. -- CUADRO DE DIALOGO
9.
10. --[[ Si quieres utilizar en la cadena las comillas que usas para asignar el string tienes que
11.      usar secuencias de escape. Las sentencias de escape son las mismas que se usan en C.
12. ]]
13. txt="In \"Users settings\" disable the option \"Hide console window\" so as to look at&&n"
14. txt=txt.."characteristics robot arms. Once it start the simulation, read which the console&&n"
15. txt=txt.."shows."
16.
17. -- BARRA DE ESTADO
18. txtc="Final Degree Project entitled:\n"
19. txtc=txtc.."\"Implementation of a weightlessness simulation environment to perform
    interaction between satellites and robots\"."
20. simAddStatusbarMessage(txtc)
21.
22.
23. -- Here we execute the regular thread code:
24. res,err=xpcall(threadFunction,function(err) return debug.traceback(err) end)
25. if not res then
26.     simAddStatusbarMessage('Lua runtime error: '..err)
27. end
```

UNIVERSIDAD DE MÁLAGA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

CREACIÓN DE ENTORNO DE SIMULACIÓN EN INGRAVIDEZ CON INTERACCIÓN ENTRE SATÉLITE Y ROBOT

La aprobación de este trabajo permite la obtención del título:

Ingeniería en Diseño Industrial y D.P.

Autor: RAFAEL CERVILLA RIVAS

Tutor: CARLOS JESÚS PEREZ DEL PULGAR MANCEBO

Cotutor -

Diciembre de 2.016