Framework de control de movimiento de plataforma de de seis grados de libertad de tipo Stewart, servicios de movimiento para vehículos en aplicaciones de simulación e integración con simuladores de vuelo

Santiago Maraggi[1], Horacio Abbate[1]

[1] CITEDEF, Instituto de Investigaciones Científicas y Técnicas para la Defensa, San Juan Bautista de La Salle 4397, B1603ALO, Villa Martelli, Buenos Aires, Argentina

{smaraggi, habbate}@citedef.gob.ar

http://www.citedef.gob.ar/

Resumen. El presente trabajo consiste en el desarrollo de un framework para control de movimiento de una plataforma de seis grados de libertad de tipo Stewart. El framework incluye además un módulo de servicios de movimiento para vehículos y un plugin de integración para simuladores de vuelo, probado con X-Plane 11 y FlightGear. El framework facilita la incorporación de movimiento sobre la plataforma como un servicio a simuladores y programas de todo tipo que involucren movimiento de vehículos.

1 Introducción

La aplicación de simuladores para entrenamiento y evaluación de desempeño en el manejo de vehículos requiere la generación de diferentes estímulos que recreen variadas sensaciones relevantes a fin de aprender y ejecutar las respuestas adecuadas a cada situación. El montaje de un simulador sobre una superficie móvil permite estimular la propiocepción del usuario acorde a la realidad simulada, más allá de la percepción visual y auditiva. Los estímulos de movimiento más relevantes a los fines de una simulación que sufren los tripulantes a bordo de un vehículo son las aceleraciones lineales y giros, particularmente notorios en aeronaves como aviones o helicópteros, pero también muy presentes en vehículos náuticos o terrestres como lanchas, camiones, ferrocarriles, tanques, vehículos blindados y automóviles, y, en menor grado, también en dispositivos móviles como grúas o elevadores.

Una "Plataforma de Stewart" [1] permite mover una superficie plana de dimensiones determinadas con seis grados de libertad: rotaciones y desplazamientos en los tres ejes, guiñada, cabeceo y alabeo y ascenso, abatimiento y avance. La plataforma es sostenida por seis actuadores prismáticos fijados en su extremo inferior al suelo y en la parte superior a la plataforma móvil mediante respectivos puntos de montaje articulados. El diseño de Stewart es suficiente a los fines de gran cantidad de aplicaciones y tiene limitaciones intrínsecas respecto de algunas muy exigentes, como podría ser la reproducción cinemática del vuelo de un avión caza de combate. La posibilidad de incorporar a un simulador, que recree estímulos visuales y sonoros, una plataforma de Stewart, permite expandir el alcance del realismo en una nueva dimensión y lograr una mayor correlación entre la simulación y la realidad para la que se entrena al piloto o conductor del vehículo.

Estos trabajos se desarrollaron en el marco del proyecto SIMMOV, financiado por el Ministerio de Defensa (PIDDEF 03/14), a cargo del Ing. Horacio Abbate, a fin de ampliar las capacidades de CITEDEF respecto de la posibilidad de controlar y utilizar en distintos contextos la plataforma de movimiento en cuestión.

2 Desarrollo de software

Los programas aquí presentados se desarrollaron sobre C++ y se han compilado para plataformas de 32 y 64 bits. El código de programa utiliza las facilidades brindadas por el estándar C++11 del lenguaje que permite escribir código con servicios de tiempo, concurrencia, hilos y varios más compatibles tanto para sistemas operativos Windows como Linux, Mac, entre otros. Se utilizó también la biblioteca multiplataforma Asio[2] de Boost[3] para el envío y recepción de paquetes UDP y la gestión del socket a tal fin. La documentación del framework se confeccionó con la herramienta Doxygen [4], que permite generarla en formatos LaTex y HTML, entre otros, y con diagramas UML. Para el control de versiones se utilizó git [5], con un servidor local como repositorio de código.

El software desarrollado consiste básicamente en un controlador primario para la plataforma de Stewart de 6 grados de libertad disponible en CITEDEF, marca MOOG [6], modelo 6DOF2000E 170-131, con una capacidad de carga de 1 tonelada sobre la que eventualmente se montará una cabina para simuladores de vuelo. Por sobre el controlador primario de la plataforma, que implementa un protocolo UDP específico del producto mencionado, se desarrolló un módulo de movimiento para vehículos que genera movimientos sobre una plataforma a partir de parámetros de aceleraciones lineales y velocidades de giro en los tres ejes. Como última capa de software se desarrolló un módulo de integración o plugin, que con apenas una variante permite la integración de los simuladores de vuelo X-Plane 11 y FlightGear indistintamente.

Con los dos simuladores mencionados se realizaron satisfactorias pruebas de integración así como se implementaron tests de software en distintos niveles y archivos de log por sesión de trabajo.

3 Controlador Plataforma Moog 6DOF2000E 170-131

Este controlador implementa una máquina de estados que replica los estados de la máquina de estados interna de la plataforma especificada por el fabricante. El controlador garantiza que las transiciones solamente ocurran entre estados válidos según esta especificación y separa las piezas de código que corresponden a la entrada, salida y permanencia de los distintos estados. La máquina de estados se implementa naturalmente con el patrón de diseño de software State [7], que atomiza al máximo las piezas de código correspondientes a cada estado y transición y aproxima al máximo en este caso la implementación y el modelo de dominio del problema estudiado. El cliente opera con la plataforma a través de una única clase denominada PlatformController o a través de su interfaz, denominada IPlatformController. Esta interfaz brinda una sencilla colección de métodos de alto nivel para un usuario que pretenda utilizar la plataforma sin inmiscuirse en la complejidad interna administrada por este controlador, en particular relacionada al protocolo de mensajes, la forma en que deben enviarse y recibirse los mismos, las secuencias válidas de envío y recepción y la interpretación de los contenidos de las respuestas recibidas desde la plataforma.

Cada uno de los 9 estados posibles de la plataforma tiene una única clase asociada en el controlador, lo mismo que cada una de las 20 transiciones. Si bien se genera de esta forma una cantidad de clases mayor, se aisló lo más posible cada bloque de operaciones sobre la plataforma, correspondiente a las distintas transiciones y operaciones sobre distintos estados, de manera de simplificar la detección de errores y facilitar la interpretación de las respuestas recibidas. Este diseño además se consideró que facilitaría el mantenimiento por utilizar los principios de la programación orientada a objetos como aislación de dependencias, clases de responsabilidad única

[8], [9], etc., frente a distintos posibles escenarios al momento incierto de desarrollo del programa.

Para facilitar la corrección de errores e interpretación y análisis de resultados se implementó un logger asincrónico junto con el controlador de la plataforma. Se optó por esta solución para no afectar la frecuencia de operación del controlador en el envío constante de mensajes a la plataforma, ya que la escritura permanente del archivo de forma sincrónica podría afectar el rendimiento y la regularidad de las comunicaciones, causando una excepción de flujo en la plataforma, que interpreta cualquier interrupción o demora considerable en el flujo de mensajes desde el controlador como un error, ante el cual detiene su movimiento, envía una señal de error y se estaciona suavemente en la posición inicial de reposo (parking). El módulo de log asincrónico genera automáticamente un nombre de archivo con la fecha y hora de ejecución del controlador y registra de forma automática todos los eventos de interés relacionados a la sesión de trabajo, tanto comandos enviados como recepción de mensajes, detección de errores, etc., hasta la finalización de la operación sobre la misma.

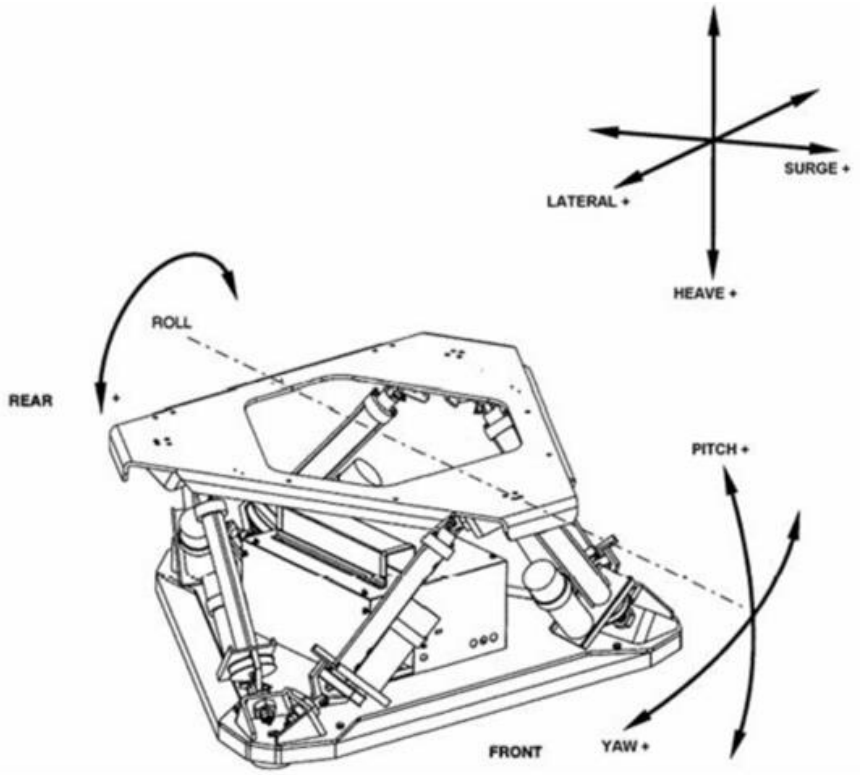

Fig. 1. Grados de Libertad de la plataforma definidos por el fabricante: yaw-pitch-roll-heave-sway-surge (guiñada-cabeceo-alabeo-elevación-abatimiento-avance) [10], [11].

4 Servicios de movimiento para vehículos

Para ensayos de laboratorio más sofisticados o integración de la plataforma con simuladores es deseable que una aplicación pueda utilizar la plataforma dentro de los márgenes especificados por el fabricante en cuanto a desplazamientos absolutos, velocidades y aceleraciones de los distintos grados de libertad y obtener a la vez el mayor rendimiento posible en cuanto a estas prestaciones y con el mayor tiempo de reacción posible.

Si bien el controlador primario de la plataforma desarrollado implementa los servicios necesarios para administrar el movimiento de la misma, delega en el usuario aspectos relacionados a la frecuencia de comunicación y espera instrucciones específicas respecto de los valores absolutos de los grados de libertad que la plataforma debe

tomar, sin tener en cuenta restricciones de velocidad ni aceleración. Es preciso para ofrecer un servicio de alto nivel, en la perspectiva tradicional del diseño de software, una capa de abstracción de nivel superior.

El módulo de servicios de movimiento para vehículos utiliza el controlador de la plataforma primario y consume del exterior las aceleraciones lineales y velocidades de rotación a las que está sujeto un vehículo genérico, de manera de mover la plataforma acorde al comportamiento del vehículo simulado y aprovechando al máximo posible las capacidades de movimiento de la plataforma.

Este módulo de control de movimiento tiene un logger asincrónico propio, similar al del controlador de nivel inferior, y facilita obtener datos relevantes posteriormente a la conclusión de la simulación relacionados a la conversión de parámetros de movimiento del vehículo en movimientos de la plataforma, volcando todo el contenido en un archivo cuyo nombre también incluye la fecha y hora del inicio de sesión de trabajo.

Este controlador de movimiento facilita al cliente inyectar valores de aceleración y rotación dinámicamente en tiempo real, sin necesidad de sincronizarlos con el envío y recepción de mensajes hacia y desde de la plataforma. Para lograr este fin el controlador de movimiento ejecuta una tarea de mantenimiento en un hilo propio que se sincroniza con la aplicación utilitaria para leer o aceptar la modificación de los parámetros dinámicos del vehículo simulado que controlan el comportamiento de la plataforma y el envío de comandos hacia la misma, aislando el mantenimiento del funcionamiento de la plataforma con su alta frecuencia de mensajes de la actualización de los parámetros de movimiento del vehículo, dado que típicamente tendrán diferentes tasas de actualización.

El módulo implementa además el algoritmo de washout, o recentrado automático, mediante la adición constante de una pequeña diferencia hacia el centro de la plataforma, sin importar la situación en que se encuentre en cada momento. Esta adición es despreciable cuando los estímulos de la plataforma son grandes y la llevan suavemente al centro cuando esos estímulos cesan o se atenúan lo suficiente, de manera de mantener a la plataforma, en la mayor medida de lo posible y sin perturbar su desempeño normal, en su centro, que es donde mayor radio de acción y velocidad de respuesta puede brindar.

4.1 Modelado de movimiento de la plataforma para vehículos

Esta solución opera sobre la plataforma mediante un protocolo que permite trabajar con los seis grados de libertad de forma sincronizada, en lugar de trabajar con las extensiones de los actuadores individualmente. En este modo de trabajo la plataforma trabaja con los parámetros de cabeceo, alabeo, guiñada, avance, abatimiento y elevación, mientras que el simulador o aplicación cliente provee las aceleraciones lineales del avión y las velocidades de rotación en cada uno de los tres ejes cartesianos.

Según Berger [12], lograr la mejor correspondencia entre la simulación de un vehículo y el movimiento de una plataforma de Stewart implica un proceso de perfeccionamiento heurístico en el que incluso es un recurso habitual la realización de encuestas a pilotos en distintas simulaciones para evaluación del realismo obtenido como valoración subjetiva y ponderando las distintas percepciones, de manera de ir perfeccionando el grado de realismo logrado a través de los movimientos.

La solución implementada y descripta en este apartado se inspiró principalmente en consideraciones generales obtenidas del manual de la plataforma Moog [11] y la publicación de Berger [12]. Parece haber cierto consenso respecto de prácticas comunes para simular movimiento de vehículos en plataformas móviles de tipo Stewart, así como también algunas discordancias, como por ejemplo entre Berger [12]

y Groen [13], respecto de la optimización de las políticas de movimiento para lograr el mayor realismo posible.

Se implementó un algoritmo de movimiento para este tipo de plataformas acorde: primero se determina el valor deseado para cada grado de libertad que se corresponda mejor con la situación actual de los parámetros de movimiento del vehículo, luego, para llegar a ese valor deseado, se tiene en cuenta la posición, velocidad y aceleración actual de cada grado de libertad, valores que se calculan mediante una ponderación de los últimos estados de la plataforma, luego se determina el signo de incremento (o decremento) de cada grado de libertad necesario, y dadas las restricciones de máxima velocidad y aceleración especificadas por el fabricante, para cada grado de libertad, se determina una velocidad de avance que permita establecer el próximo cuadro de situación a enviar como parámetro a la plataforma. En esta implementación se adiciona una constante pequeña como algoritmo de washout o recentrado para mantener una tendencia de la misma hacia la posición central.

Los valores deseados de los grados de libertad hacia los que se mueve la plataforma en cada instante no pretenden copiar la actitud del vehículo respecto de su espacio virtual, sino que se busca generar en el tripulante la percepción de las aceleraciones lineales y velocidades de giro como resultado de los distintos movimientos, para lo cual se aplica una coordinación de giros en la plataforma para emular las aceleraciones lineales frontal y lateral que se desea hacer percibir al tripulante, en combinación con una velocidad de giro en cada eje proporcional a la velocidad de giro del vehículo simulado en ese eje, junto con algunos movimientos lineales cortos que puedan aumentar el realismo percibido, en particular para la aceleración lineal vertical. Los resultados finales en cuanto al realismo de estos movimientos son óptimos cuando el tripulante opera en una cabina cerrada que brinde los estímulos visuales correspondientes y aísle al tripulante del exterior, ya que se combinan las percepciones visuales exclusivamente inherentes a la simulación (de las cuales la propia cabina forma parte también) con la propiocepción estimulada por la plataforma de movimiento, lográndose en general de esta manera inducir al cerebro una percepción alterada respecto de la propia orientación espacial.

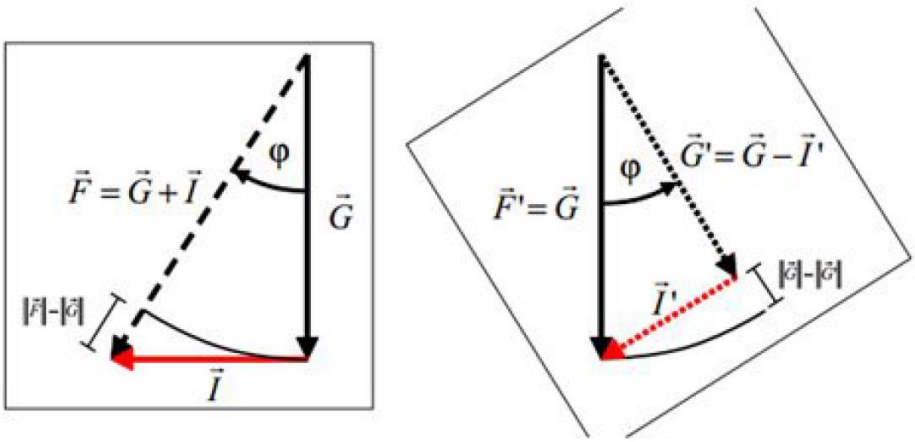

Fig. 2. Coordinación de giro a fin de alinear la gravedad *G* con la aceleración resultante percibida por el piloto *F* ante una aceleración lineal en el plano horizontal *I* [12].

El módulo contempla además la ocurrencia de errores en la plataforma y que son reportados por el controlador primario de la plataforma, en caso de ocurrir. Ante un error la plataforma detiene sus movimientos y se ubica en posición de estacionamiento suavemente de forma automática.

5 Integración con simuladores de vuelo

La última capa para la utilización del servicio de movimiento de la plataforma como un servicio consiste en el desarrollo de un plugin específico para captura de parámetros de movimiento de vehículos. Para el presente trabajo se desarrolló una aplicación con dos variantes apenas diferentes que permiten integrar los simuladores de vuelo XPlane-11 [14] y FlightGear [15] con los servicios de movimiento de la plataforma para vehículos.

Esta aplicación consiste en un socket UDP que recibe datagramas específicos del simulador en cuestión con los parámetros de interés para la plataforma (aceleraciones lineales y velocidades de giro en los tres ejes). Para la instanciación del socket se utilizó la biblioteca Asio[2] de Boost[3] para mantener la total portabilidad del código entre distintos sistemas operativos. Este programa es mínimo ya que aprovecha los servicios de movimiento para vehículos y el controlador de la plataforma previamente mencionados, de manera que escribir plugins para otros simuladores o programas propios que quieran utilizar de forma similar la plataforma es simple en tanto se puedan extraer del mismo los datos relevantes.

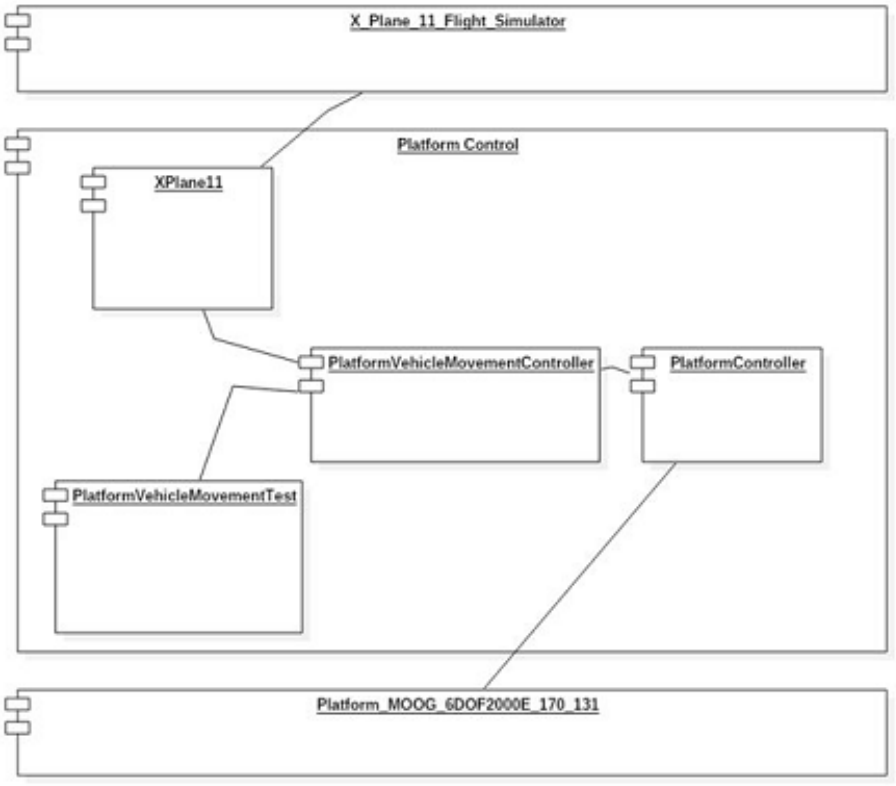

Fig. 3. Diagrama de Componentes que ilustra la relación entre el simulador X-Plane 11, los módulos de software desarrollados y la plataforma de movimiento.

6 Resultados obtenidos

Por el diseño realizado entre los distintos componentes es posible cambiar cualquiera de ellos dado que se mantienen interfaces pequeñas y claras. Esta practicidad es particularmente interesante en el nivel superior, dado que facilita la incorporación de la plataforma de movimiento con cualquier simulador o software que provea los parámetros de movimiento de un vehículo en algún formato conocido, mientras que el controlador y el servicio de movimiento para vehículos aíslan y mantienen coherente la complejidad del desarrollo.

Los archivos de log registran las sesiones de trabajo y los resultados obtenidos en las distintas ejecuciones para posteriores análisis.
Los algoritmos del servicio de movimiento para vehículos están parametrizados con constantes para facilitar ensayos con distintas configuraciones de movimiento, de manera de posibilitar el uso de archivos de texto de configuración externos para su definición y facilitar sucesivos ensayos con variantes sin requerir la recompilación del software, sino apenas el reemplazo o la edición de los mencionados archivos.
Los primeros resultados obtenidos se extrajeron directamente de casos de testing unitario de software desarrollado para comprobar el correcto funcionamiento de piezas de código aisladas. Un nivel más arriba de pruebas de código aisladas se encuentran las pruebas de integración sobre el controlador primario de la plataforma. Estos casos de prueba implementan rutinas de movimiento predefinidas con la plataforma y durante su ejecución se han obtenido las primeras experiencias moviendo la plataforma real en toda su amplitud. Los archivos de log del controlador se han guardado y analizado en las sucesivas iteraciones.
Para comprobar el funcionamiento del módulo de servicio para vehículos de la plataforma también se desarrollaron casos de test específicos, que en vez de inyectar valores de movimiento predefinidos para la plataforma generaban juegos de valores de aceleraciones lineales y velocidades de rotación para comprobar los movimientos correspondientes generados en la plataforma. En estos casos se generaban los archivos de log del controlador primario de la plataforma y log de servicios de movimiento, ya que registran eventos de diferente índole, aunque guardan una correlación natural.
Previo a la integración con simuladores de vuelo también se hicieron pruebas aisladas con los simuladores para comprobar la recepción correcta de los parámetros de interés para el control de los movimientos de la plataforma.

6.1 Resultados obtenidos con simuladores de vuelo

Aunque hay una aparente diferencia entre los dos simuladores utilizados, concretamente FlightGear parece producir movimientos más suaves, mientras que X-Plane 11 parece brindar parámetros de movimiento más intensos, en esencia se puede decir se obtuvo un resultado satisfactorio a los fines propuestos de este trabajo en ambos casos. Se probaron vuelos con distintos tipos de aeronaves, grandes aviones de pasajeros como Boeing 737, aviones militares de distinto tipo como cazas (A-4 Skyhawk, F-4 Phantom), de transporte (C-130 Hércules) y espías (SR-71 Blackbird) como helicópteros (Alouette III, Sikorsky S-76), aunque la mayoría de las pruebas se realizaron sobre el avión monomotor Cessna 172, disponible en ambos simuladores. Cabe mencionar al respecto que se utilizó ampliamente en mayor medida el simulador X-Plane 11, por estar requerido de esta manera.
La complejidad del modelo de vuelo de un simulador, que incluye varias fuerzas cambiantes como el viento y las reacciones de las distintas superficies de la aeronave respecto del flujo de aire que incide en estas, en ocasiones dificulta asociar en forma directa las reacciones de movimiento de la plataforma a algunos estímulos que son difíciles de percibir en la pantalla, en particular cuando son generados por agentes externos al piloto, como por ejemplo turbulencias. Aún con estas complejidades propias del modelo de vuelo se puede apreciar de forma general la correspondencia entre los movimientos de la aeronave y los de la plataforma.
Algunas situaciones en que se aprecia especialmente la correspondencia entre la situación de vuelo y los movimientos de la plataforma son las siguientes: *rodaje* (vibraciones, movimientos por reacción de los amortiguadores al doblar), *carreteo en pista* (vibraciones y cabeceo hacia atrás/arriba por aceleración hacia adelante), *frenada en pista* (vibraciones y cabeceo hacia abajo), *despegue* (desaparecen vibraciones de carreteo, cabeceo acorde y elevación de la plataforma por aceleración

vertical del vehículo), *vuelo* (turbulencias, giros y distintos cambios en las aceleraciones mueven la plataforma como si estuviera flotando), *aterrizaje* (vibraciones y cambios en la elevación de la plataforma), *giro* (cambios en guiñada y altura y rolido, en pugna por cambios en velocidad de rolido del vehículo y aceleración lineal lateral), *giro coordinado* (se puede percibir el equilibrio entre las fuerzas propias del giro si se realiza correctamente) y *giro tonel* (combinaciones de rolido, desplazamientos laterales y otras componentes principalmente causadas por cambios en aceleraciones lateral y vertical, así como velocidad de rolido y de cabeceo del vehículo).

7 Conclusiones

El presente trabajo conforma un framework para la incorporación de movimiento sobre la plataforma de Stewart a cualquier clase de simulador o programa que provea los parámetros de movimiento de un vehículo, pudiendo ser estos vehículos aéreos, terrestres o navales, independientemente del lenguaje de programación, herramientas de desarrollo y entorno, en tanto se escriban los parámetros correspondientes en un socket UDP. El buen diseño de software, los criterios de documentación y testing, el versionado de código en git, la utilización de características incorporadas a C++11 y propiedades de la biblioteca Asio de Boost para la compatibilidad del software en diferentes plataformas dotan a este framework de una gran flexibilidad para su mantenimiento y rápido despliegue e integración en diferentes entornos y con distintos productos, a la vez que facilitan comprobar la calidad en el proceso de desarrollo. Este trabajo agrega valor en el ámbito de la simulación posibilitando la integración de las capacidades de la plataforma subyacente con todo tipo de programas y simuladores, a la vez que abre las puertas para trabajos de investigación para el refinamiento del movimiento de la misma respecto de distintas situaciones de vuelo y operación de vehículos o piezas móviles en un espacio simulado, así como también ensayos de laboratorio que requieran movimientos controlados de precisión.

Referencias

1. Stewart: “A platform with 6 degrees of freedom”
2. http://www.boost.org/doc/libs/master/doc/html/boost_asio.html
3. http://www.boost.org/
4. http://www.doxygen.org
5. https://git-scm.com/
6. http://www.moog.com/
7. ErichGamma, Richard Helm, Ralph Johnson, John Vlissides: “Design Patterns: Elements of Reusable Object-Oriented Software”.
8. Martin, Robert C, ArticleS.UncleBob: "The Principles of OOD" (2005).
9. Martin, Robert C. : “Agile Software Development, Principles, Patterns, and Practices.” (2003 Prentice Hall. p. 95. ISBN 978-01 35974445).
10. Moog 6DOF2000E Model 170-131 Motion System User’s Manual (CDS7238, Rev. 18 de julio de 2008)
11. Moog 6DOF2000E Model 170-131 Motion System Interface Definition Manual (CDS7238, Rev. B 16 de mayo de2008)
12. Daniel R. Berger, Jorg Schulte-Pelkum, Heinrich H. Bulthoff : “Simulating believable forward accelerations on a Stewart motion platform”
13. Eric L. Groen, Mario S. V. Valenti Clari, and Ruud J. A. W. Hosman : “Psychophysical thresholds associated with the simulation of linear acceleration.” AIAA Modeling and Simulation Technologies Conference, Denver (CO), (paper number AIAA 2000-4294), pages 1–9. TNO Human Factors, American Institute of Aeronautics and Astronautics, Inc., 2000
14. http://www.x-plane.com/
15. http://www.flightgear.org