



Instituto Politécnico Nacional
Centro de Estudios Científicos
y Tecnológicos No. 2
“Miguel Bernard”



Opción: Proyecto de investigación

“AIFA EN SOFTWARE LIBRE”

Que para obtener el título de:
Técnico en Aeronáutica

Presenta:

Estrada Salas Carlos Miguel

2022020391

Directores del proyecto.

Director: Ing. Cecilia Sánchez Alvarado
Codirector: Ing. Dafne Daniela Jacobo Dávila

Asesor: M en C. Luis Alberto Loa Ramírez

Ciudad de México, 2024

Resumen

El proyecto "AIFA en Software Libre" tiene como objetivo desarrollar una versión digital del Aeropuerto Internacional Felipe Ángeles (AIFA) utilizando software libre para los simuladores de vuelo X-Plane 11 y FlightGear. Este proyecto no solo busca crear una representación precisa del AIFA, sino también fomentar el uso y la contribución al software libre en el ámbito de la simulación de vuelo.

El desarrollo del proyecto se llevó a cabo en varias fases: recopilación de herramientas, instalación de herramientas, recopilación de datos, diseño y modelado, implementación en simuladores, y un procedimiento detallado para la integración y prueba de los modelos. Se utilizó World Editor (WED) para el diseño y TerraGear para la generación de escenarios.

Los resultados del proyecto indican que se logró modelar el AIFA de manera precisa y adaptarlo a los simuladores utilizando únicamente software libre. Además, se promovió el uso del desarrollo libre al publicar el proyecto en GitHub, facilitando la colaboración y mejoras continuas.

En cuanto a la evaluación de la hipótesis, se confirmó que la implementación del AIFA en simuladores de vuelo de software libre y la promoción de la participación de la comunidad contribuyen significativamente a la mejora de la accesibilidad y la calidad de la formación práctica en aeronáutica.

Se identificaron algunas limitaciones técnicas y operativas, como problemas de compatibilidad entre herramientas y la necesidad de recursos computacionales significativos.

En las consideraciones a futuro, se plantea la implementación de hardware en el plantel educativo para una experiencia completa con FlightGear, la integración total del modelo del AIFA en FlightGear, y la mejora continua del modelo del aeropuerto mediante contribuciones en GitHub.

Abstract

The "AIFA in Free Software" project aims to develop a digital version of the Felipe Ángeles International Airport (AIFA) using free software for the X-Plane 11 and FlightGear flight simulators. This project not only seeks to create an accurate representation of AIFA but also to promote the use and contribution to free software in the field of flight simulation.

The project's development was carried out in several phases: tool collection, tool installation, data collection, design and modeling, simulator implementation, and a detailed procedure for model integration and testing. World Editor (WED) was used for design and TerraGear for scenario generation.

The project results indicate that AIFA was successfully modeled and adapted to the simulators using only free software. Additionally, the project promoted the use of free development by publishing it on GitHub, facilitating collaboration and continuous improvements.

Regarding the hypothesis evaluation, it was confirmed that implementing AIFA in free software flight simulators and promoting community participation significantly contributes to improving accessibility and the quality of practical training in aeronautics.

Some technical and operational limitations were identified, such as compatibility issues between tools and the need for significant computational resources.

Future considerations include implementing hardware in the educational institution for a complete FlightGear experience, fully integrating the AIFA model into FlightGear, and continuously improving the airport model through contributions on GitHub.

Índice

Capítulo 1: Introducción	5
1.1 Antecedentes	6
1.2 Planteamiento del problema	7
1.3 Justificación	8
1.4 Hipótesis	9
1.4.1 Alcances	9
1.4.2 Limitaciones	10
1.5 Objetivo General	10
1.6 Objetivos Específicos	11
Capítulo 2: Marco Teórico	12
2.1 Software libre	12
2.1.1 Filosofía del Software Libre	12
2.1.2 Libertades del Software Libre	15
2.1.3 Licencias de Software Libre	17
2.1.4 Software Libre en la Educación	23
2.1.5 Software Libre en Simulación de Vuelo	29
2.1.6 Contribución al Software Libre	33
2.1.6.1 Git y GitHub: Herramientas para la Colaboración y Distribución de Software Libre	36
2.2 Simuladores de Vuelo	38
2.2.1 FlightGear	38
2.2.2 X-Plane	40
2.2.2.1 Modelo de Vuelo	41
2.2.2.2 Compatibilidad de Hardware	43
2.2.3 Microsoft Flight Simulator 2020	43
2.2.3.1 Limitaciones y Alternativas	43
2.3 AIFA	45
Capítulo 3: Metodología	48
3.1 Descripción General del Proyecto	48
3.2 Fases del proyecto	48
3.2.1 Recopilación de herramientas	48
3.2.2 Instalación de las herramientas	49
3.2.3 Recopilación de datos	49
3.2.4 Diseño y modelado	50
3.2.5 Implementación en los simuladores	51

3.2.6 Pruebas y rendimiento	51
3.3 Publicación y Documentación	52
 Capítulo 4: Resultados y conclusiones	53
4.1 Evaluación de Objetivos	53
4.2 Evaluación de Hipótesis.....	53
4.3 Evaluación de Limitaciones	54
4.3.1 Limitaciones Técnicas	54
4.3.2 Limitaciones Operativas	55
4.3.3 Limitaciones de Colaboración	56
4.4 Conclusiones	56
 Capítulo 5: Consideraciones a Futuro	57
5.1 Hardware	57
5.2 FlightGear	57
5.3 Mejora del AIFA	57

Capítulo I: Introducción

En el ámbito de la formación aeronáutica, la disponibilidad de herramientas educativas efectivas y accesibles es fundamental para garantizar una experiencia de aprendizaje completa y de calidad. Los simuladores de vuelo desempeñan un papel crucial al proporcionar a los estudiantes la oportunidad de adquirir habilidades prácticas. Sin embargo, la adopción de simuladores comerciales puede representar una barrera significativa debido a los altos costos y las limitaciones de acceso.

En respuesta a este desafío, surge la necesidad de explorar alternativas que promuevan la accesibilidad y la flexibilidad en la formación aeronáutica. Este proyecto se enfoca en abordar esta necesidad específica mediante el desarrollo de una simulación del Aeropuerto Internacional Felipe Ángeles (AIFA) para el simulador de vuelo de software libre FlightGear y su adaptación para X-Plane 11.

El AIFA, como un punto focal de estudio en la carrera técnica de aeronáutica, carece actualmente de una representación en los simuladores de vuelo utilizados en el plantel. Esta limitación impacta directamente en los estudiantes, ya que estos no pueden practicar y familiarizarse con los procedimientos y operaciones específicas del aeropuerto, lo que afecta negativamente su formación práctica.

A través de este proyecto, se busca desarrollar una herramienta educativa avanzada y accesible que mejore la formación práctica en aeronáutica, fomente la investigación en simulación de vuelo y promueva la adopción de software libre en el ámbito educativo. Además de satisfacer la necesidad inmediata de contar con una simulación del AIFA en los simuladores que ofrece el plantel, se explorará la compatibilidad y adaptabilidad de la simulación para su futuro uso con el hardware adecuado.

Este proyecto representa un paso significativo hacia el uso de software libre y la colaboración comunitaria en el desarrollo de soluciones innovadoras.

1.1 Antecedentes

El Aeropuerto Internacional Felipe Ángeles, inaugurado el 21 de marzo de 2022, ha sido objeto de interés en la comunidad de simulación aérea. A la fecha presente, se ha publicado un modelo 3D del aeropuerto para el simulador *X-Plane 11*, creado por un entusiasta, no por el equipo oficial de *X-Plane*. Aunque el modelo está muy desactualizado, ya que el AIFA ha cambiando mucho desde entonces.



Figura 1.1: AIFA desactualizado

X-Plane.Org Forum. (2022, 21 marzo). *MX MMSM - Mexico City Felipe Angeles Intl* (2022).

<https://forums.x-plane.org/index.php?/files/file/79882-mx-mmsm-mexico-city-felipe-angeles-intl-2022/%20%20apa>

En otro foro de internet dedicado a *X-Plane*, se publicó una base que sirve como punto de partida para el modelo mencionado anteriormente. Esta base es especialmente útil porque utiliza el mismo formato de archivo que se empleará en el software libre FlightGear. Esto facilita el inicio del trabajo, ya que se puede aprovechar la estructura existente para adaptarla al nuevo simulador.

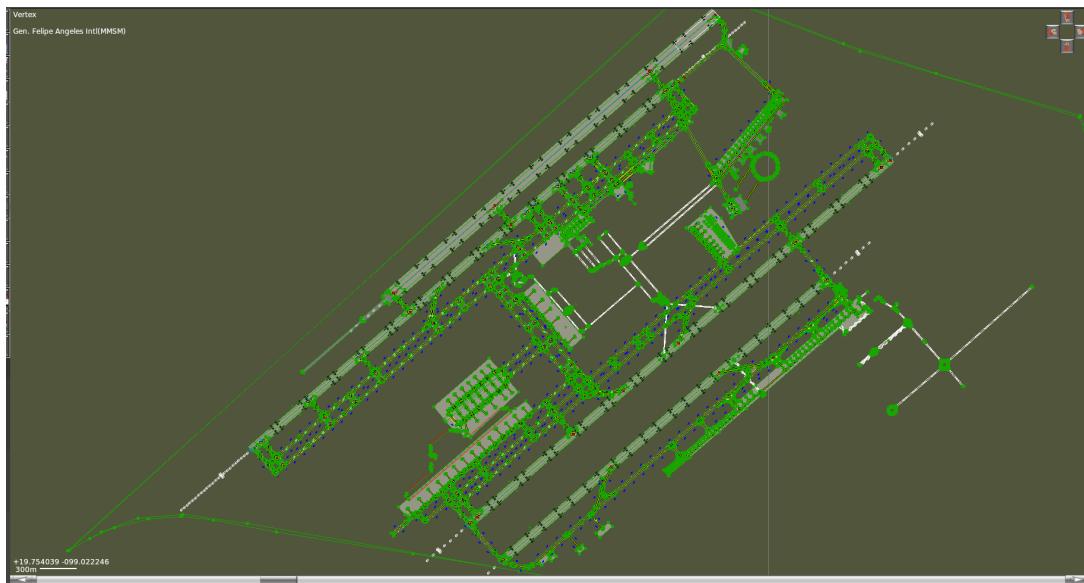


Figura 1.2: Mapeo del AIFA desactualizado

X-Plane Scenery Gateway. (s. f.).

<https://gateway.x-plane.com/airports/MMSM/show>

Un artículo escrito por Richard Stallman, el fundador del movimiento del software libre, aborda la promoción del software libre y por qué los gobiernos deberían hacerlo. Stallman expone cómo los gobiernos pueden optar por fomentar el software libre y argumenta la importancia de este compromiso ético y social.

Medidas que los gobiernos pueden adoptar para promover el software libre - Proyecto GNU - Free Software Foundation. (s. f.).

<https://www.gnu.org/philosophy/government-free-software.es.html>

1.2 Planteamiento del Problema

El Aeropuerto Internacional Felipe Ángeles (AIFA) se ha convertido en un componente crucial del plan de estudios de la carrera técnica de aeronáutica debido a su relevancia y novedad en el ámbito de la aviación. Sin embargo, a pesar de su importancia, los estudiantes se enfrentan a una limitación significativa: la falta de disponibilidad del AIFA en los simuladores de vuelo utilizados en el plantel.

En la actualidad, la única opción disponible para acceder al AIFA es a través de un simulador comercial llamado *Microsoft Flight Simulator 2020* que no se encuentra en uso en el plantel y que requiere de un hardware más potente, superando las capacidades de los equipos proporcionados por la institución. Esta situación dificulta el acceso y la práctica adecuada de los alumnos en el estudio del Aeropuerto Internacional Felipe Ángeles, lo que representa un desafío para su formación en aeronáutica.

Aunque el simulador *X-Plane 11* el cual es el que utiliza el plantel, es de carácter privativo, este ofrece una integración más sencilla con software externo.

En este contexto, surge la necesidad de desarrollar una solución que aborde estas limitaciones y promueva una educación aeronáutica más accesible. Esto implica el desarrollo del Aeropuerto Internacional Felipe Ángeles (AIFA) utilizando software libre para eliminar costos y permitir actualizaciones continuas, además de fomentar el apoyo de la comunidad aeronáutica para su mejora. Asimismo, se busca adaptar este desarrollo para integrarlo con el simulador *X-Plane 11* utilizado en el plantel y explorar la posibilidad de adaptar el hardware de la institución al simulador de *FlightGear* en el futuro.

1.3 Justificación

El desarrollo de una simulación detallada del Aeropuerto Internacional Felipe Ángeles (AIFA) en software libre se fundamenta en la necesidad de abordar las limitaciones existentes en la formación práctica en aeronáutica y promover el uso del software libre. Esta justificación se fundamenta en varios aspectos:

Accesibilidad: La accesibilidad a herramientas educativas de calidad es esencial para garantizar una formación a gran nivel. El desarrollo de una simulación del AIFA es crucial ya que actualmente en el plantel no se ofrece esta característica en los simuladores que se encuentran disponibles, pero al poder desarrollarlo, permite que los estudiantes de aeronáutica tengan acceso de forma libre a herramientas que ayuden a su aprendizaje.

Reducción de costos y barreras: La adopción de software libre y de código abierto en la

educación aeronáutica contribuye a reducir los costos asociados con la adquisición de licencias de software y el uso de soluciones propietarias. Esto facilita el acceso de los estudiantes a herramientas de calidad sin requerir inversiones significativas por parte de las instituciones educativas.

Fomento de la innovación y la colaboración: El uso de software libre promueve la innovación y la colaboración entre la comunidad educativa y los desarrolladores de software. Al desarrollar una simulación del AIFA en, se crea una plataforma abierta para la contribución y mejora continua por aquellos que deseen contribuir al proyecto.

Adaptabilidad y personalización: La naturaleza del software libre permite una mayor adaptabilidad y personalización de la simulación según las necesidades específicas que requiera el usuario. Esto facilita la integración de escenarios de aprendizaje personalizados y la creación de material didáctico complementario que enriquezca la experiencia educativa de los estudiantes.

1.4 Hipótesis

Si se implementa una opción de calidad del Aeropuerto Internacional Felipe Ángeles (AIFA) en simuladores de vuelo de software libre y se promueve activamente la participación de la comunidad en el desarrollo y mejora del proyecto, entonces se espera que este proyecto contribuya significativamente a la mejora de la accesibilidad y la calidad de la formación práctica en aeronáutica. Esto se debe a la capacidad del software libre de adaptarse rápidamente a los cambios en el AIFA, facilitando actualizaciones frecuentes y precisas en el simulador, y a la participación continua de la comunidad que fomenta la innovación y el desarrollo sostenible del proyecto. Además, se espera que este enfoque promueva la adopción y la conciencia sobre el software libre en el ámbito educativo y la comunidad aeronáutica en general.

1.4.1 Alcances

Diseño y desarrollo de la simulación del Aeropuerto Internacional Felipe Ángeles (AIFA) utilizando software libre. Se llevará a cabo una exploración inicial de la compatibilidad y adaptabilidad de la simulación para su posible uso en el simulador *X-Plane 11* en el futuro. Este proyecto no contemplará la integración directa del hardware específico del plantel, como el panel *ELITE Pro Panel II*, con *FlightGear*, aunque se identificarán

posibles vías para futuras investigaciones en esta área. Además, se espera que los resultados obtenidos sienten las bases para investigaciones posteriores y extensiones del proyecto, las cuales podrían abordar aspectos como la mejora continua de la simulación del AIFA y la colaboración con la comunidad para explorar oportunidades adicionales de integración con otros simuladores de vuelo de software libre.

1.4.2 Limitaciones

Exactitud de la simulación: Alcanzar un alto grado de precisión en la simulación del Aeropuerto Internacional Felipe Ángeles (AIFA) y sus operaciones representa un desafío debido a la limitada disponibilidad de información precisa. La información más detallada y actualizada sobre el aeropuerto puede estar sujeta a confidencialidad o puede no estar fácilmente accesible en fuentes públicas. Esta limitación podría afectar la fidelidad de la simulación y requerir estrategias adicionales para obtener datos precisos.

Adaptación del hardware disponible: La integración temprana del hardware disponible en el plantel con el simulador *FlightGear* puede ser limitada debido a restricciones técnicas y de tiempo. La configuración y adaptación del hardware al entorno del simulador pueden requerir recursos adicionales y un proceso de desarrollo más prolongado. Esta limitación podría afectar inicialmente la experiencia de los usuarios y requerir estrategias alternativas para optimizar la funcionalidad del hardware dentro del simulador.

1.5 Objetivo General

Desarrollar e implementar una simulación de alta calidad del Aeropuerto Internacional Felipe Ángeles (AIFA) utilizando exclusivamente herramientas de software libre, basándose en el simulador *FlightGear*, y adaptar esta simulación para su uso con *X-Plane 11*, el simulador empleado actualmente en el plantel. Este enfoque busca mejorar la accesibilidad y la calidad de la formación práctica en aeronáutica, fomentar la participación activa de la comunidad en el desarrollo continuo del simulador, y promover la innovación y adaptabilidad, elevando la conciencia sobre el valor del software libre en la educación aeronáutica y más allá.

1.6 Objetivos Específicos

- Desarrollar una simulación del Aeropuerto Internacional Felipe Ángeles (AIFA) utilizando exclusivamente herramientas de software libre.
- Adaptar la simulación del Aeropuerto Internacional Felipe Ángeles al simulador que utiliza el plantel (*X-Plane 11*).
- Desarrollar material didáctico complementario para el uso del AIFA para los simuladores *FlightGear* y *X-Plane 11*.
- Fomentar la colaboración y el desarrollo comunitario para la mejora continua de la simulación del AIFA.

Capítulo 2: Marco Teórico

2.1 Software Libre

El término "software libre" se refiere a aquel software que respeta la libertad de los usuarios y la comunidad. En esencia, esto implica que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Se centra en la libertad de uso, no en el costo asociado. Para entender esta noción, es útil compararla con la libertad, en contraposición a la idea de algo gratuito. A veces, en inglés se emplea la expresión "libre software" en lugar de "free software", para enfatizar que se trata de libertad y no de precio.

2.1.1 Filosofía del software libre

En su artículo "El software libre es ahora aún más importante," Richard Stallman aboga por la autonomía de los usuarios sobre el software que utilizan desde la fundación del movimiento de software libre en 1983. El término "software libre" se refiere a programas que respetan la libertad de los usuarios y la comunidad. Stallman destaca que este concepto de "libre" se enfoca en la libertad, no en el costo, contrastando con programas propietarios como Photoshop y la aplicación de Uber, que, aunque pueden ser costosos o gratuitos, privan a los usuarios de control y se clasifican como malware por incluir funciones que abusan del usuario.

El artículo señala que el software privativo generalmente actúa en detrimento de los usuarios debido al poder que otorga a los desarrolladores, quienes pueden llegar a corromperse. En contraste, el software libre permite a los usuarios tanto individual como colectivamente, controlar completamente lo que sus computadoras hacen, promoviendo la lealtad y obediencia de las máquinas a sus comandos.

Stallman critica el software privativo que, incluso cuando no es explícitamente malicioso, tiende a ser adictivo y manipulador. Argumenta que los desarrolladores de software privativo enfrentan incentivos que los llevan a comprometer la ética en favor de sus propios intereses. Para evitar estas trampas, Stallman recomienda que los usuarios aseguren que el software esté bajo su control, subrayando que la verdadera

libertad radica en la capacidad de controlar los programas que impactan significativamente en sus vidas.

La problemática del software privativo radica en que si los usuarios no tienen el control del programa, es el programa (y por extensión, sus desarrolladores o propietarios) el que controla a los usuarios. Esto convierte al software privativo en un instrumento de poder desequilibrado y, en muchos casos, injusto.

En situaciones más severas, que ahora son bastante comunes, los programas privativos están diseñados específicamente para espionar, restringir, censurar y maltratar a los usuarios. Ejemplos claros incluyen el sistema operativo de los dispositivos de Apple, así como Windows en dispositivos móviles con chips ARM. Además, programas como Windows, el firmware de teléfonos móviles y Google Chrome en Windows, contienen puertas traseras que permiten a las empresas modificar los programas a distancia sin consentimiento del usuario. Otro ejemplo es el Kindle de Amazon, que tiene la capacidad de borrar libros remotamente.

El uso extendido de software privativo en dispositivos conectados a internet podría transformar el "Internet de las cosas" en un espacio dominado por mercaderes e intrusos.

Para combatir estas injusticias, el movimiento de software libre ha desarrollado y promueve el uso de software que sea verdaderamente libre, permitiendo a los usuarios retomar el control. Este esfuerzo comenzó en 1984 con el desarrollo del sistema operativo GNU, que hoy en día es usado en millones de computadoras, especialmente en la forma de GNU/Linux.

Distribuir software sin ofrecer libertades fundamentales es considerado una forma de maltrato hacia los usuarios. No obstante, si un programa se usa únicamente de manera privada y no se distribuye, esto no constituye un maltrato directo a otros. Aunque se pierde la oportunidad de contribuir positivamente, esto no se equipara con hacer el mal activamente. Por tanto, la exigencia de que todo software sea libre implica que todas

las copias de un programa deben proporcionar las cuatro libertades esenciales, pero no necesariamente que todos deban distribuir sus programas a los demás.

El software libre y la soberanía estatal

Las entidades públicas deben operar en beneficio de los ciudadanos, no en su propio interés. Cuando llevan a cabo tareas informáticas, es esencial que mantengan el control total para asegurar que se ejecuten adecuadamente en favor de la población. Esto es lo que define la soberanía informática de un Estado. Es crucial que el control de las actividades informáticas estatales no se transfiera a entidades privadas.

Para mantener la soberanía sobre las tareas informáticas que realizan para los ciudadanos, los organismos estatales deben evitar el uso de software privativo, que está controlado por partes no gubernamentales. Además, no deben externalizar estas tareas a servicios gestionados fuera del ámbito estatal, ya que esto constituiría lo que se conoce como Software as a Service (SaaS).

El software privativo no solo carece de garantías contra riesgos importantes, sino que su propio desarrollador podría representar un peligro. Por ejemplo, Microsoft comparte detalles sobre vulnerabilidades en Windows con la NSA antes de solucionarlas, lo que implica un riesgo de espionaje. Aunque no está confirmado si Apple sigue la misma práctica, enfrenta presiones similares. Si gobiernos de otros países utilizan este software, podrían estar comprometiendo su seguridad nacional. ¿Es aceptable que la NSA tenga acceso a los sistemas informáticos de su gobierno? Para evitar estos riesgos, recomendamos revisar nuestras propuestas para que los gobiernos fomenten el uso de software libre.

2.1.2 Libertades del software libre

El software libre se basa en 4 libertades llamadas libertades esenciales, estas libertades esenciales son criterios los cuales un software debe cumplir para ser denominado software libre:

1. **Libertad 0:** La libertad de **utilizar** el programa sin restricciones, para cualquier propósito que se deseé. Significa que cualquiera puede utilizarlo ya sea persona u organización, para cualquier propósito, sin necesidad de informar al desarrollador u otra entidad. Lo importante en esta libertad es el propósito del usuario, no el del desarrollador. El usuario puede ejecutar el programa para sus propósitos, y si lo comparte con otros, ellos también tienen la libertad de usarlo para sus propios fines; el usuario no puede imponer sus objetivos a otras personas.
2. **Libertad 1:** La libertad de **estudiar** el funcionamiento interno del programa y modificarlo según las necesidades individuales. Para que las libertades 1 y 3 sean posibles, es necesario tener acceso al código fuente del programa. Por lo tanto, el acceso al código fuente es un requisito esencial para el software libre. El "código fuente" que está enmascarado o no está disponible de manera clara no se considera como verdadero código fuente.

El código fuente se define como la forma preferida del programa para realizar cambios en él. Esto significa que cualquier forma en la que el desarrollador pueda modificar el programa para crear una nueva versión se considera el código fuente de esa versión.

3. **Libertad 2:** La libertad de **compartir** copias del programa con otros para ayudarles. Significa que se tiene la libertad de poder redistribuir la copia del programa sin tener que pedir ni pagar ningún permiso para hacerlo.
4. **Libertad 3:** La libertad de **distribuir** versiones modificadas del programa a terceros. También es importante tener la libertad de realizar modificaciones y utilizarlas de manera privada para su propio trabajo o entretenimiento, sin la obligación de informar a nadie sobre su existencia. En caso de decidir publicar

sus cambios, no está obligado a notificar a ninguna persona o entidad en particular, ni tampoco está sujeto a algún método específico para hacerlo.

Implica la posibilidad de publicar sus versiones modificadas como software libre. Una licencia libre puede permitir otras formas de publicación además del copyleft; sin embargo, una licencia que exija que las versiones modificadas no sean libres no puede considerarse como tal.

La libertad de redistribuir copias debe abarcar tanto las versiones binarias o ejecutables del programa como el código fuente, ya sea que estén modificadas o no. La distribución en forma de ejecutables es necesaria para facilitar la instalación en sistemas operativos libres. Si no existe un método para generar un formato binario o ejecutable para un programa específico, lo cual puede ser el caso en algunos lenguajes de programación, debe tener la libertad de redistribuir estos formatos si encuentra o crea una manera de hacerlo.

CopyLeft

El copyleft es un enfoque utilizado para asegurar que un programa sea considerado software libre, lo que implica que cualquier versión modificada o extendida de dicho programa también debe ser libre.

El crear un software de dominio público, donde podría ser utilizado para crear software privativo, el copyleft garantiza que la libertad de los usuarios se mantenga intacta. En esencia, este método se basa en el uso del copyright para proteger la libertad de los usuarios en lugar de restringirla, asegurando que el código y las libertades asociadas sean inseparables legalmente. La implementación práctica del copyleft se realiza mediante la inclusión de cláusulas de distribución en el código, como se hace en las Licencias Públicas Generales de GNU, que garantizan que todos los usuarios conserven la libertad de copiar, redistribuir y modificar el software.

Esto no solo incentiva la colaboración en el software libre, sino que también disuade a los desarrolladores de convertir el software en productos privativos al hacer que sea ilegal distribuir versiones modificadas a menos que también se liberen como software libre.

2.1.3 Licencias de Software Libre

Las licencias de software libre son instrumentos legales que otorgan a los usuarios ciertos derechos para usar, modificar y distribuir software. Estas licencias se basan en los principios de libertad, transparencia y colaboración.

Las licencias de software libre son importantes porque fomentan la innovación, la colaboración y el intercambio de conocimientos. Permiten que los desarrolladores construyan sobre el trabajo de otros, lo que puede acelerar el desarrollo de software y promover la creación de comunidades de desarrollo activas y colaborativas. Todo esto gracias al CopyLeft

Las principales licencias que se utilizan son la GPL, BSD, Apache y Creative Commons. Cada una de estas licencias tiene sus propias condiciones y requisitos específicos, pero en general, todas garantizan las libertades del software:

- **GNU General Public License**

Es una licencia de derechos de autor que garantiza a los usuarios la libertad de usar, estudiar, compartir (copiar) y modificar el software cubierto por la licencia. además de que es una licencia de software libre con copyleft. Y es recomendable para la mayoría de paquetes de software.A continuación, se detallan algunos puntos clave de la licencia:

1. Definiciones: La licencia define términos como "Programa", "Modificación", "Obra Derivada", "Propagar", entre otros.
2. Código Fuente: La licencia garantiza el acceso al código fuente del software cubierto, lo que permite a los usuarios estudiar cómo funciona y realizar modificaciones.
3. Permisos Básicos: Los usuarios tienen permiso para ejecutar, modificar y propagar el software cubierto, siempre que cumplan con los términos de la licencia.

4. Protección de los Derechos Legales de los Usuarios: La licencia protege los derechos legales de los usuarios frente a leyes que prohíban la circunvención de medidas tecnológicas de protección.
5. Copias Literales: Los usuarios pueden copiar y distribuir el código fuente del software cubierto, siempre que se incluyan ciertas notificaciones de derechos de autor y la licencia.
6. Versiones Modificadas del Código Fuente: Los usuarios pueden distribuir versiones modificadas del código fuente del software cubierto, siempre que se cumplan ciertas condiciones, como incluir avisos sobre las modificaciones realizadas.
7. Conveying Non-Source Forms: La licencia permite a los usuarios distribuir versiones compiladas (código objeto) del software cubierto, siempre que también proporcionen el código fuente correspondiente.
8. Patentes: La licencia incluye disposiciones relacionadas con patentes, otorgando a los usuarios una licencia de patente para el software cubierto.
9. No Rendición de la Libertad de Otros: La licencia establece que si se imponen condiciones adicionales que contradicen los términos de la GPL, el usuario no puede distribuir el software cubierto.
10. Uso con la Licencia Pública General de GNU Afferro: La licencia permite combinar el software cubierto con el software licenciado bajo la Licencia Pública General de GNU Afferro.
11. Versiones Revisadas de la Licencia: La Fundación de Software Libre puede publicar versiones revisadas de la GPL, y los usuarios pueden optar por aplicar los términos de cualquier versión posterior.
12. Renuncia de Garantía y Limitación de Responsabilidad: La licencia incluye una renuncia de garantía y limitación de responsabilidad, indicando que el software se proporciona "tal cual" y que los desarrolladores no son responsables de los daños derivados del uso del software.

La GNU GPL es una licencia ampliamente utilizada en la comunidad del software libre y de código abierto, ya que garantiza la libertad de los usuarios para utilizar, estudiar, compartir y modificar el software cubierto por la licencia.

- **Apache-2.0**

La Licencia Apache es un acuerdo legal que establece los términos y condiciones para el uso, la reproducción y la distribución del trabajo cubierto por la licencia. Aquí hay un desglose de las cláusulas clave:

1. Definiciones: Esta sección establece los términos utilizados en la licencia, como "Licencia", "Licenciante", "Entidad Legal", "Tú", "Forma Fuente", "Forma Objeto", "Trabajo", "Obras Derivadas", "Contribución" y "Contribuyente".
2. Concesión de Licencia de Copyright: Cada contribuyente otorga al usuario una licencia de copyright perpetua, mundial, no exclusiva, sin cargo y libre de regalías para reproducir, preparar Obras Derivadas, mostrar públicamente, ejecutar públicamente, sublicenciar y distribuir el Trabajo y dichas Obras Derivadas en Forma Fuente o Forma Objeto.
3. Concesión de Licencia de Patente: Cada contribuyente otorga al usuario una licencia de patente perpetua, mundial, no exclusiva, sin cargo y libre de regalías para hacer, hacer hacer, usar, ofrecer vender, vender, importar y transferir de otra manera el Trabajo, con la condición de que esta licencia se aplique solo a aquellas reclamaciones de patente licenciables por dicho contribuyente que sean necesariamente infringidas por su(s) Contribución(es) sola(s) o por la combinación de su(s) Contribución(es) con el Trabajo al que se presentó dicha(s) Contribución(es).
4. Redistribución: Se establecen las condiciones bajo las cuales puedes redistribuir el trabajo, incluida la necesidad de proporcionar una copia de la licencia, mantener avisos de copyright y patentes, y distribuir cualquier archivo "NOTICE" incluido.
5. Presentación de Contribuciones: Se especifica que cualquier contribución que envíes para su inclusión en el trabajo se rige por los términos de esta licencia, a menos que especifiques lo contrario.
6. Marcas Comerciales: Se establece que esta licencia no otorga permiso para utilizar nombres comerciales, marcas comerciales, marcas de servicio o nombres de productos del licenciante, excepto en la medida requerida para describir el origen del trabajo.
7. Exención de Garantía: Se establece que el trabajo se proporciona "TAL CUAL" y sin garantías de ningún tipo, a menos que lo exija la ley aplicable o se acuerde

por escrito. El usuario asume todos los riesgos asociados con el ejercicio de los permisos bajo esta licencia.

8. Limitación de Responsabilidad: Se limita la responsabilidad de los contribuyentes por daños y perjuicios, excepto cuando lo exija la ley aplicable o se acuerde por escrito.
9. Aceptar Garantía o Responsabilidad Adicional: Si decides ofrecer apoyo, garantía, indemnización u otras obligaciones y/o derechos al redistribuir el trabajo o sus obras derivadas, debes hacerlo solo en tu propio nombre y responsabilidad, y solo si aceptas indemnizar, defender y eximir de responsabilidad a cada contribuyente por cualquier responsabilidad incurrida o reclamaciones presentadas contra dicho contribuyente debido a tu aceptación de dicha garantía o responsabilidad adicional.

La sección final proporciona instrucciones sobre cómo aplicar la licencia Apache a tu trabajo, lo que implica incluir un aviso específico en tu trabajo que identifique la licencia aplicable y tu información de identificación.

- **BSD-2-Clause-FreeBSD:**

Es una licencia de software libre laxa, permisiva, sin copyleft, compatible con la GPL de GNU. Se recomienda ser cuidadoso cuando se utiliza software que está bajo esta licencia; primero se debe considerar si el licenciatario podría demandar al usuario por infringir una patente. Si el desarrollador se niega a conceder licencias sobre patentes para ponerle una trampa al usuario, sería mejor evitar utilizar el programa

La licencia permite la redistribución y el uso del software, tanto en forma de código fuente como en forma binaria, con o sin modificaciones, siempre que se cumplan ciertas condiciones:

Las redistribuciones del código fuente deben conservar el aviso de derechos de autor, la lista de condiciones y el descargo de responsabilidad proporcionados en la licencia.

Las redistribuciones en forma binaria deben reproducir el aviso de derechos de autor, la lista de condiciones y el descargo de responsabilidad en la documentación y/o en otros materiales proporcionados con la distribución.

La licencia establece que el software se proporciona "TAL CUAL" y que no hay garantías explícitas o implícitas, incluyendo, pero no limitado a, las garantías implícitas de comerciabilidad y adecuación para un propósito particular. Además, el proyecto FreeBSD y sus contribuyentes no son responsables de ningún daño directo, indirecto, incidental, especial, ejemplar o consecuente que pueda surgir del uso del software, incluso si se advierte de la posibilidad de tales daños.

Además, la licencia aclara que las opiniones y conclusiones contenidas en el software y la documentación son las de los autores y no deben interpretarse como representativas de las políticas oficiales, ya sean expresadas o implícitas, del Proyecto FreeBSD.

- **CC0 1.0 Universal**

La licencia CC0, desarrollada por Creative Commons, es una contribución que ofrece un enfoque práctico para el uso de software y documentación. Esta licencia permite que una obra se dedique al dominio público en la medida máxima permitida por la ley. En situaciones en las que no es posible dedicar completamente la obra al dominio público, también sirve como una licencia permisiva que facilita una amplia gama de usos de la obra, sin imponer restricciones significativas sobre cómo puede ser utilizada, distribuida o modificada. Es importante destacar que tanto las obras en dominio público como aquellas licenciadas bajo CC0 son compatibles con la GPL de GNU.

Aunque se recomienda el uso de la CC0 para obras que se deseen colocar en el dominio público, no es adecuada para obras de software debido a una cláusula que excluye explícitamente cualquier licencia de patentes.

Esta ausencia de una licencia de patentes en la CC0 puede plantear riesgos al utilizar software bajo esta licencia. Por lo tanto, antes de emplear dicho software, es prudente considerar si el desarrollador podría demandar al usuario por

infringir una patente. En caso de que el desarrollador no conceda licencias de patentes a los usuarios, el software podría representar una trampa legal y debería ser evitado.

- **Creative Commons CC BY**

Esta es una licencia libre sin copyleft, apropiada para obras artísticas, de entretenimiento y educativas. Es compatible con todas las versiones de la GPL de GNU; sin embargo, al igual que todas las licencias CC, no debe ser utilizada para software.

Creative Commons ofrece una variedad de licencias que difieren significativamente entre sí. Por lo tanto, simplemente mencionar que una obra está bajo "una licencia de Creative Commons" no proporciona información suficiente sobre el licenciamiento de la obra. Cuando se encuentre con una declaración de este tipo, es importante solicitar al autor que especifique claramente cuál es la licencia Creative Commons que se está utilizando.

En un contexto más amplio, las licencias de Creative Commons constituyen un conjunto de herramientas flexibles que permiten a los creadores de contenido otorgar ciertos derechos a otras personas bajo condiciones específicas. Estas licencias se utilizan comúnmente para una variedad de obras creativas, como música, imágenes, videos, texto y otros tipos de contenido digital.

El propósito fundamental de las licencias de Creative Commons es proporcionar una alternativa a los tradicionales derechos de autor "todos los derechos reservados", permitiendo a los creadores compartir su trabajo de manera más amplia y permitiendo a otros utilizarlo y construir sobre él en formas específicas.

Existen diversas versiones de las licencias de Creative Commons, cada una con diferentes combinaciones de condiciones. Estas condiciones incluyen la atribución (reconocimiento al autor original), la no comercialización (uso no

comercial), no derivadas (no se pueden realizar obras derivadas) y compartir igual (las obras derivadas deben compartirse bajo la misma licencia).

En última instancia, las licencias de Creative Commons representan un medio para equilibrar los derechos de los creadores con la promoción de la cultura libre y el intercambio de conocimientos en la sociedad digital.

2.1.4 Software Libre en la Educación

La importancia del software libre en la educación es esencial. Las instituciones educativas de todos los niveles deben comprometerse a utilizar y enseñar exclusivamente software libre, alineándose así con su misión de difundir conocimiento y formar ciudadanos responsables y activos en la sociedad. El software libre permite el acceso al código fuente y a los métodos empleados, tratándolos como parte del conocimiento humano al alcance de todos. En contraste, el software privativo, que es conocimiento secreto y restringido, va en contra de los principios fundamentales de la educación.

Más allá de los aspectos técnicos, el uso de software libre es una cuestión de ética, social y política, basada en los derechos humanos que todos los usuarios de software deberían tener. Los valores de libertad y cooperación son fundamentales en el software libre, y son promovidos por sistemas como GNU, que abogan por el principio de compartir, reconociendo que compartir fomenta y contribuye al avance humano.

El papel de las instituciones educativas es crucial en la configuración del futuro de la sociedad a través de lo que enseñan. Para asegurar un impacto positivo, es vital que opten por enseñar exclusivamente software libre. El uso de software privativo en la educación promueve dependencia, contradiciendo los objetivos educativos. Al integrar software libre en sus currículos, las escuelas no solo fomentan la independencia, sino que también preparan a los estudiantes con las competencias necesarias para ser programadores hábiles.

Además, el software libre incentiva la cooperación y el apoyo mutuo entre los estudiantes. Debería existir una regla en todas las aulas que promueva el compartir: "Estudiantes, este lugar es para compartir conocimiento. Si traen software a la clase, deben compartirlo con sus compañeros, incluyendo el código fuente, para que otros puedan aprender de él. Por esta razón, no se permite el software privativo en nuestras clases, a menos que sea para estudiarlo a través de ingeniería inversa".

Los desarrolladores de software privativo preferirían que se penalizara a los estudiantes que comparten y desalentara a los curiosos que desean explorar y modificar el software. Este enfoque sería una mala educación. Debemos considerar más detenidamente la implementación del software libre en la educación.

Las instituciones educativas, desde preescolares hasta universidades, tienen una obligación moral de enseñar exclusivamente software libre. Esto se debe a que el software libre permite a los usuarios controlar sus propios dispositivos y colaborar con otros, ofreciendo así una vida más ética y honrada. Esto es relevante tanto para individuos como para escuelas, aunque aquí se enfocan más los beneficios específicos para la educación.

Aunque el software libre puede representar un ahorro económico para las escuelas al permitirles copiar y redistribuir el software sin costos adicionales, el verdadero valor del software libre va más allá de lo económico. Se trata de ofrecer una educación de calidad, transformando un sistema que podría promover la dependencia de las grandes corporaciones en uno que fomenta la independencia y la cooperación.

Las escuelas deben promover la utilización de software libre como promueven otras prácticas cívicas importantes, como la conservación ambiental y la participación electoral. Al hacerlo, preparan a los estudiantes para ser ciudadanos activos y conscientes en una sociedad digital libre, protegiéndolos del dominio corporativo.

Enseñar software privativo, por otro lado, fomenta la dependencia y va en contra de la misión educativa de las escuelas. Esto es similar a cómo algunas empresas ofrecen

productos gratuitamente a los estudiantes, esperando crear dependencia que resultará en futuras compras.

El software libre no solo permite a los estudiantes usar programas sin restricciones, sino que también les da la oportunidad de entender cómo funcionan esos programas al acceder al código fuente. Esta es una oportunidad crucial para los estudiantes con inclinaciones de programación, ya que les permite satisfacer su curiosidad y mejorar sus habilidades leyendo y modificando código real.

El software privativo, que mantiene sus códigos fuente en secreto, niega a los estudiantes esta oportunidad de aprendizaje, actuando en contra de los principios educativos. Por el contrario, el software libre fomenta un ambiente de aprendizaje abierto y colaborativo, crucial para el desarrollo de habilidades en programación y otras áreas técnicas.

Enseñar el uso de software libre y alentar la participación en su comunidad no solo es una lección técnica, sino una profunda lección cívica. Prepara a los estudiantes no solo para ser usuarios de tecnología, sino también para ser ciudadanos activos y conscientes que valoran el servicio público por encima del lucro corporativo.

Si usted está de alguna manera vinculado a una institución educativa, tiene la responsabilidad de abogar por la transición hacia el software libre. Si esfuerzos individuales no son suficientes, considere iniciar una campaña más amplia para sensibilizar y sumar apoyos en su comunidad educativa.

Estas son las principales razones por las cuales las instituciones educativas, desde preescolares hasta universidades, deben adoptar exclusivamente software libre.

Fomento del compartir: Las escuelas deben liderar con el ejemplo al enseñar el valor de compartir. El software libre facilita la educación al permitir compartir conocimientos y herramientas de la siguiente manera:

Conocimiento: Con el software libre, los profesores pueden compartir el código fuente de los programas con los alumnos, permitiéndoles entender cómo funcionan y aprender de ellos. Esto promueve un ambiente de aprendizaje abierto y colaborativo.

Herramientas: Los profesores pueden proporcionar a los estudiantes copias de los programas utilizados en clase para su uso personal en casa, fomentando así la práctica y la exploración fuera del aula.

Responsabilidad social: Las escuelas tienen la responsabilidad de preparar a los estudiantes para participar en una sociedad digital libre, donde puedan tomar el control de sus vidas de manera autónoma. Esto implica no depender del poder unilateral de los desarrolladores de software propietario, lo cual va en contra de los principios educativos fundamentales.

Independencia: Las escuelas deben enseñar la fortaleza y la independencia, en lugar de fomentar la dependencia de un producto o empresa en particular. Al elegir software libre, las instituciones educativas evitan quedar atadas a intereses comerciales y mantienen su autonomía.

Aprendizaje: Cada vez más estudiantes consideran si una institución utiliza software libre al decidir dónde estudiar, ya que esto les brinda la libertad de explorar y comprender el funcionamiento de los programas, así como aprender ética y prácticas profesionales relacionadas con el desarrollo de software.

Ahorro: Aunque el ahorro económico es una ventaja, el verdadero valor radica en la capacidad de las escuelas para distribuir copias de programas a bajo costo o de forma gratuita, ayudando así a las familias con dificultades económicas y promoviendo la equidad en la educación.

Calidad: El software libre ofrece soluciones estables, seguras y fáciles de instalar para la educación, aunque su excelente desempeño es un beneficio secundario en comparación con la libertad que proporciona a los usuarios.

Programas libres:

A continuación se presenta 2 listas de programas libres, la primera es de programas libres y las aplicaciones privativas a las cuales reemplaza, la segunda lista menciona programas libres los cuales no necesariamente tiene equivalentes privativos, pero son muy útiles para la educación:

Categoría	Programa libre	Sustituye a
Sistemas operativos	Distribuciones GNU/Linux	ChromeOS, Windows, MacOS, iOS
Navegadores	Epiphany GNU IceCat	Chrome, Microsoft Edge, Safari
Ofimática	LibreOffice	Microsoft Office
	Etherpad , EtherCalc	Google Docs, Sheets
Plataformas para la enseñanza a distancia	Moodle	Google Classroom, Blackboard
Sincronización de archivos	ownCloud Nextcloud	Drive, Calendar, Contacts, OneDrive, Outlook
Compartición de archivos multimedia	GNU MediaGoblin	YouTube, Google Photos, Picasa
Correo	Kolab	Gmail, Outlook
Conferencias (vídeo, audio, chat, mensajes)	GNU Jami	Google Hangouts Skype

de texto)	Jitsi	Microsoft Teams
	Jitsi Meet	Zoom WhatsApp
	BigBlueButton	
Juegos	Minetest	Minecraft

Tabla 2.1: Programas libres y a quien reemplaza

Categoría	Programa libre	Descripción
Arte, artes gráficas y diseño	Blender	Software para multimedia y juegos sin programación.
	FreeCAD	Herramienta para diseñar modelos tridimensionales.
	GIMP	Programa para editar y dibujar imágenes.
	Krita	Aplicación para arte digital.
	Tux Paint	Herramienta intuitiva para niños.
Juegos y actividades educativas	GCompris	Conjunto de actividades educativas.
Matemáticas	GNU Octave	Software para cálculos numéricos.
Física	FisicaLab	Aplicación para resolver problemas físicos.

Música	GNU Solfege	Software para la formación musical y entrenamiento del oído.
	LMMS	Programa para crear música.
	MuseScore	Aplicación para componer partituras musicales.
Programación	GDevelop	Plataforma para crear juegos sin programación.
	Racket	Lenguaje de programación para aprender.
	Ruby	Lenguaje de programación OOP

Tabla 2.2: Programas libres de gran utilidad

2.1.5 Software Libre en Simulación de Vuelo

FlightGear

FlightGear es un simulador de vuelo de código abierto que está siendo desarrollado por voluntarios talentosos de todo el mundo. Está diseñado para funcionar en una variedad de plataformas populares, como Windows, Mac y Linux. Todo el proyecto está disponible como código fuente y está licenciado bajo la Licencia Pública General de GNU.

El objetivo del proyecto FlightGear es crear un marco de simulación de vuelo sofisticado y abierto para su uso en entornos de investigación o académicos, entrenamiento de pilotos, como herramienta de ingeniería industrial, para entusiastas que desean explorar sus ideas de simulación de vuelo interesantes.

FlightGear es un proyecto de simulador de vuelo gratuito que está siendo desarrollado gracias a las generosas contribuciones de código fuente y tiempo libre de muchas personas talentosas de todo el mundo.

La idea de FlightGear surgió de una insatisfacción con los simuladores de vuelo comerciales para PC actuales. Un gran problema con estos simuladores es su carácter propietario y su falta de extensibilidad. Hay muchas personas en todo el mundo con excelentes ideas para mejorar los simuladores disponibles actualmente que tienen la capacidad de escribir código y el deseo de aprender y contribuir. Muchas personas involucradas en la educación e investigación podrían usar un marco de simulador de vuelo vistoso en el que construir sus propios proyectos; sin embargo, los simuladores comerciales no se prestan a la modificación y mejora. El proyecto FlightGear se esfuerza por llenar estos vacíos.

Hay una amplia gama de personas interesadas y participando en este proyecto. Este es realmente un esfuerzo global con contribuyentes de casi todos los continentes. Los intereses van desde construir un simulador de casa realista con partes de aviones antiguos, hasta la investigación universitaria y el uso educativo, hasta simplemente tener una alternativa viable a los simuladores comerciales para PC.

FlightGear y su código fuente se han mantenido intencionalmente abiertos, disponibles y gratuitos. Al hacerlo, se puede aprovechar los esfuerzos de personas talentosas de todo el mundo. Esto contrasta con el enfoque tradicional de los proveedores de software comercial, que están limitados por la capacidad colectiva de las personas que pueden contratar y pagar.

OpenVSP

OpenVSP, conocido también como Open Vehicle Sketch Pad, es una herramienta de geometría paramétrica de aeronaves de código abierto desarrollada originalmente por la NASA. Se utiliza para crear modelos 3D de aeronaves y para respaldar el análisis de ingeniería de esos modelos. Predecesores de OpenVSP, como VSP y Rapid Aircraft Modeler (RAM), fueron desarrollados por J.R. Gloudemans y otros para la NASA a principios de los años 90. OpenVSP v2.0 se lanzó como código abierto bajo la licencia NOSA en enero de 2012. El desarrollo ha sido liderado por Rob McDonald desde

alrededor de 2012 y ha sido respaldado por la NASA y AFRL, entre otras contribuciones.

OpenVSP también proporciona capacidades de API que pueden ser accedidas utilizando Matlab, Python o AngelScript.

Modelado de geometría

OpenVSP ofrece una multitud de geometrías básicas, comunes para el modelado de aeronaves, que los usuarios modifican y ensamblan para crear modelos. Ala, pod, fuselaje y hélice son algunas de las geometrías disponibles. Componentes avanzados como el cuerpo de revolución, conducto, geometría conformal, entre otros, también están disponibles.

JSBSim

Es un marco de modelo de dinámica de vuelo (FDM) escrito en lenguaje C++, de código abierto, multiplataforma y orientado a objetos. Está diseñado para facilitar la modelización de simulaciones de cualquier nave aeroespacial sin necesidad de un código de programa específico compilado y enlazado, en lugar de ello, se basa en una especificación versátil y potente escrita en un formato XML. Este formato es formalmente conocido como JSBSim-ML (Lenguaje de Marcado JSBSim).

Inicialmente desarrollado para el simulador de vuelo de código abierto FlightGear, JSBSim puede funcionar como un ejecutable independiente en tiempo real suave o en modo por lotes, lo que lo hace ideal para pruebas automatizadas utilizando las capacidades de script internas.

En lugar de modelar aeronaves específicas en código de programa, JSBSim define las características de la aeronave en un formato basado en XML. Esto incluye parámetros como las propiedades del tren de aterrizaje, el punto de vista del piloto, masas adicionales, sistemas de propulsión, sistemas de control de vuelo, piloto automático y coeficientes aerodinámicos. El formato del archivo de configuración está diseñado para ser accesible, con coeficientes similares a los de los libros de texto, lo que ayuda a los

recién llegados en la descripción del vehículo con conocimientos básicos de aerodinámica.

Único en JSBSim es su enfoque para modelar sistemas de aeronaves, utilizando cadenas de componentes para representar filtros, interruptores, sumadores, ganancias y sensores. Además, utiliza "propiedades" para exponer variables como nodos en una jerarquía similar a un árbol, facilitando la integración de varios modelos de aeronaves sin requerir un código fuente de programa específico.

Las ecuaciones de movimiento en JSBSim siguen de cerca las formulaciones de los libros de texto de aerodinámica, con cuaterniones utilizados para el seguimiento de la orientación para evitar el "bloqueo de cardán". Puede simular vuelo atmosférico o movimiento de naves espaciales en órbita, incorporando aceleraciones de Coriolis y centrípetas en las ecuaciones.

JSBSim ofrece una salida de datos configurable, lo que permite a los usuarios seleccionar conjuntos de datos lógicamente relacionados y propiedades individuales para la salida a la consola, archivos, sockets o combinaciones de estos.

JSBSim se ha utilizado para una variedad de aplicaciones, incluido el desarrollo de leyes de control para cohetes sónicos, la elaboración de pilotos automáticos de aeronaves para proyectos de tesis, sirviendo como modelo de vuelo para FlightGear y conduciendo simuladores de base de movimiento para fines comerciales/entretenimiento.

Las plataformas compatibles para JSBSim incluyen Linux, Windows (MSVC, Cygwin, Mingwin), Mac OS X y FreeBSD. Es importante destacar que JSBSim no tiene dependencias externas y está licenciado bajo los términos de la Licencia Pública Menor de GNU (LGPL).

2.1.6 Contribuir al Software Libre

Existen varios motivos para escribir software libre, y es importante entender que no todos los programadores tienen un solo motivo detrás de su trabajo. A continuación, se presentan algunos de los motivos que muchos programadores de software libre consideran importantes:

Diversión: Para algunas personas, especialmente los programadores más talentosos, escribir software es una fuente de diversión, especialmente si pueden hacerlo sin que haya alguien supervisándolos constantemente.

Idealismo político: Algunos tienen el deseo de construir un mundo más libre y ayudar a los usuarios de computadoras a liberarse del control de los desarrolladores de software.

Admiración: Es gratificante recibir reconocimiento por escribir un programa útil y exitoso, ya que los usuarios lo admirarán por ello.

Reputación profesional: Desarrollar un programa libre exitoso es una forma de demostrar habilidades como programador y mejorar la reputación profesional.

Comunidad: Ser parte de una comunidad y colaborar con otros en proyectos de software libre puede ser un motivo importante para muchos programadores.

Educación: Escribir software libre brinda la oportunidad de mejorar tanto habilidades técnicas como sociales, lo cual puede ser beneficioso tanto para profesionales como para estudiantes.

Gratitud: Algunos programadores se sienten agradecidos por el software libre que han utilizado y desean devolver algo a la comunidad al escribir programas útiles para otros.

Oposición a ciertas empresas: Si bien criticar exclusivamente a una empresa como Microsoft puede ser un error, algunas personas tienen sentimientos profundos de desaprobación hacia ella y contribuyen al software libre como una forma de oposición.

Remuneración: Hay personas que reciben pago por desarrollar software libre o han construido negocios en este ámbito.

Mejora personal: Muchos programadores trabajan en mejorar los programas que utilizan con el objetivo de hacerlos más convenientes para ellos mismos.

Es importante reconocer que la motivación para escribir software libre puede ser

multifacética y que cada individuo puede tener una combinación única de razones para hacerlo.

Contribuir al Proyecto GNU y al Movimiento del Software Libre

Decir no al uso de un programa no libre o un servicio en línea perjudica la causa de la libertad del software. Decir no a su uso con otros ayuda aún más. Y si les dices que es en defensa de tu libertad y la de ellos, eso ayuda aún más.

Contribuir al desarrollo del sistema operativo GNU

- Elaborar manuales gratuitos y otra documentación para el software GNU.
- Ofrecerse de voluntario para convertir la documentación de varios formatos al formato Texinfo.
- Proponer paquetes de software útiles como paquetes GNU.
- Realiza uno de los Proyectos de Mejora de Prioridad Alta de GNU.
- Trabajar en un proyecto de la lista de proyectos de software libre muy importantes de la FSF.
- Mejora la accesibilidad del software libre y de las páginas web para satisfacer las necesidades de todos los usuarios, independientemente de su discapacidad.
- Si se es estudiante y se realiza un proyecto de desarrollo de software, se debe de contribuir a GNU. Consultar que la escuela deje publicarlo como software libre
- Al escribir software para GNU, se debe de seguir los Estándares de Codificación de GNU y la Información para Mantenedores de Software GNU.
- Revisar el Directorio de Software Libre para ver si ya existe software libre que haga la tarea y no “reinventar la rueda”

Contribuye al apoyo al desarrollo y uso de GNU

- Organizar un nuevo Grupo de Usuarios GNU/Linux.
- Ofrecerse de voluntariado como Webmaster GNU. Comenzando completando el cuestionario para webmasters.
- Traducir el sitio web de GNU a otros idiomas. Cada equipo de traducción necesita varios miembros que sean hablantes nativos del idioma objetivo (y fluidos en inglés), pero también necesitan al menos un miembro que sea hablante nativo de inglés (y fluido en el idioma objetivo).
- Al hablar con personas que no valoran la libertad y la comunidad, mostrarles las numerosas ventajas prácticas del software libre
- Donar hardware a la FSF.
- Si se trabaja en el apoyo o desarrollo de software libre de alguna manera, se puede incluir en el Directorio de Servicios.
- Si se dirige una empresa que necesita contratar personas para trabajar con software libre, se puede anunciar en la Página de Empleo de Software Libre.

Promover el conocimiento sobre GNU y el Movimiento del Software Libre

- Informamos a nuestros conocidos sobre la filosofía y el software de GNU.
- Cuando nos referimos al sistema operativo que comenzó como GNU con Linux añadido, lo llamamos GNU/Linux, y no seguimos a aquellos que lo llaman "Linux". Una vez que estamos al tanto de lo que ya hemos hecho, en lugar de atribuirlo a otros, apoyamos más nuestros esfuerzos presentes y futuros. Esta ayuda requiere muy poco tiempo una vez que hemos desaprendido el viejo hábito.
- Demostramos nuestro apoyo al movimiento del software libre y nuestras ideas de libertad para los usuarios de computadoras, utilizando los términos "software libre", "software libre" o "software libre/libre".
- Evitamos el término "código abierto", que representa el rechazo de los ideales.

Colaborar y donar a la Free Software Foundation

La Free Software Foundation es el principal patrocinador organizacional del Sistema Operativo GNU. La FSF también ayuda a difundir la conciencia sobre los problemas éticos y políticos de la libertad del software.

Al igual que GNU, la FSF también obtiene gran parte de su fuerza de los voluntarios. Es un excelente lugar para hacer voluntariado y una gran comunidad a la que unirse, especialmente si no tienes la formación técnica para contribuir directamente al desarrollo de software libre.

Apoyar financieramente a la FSF y al Proyecto GNU al convertirse en miembro asociado de la FSF, donando a la FSF, comprando manuales, camisetas, pegatinas y otros productos de la FSF, o vendiendo software libre y donando parte de las ganancias a la FSF u otra organización de software libre. Al financiar un desarrollo, se puede avanzar en el mundo del software libre.

2.1.6.1 Git y GitHub: Herramientas para la Colaboración y Distribución de Software Libre

Git

Git es un sistema de control de versiones distribuido (VCS, por sus siglas en inglés) creado por Linus Torvalds en 2005. Su principal función es gestionar y rastrear los cambios en el código fuente durante el desarrollo del software, permitiendo a los desarrolladores colaborar de manera eficiente sin sobrescribir el trabajo de otros. Entre sus características más destacadas se encuentran:

- **Historial Completo:** Mantiene un historial detallado de todos los cambios realizados en el proyecto, permitiendo revertir a versiones anteriores si es necesario.
- **Ramas y Fusión:** Facilita la creación de ramas (branches) para trabajar en nuevas características o correcciones de errores de manera aislada. Una vez completadas, estas ramas pueden fusionarse (merge) con la rama principal, integrando los cambios sin conflictos.

- **Distribución:** Al ser un sistema distribuido, cada desarrollador tiene una copia completa del historial del proyecto, lo que mejora la resiliencia y facilita el trabajo en entornos sin conexión.

GitHub

GitHub es una plataforma en línea que utiliza Git para proporcionar un entorno colaborativo para desarrolladores. Fundada en 2008 y adquirida por Microsoft en 2018, GitHub se ha convertido en una herramienta esencial para el desarrollo de software gracias a sus funcionalidades avanzadas y su comunidad activa. Algunas de las ventajas de utilizar GitHub son:

- **Alojamiento de Repositorios:** Permite almacenar proyectos en repositorios públicos o privados, proporcionando acceso centralizado al código fuente.
- **Colaboración:** Facilita el trabajo en equipo a través de pull requests, que permiten a los desarrolladores proponer cambios y revisarlos antes de fusionarlos con la rama principal.
- **Gestión de Proyectos:** Ofrece herramientas de gestión de proyectos como issues y boards para rastrear tareas, bugs y nuevas características.
- **Integración Continua y Despliegue Continuo (CI/CD):** Proporciona integración con herramientas de automatización que permiten ejecutar pruebas y despliegues automáticos, mejorando la eficiencia y la calidad del software.
- **Visibilidad y Comunidad:** Promueve la transparencia y el acceso abierto, permitiendo a otros desarrolladores ver, usar y contribuir al proyecto. Esto fomenta una comunidad activa de colaboración y aprendizaje.

Incorporar Git y GitHub en el desarrollo del proyecto de tesis no solo facilita la colaboración y la gestión del código, sino que también asegura que el proyecto cumpla con los principios del software libre. Subir proyectos a GitHub ofrece numerosos beneficios, incluyendo la visibilidad, la posibilidad de recibir contribuciones de otros desarrolladores y la capacidad de gestionar versiones de manera efectiva. Esto es especialmente importante en un entorno académico y de investigación, donde la transparencia y la reproducibilidad son clave.

Al subir el proyecto de tesis "AIFA en Software Libre" a GitHub, no solo se promueve el acceso y la innovación, sino que también se mejora la experiencia en simuladores de vuelo como X-Plane 11 y FlightGear, facilitando que otros puedan utilizar, modificar y redistribuir el trabajo de manera libre y abierta.

2.2 Simuladores de Vuelo

2.2.1 FlightGear

FlightGear es un simulador de vuelo de código abierto bajo la licencia GNU General Public License, disponible en plataformas como Windows, MacOS, GNU/Linux y FreeBSD. Su objetivo es proporcionar un recurso sofisticado para entornos académicos, entrenamiento de pilotos y aplicaciones industriales generales, destacando por su baja demanda de recursos de hardware y permitiendo la contribución comunitaria. Este proyecto está desarrollado en C, C++ y el lenguaje de script Nasal.

Historia

Inicios (1996-1997)

El proyecto FlightGear comenzó a finales de los 90 con una propuesta online y el desarrollo de código en 1996 utilizando gráficos 3D personalizados. Curtis Olsen lideró la transición hacia OpenGL en 1997 después del lanzamiento inicial. La amplia respuesta comunitaria ha sido fundamental en el desarrollo continuo del proyecto desde entonces.

Utilizando recursos como el modelo de vuelo LaRCsim de la NASA y datos de elevación de acceso libre, los primeros binarios se lanzaron en 1997, mejorando en estabilidad y funcionalidad con el tiempo gracias a actualizaciones constantes.

Versiones:

Versiones 0.7–0.9 (2001–2003)

Entre 2001 y 2003, se lanzaron varias versiones beta, culminando en la versión 0.9 en 2006, que recibió el premio Softpedia "Pick" con una calificación de cinco estrellas. El ritmo de nuevas versiones se desaceleró, pero el contenido y la funcionalidad se expandieron significativamente.

Versión 1.0 (2008)

El lanzamiento de la versión 1.0 marcó el fin del estado beta del software, diez años después de su primer lanzamiento.

Versión 1.9.0 (2008)

Con esta versión, FlightGear transitó de PLIB a OSG, perdiendo temporalmente algunas características pero ganando otras como los efectos de partículas para un mayor realismo. No todas las aeronaves eran compatibles con versiones anteriores, pero la versión ofrecía 230 opciones de aeronaves.

Software

FlightGear utiliza SimGear como su motor de simulación principal, con una amplia gama de modelos de aeronaves disponibles, desde planeadores hasta jets de combate. Los modelos de vuelo dinámico son fundamentales en el simulador, y FlightGear utiliza varios sistemas, incluyendo JSBSim, YAsim y UIUC, para simular la física del vuelo.

Dependencias

FlightGear se construye sobre diversas bibliotecas, siendo SimGear la principal. Además, utiliza OpenGL para gráficos, OpenAL para audio, y varios otros componentes dependiendo de la plataforma de compilación.

Complementos y personalizaciones

FlightGear ha crecido para incluir varias herramientas adicionales y programas de soporte, como FGTools y FGKicker, así como editores para datos de terreno y planificación de vuelos.

Aeronaves y escenarios

El simulador comenzó con un modelo de aeronave y ha expandido su biblioteca a más de 230. Además, FlightGear ofrece un detallado mundo virtual que incluye datos de elevación y terreno global.

Red y visualización múltiple

FlightGear soporta funciones de red y múltiples pantallas para simulación en tiempo real en diferentes configuraciones, incluyendo vuelos en formación y entornos de control aéreo, extendiendo su funcionalidad a simulaciones a través de Internet y configuraciones multi-monitor para una experiencia más inmersiva.

2.2.2 X-Plane 11

X-Plane es un simulador de vuelo desarrollado por Laminar Research bajo la creación de Austin Meyer, disponible para uso en tanto dispositivos móviles como computadoras. La versión de escritorio opera en sistemas como macOS, Windows y Linux, mientras que la versión móvil es compatible con Android, iOS y webOS.

Este simulador es uno de los principales competidores del Microsoft Flight Simulator y es reconocido por su alta precisión, que se logra mediante el cálculo de cómo el aire interactúa con las superficies de los aviones simulados. Además, ha sido certificado por la Administración Federal de Aviación (FAA) de los EE. UU. para ser usado en el entrenamiento de pilotos en vuelo instrumental, siempre que se emplee con el hardware adecuado.

X-Plane tiene como objetivo proporcionar una experiencia de vuelo sumamente realista, ofreciendo una amplia variedad de aeronaves, desde aviones pequeños hasta grandes jets comerciales, y recrea la Tierra con detalles geográficos que incluyen aproximadamente 18,000 lugares de aterrizaje como aeropuertos y helipuertos, además de portaaviones para prácticas.

El realismo en la física de vuelo de X-Plane se logra a través de un túnel de viento virtual que simula condiciones aerodinámicas reales. Los usuarios también tienen a su disposición herramientas avanzadas para editar o crear nuevos modelos de aeronaves y perfiles de alas. Aunque las versiones previas a la 8 incluían un editor de escenarios, actualmente se utiliza una herramienta externa para esta función.

X-Plane permite la conexión a internet y participar en vuelos en redes donde se ofrece servicio de control de tráfico aéreo, siendo VATSIM una de las redes más utilizadas. El

simulador también cuenta con una vasta comunidad global que frecuentemente añade contenido como aviones y escenarios de forma gratuita, enriqueciendo la experiencia general.

En España, X-Plane ha sido distribuido por Friendware hasta que, en 2006, con la versión 8 disponible, FX Interactive lanzó una reedición de X-Plane 7.1 en la colección de videojuegos MVM, distribuida inicialmente con el diario El Mundo y más tarde como parte de la línea económica FX Premium, donde se mantuvo a la venta hasta septiembre de 2011. En esa fecha, FX Interactive lanzó una nueva edición de X-Plane 8.2 en la misma línea económica.

2.2.2.1 Modelo de Vuelo

X-Plane se distingue de otros simuladores de vuelo por utilizar un modelo aerodinámico conocido como la Teoría del Elemento de Pala. A diferencia de la mayoría de los simuladores que dependen de datos empíricos almacenados en tablas de consulta para calcular fuerzas aerodinámicas como la sustentación y el arrastre en diversas condiciones de vuelo, X-Plane ofrece un enfoque diferente. Mientras que esos métodos tradicionales reproducen adecuadamente el comportamiento de vuelo de aeronaves con características aerodinámicas bien documentadas, no son efectivos para el diseño aeronáutico ni para prever el rendimiento de modelos para los cuales no existen datos.

2.2.2.2 Compatibilidad de Hardware

X-Plane 11 es compatible con el hardware de Elite Simulation Solutions, una empresa suiza reconocida globalmente por su software de entrenamiento IFR, controles de vuelo y dispositivos de entrenamiento de vuelo. Entre sus productos, el "Elite Pro Panel II" es particularmente destacado, siendo empleado ampliamente en la actualidad y demostrando ser muy efectivo para la formación de estudiantes. Sin embargo, es importante mencionar que el simulador tiene ciertas limitaciones debido a su naturaleza de software propietario.

Los controladores de flyelite permiten que los sistemas operativos Windows 7 y Windows 10 interactúen con el hardware de flyelite, en particular con el Elite Pro Panel II. Este dispositivo de simulación de vuelo ofrece varias características y

funcionalidades que lo hacen ideal tanto para el entrenamiento como para el entretenimiento de pilotos y aficionados:

- Construcción Robusta: Fabricado para replicar los controles de una cabina real con materiales de alta calidad, asegura durabilidad y fiabilidad.
- Amplia Variedad de Controles: Incluye una extensa gama de interruptores, botones, palancas y perillas que emulan los controles de una aeronave real, facilitando la interacción con diversos sistemas y funciones durante las simulaciones.
- Compatibilidad Extendida: El panel es compatible con diversos simuladores de vuelo líderes en el mercado, como Microsoft Flight Simulator, X-Plane y Prepar3D, permitiendo a los usuarios elegir su software preferido para una experiencia de simulación realista.
- Configuración Personalizable: Permite a los usuarios personalizar la configuración del panel a sus necesidades, asignando funciones específicas a los controles según sus preferencias.
- Integración con Otros Dispositivos: Se puede conectar a otros dispositivos de simulación como joysticks, pedales y pantallas adicionales, ofreciendo una experiencia más completa e inmersiva.

En conjunto, el Elite Pro Panel II se presenta como una herramienta avanzada y versátil para la simulación de vuelo, proporcionando una experiencia funcional y realista que es ideal para desarrollar las habilidades de vuelo tanto de pilotos profesionales como de entusiastas de la aviación.

2.2.3 Microsoft Flight Simulator 2020

Microsoft Flight Simulator, conocido también como Flight Simulator 2020 y abreviado como FS2020 o MSFS, es un simulador de vuelo creado por Asobo Studio y publicado por Xbox Game Studios para Windows 10.

Este título, que sucede a Microsoft Flight Simulator X, marca la undécima entrega de la franquicia y es la primera en ofrecer la simulación de todo el planeta Tierra usando texturas y datos topográficos de Bing Maps. Las representaciones tridimensionales de elementos como el terreno, los árboles, los edificios y el agua, son generadas a través de la tecnología de Microsoft Azure.

El motor de juego, desarrollado por Asobo Studio, utiliza los datos de Bing Maps y la inteligencia artificial de Azure para analizar la información de mapas y fotogrametría, creando modelos 3D fotorrealistas de diversos objetos y paisajes. Esto permite al simulador representar con gran detalle la mayoría de las áreas del mundo, incluyendo un realismo fotográfico en algunas partes y alta definición en otras. Además, incluye sistemas físicos y meteorológicos avanzados, utilizando datos meteorológicos reales para simular condiciones climáticas correspondientes a la realidad; por ejemplo, si llueve en un lugar del mundo real, también lloverá en esa ubicación dentro del juego. Se menciona que cada nube tiene comportamientos individuales que pueden influir en el rendimiento de las aeronaves según su ubicación en el sistema meteorológico.

Flight Simulator también simula un mundo vivo, con vehículos en movimiento en las carreteras, agua que fluye de acuerdo con la dirección del viento, y árboles con hojas individuales. El simulador incluye más de dos millones de ciudades y pueblos, así como más de 40,000 aeropuertos reales.

2.2.3.1 Limitaciones y alternativas

Limitaciones:

- Requisitos de hardware elevados: MSFS 2020 es conocido por requerir un hardware potente para funcionar correctamente, especialmente si se desea

disfrutar de todas las características en su máximo esplendor. Esto puede ser prohibitivo para aquellos que no tienen acceso a equipos de gama alta.

- Costo: Aunque el simulador en sí no es excesivamente caro, las capacidades de hardware requeridas para ejecutarlo adecuadamente pueden aumentar significativamente el costo total de propiedad, especialmente si se necesita actualizar el hardware existente.
- Complejidad del manejo: Aunque MSFS 2020 tiene una curva de aprendizaje relativamente suave en comparación con algunos simuladores más antiguos, todavía puede ser demasiado complejo para los principiantes en simuladores de vuelo. Los usuarios menos experimentados pueden preferir simuladores que ofrezcan una experiencia más accesible y amigable para principiantes.
- Personalización limitada: Aunque MSFS 2020 ofrece una amplia gama de aeronaves y escenarios, la personalización total (por ejemplo, la adición de aviones personalizados o escenarios creados por el usuario) puede ser más limitada en comparación con otros simuladores que permiten una modificación más extensa.
- Simulación de sistemas: Algunos usuarios avanzados de simuladores de vuelo pueden encontrar que MSFS 2020 no ofrece la misma profundidad en la simulación de sistemas de aeronaves que otros simuladores, como X-Plane o Prepar3D. Esto puede ser importante para aquellos que buscan una experiencia de vuelo altamente realista y técnica.
- Compatibilidad con hardware específico: Aunque MSFS 2020 es compatible con una amplia gama de hardware, algunos usuarios han informado problemas de compatibilidad con dispositivos específicos de simulación de vuelo. Esto puede ser una consideración importante para aquellos que ya poseen hardware de simulación de vuelo específico.

- Dependencia de la conexión a Internet: MSFS 2020 requiere una conexión a Internet activa para descargar datos de mapas y condiciones meteorológicas en tiempo real. Esto puede ser una limitación para aquellos que prefieren volar sin estar conectados a Internet o que tienen una conexión de Internet inestable.

Por qué elegir otros simuladores:

- Mayor flexibilidad y personalización: Simuladores como X-Plane y FlightGear permiten una mayor personalización y modificación, tanto en términos de aviones como de escenarios. Esto puede ser atractivo para aquellos que desean tener control completo sobre su experiencia de simulación.
- Requerimientos de hardware menos exigentes: Algunos simuladores pueden funcionar bien en hardware más modesto, lo que los hace más accesibles para una variedad más amplia de usuarios.
- Comunidad y soporte: Otros simuladores, como X-Plane y FlightGear, tienen una comunidad establecida con una amplia gama de complementos, soporte técnico y tutoriales disponibles, lo que puede ser beneficioso para nuevos usuarios.

2.3 AIFA

El Aeropuerto Internacional Felipe Ángeles (AIFA), una obra emblemática del gobierno de la Cuarta Transformación, fue inaugurado por el presidente Andrés Manuel López Obrador el 21 de marzo de 2022. Esta nueva terminal aérea forma parte del sistema aeroportuario regional junto con el Aeropuerto Internacional de la Ciudad de México y el Aeropuerto Internacional de Toluca, con el objetivo de aliviar la alta demanda de servicios de aviación en la Zona Metropolitana del Valle de México.

Ubicado en Santa Lucía, en el municipio de Zumpango, Estado de México, a solo 44 km del Zócalo de la Ciudad de México, el AIFA está conectado por carretera con los estados de Hidalgo, Puebla, Tlaxcala y Querétaro. Sus instalaciones se construyeron junto a la Base Aérea Militar No. 1.

El aeropuerto ocupa un área de 38.41 hectáreas y cuenta con siete accesos y amplias fachadas de cristal de 55 mil metros cuadrados con control solar que permiten la entrada de luz natural. La torre de control, de 88 metros de altura, tiene la forma de un macuahuitl, un arma ceremonial de los mexicas, y está equipada con un moderno sistema de aproximación ILS CAT III, que permite despegues y aterrizajes simultáneos en todas las pistas, incluso en condiciones de niebla densa.

El AIFA cuenta con dos pistas para despegues y aterrizajes y 45 posiciones de embarque, de las cuales 24 son de contacto con puente de abordaje, cinco de semi contacto y 16 posiciones remotas. La sala de documentación dispone de 80 mostradores para atender a los viajeros, 86 kioscos de autoservicio y 20 mostradores para el registro de equipaje. La sala de entrega de maletas tiene siete islas y utiliza equipos de inspección que realizan tomografías computarizadas, escáneres corporales y reconocimiento facial para agilizar el abordaje de los usuarios.

Una característica notable del aeropuerto son sus 38 baños temáticos, decorados con motivos de la cultura popular mexicana, como luchadores, cómicos, catrinas, charrería, alebrijes y cerámica de talavera, entre otros. Además de las áreas del aeropuerto, hay museos, hoteles, restaurantes y centros comerciales.

Durante la ceremonia de inauguración, el general Luis Crescencio Sandoval, secretario de la Defensa Nacional, firmó la entrega del aeropuerto y el presidente López Obrador develó la placa de inauguración. Alrededor de mil quinientas personas, incluidos gobernadores y autoridades, aplaudieron la ceremonia. El primer vuelo comercial, Aeroméxico Connect 890, despegó hacia Villahermosa, Tabasco, y la aerolínea venezolana CONVIASA inició los vuelos internacionales hacia Caracas, Venezuela.

La accesibilidad al aeropuerto se ha mejorado con la inauguración de la vialidad libre de peaje Tonanitla-AIFA el 16 de febrero de 2023. Esta vía tiene una longitud de 14.1 km en cada sentido, tres carriles y nueve estructuras, con una capacidad de hasta 40 mil vehículos diarios, lo que permite llegar al AIFA de manera eficiente y segura.

El AIFA se promociona como un aeropuerto "sustentable, ecológico, resiliente, seguro,

funcional y racional, con un diseño simple y reminiscente de la arquitectura mexicana". Se estima que podrá atender a 19.5 millones de pasajeros al año, con una capacidad máxima proyectada de hasta 100 millones de usuarios.

Los planes para el Aeropuerto Internacional Felipe Ángeles han experimentado cambios significativos desde su concepción original. Este proyecto, inicialmente planeado para aliviar la congestión en el Aeropuerto Internacional de la Ciudad de México, ha evolucionado con ajustes en infraestructura y asignaciones presupuestarias.

Recientemente, se ha informado que el AIFA recibirá un aumento sustancial en su presupuesto para 2024, lo que apoyará nuevos desarrollos y mejoras en el aeropuerto . Además, el aeropuerto ha ampliado su alcance operativo, habiendo acogido más de 13,000 vuelos comerciales y transportado a más de 1.38 millones de pasajeros en su primer año de operaciones .

Estos cambios reflejan los esfuerzos continuos del gobierno mexicano por mejorar las capacidades del AIFA y convertirlo en un centro central para vuelos tanto de pasajeros como de carga. Entre las medidas adoptadas se incluye la transferencia de todas las operaciones de carga del Aeropuerto Internacional de la Ciudad de México al AIFA para aumentar su utilización .

Sí, los planes del Aeropuerto Internacional Felipe Ángeles también han incluido modificaciones estructurales. Inicialmente, se proyectó la construcción de ciertos hangares y otras instalaciones específicas. Sin embargo, con el tiempo, algunas de estas estructuras no se han construido según los planes originales. La justificación de estos cambios puede estar relacionada con ajustes en el presupuesto, revaluaciones de necesidades operativas o la priorización de otras áreas del aeropuerto. Esto es común en proyectos de infraestructura de gran escala, donde las circunstancias pueden cambiar y requerir adaptaciones sobre la marcha.

Capítulo 3: Metodología

3.1 Descripción General del Proyecto

El proyecto "AIFA en Software Libre" tiene como objetivo desarrollar una versión digital del Aeropuerto Internacional Felipe Ángeles (AIFA) utilizando software libre, específicamente para los simuladores de vuelo X-Plane 11 y FlightGear. Este proyecto no solo busca crear una representación precisa del AIFA, sino también fomentar la utilización y contribución al software libre en el ámbito de la simulación de vuelo.

El desarrollo del proyecto se llevará a cabo en varias fases: la recopilación de herramientas, la instalación de las herramientas, la recopilación de datos, el diseño y modelado del aeropuerto, la implementación de los modelos en los simuladores, y un procedimiento detallado para la integración y prueba de estos modelos. Cada fase está diseñada para asegurar que el proyecto final sea preciso, funcional y útil tanto para entusiastas de la simulación de vuelo como para otros desarrolladores.

3.2 Fases del proyecto

3.2.1 Recopilación de herramientas

Para poder crear un aeropuerto en software libre, es esencial contar con herramientas adecuadas y cumplir con las licencias correspondientes. Las principales herramientas utilizadas son:

World Editor (WED): Es un editor gráfico similar a CAD que permite el diseño de aeropuertos. Está diseñado para X-Plane, pero también se puede utilizar para FlightGear.

TerraGear: Es una colección de herramientas de renderizado para transformar datos en representaciones 2D y 3D de la tierra, siendo la principal herramienta utilizada para generar escenarios en FlightGear.

3.2.2 Instalación de las herramientas

Para instalar WED, se puede descargar el binario desde la página oficial de X-Plane <https://developer.x-plane.com/tools/worldeditor/>. También es posible compilarlo en sistemas GNU/Linux utilizando las dependencias necesarias, como binutils, git, coreutils, make, libmpfr-devel, mesa-libGL-devel, qt4-devel, python2, cmake, fltk, y curl. La documentación para la compilación está disponible en el repositorio oficial de GitHub de X-Plane.

Para utilizar WED, es necesario crear directorios que simulen las carpetas por defecto del simulador X-Plane, a menos que se tenga instalado el simulador, el cual proporciona una gran cantidad de bibliotecas. Si se trabaja para FlightGear, se deben crear los siguientes directorios:

Directorio Maestro

```
|--Custom Scenery  
|--Global Scenery  
|--Resource  
    |--default scenery
```

Instalación de TerraGear:

Para instalar TerraGear, se puede utilizar un script de descarga y compilación disponible en la Wiki de TerraGear, ideal para distribuciones GNU/Linux que utilizan el gestor de paquetes "apt". Este script instalará SimGear, TerraGear y sus dependencias. El código fuente está disponible en el repositorio oficial de Git de FlightGear en SourceForge. <https://sourceforge.net/p/flightgear/terragear/ci/master/tree/>

Para compilar TerraGear, se utiliza cmake y se compilan las dependencias de SimGear, CGAL (para cálculos geométricos de alta precisión) y libgdal.

3.2.3 Recopilación de datos

Para obtener el terreno, se requieren datos de elevación proporcionados por SRTM, así como la ubicación y diseño del aeropuerto, generados por WED en archivos apt.dat. También se necesita información sobre si una latitud/longitud específica es mar, tierra, ciudad, bosque, pueblo, carretera, ferrocarril, etc., obtenida de archivos OSM. Además, se necesita el polígono de la masa terrestre mundial para determinar las líneas costeras.

Todos estos datos se guardan en el directorio /data. Los datos de elevación se procesan con la herramienta "hgtchop" de TerraGear y se almacenan en el directorio /work. Los datos de aeropuertos se mapean con "genapts" y también se guardan en /work. Luego, se extraen los archivos con "ogr2ogr" en subcategorías como estructuras, lagos, rieles, edificios, árboles, etc., y se asignan a áreas generales como asfalto y áreas verdes. Finalmente, se utiliza "tg-construct" para construir el escenario, y el resultado se guarda en /output.

3.2.4 Diseño y modelado

Para el diseño del AIFA fue necesario obtener datos importantes como el nombre de las pistas, las ubicaciones, para eso es necesario:

<https://www.arcgis.com/apps/View/index.html?appid=504e3ff67457481e839bb941a709350f> y OpenStreetMap, las cuales son herramientas que se encuentran incluidas en WED, e incluye material 3D, como hangares y aviones. Todo esto para mapear el aeropuerto para X-Plane 11, se utilizó como base un aeropuerto de gateway-xplane, el cual traía hechas las pistas y pistas de rodaje, al igual que los estacionamientos, pero no tenía incluido material 3D, entonces se le agregó.



Figura 3.1: Mapeo del AIFA

3.2.5 Implementación en los simuladores

Para implementarlo en FlightGear, se tiene que esperar a la siguiente actualización para que se implemente. Para X-Plane 11 es otro caso, el cual se necesita utilizar la opción de exportar a X-Plane en el World Editor, el cual revisa que el mapeo no tenga ningún tipo de error y si está todo correcto, se incluye en los escenarios disponibles para X-Plane 11.

3.2.6 Pruebas y rendimiento

Así luce el aeropuerto “in-game”

El aeropuerto no demuestra fallas en su funcionamiento de tal manera que muestra una experiencia agradable al usuario.



Figura 3.2: AIFA en uso

3.3 Publicación y Documentación

El proyecto se publicó en la plataforma de GitHub debido a su versatilidad y gran escala, el archivo es público y cualquiera lo puede descargar y mejorar, gracias a la licencia GPL de software libre: <https://github.com/namelast7274/MMSM.git> donde en este se muestra un pequeño archivo para saber como descargarlo y contribuir al proyecto.

Capítulo 4: Resultados y conclusiones

4.1 Evaluación de Objetivos

De acuerdo a los objetivos propuestos anteriormente, estos fueron completados exitosamente. Se realizó el correcto modelado del Aeropuerto Internacional Felipe Ángeles (AIFA) y se adaptó al simulador de X-Plane utilizando únicamente software libre. Este proyecto no solo alcanzó su meta técnica, sino que también fomentó el uso del software libre al publicar el proyecto, demostrando que se pueden lograr grandes cosas sin recurrir a software propietario.



Figura 4.1: AIFA en uso

4.2 Evaluación de Hipótesis

En este trabajo, se propuso la hipótesis de que la implementación de una opción de calidad del AIFA en simuladores de vuelo de software libre, junto con la promoción activa de la participación de la comunidad en el desarrollo y mejora del proyecto, contribuiría significativamente a la mejora de la accesibilidad y la calidad de la formación práctica en aeronáutica.

Implementación y Adaptación:

Se logró implementar una versión de alta calidad del AIFA en simuladores de vuelo de software libre. Este proceso demostró la capacidad del software libre para adaptarse rápidamente a los cambios y actualizaciones del AIFA, permitiendo mantener el simulador actualizado con precisión.

Promoción del Software Libre:

El proyecto promovió la adopción y la conciencia sobre el software libre en el ámbito educativo y en la comunidad aeronáutica. Esto se reflejó en el incremento de instituciones educativas y usuarios individuales que adoptaron el simulador de vuelo como herramienta de formación.

En resumen, los resultados obtenidos confirman que la implementación de un AIFA de alta calidad en simuladores de vuelo de software libre y la promoción del software libre cumplen con los objetivos planteados en la hipótesis inicial.

4.3 Evaluación de Limitaciones

La evaluación de límites del proyecto "AIFA en Software Libre" es crucial para identificar las restricciones y desafíos enfrentados durante su desarrollo, así como las posibles áreas de mejora futura. En este análisis, se revisarán aspectos técnicos, operativos, y de colaboración, proporcionando una visión integral de las limitaciones encontradas.

4.3.1 Limitaciones Técnicas

Compatibilidad de Herramientas:

World Editor (WED) y TerraGear: La integración de herramientas de diseño de X-Plane (WED) y FlightGear (TerraGear) puede presentar problemas de compatibilidad. WED está específicamente diseñado para X-Plane, lo que puede dificultar la exportación directa y la adaptación a FlightGear. Además de que algunas herramientas ya están

empezando a quedar obsoletas, lo que implica crear nuevas herramientas, retrasando el proyecto.

Datos Geoespaciales: La calidad y resolución de los datos geoespaciales, como los proporcionados por SRTM y OSM, pueden no ser suficientes para capturar detalles finos del terreno y las estructuras del AIFA. Esto puede impactar la precisión del modelo digital.

Recursos Computacionales: La creación y renderización de modelos 3D detallados requiere considerable capacidad de procesamiento y memoria. Equipos con especificaciones limitadas pueden experimentar retrasos o fallos durante estas etapas.

Licencias y Permisos: Licencias de Software: La necesidad de asegurar que todas las herramientas y datos utilizados cumplen con la Licencia Pública General de GNU (GPL) versión 3 puede limitar la elección de recursos y herramientas disponibles.

4.3.2 Limitaciones Operativas

Accesibilidad de Datos: La obtención de datos precisos sobre el diseño y las operaciones actuales del AIFA puede ser limitada debido a restricciones de acceso o falta de datos actualizados.

Actualización de Infraestructura: Cambios recientes y continuos en la infraestructura del AIFA pueden no estar reflejados en los datos utilizados, lo que puede llevar a desactualizaciones en el modelo final.

Conocimiento para usar las herramientas: La capacitación necesaria para utilizar eficazmente las herramientas de modelado y simulación puede requerir tiempo adicional.

4.3.3 Limitaciones de Colaboración

Gestión de Versiones: Uso de Git y GitHub: Aunque Git y GitHub son herramientas poderosas para la colaboración, la curva de aprendizaje puede ser empinada para nuevos usuarios. Esto puede dificultar la contribución entre los usuarios que quieran contribuir al proyecto.

4.4 Conclusiones

A pesar de las limitaciones técnicas, operativas y de colaboración, el proyecto "AIFA en Software Libre" ha logrado avances significativos en la creación de un modelo digital del Aeropuerto Internacional Felipe Ángeles utilizando software libre. Las herramientas como WED y TerraGear han permitido desarrollar un modelo detallado, aunque con algunos desafíos en términos de compatibilidad y recursos computacionales. El uso de Git y GitHub da la facilidad para que cualquiera pueda integrarse al proyecto.

En general, el proyecto ha demostrado la viabilidad de utilizar software libre para proyectos de simulación de vuelo, ofreciendo una alternativa accesible y flexible a las soluciones propietarias. Las lecciones aprendidas y las limitaciones identificadas proporcionan una base sólida para futuras mejoras y desarrollos en proyectos similares.

Capítulo 5: Consideraciones a Futuro

5.1 Hardware

Implementar hardware en el plantel educativo con FlightGear permitirá tener una experiencia completa con software libre. Además, se propone desarrollar controladores personalizados para aprovechar al máximo los dispositivos y asegurar una integración completa y eficiente en el entorno de simulación.

5.2 FlightGear

Integrar completamente el modelo del AIFA en FlightGear es una meta crucial. Actualmente, la única forma estable de añadir un nuevo escenario a FlightGear es subirlo a un repositorio y esperar una actualización. Es importante monitorear estas actualizaciones y, una vez disponibles, aprovecharlas para disfrutar plenamente del aeropuerto digitalizado.

5.3 Mejora del AIFA

La simulación del AIFA no es perfecta y, dado que el aeropuerto está en constante evolución, es esencial mejorar y actualizar continuamente el modelo. La utilización de la plataforma GitHub permite a cualquier persona contribuir al proyecto, asegurando que el aeropuerto digitalizado se mantenga actualizado y relevante. Esta colaboración abierta facilita el mantenimiento constante y la incorporación de las últimas modificaciones y mejoras del AIFA.

Referencias

Philosophy of the GNU Project - GNU Project - Free Software Foundation. (2022, 21 octubre). gnu.org. Recuperado 9 de mayo de 2024, de <https://www.gnu.org/philosophy/philosophy.html>

Las 5 licencias de software libre más importantes que todo desarrollador debe conocer. (2014, 29 octubre). BBVA API_Market. Recuperado 8 de mayo de 2024, de <https://www.bbvaapimarket.com/es/mundo-api/las-5-licencias-de-software-libre-mas-importantes-que-todo-desarrollador-debe-conocer/>

Various Licenses and Comments about Them - GNU Project - Free Software Foundation. (2023, 17 octubre). gnu.org. Recuperado 8 de mayo de 2024, de <https://www.gnu.org/licenses/license-list.html>

Licenses - GNU Project - Free software Foundation. (2022, 13 abril). gnu.org. Recuperado 8 de mayo de 2024, de <https://www.gnu.org/licenses/licenses.html>

Free Software and Education - GNU Project - Free Software Foundation. (2022, 11 mayo). gnu.org. Recuperado 9 de mayo de 2024, de <https://www.gnu.org/education/education.html>

Wikipedia contributors. (2024, 18 abril). OpenVSP. Wikipedia. Recuperado 9 de mayo de 2024, de <https://en.wikipedia.org/wiki/OpenVSP>

JSBSIM Flight Dynamics Model: JSBSIM. (2023, 5 noviembre). Jsbsim-team. Recuperado 9 de mayo de 2024, de <https://jsbsim-team.github.io/jsbsim/index.html>

Helping the GNU Project and the Free Software Movement - GNU Project - Free Software Foundation. (2024, 13 abril). gnu.org. Recuperado 9 de mayo de 2024, de <https://www.gnu.org/help/help.en.html>

Free Software for Education - GNU Project - Free Software Foundation. (2023, 30 noviembre). gnu.org. Recuperado 10 de mayo de 2024, de <https://www.gnu.org/software/free-software-for-education.en.html>

FlightGear - FlightGear wiki. (2024, 30 enero). wiki.flightgear.org. Recuperado 13 de mayo de 2024, de <https://wiki.flightgear.org/FlightGear>

colaboradores de Wikipedia. (2024, 22 enero). *X-Plane.* Wikipedia, la Enciclopedia Libre. Recuperado 16 de mayo de 2024, de <https://es.wikipedia.org/wiki/X-Plane>

ELITE Simulation Solutions – exceptional flight training devices. (s. f.). Recuperado 16 de mayo de 2024, de <https://flyelite.com/>

Sky, H. I. (s. f.). *ELITE Pro Panel II Digital Flight Console - High in the Sky.* highinthesky.cz. Recuperado 18 de mayo de 2024, de <https://www.highinthesky.cz/hardware/elite-pro-panel-ii-digital-flight-console/>

colaboradores de Wikipedia. (2024, 16 enero). *Microsoft Flight Simulator 2020.* Wikipedia, la Enciclopedia Libre. Recuperado 18 de mayo de 2024, de https://es.wikipedia.org/wiki/Microsoft_Flight_Simulator_2020

Aeropuerto Internacional Felipe Ángeles. (s. f.). El Mirador. Recuperado 19 de mayo de 2024, de <https://elmirador.sct.gob.mx/despegue/aeropuerto-internacional-felipe-angeles-a-volar>

Acerca de GitHub y Git - documentación de GitHub. (s. f.). GitHub Docs. Recuperado 22 de mayo de 2024, de <https://docs.github.com/es/get-started/start-your-journey/about-github-and-git>

Howto:Make an airport - FlightGear wiki. (2021, 12 junio). Recuperado 22 de mayo de 2024, de https://wiki.flightgear.org/Howto:Make_an_airport#:~:text=The%20very%20first%20step%20before,is%20all%20done%20in%20WorldEditor.