Aircraft Simulators with Unparalleled Capabilities

FLAMES® and Unreal® Engine can be used to develop aircraft simulator systems, including computer generated forces simulations, that provide unparalleled capabilities and that can be developed quickly and inexpensively.

FLAMES® Simulation Framework

UNREAL ENGINE

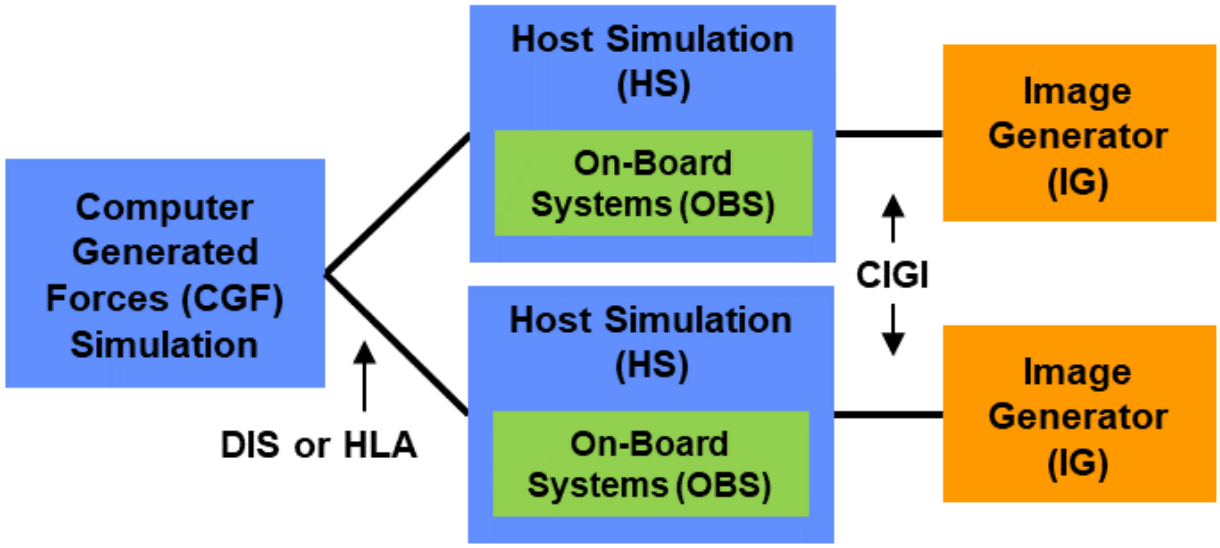
TERNION® CORPORATION
Simulations That Work℠

Aircraft Simulators with Unparalleled Capabilities

This paper compares typical aircraft simulators with enhanced aircraft simulators developed using FLAMES and Unreal Engine. The design principles presented in this paper also apply to simulators of vehicles that operate in other domains, such as ground vehicle and ship simulators.

Typical Aircraft Simulator

The figure below is a diagram of the design of a typical aircraft simulator. The sections that follow describe each of the major subsystems of this simulator.



Typical Aircraft Simulator

Not all aircraft simulators are designed exactly as shown above, but many have a very similar design. Examining this design will lay the foundation for understanding how aircraft simulators developed using FLAMES and Unreal Engine are different from all other simulators.

Image Generator (IG)

All aircraft simulators have some system for rendering the scene that is visible by the pilot. Typically, this scene is rendered by an image generator (IG), a stand-alone system designed specifically for rendering 3-dimensional (3D) scenes as realistically as possible. The scene is rendered on some type of display device or set of devices, such as one or more monitors or projectors or a virtual-reality (VR) headset.

Host Simulation (HS)

The host simulation (HS) is typically a stand-alone system that is responsible for simulating the host (motion) of the aircraft that is being flown by the pilot. This system responds to controls, such as a flight stick and throttle, that the pilot uses to fly the aircraft. High-fidelity systems may include a motion platform that helps give the pilot the sense that the aircraft is moving.

Computer Generated Forces (CGF) Simulation

The constructive or computer generated forces (CGF) simulation is responsible for simulating all of the entities in the simulation other than the aircraft being flown by the pilot. These could include friendly and hostile aircraft, ground threats (such as surface-to-air missile (SAM) sites), and stationary or moving ground targets. The sensors, weapon systems, munitions, jammers, and communication devices, etc. of these entities are also modeled by the CGF simulation.

On-Board Systems (OBS)

The host simulation (HS) is usually responsible for modeling the on-board systems (OBS) that exist on-board a real aircraft, such as sensors, weapon systems, munitions, jammers, and communication devices. These systems interact with the entities that are simulated within the computer generated forces (CGF) simulation.

Connectivity

The subsystems of typical aircraft simulators usually execute on separate computers. Therefore, network-based communication systems are used to allow the subsystems to connect to and exchange data with each other. Often, the following standard network communication protocols are often used:

- The HS and IG often communicate using the Common Image Generator Interface (CIGI). The state of the host and of all of the entities simulated within the CGF must be transmitted to the IG using CIGI.

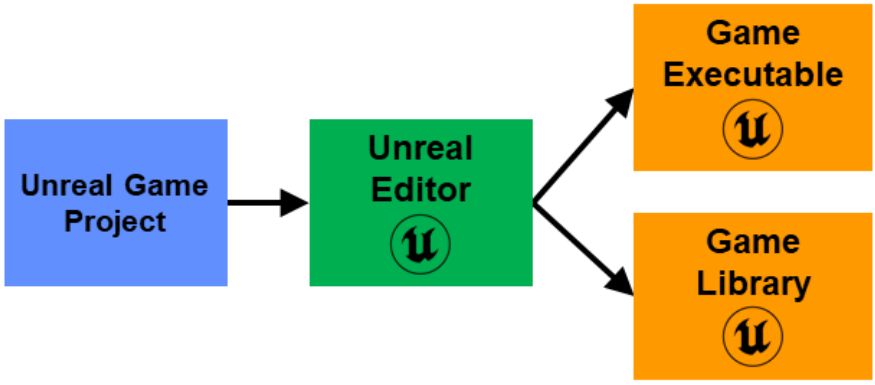
- The CGF and HS often communicate using the Distributed Interactive Simulation (DIS) protocol or the High Level Architecture (HLA). In order to allow the HS on-board systems (OBS) to interact with the entities in the CGF simulation, the properties and state of all the entities in the CGF simulation must be continuously transmitted to the HS.

Using FLAMES and Unreal Engine

The remainder of this paper describes how enhanced aircraft simulators with unparalleled capabilities can be developed using FLAMES and Unreal Engine. To provide some background,

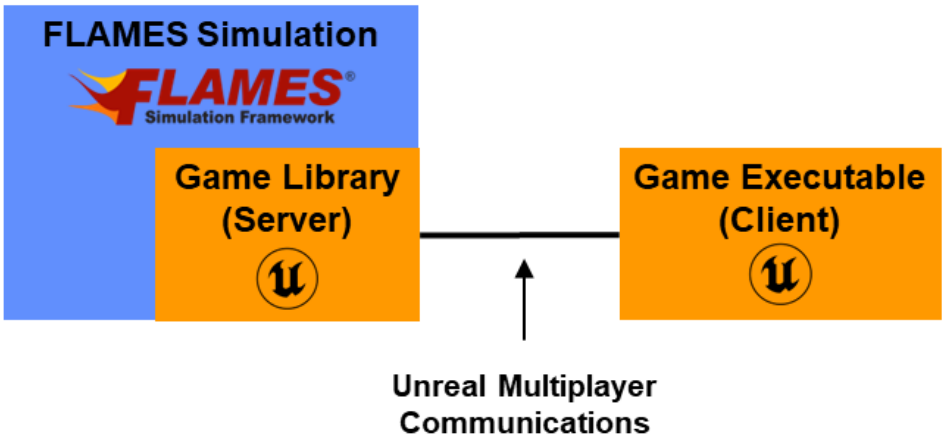
this section provides a brief overview of how games developed using Unreal Engine can be directly integrated into FLAMES simulations.

There is a little-known but extremely important fact about the Unreal Editor (the application from Epic Games that is used to develop Unreal games). In addition to building games as executable programs, the Unreal Editor can also build games as libraries (specifically dynamic link libraries (DLLs)). To integrate a game into a FLAMES simulation, the same game project that is used to build a game executable program is built a second time to create a game library.



Building a Game as a Library

The FLAMES Unreal Engine option allows a game library to be directly integrated into a FLAMES simulation. When the simulation is executed, the game library is executed within the simulation as an Unreal game server. As a result, the simulation can exchange data using Unreal Multiplayer Communications with one or more game executables that are executed as Unreal game clients.

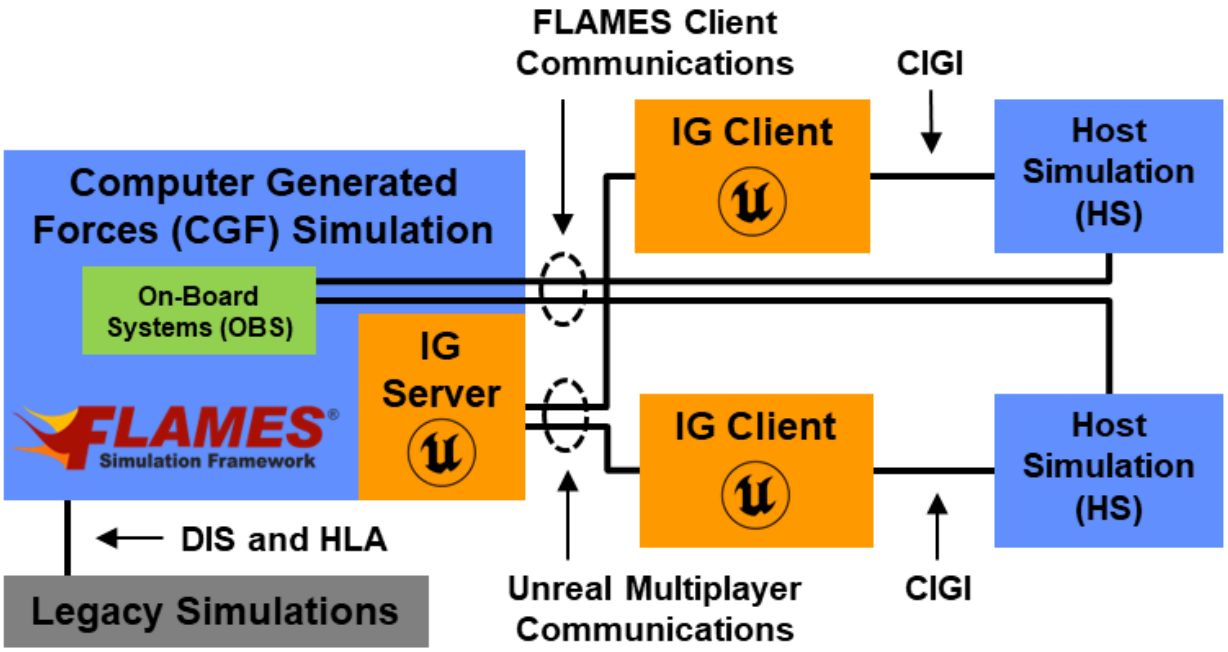


Integrating a Game Library into a FLAMES Simulation

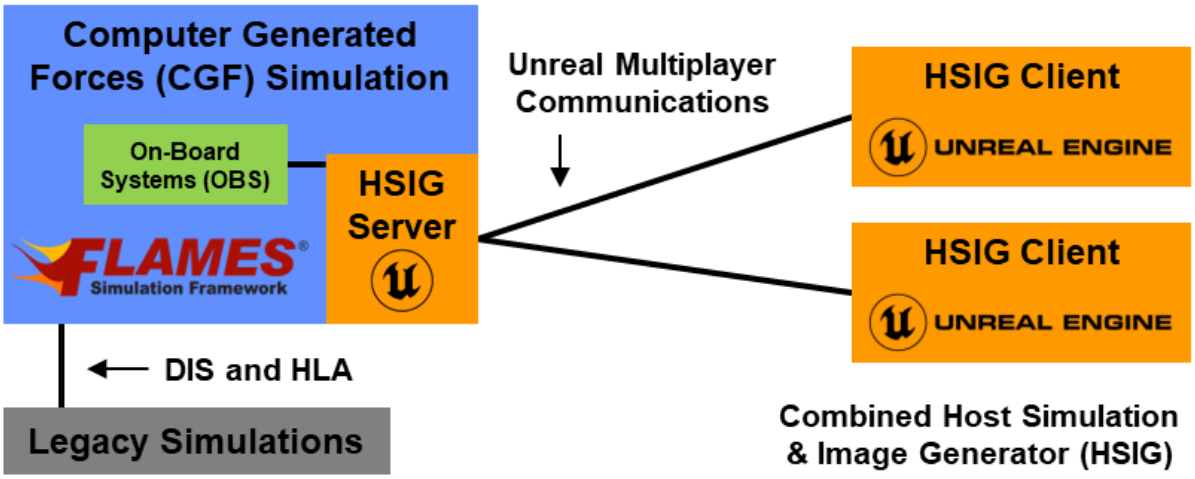
The ability to integrate Unreal games directly into FLAMES simulations allows fundamental changes in the design of aircraft simulators, as described below.

Enhanced Aircraft Simulators

The figures below illustrate two of the ways that FLAMES and Unreal Engine can be used to enhance the capabilities of aircraft simulators. One simulator uses an IG that communicates with the host simulation (HS) using CIGI. In the other simulator, the HS and the IG are combined into a single application. The sections that follow describe the major subsystems of these enhanced simulators.



First Enhanced Simulator – IG That Uses CIGI



Second Enhanced Simulator – Combined HS and IG

Image Generator (IG) That Uses CIGI

In the first aircraft simulator displayed above, the IG is developed using Unreal Engine. This allows the IG to exploit the industry-leading 3D content creation and 3D rendering capabilities of Unreal Engine. The IG continues to communicate with the HS using CIGI and thereby allows the use of existing, CIGI-compatible host simulations (HS).

Host Simulation (HS)

In the first aircraft simulator displayed above, the host simulation (HS) is very similar to the HS of a typical aircraft simulation. However, the HS is simplified and enhanced by moving the simulation of the on-board systems (OBS) to the CGF, as described below.

Combined Host Simulation and Image Generator (HSIG)

In the second aircraft simulator displayed above, the host simulation (HS) and the image generator (IG) are developed as a single, combined application (HSIG) using Unreal Engine. This allows host simulation and image generation to be performed in the same application without any intervening network communications (which means the use of CIGI is eliminated). In this simulator, the simulation of the on-board systems (OBS) is also moved to the CGF, as described below.

Computer Generated Forces (CGF) Simulation

The computer generated forces (CGF) simulation is developed using Ternion's FLAMES Simulation Framework. As in a typical aircraft simulator, the CGF simulation is responsible for simulating all of the entities in the simulation other than the aircraft being flown by the pilot.

One of the unique capabilities of FLAMES that is exploited in the CGF simulation is FLAMES' ability to directly integrate and execute a game developed using Unreal Engine (as described earlier). One or more instances of the Unreal-based IG or HSIG are executed as Unreal game clients. Another instance of the same game is executed directly within the FLAMES-based CGF as an Unreal game server. Within the CGF simulation, there is zero-latency, near seamless interaction between the modeling processes taking place in the FLAMES simulation and the processing taking place in the IG or HSIG game server.

On-Board Systems (OBS)

In the enhanced simulators illustrated above, the on-board systems (OBS) that exist on-board a real aircraft, such as sensors, weapon systems, munitions, jammers, and communication devices, are simulated within the CGF simulation. This allows the OBS to have direct, zero-latency interaction with the entities that are simulated within the CGF simulation. This also eliminates that need to transmit the properties and state of the entities in the CGF simulation to the HS or HSIG.

Connectivity

In both enhanced aircraft simulators illustrated above, the IG/HSIG server and IG/HSIG clients communicate using the Unreal Multiplayer Communication system. All of the state information for all of the entities in the CGF is sent to the IG/HSIG clients using this system. In the first simulator, CIGI is still used to send host state information to the IG.

In both aircraft simulators, the state information for all of the entities in the CGF does not need to be sent to the IG/HSIG using DIS or HLA. DIS or HLA is used only to communicate with external, legacy simulations.

In the first aircraft simulator, the one that uses CIGI, the FLAMES client communication system is used to allow the HS to interact with and control the on-board systems (OBS). In the second simulator, Unreal Multiplayer Communications is used to allow the HS to interact with and control the on-board systems (OBS).

Features and Benefits

This section compares the design of typical aircraft simulators with the design of enhanced aircraft simulators and explains the features and benefits of the enhanced design.

Open Architecture

The enhanced aircraft simulators are developed using FLAMES and Unreal Engine. Both products are application-independent and provide an open-architecture framework/platform for developing custom applications. In addition, both the FLAMES Developer and the Unreal Engine Editor are available for FREE. Further, both FLAMES and Unreal Engine are mature, polished development frameworks that greatly simplify and reduce the cost of the development of complex applications. Developing extremely capable applications with FLAMES and Unreal Engine is very simple and inexpensive when compared to attempting to develop similar applications from scratch.

Virtual World Modeling and Rendering

The virtual world (terrain, buildings, roads, trees, bodies of water, etc.) must be modeled in each subsystem of an aircraft simulator. In the IG, the virtual world must support high-framerate, visually realistic 3D rendering. In the HS and CGF, the virtual world must support many different types of mathematical operations, such as height-above-terrain, terrain intersection, and line-of-sight masking calculations. The virtual world must also support the calculations that are performed by ground-vehicle models.

Virtual world modeling presents some very difficult challenges for aircraft simulators. The virtual world must exist in a format that is optimized for each subsystem and that is compatible with

the different internal architectures of each subsystem. However, the virtual world geometry must be perfectly correlated in each subsystem, otherwise calculations that include the geometry of the virtual world will produce different results.

There are many approaches to addressing these virtual world modeling challenges, but no approach is more advanced and more elegant than the approach employed by the enhanced aircraft simulators described above. In these aircraft simulators, the virtual world in the IG/HSIG is defined in and managed by Unreal Engine. Because the IG/HSIG is executed directly within the CGF, the CGF is able to directly use the virtual world in the IG/HSIG. Therefore, there is only one virtual world that is used in the entire aircraft simulator system, and virtual world geometry correlation issues do not exist.

Connectivity

As has been explained, typical aircraft simulators often use CIGI, DIS, and HLA protocols to allow data to be exchanged between the IG, HS, and CGF. Developing applications that support CIGI, DIS, and HLA (especially HLA) is quite complicated, which makes the development expensive, time consuming, and prone to error. However, complexity is not the worst aspect of these protocols. Much more troublesome is their lack of capability. These protocols can work fine for exchanging the spatial state of entities between applications. But they do not provide good support for exchanging the detailed data that is necessary to support the high-fidelity events and interactions that take place in the real world.

For example, it is difficult to send enough data between the CGF and the HS to allow electronic warfare to be simulated (such as a self-protection jammer on the aircraft in the HS jamming the radar of an enemy surface-to-air missile system). As another example, it is difficult to get an explosion that is taking place in the CGF to be communicated to the HS and then to the IG such that the explosion is rendered realistically in the IG. As still another example, if there is a ground vehicle moving in the CGF, it is difficult to communicate the state of the vehicle to the HS and the IG such that the IG can display the wheels of the ground vehicle turning and bouncing over rocks. It is easy to come up with many other examples of detailed data that is difficult to exchange between the subsystems.

In the enhanced aircraft simulators, the exchange of complex data such as that described in the previous paragraph is not only possible, it is also relatively simple to do because of the unique and advanced aspects of the design:

- In the case of the first enhanced simulator, the one that uses CIGI, CIGI is used only to transmit the state of the host. No other information is transmitted using CIGI. In the case of the second enhanced simulator, there is no network communication at all between the HS and the IG, since these two subsystems are combined into one.

- Network communication between the CGF and the IG/HSIG is performed using Unreal's multiplayer communication system. This system is much more advanced than DIS, HLA, and CIGI and can easily send the data necessary to render such things as explosions, dust clouds, and bouncing wheels. This system is also much easier to program due to the extensive, built-in support provided by Unreal.

- Perhaps most important, the host on-board systems (OBS), such as the sensors, weapon systems, munitions, jammers, and communication devices, are modeled in the CGF together with the systems on-board all the constructive entities in the CGF. Therefore, all of these systems are able to interact with each other directly without any latency or intervening network communications. This allows higher fidelity and higher performance modeling of complex processes, such as weapon engagements and electronic warfare. It also allows the OBS and the systems on all the constructive entities to be modeled in the same way, thereby creating a "level playing field" for modeling.

FLAMES Demonstration Aircraft Simulator

Beginning with version 22.1 of FLAMES, a complete aircraft simulator with a combined host simulation and image generator (HSIG) and a FLAMES CGF simulation as described in this paper will be available for FREE in the FLAMES Store. The simulator will be available in a FLAMES content item that can be downloaded and executed immediately. It will also be available in source form, including the source code to all of the FLAMES models and the source to the entire Unreal Engine game project. This source can be used to learn how to modify your existing aircraft simulators to support the enhanced capabilities described in this paper, or it can be used as the starting point for developing your own, new aircraft simulator.

Conclusion

FLAMES and Unreal Engine allow constructive and virtual simulations to be developed that support capabilities that are not available in other simulations. In addition, the open and mature architectures of FLAMES and Unreal Engine allow simulations to be developed relatively quickly and inexpensively. Learn more about FLAMES and the FLAMES Unreal Engine option, and download the FLAMES Developer for FREE, at flamesframework.com.

About FLAMES®

FLAMES is a family of commercial off-the-shelf (COTS) software products that provide a framework for developing custom constructive and virtual simulations and interfaces between live, virtual and constructive (LVC) simulations. The optional integration with Unreal® Engine extends FLAMES to provide the ultimate framework for the creation of serious games and visually stunning, entity-level constructive and virtual simulations. For more information on FLAMES, visit flamesframework.com.

About Ternion Corporation

Ternion Corporation is the developer of FLAMES and an expert in developing custom, FLAMES-based simulations for government and commercial organizations worldwide. To learn more about Ternion's past projects and how Ternion can help you build your constructive and virtual simulations or build them for you, visit ternion.com.