FLAMES®
Simulation Framework

THE FLAMES UNREAL ENGINE OPTION

THE FLAMES® UNREAL ENGINE OPTION TIGHTLY INTEGRATES FLAMES, THE WORLD'S MOST CAPABLE CONSTRUCTIVE SIMULATION FRAMEWORK, WITH UNREAL® ENGINE, THE WORLD'S MOST OPEN AND ADVANCED REAL-TIME 3D CREATION TOOL, TO CREATE THE ULTIMATE FRAMEWORK FOR THE CREATION OF SERIOUS GAMES AND 3D, ENTITY-LEVEL CONSTRUCTIVE AND VIRTUAL SIMULATIONS.



PRECISE SYSTEMS
SCALE YOUR FUTURE

THE FLAMES UNREAL ENGINE OPTION

The FLAMES Unreal Engine option allows games developed using Unreal Engine to be directly integrated into simulations developed using FLAMES. Data is automatically shared between the simulation and the game to create a single, shared synthetic environment. This near-seamless integration of FLAMES and Unreal Engine makes all the features of FLAMES and Unreal Engine available simultaneously in a single application. The powerful, flexible, and open architectures of the FLAMES Developer and the Unreal Engine Editor allow you to quickly create your own serious games and constructive and virtual simulations with stunning capabilities.

This paper begins with a high-level overview of some of the features and benefits of using FLAMES and Unreal Engine together. Later in the paper, more detailed information is provided about how to use the FLAMES Unreal Engine option, how FLAMES and Unreal Engine work together, and how to create your own FLAMES simulation that integrates an Unreal Engine game.

WHAT UNREAL ENGINE BRINGS TO FLAMES

The following is a list of some of the features and benefits of Unreal Engine that are available when an Unreal Engine game is integrated into a FLAMES simulation.

- Powerful tools for creating 3D worlds and 3D objects
- High-performance 3D rendering capability, including stunning visual and audio effects
- Detailed physics functions for simulating the movement of objects, especially ground vehicles and humans
- Detailed physics functions for simulating collisions, blast effects, environmental effects, and more
- Robust support for multi-player games which allows Unreal Engine-based virtual simulators to integrate with FLAMES-based constructive simulations with a level of fidelity, realism, and performance that is not possible with DIS and HLA
- A free game development system, the Unreal Editor, available for download from unrealengine.com
- An immense library of existing, editable 3D content
- A massive support community and a wide variety of compatible third-party products
- All the other powerful features and capabilities of Unreal Engine

WHAT FLAMES BRINGS TO UNREAL ENGINE

The following is a list of some of the features and benefits of FLAMES that are available when an Unreal Engine game is integrated into a FLAMES simulation.

• Robust support for simulating the movement of air and space-based vehicles and munitions Robust support for simulating sensors, data processors, weapon systems, and electronic warfare

• Robust support for simulating command, control, communications (C3), and logistics

• Human cognition modeling that realistically simulates real-world behaviors and tactics

• Robust support for interfaces to external and real-world systems, including live C3 systems

• A powerful editor for creating and executing scenarios without the need to rebuild the Unreal Engine game (resulting in infinite use cases for a single game)

• Flexible, interactive control over players during scenario execution

• Simultaneous 2D and 3D views in the editor during scenario editing and execution, both rendered by Unreal Engine

• Complete elimination of “terrain correlation” issues; FLAMES simulations use the Unreal Engine world as the terrain

• A free simulation development system, the FLAMES Developer, available for download from flamesframework.com

• A library of existing, editable modeling classes and scenarios

USING THE FLAMES UNREAL ENGINE OPTION

The following is a list of some of the features and benefits of FLAMES that are available when an Unreal Engine game is integrated into a FLAMES simulation.

1. Follow the instructions at https://flamesframework.com/getting-started/ to download and install the FLAMES Launcher and the FLAMES Developer on your computer.

2. Follow the instructions at https://flamesframework.com/support/installation-and-setup/installing-flames-content/ to download and install the FLAMES Starter Content and one of the Unreal content items that contains an example, FLAMES-compatible Unreal Engine game, such as the Unreal Tutorial Content or the Unreal Camp Pendleton Content. As described in these instructions, use the FLAMES Launcher to enable the FLAMES Unreal Engine option and select the Unreal example game that you installed. (Note that Unreal Engine does not need to be installed on your computer to execute FLAMES with an Unreal game.)

3. Start FORGE, the FLAMES scenario editor, and open one of the scenarios that was installed when you installed the Unreal content. Click the Play buton in FORGE to execute the scenario. (Detailed instructions can be found in the Unreal Example Content Documentation in the topic titled “Executing Unreal Example Scenarios”. This documentation is included in all Unreal content.)

4. Watch “Your First Hour with FLAMES” training videos on the flamesframework website to learn how to edit FLAMES scenarios using FORGE. At any time during scenario editing, you can click the Play buton to execute the scenario. The Unreal game does not need to be modified or re-built.

“UNDER THE HOOD” OF THE FLAMES UNREAL ENGINE OPTION

This section provides a high-level overview of how to create a FLAMES-compatible Unreal Engine game and how a game is integrated and used in FLAMES.

CREATING A FLAMES-COMPATIBLE UNREAL ENGINE GAME

FLAMES-compatible Unreal games are created using Epic Games’ Unreal Editor, just like any other Unreal game. However, they are packaged differently. Normally, the output from packaging a game using the Unreal Editor is an executable program. However, to integrate a game into FLAMES, the game must be packaged as a dynamic link library (DLL).

The binary version of the Unreal Editor available on the unrealeditor.com website is not able to package a game as a DLL. To get a version of the Unreal Editor that can package a game as a DLL, you must download and build the source code to Unreal Engine. Instructions for how to do this are available in the FLAMES Documentation in the topic titled “Building Unreal Engine”.

The best way to begin creating a FLAMES-Compatible game is to start from the Unreal project (in source form) for one of the example FLAMES-compatible Unreal games. You can download these projects from the FLAMES Store. Two of the projects available in the Store are the Unreal Tutorial Game Project and the Unreal Camp Pendleton Game Project. Detailed instructions for downloading and building these projects can be found in the Unreal Example Content Documentation in the topic titled “Downloading and Building Example Games”.) The Unreal Example Content Documentation provides a detailed description of the Tutorial game project beginning in the topic titled “FLAMESTutorial Game Overview”. You can study this documentation and the Unreal project source to get detailed information about the things that make a game FLAMES-compatible.

INTEGRATING AN UNREAL GAME IN FLAMES

As explained above, the output from packaging a FLAMES-compatible game using the Unreal Editor is a dynamic link library (DLL). When you download and install Unreal content from the FLAMES Store, the content includes a packaged game DLL. If you package a custom game using the Unreal Editor, you create your own game DLL. In either case, when you use the FLAMES Launcher to enable the FLAMES Unreal Engine option, you specify the game (i.e., the game DLL) that you want to use. Then, when a FLAMES application starts, it automatically loads the game DLL and integrates the game in the application.

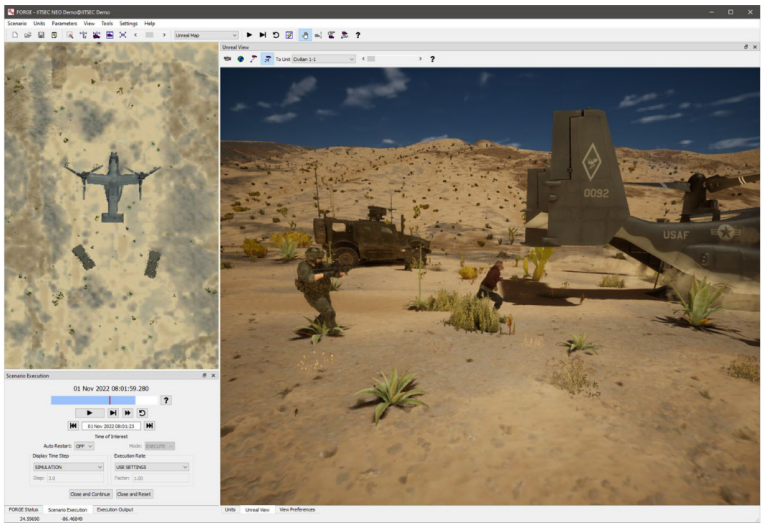


Figure 1: Simultaneous 2D and 3D Visualization in FORGE Rendered by Unreal Engine

HOW AN UNREAL GAME IS USED IN FLAMES

When an Unreal game is loaded and integrated in FORGE, the FLAMES scenario editor, several additional features and capabilities are available, including the following:

• The virtual world defined in the game (i.e., the game "level") is used directly as the terrain in the simulation. Because only one world/terrain is defined, "terrain correlation" issues do not exist.

• An additional window is available in FORGE called the Unreal View. This window displays the virtual world in 3D and is rendered directly by the game. This is the standard Unreal Engine 3D visualization. The Unreal View window is available during scenario editing (before scenario execution starts) and during scenario execution. The "camera" of the Unreal View can be controlled interactively, and it can also be controlled dynamically by players in the scenario during scenario execution.

• The 2D view of the scenario in the FORGE main window, the window where most of the scenario editing takes place, is also rendered by the game. This 2D view is also available during scenario editing and scenario execution. Therefore, FORGE provides a 2D and 3D view of the scenario simultaneously.

• Players in a scenario can be added, edited, and removed using the powerful and friendly graphical user interface provided by FORGE. Players exist simultaneously in FLAMES and in the game and remain fully synchronized at all times. Hence, FORGE provides an interactive way to edit a game without having to re-package the game. This can reduce game development time by days, weeks, or even months and is truly a "game changer" (pardon the pun).

• Three different types of players can be created, each of which is described in the next section.

• FLAMES effects (a standard modeling class supported by FLAMES) can directly generate Unreal damage, impulse, and visual effects within the game.

An Unreal game can be used in another FLAMES application called FIRE. FIRE has no graphical user interface and is designed to execute scenarios in batch mode or on a server. Unreal is used in FIRE just like it is used in FORGE except for graphics related operations.

THREE TYPES OF PLAYERS

When an Unreal game is used in FLAMES, three types of players are supported. In FLAMES, players are referred to as "Units" (not to be confused with a military unit such as a company or squadron). For the purposes of this paper, players in an Unreal game are referred to as "entities". For each Unit that exists in a FLAMES scenario, an entity of the corresponding type exists in the game (provided the FLAMES scenario and game are defined properly). The three types of Units/entities can be used in any combination.

This section provides an overview of each type of Unit/entity.

FLAMES AUTONOMOUS UNITS (UNREAL SURROGATE ENTITIES)

FLAMES Autonomous Units are created in FLAMES and are controlled completely by models that execute in FLAMES. A corresponding Unreal Surrogate entity is created automatically in the Unreal game when the Unit is created. During scenario editing and execution, FLAMES automatically updates the spatial state (position, orientation, velocity, etc.) of the entity in the game to mirror the state of the FLAMES Unit.

FLAMES Autonomous Units can take advantage of the excellent support provided by FLAMES for modeling sensors, data processors, weapon systems, munitions, electronic warfare, command, control, communications (C3), human behavior, tactics, and logistics. Autonomous Units can also interact with external and real-world systems, including live C3 systems.

FLAMES also has excellent support for modeling the movement of vehicles that do not have complex interactions with the terrain and that do not collide with other vehicles and objects. Therefore, it is often recommended to simulate air and space vehicles and munitions (missiles, bombs, etc.) as FLAMES Autonomous Units.

Autonomous Units can be created and edited in a scenario without having to rebuild the game.

FLAMES HYBRID UNITS (UNREAL HYBRID ENTITIES)

FLAMES Hybrid Units are very similar to FLAMES Autonomous Units. Hybrid Units are created in FLAMES and are controlled by models that, with the exception of the vehicle/character motion model, execute in FLAMES. A corresponding Unreal Hybrid entity is created automatically in the Unreal game when the Unit is created.

The vehicle/character motion model of a Hybrid Unit is simulated in the game by the corresponding Hybrid entity. During scenario editing and execution, FLAMES automatically updates the spatial state (position, orientation, velocity, etc.) of the Hybrid Unit in the scenario to mirror the state of the Hybrid entity in the game.

FLAMES Hybrid Units have all the advanced modeling capability of Autonomous Units with the added benefits of modeling vehicle movement in Unreal. Unreal has exceptional support for modeling the movement of vehicles and characters that have complex interactions with the terrain and that can collide with other vehicles and objects. Therefore, it is often recommended to simulate vehicles and humans that move on the ground/water as FLAMES Hybrid Units.

Hybrid Units can be created and edited in a scenario without having to rebuild the game.

FLAMES HYBRID UNITS (UNREAL HYBRID ENTITIES)

Unreal Autonomous entities are created in the Unreal game and are controlled by software that executes in the game. They can be Unreal Players (controlled by a human) or Unreal AI Entities (controlled by software). FLAMES automarically creates corresponding FLAMES Surrogate Units for each Unreal Autonomous entity defined in the game. During scenario editing and execution, FLAMES automatically updates the spatial state (position, orientation, velocity, etc.) of the Surrogate Unit in the scenario to mirror the state of the Autonomous entity in the game.

The vehicle/character mo□on model of a Hybrid Unit is simulated in the game by the corresponding Hybrid en□ty. During scenario edi□ng and execu□on, FLAMES automa□cally updates the spa□al state (posi□on, orienta□on, velocity, etc.) of the Hybrid Unit in the scenario to mirror the state of the Hybrid en□ty in the game.

FLAMES Hybrid Units have all the advanced modeling capability of Autonomous Units with the added benefits of modeling vehicle movement in Unreal. Unreal has excep□onal support for modeling the movement of vehicles and characters that have complex interac□ons with the terrain and that can collide with other vehicles and objects. Therefore, it is o□en recommended to simulate vehicles and humans that move on the ground/water as FLAMES Hybrid Units.

Hybrid Units can be created and edited in a scenario without havin

FLAMES SURROGATE UNITS (UNREAL AUTONOMOUS ENTITIES)

Unreal Autonomous en□□es are created in the Unreal game and are controlled by so□ware that executes in the game. They can be Unreal Players (controlled by a human) or Unreal AI En□□es (controlled by so□ware). FLAMES automa□cally creates corresponding FLAMES Surrogate Units for each Unreal Autonomous en□ty defined in the game. During scenario edi□ng and execu□on, FLAMES automa□cally updates the spa□al state (posi□on, orienta□on, velocity, etc.) of the Surrogate Unit in the scenario to mirror the state of the Autonomous en□ty in the game.

FLAMES Surrogate Units can be detected and engaged by other Units in FLAMES. FLAMES models other than platform models can also be atached to Surrogate Units which allows FLAMES modeling to be performed on behalf of the corresponding Unreal Autonomous entity. The main disadvantage of Autonomous entities is that they cannot be created or edited without rebuilding the game. Therefore, the use of Unreal Autonomous entities is usually not recommended. One notable exception is player-controlled entities. If the entity is intended to be controlled by a human player in an Unreal client application, the entity must be an Unreal Autonomous entity.

MULTIPLAYER GAMES

The FLAMES Unreal Engine option supports the integration of Unreal games that make use of almost every available feature of Unreal Engine including Unreal's extensive support for multiplayer games. In multiplayer games, the game DLL loaded in FORGE or FIRE is executed as an Unreal "listen server". During execution, multiple stand-alone Unreal game client applications can be executed and connected to the server. The server and the clients communicate using Unreal multiplayer network communications, which is superior to DIS and HLA.

State-of-the art virtual simulations can be developed using Unreal Engine. Executing virtual simulations as clients to a FLAMES-based constructive simulation server provides exceptional capabilities. Contact Ternion for more information.

CONCLUSION

FLAMES and Unreal Engine are both leading products in their respective industries. The FLAMES Unreal Engine option tightly integrates these products to allow the strengths of each product to be employed in the creation of serious games and 3D, entity-level constructive and virtual simulations. A more open, powerful, and flexible combination of simulation development tools is not available anywhere in the world. Download the FLAMES Developer and Unreal Engine, for free, and see for yourself.

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 or to download the FREE FLAMES Developer, visit flamesframework.com.

ABOUT TERNION, A PRECISE SYSTEMS COMPANY

Ternion, a Precise Systems Company is the developer of FLAMES and an expert in developing custom, FLAMESbased 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.

FLAMES® Simulation Framework

TERNION® A PRECISE SYSTEMS COMPANY® Simulations That Work℠

UNREAL ENGINE


PRECISE
SYSTEMS
SCALE YOUR FUTURE