**Unreal Engine Report**

Liam Whitehouse – 103862481

**Overview:**

For this report I will be writing about Unreal Engine. This is a very popular commercial Engine in which multiple games have been made with, most notable are Fortnight, Payday 3, Tekken, Sea of Thieves, Star Wars Jedi: Fallen Order and many, many more. On top of these titles, Unreal Engine can create and build out games on multiple platforms such as Xbox, PlayStation and PC.

With the vast array of games Unreal Engine has made, the Engine has a multitude of Libraries and Frameworks that helps developers create and produce these amazing games and I will be going over a couple of them but know there are many more.

**Section (1)**

**Gameplay Ability System Framework:**

Overview:

This is a very powerful gameplay framework developed by Unreal Engine during the development of Paragon that focuses on handling complex gameplay mechanics regarding abilities, attributes, gameplay effects, visual effects and server replication. Due to Paragon’s development being cancelled, Unreal Engine released this framework as a downloadable plugin for developers to use to enhance their own games. Fortnight implements the Gameplay Ability System for all abilities, guns and effects that occur during gameplay and replicating those events to the server.

Components of the Gameplay Ability System:

The GAS is comprised of multiple components that allow it to perform its required tasks, these are:

1. Gameplay Abilities.
   1. This is a component that allows the character to cast Abilities. These abilities can be passive and provide buffs or de-buffs to a character or they can be activated and perform an effect on either a target, in an AOE or on the character that casts it. This is a very robust system in GAS as it allows the characters attributes to be altered and changed.
   2. Gameplay Abilities are extremely versatile and are derived from UObjects and can be activated on either the C++ side or in Blueprints.
2. Gameplay Attributes.
   1. These are the raw attributes the character has, these attributes normally consist of Health, Mana, Strength, Speed, Critical Damage, Damage and many more, these attributes are created by the developer and cater towards the type of game that they are creating. These attributes are tied to the characters Ability System and when they are affected, it can cause the character to either get stronger, die, resurrect and provide many more benefits. This is also replicated to provide consistency across both server and clients.
3. Gameplay Effects.
   1. Gameplay Effects are what changes the values in the characters attributes, whether it be a permanent change like adding 10 to health, a timed change, or increasing an attribute, Gameplay Effects cause this reaction to occur to the Attribute Set on the character.
4. Gameplay Cues.
   1. Gameplay Cues are the visual and audio effects of the GAS and are the physical representations of the ability, example is firing a Gameplay Ability that is a firebolt, a Gameplay Cue is used to provide the sound effects of the projectile and even the particle effects that are emitted from the projectile. These Gameplay Cues are linked closely to Gameplay Effects and are triggered when the Gameplay Effects are.

Flow of GAS (Gameplay Ability System):

The Ability System Component is the core of the GAS, this holds the characters Attribute Set, Abilities and handles replication. This Ability System Component is created in C++ but has a lot of helper functionality in Blueprints to make is easier for developers to be able to activate and use abilities. This is essentially the brain of GAS.

The Attribute Set of the Ability System component holds the characters attributes, these attributes are defined and created by the developer, these generally consist of Health, Mana, Strength and so on. This controls the characters power, if they have the necessary resources to cast an ability. These values can be used anywhere in code or in blueprints and is used to control the characters flow and these values are all replicated to the server as well, supporting online gameplay. An attribute set is created in C++ and as I mentioned can be called in other C++ or blueprints.

The Gameplay Ability are the abilities themselves; these are what get activated based on either a character’s input or condition happening and as for what they can do is entirely up to the developer. These abilities will have a cost and cooldown tied to them which are both Gameplay Effects. These abilities are given to an Ability System Component and can be activated either in C++ or in Blueprints making it extremely versatile. You can create hundreds of Gameplay Abilities that perform widely different actions like spawning a projectile, playing effects, altering an area and so on.

The Gameplay Effect class controls what attributes in the attribute set are affected and altered by either a Gameplay Ability or they can be executed on the Attribute Set itself. These effects have three types being:

* Infinite: The Gameplay Effect occurs until the Gameplay Effect is removed.
* Has Duration: The Gameplay Effect occurs until the duration is over.
* Instant: Instantly alters the attribute.

Gameplay Effects are also used to control the cost of Abilities the character casts and their cooldown timer. The Gameplay Effect is extremely powerful and versatile and can be created in either C++ or Blueprints.

Overall, this is a very powerful and robust Framework. The notable game that uses this Framework is Fortnight, the characters health, weapons, effects, replication are all handled by the Gameplay Ability System which creates an extremely fun experience for the players.

**Gameplay Framework:**

Overview:

This is the core games system of Unreal Engine and consists of the Game Mode, Player State, Controllers, Pawns, Cameras, Actors and many more. It is a collection of core classes that provides the developers with a modular framework and foundation to work upon to build a unique gameplay experience catered towards the game the develop is making.

Components of the Gameplay Framework:

1. The Game Instance.
   1. This is created and remains active and alive until the game is shut down. The Game Instance is a manager class that can be overridden that tracks gameplay and executes code when required. This is not replication therefore it exists independently from the server. This is primary where developers would put the functionality for game saving and for anything to persist between level loading should be in the Game Instance.
2. The Game Mode.
   1. This is instantiated (created) right after the level is loaded in the Game Engine and the world is created. The Game Mode is a server-based manager class that is inherited from the actor class. The Game Mode does not persist across different levels and is the heart of the Gameplay Framework managing the overall rules and structure of the gameplay session and instantiating the remaining actors upon its creation.
3. The Game State/Player State.
   1. This is a non-physical actor that is designed to track the state of the game and players in it. These classes replicate their state information between an authoritative server and all connected clients in the network. The Game State contains all logic and data relevant to all players in the game, such as scores, objectives, list of players and their respective states and so on. The Player State however handles data and logic relevant to its associated player, such as health, mana, ammo count and inventory (anything the developer wishes the player state to keep track of). A single Game State is created by the Game Mode and a Player State is created for each player that exists in the game.
4. The Game Mode.
   1. The Game Mode spawns in the players when they join or enter the game. The player primarily consists of a controller and a pawn. The Controller class handles the logic that dictates the players movement and actions in the game. The Pawn class is the physical manifestation of the player/character in the game world. A Controller possess a Pawn and directs the Pawns actions in the game. This is for both the Player and AI is there is an AI Controller with its own unique set of functionalities.
5. The Actor.
   1. The Actors class is the fundamental building blocks of a game made in Unreal Engine. Actors are an object that can be placed in a level such as a Camera, Static Mesh, Player, AI, or a Start Location. Actors support transformations, rotation, scaling and are spawned and destroyed through gameplay code. Actors are also containers that hold special types of objects called actor components. Different types of components are used to control how the actors move, react and behave in the game. The other main functionality of an Actor is the replication properties and functions call across the network during gameplay.

A screenshot of a computer

Description automatically generated

**Media Framework:**

Overview:

The Media Framework uses several assets to enable the playback of videos inside Unreal Engine. Videos can be scrubbed, paused, or rewound inside the Media Player asset and controlled through C++ or Blueprints.

Components of the Media Framework:

1. Media Source Types.
   1. There is a File Media Source which is used for media files stored on your personal device or in a shared local network file. This can load the entire media file into memory before playing it. Based on the size of the file, caching time may vary.
   2. Image Media Source is an asset that can be used for playback of image sequences on your device or a shared local network. The images must be in a supported format and name sequentially (MyImage01) for the Engine to discover and populate the remaining images in the sequence.
   3. Platform Stream Media supports overriding a Media Source on a per-platform basis. Suppose you want a video to play only on a given device like PS4, in the Media Sources section, you can designate which videos to play on which platforms. When using the Platform Media Source, a media source must be selected for every platform.
2. Media Sound Component.
   1. To hear any audio associated with a given video, we will need to create a Media Sound. The Media Sound provides channels, attenuation, concurrency and other audio related settings that you can use to define how the sound is perceived. When linked to a Media Player Asset, audio is attached to a video source will automatically playback along with the video.
3. Media Textured.
   1. The Media Texture Component provides the visuals from the Media Source assets. This can be included within materials that can then be applied to meshes in the levels such as a billboard, TV or monitor to make it appear as if a video is playing on that object.

**Section (2)**

**Unreal Engine:**

Overview:

Unreal Engine is a 3D Graphics Game Engine developed by Epic Games. This is an extremely powerful and widely used engine for not only Video Games but also Films and Media as well. Unreal Engine was originally showcased in 1998 with the game Unreal. Now, hundreds of developers use this Engine to not only release games but also prototype them as well.

Features:

Unreal Engine boasts a very large number of Features, ranging from Design Visualizations and Cinematic Experiences to very high-quality games across PC, Console, Mobile, VR, and AR. Unreal Engine gives you all the tools necessary to start, ship and grow a product. Some of these features that the engine includes are:

1. Pipeline Integration.
   1. Unreal Engine connects media production pipelines, with support for FBX, USD, and Alembic. Unreal Engine can read USD files from anywhere on disk without a time-consuming full import and write back changes to the file as it overrides.
   2. Unreal Engine supports Python scripting, enabling the developer to automate their workflow, construct asset management pipelines, automate data preparation workflows and procedurally lay out content in a level.
   3. Datasmith and seamless data conversion, Unreal Engine can convert entire scenes (including animation and metadata) from 3DS Max, Revit, SketchUp Pro, Cinema 4D and other DCC, CAD and BIM formats at a high fidelity with Datasmith.
   4. Visual Dataprep, Unreal Engine allows for developers to easily automate data preparation workflows with simple visual tools that lets us create a “recipe” of filters and operators that we can save and reuse on other scenes or projects. Make LODs, set up Lightmap UVs, substitute materials, and delete or merge objects together.
   5. ShotGrid Integration, this offers a streamlined connection with both upstream 3D asset data created by artists in other application like Maya, and downstream image data that needs to be reviewed by supervisors and directors.
   6. LiDAR point cloud support, this aggregates and uses huge datasets captured from the real world, with the ability to import, visualize, edit and interact with point clouds acquired from laser scanning devices directly within Unreal Engine.
2. World Building.
   1. The Unreal Editor, this is an integrated development environment available on Linux, MacOS, and Windows for content authoring and game level development. With support for Multi-User Editing, artists, designers, and developers can simultaneously make changes to the same Unreal Engine project in a safe and reliable way.
   2. Modelling, UVs and baking, this is built into the Unreal Engine as an extensive mesh creation and editing toolset that includes subdivision operations, dynamic sculpting tools, and geometry scripting. In addition to these, there’s a robust suite of UV creation and editing tools, and a useful selection of tools for baking textures and transferring mesh attributes.
   3. Landscape and terrain tools, this allows developers to create massive-scale open world environments and terrains with mountains, valleys, and even caves with the Landscape system. Add multiple height maps and paint layers and sculpt and paint them independently of each other. Developers can also non-destructively edit their landscape with a layer reserved for splines and create unique custom brushes in Blueprint.
   4. Procedural Content Generation, this enables developers to define rules and parameters to populate large scenes with Unreal Engine assets of their choice. In-editor tools make it fast and easy to create large worlds, while a runtime component means they it can react to gameplay or geometry.
   5. World Partition, this allows developers to automatically divide the world into a grid and streaming only necessary cells. Teams can work simultaneously on the same region of the world without treading on each other thanks to a One File Per Actor system.

Platforms:

Unreal Engine supports a large variety of platforms to provide a global experience. The platforms it allows developers to deploy projects on are Windows PC, PlayStation 5, PlayStation 5, Xbox Series X, Xbox Series S, Xbox One, Nintendo Switch, macOS, iOS, Android, ARKit, ARCore, OpenXR, SteamVR, Oculus, Linux and SteamDeck.

Console tools and code are provided for free to developers who are registered developers for their respective platforms.

Unreal Engine Versions:

Unreal Engine has gone through many iterations, updates and changes. We are currently on Unreal Engine 5.4.2 and throughout the Engines lifecycle it has been constantly upgraded and has additions added to it. These are the Engine versions and what were added to them.

|  |  |
| --- | --- |
| Version 5 | This is the latest major version of Unreal Engine, This Engine introduces major features such as Nanite, which allows for high-detailed photographic source material to be imported into games and makes it much easier to make detailed game worlds without having to spend excessive time creating new, detailed assets. Nanite also handles LODs but can only be used for Static Meshes.  Lumen lighting was introduced as a fully dynamic global illumination solution that immediately reacts to scene and light changes. Lumen eliminates the need for artists to craft a lightmap for a given scene and instead calculates the light reflections and shows on the go thus allowing for real-time behaviour of light sources.  Unreal Engine 5 also includes other addition features such as Virtual Show Maps which is a new shadow mapping method used to deliver consistent, high-resolution shadowing that works with film-quality assets and large, dynamically lit open worlds.  Some games on Unreal Engine 5 were Fortnight, Dark and Darker, Gears of War: E-Day, Satisfactory, Tekken 8, Ashes of Creation and Black Myth: Wukong. |
| Version 4 | Unreal Engine 4 was the first version to include a new visual programming language called Blueprints, this allowed for rapid development of game logic without using code, resulting in less of a divide in technical artists, designers and programmer. A UI toolkit was also introduced that simplified the User Interface and finally they introduced Physically based materials which allowed for Unreal Engine to better handle its shading model and embrace a more physically based material workflow.  Some games developed with unreal Engine 4 were Darksiders 3, Dead Island 2, Deep Rock Galactic, Ark: Survival Evolved, Astroneer and Days Gone. |
| Version 3 | Unreal Engine 3 started to make use of Multiple-Threading, adopted DirectX 9 and took advantage of fully programmable shader hardware which meant all lighting and shadowing calculations were done per pixel instead of per vertex.  Some games made with Unreal Engine 3 were Homefront, Gears of War 2, Gears of War 3, Gears of War Judgment, Injustice 2, Infinity Blade and Batman: Arkham City. |
| Version 2 | Unreal Engine 2 provided GPU acceleration and was the first to make use of Direct X, Multiple platform support (now supported PlayStation 2, Xbox, and GameCube), and also included particle systems and cinematic editing tools.  Some games made with Unreal Engine 2 were Tom Clancy’s Ghost Recon 2, Thief: Deadly Shadows and Alien Swarm. |
| Version 1 | This was the first generation of Unreal Engine. This Engine included Rendering, Collision Detection, Coloured Lighting, limited form of Texture Filtering and had a Level Editor.  Some games built with Unreal Engine 1 were Harry Potter and the Sorcerer’s Stone, Unreal Tournament and Dues Ex. |

These are the main versions of Unreal Engine, but with each main version release, there have been updates to these versions with the current being Unreal Engine 5.4.2. The Engine is always being updated and worked on to provide developers with the best possible tools for developing and producing not inly video games but media as well.