DEMON SLAYERS

Unreal Assignment

A horde-based FPS, with split-screen support

Daniel Bellido, Daniel Khatkar, Ollie Dunn

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

Contents

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

DESCRIPTION OF THE GAME

This video game, developed in Unreal, is part of the popular, First Person Shooter genre.The gameplay is based on shooting, managing resources, surviving, eliminating hordes of enemies and defeating the final boss.

Regarding the narrative; the player investigates a facility where terrible experiments are carried out with living beings, and a demonic entity has taken control of these creatures- endowing them with a diabolical mutation and a thirst for revenge.

The objective of the game is to exterminate the presence of all creatures that extend over two levels, as their existence endangers the survival of the human species.

In each level the player will have to survive a total of eight waves of enemies, in order to face the demon that creates and controls the enemies. The player will have to manage his most precious resources such as his own health and ammunition since this is finite and must be used wisely.

Their only aid, the player possesses a technology that slowly regenerates their shield and on some occasions the player will need to run away in order to regenerate this shield before facing the horde again.

In this version of the demo, the player only has an automatic rifle but the game is designed so that the player has access to other types of weapons, such as: shotguns, grenades; and melee weapons. All of which can be ‘bought’ with the points obtained by eliminating enemies.

The target platform of this project is designed to work mainly in systems that have a controller since the game is designed to meet this challenge cooperatively with another player. Some examples of these systems are; Nintendo Switch, PlayStation 4, Xbox and PC.

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

VISUAL DESIGN, LEVEL DESIGN AND GAME MECHANICS

The visual design is intended to be one of the main attractions for older users since it presents a realistic style, within a dark science fiction world that aims to inspire horror.

The aesthetics of the selected assets have a horrific theme, and an attempt has been made to correlate their appearance with a dangerous, cruel, and bloody world, fullof violence and action. Resources such as ambient music and SFX typical of horror movies have been included.

The user interface and the use of fonts do not adjust to the level proposed by the level of other elements such as the Main Menu, the animations and the aesthetics of the characters present a very attractive visual aspect for lovers of First-Person games shooters. In addition to enemies, there are environmental hazards such as fire that can damage the player which is visually communicated through the user interface with a bloody vignette indicating the player is taking damage.

Regarding the level design, an asymmetrical main level composed of different areas has been created. The facilities have a labyrinthine hangar with two levels, and a basement that communicates with an outside area creating a fully connected comfortable area so that the player cannot get trapped or hide from enemies.

The player has at his disposal, a series of basic mechanics that are easy to comprehend.Such as: aiming, shooting, reloading, jumping, crouching; and running. There is no explicit tutorial that indicates what the mechanics are, though the controls were designed to be intuitive, and follow the standard set by many games of the genre.The player starts the level above the ground floor, next to a railing where he sees the horde of enemies coming, giving him the option to shoot from a safe starting position. After a few moments the horde reaches the player's position leaving him cornered and he has the option to jump over the railing to run and find another safe position.

The indicator that the horde is approaching is conveyed through sound as when a wave of enemies approaches, they can be heard growling and shouting. Upon killing all enemies in the wave, the sound stops momentarily until a new wave respawns.

The arrival of the final boss is also announced with dialogue and with a particle effect and a unique character model that indicates that the enemy is a final boss.

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

The ranged enemy, this model has a larger acceptance radius than the others. Once the player is within range, the enemy's 'shout' animation (as seen in the image) is triggered. This contains an animation notify, which spawns a projectile socketed to the mouth of the middle head, and the projectile flies toward its current target.

On death, this enemy has ragdoll physics.

Health:100

Zombie mutant 2 in Characters - UE Marketplace (unrealengine.com)

The melee enemy, once it reaches the player it plays an attack animation montage, this contains an animation notify state which controls the opening and closing of an attack box at the appropriate animation frames. This specifically causes damage to the player, also avoidable.

On death, this enemy has ragdoll physics.

Health: 100
Damage: 0.25F

Monster_Flesh in Characters - UE Marketplace (unrealengine.com)

The explosive enemy, once the player is within its movement acceptance radius, a particle explosion is spawned, and radial damage is applied to all players within reach- this is avoidable.

On death, this enemy also explodes.

Health: 100
Damage: 0.4F

Human mutant 3 in Characters - UE Marketplace (unrealengine.com)

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

The projectile of the ranged enemy. Once spawned, it flies toward the targeted player. Upon collision it is destroyed after a brief delay, if the object it collides with is a player, it also deals damage. This allows it to be blocked by obstacles, as well as avoided with quick movements.

Damage: 0.1f

The player model- this character consists of fully blended running animations based upon both speed and direction of travel. It is the actor controlled by the player and the target of the enemy characters.

Armour: 1.0f
Health: 1.0f

Mixamo

The players weapon- the spawning of projectiles and amount of ammunition are controlled within this object. Its purpose is to provide the player with the ability to engage the enemies in combat. Each weapon contains a reference to the owning player, to allow this information to be passed onto the spawned projectiles.

https://virtushub.com/p/resources

The players projectile- these spawn at the muzzle section of the gun and flies in the direction the player shoots. Upon collision with enemy characters, it acts as the trigger for their receiving of damage. The spawn position is also used as a parameter for the rag doll effectto ensure the impulse is applied from the appropriate angle. Each projectile has their owner registered, which allows for the game to acknowledge which player fired the killing blow, and adjust their score accordingly

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

Sevarog, the boss enemy. Upon spawning, it rises from the ground at a designated point, accompanied with a particle effect and sound cue, which were provided in the same pack. The boss has a melee attack using a similar method as the base melee enemy, it's second stage was cut from the project, though is still available in the BP.

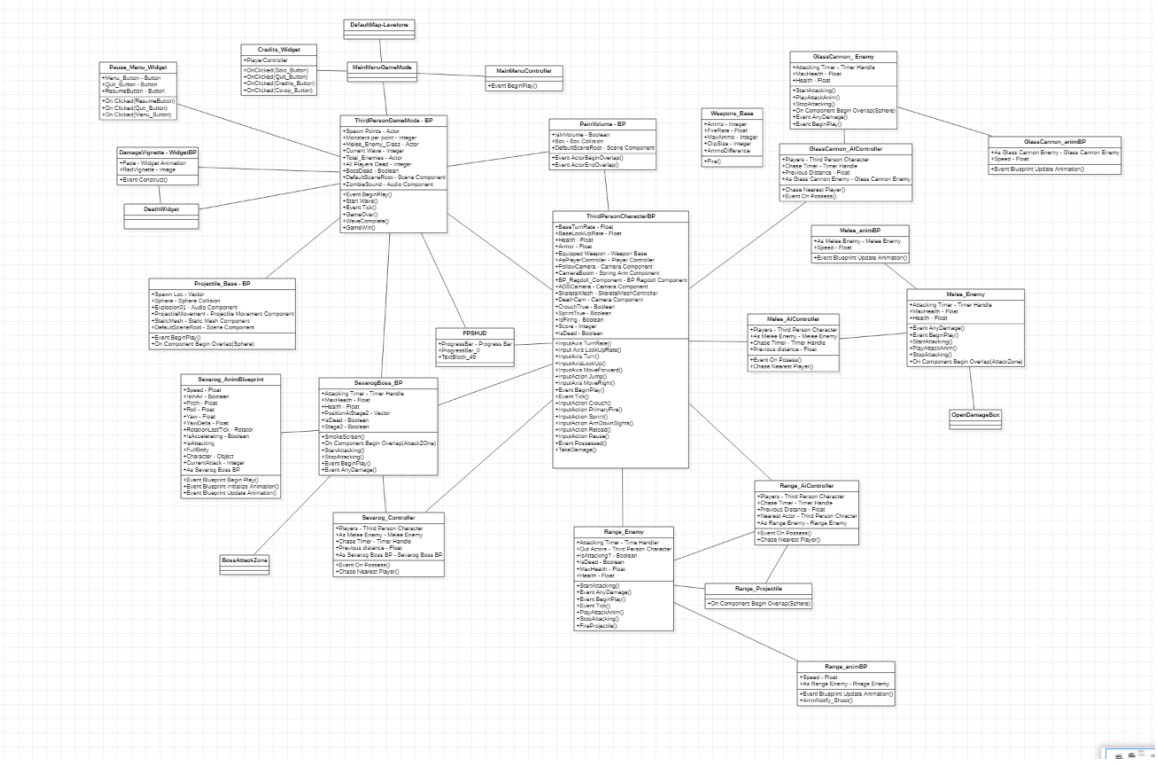
On death, the boss plays an animation, and dialogue as fire rises from the ground. Shortly after, the particle effect fades out and the actor is destroyed.

Health: 1000
Damage: 0.5f

Paragon: Sevarog in Epic Content - UE Marketplace (unrealengine.com)

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

TECHNICAL IMPLEMENTATION



As the diagram demonstrates, the project is the total sum of all its moving parts. Very few elements work in isolation from other blueprints, even if that relationship is as basic as passing information regarding a collision, to the relevant actors.
Central to the architecture is the ThirdPersonCharacter blueprint. While the GameMode controls spawning, win/lose conditions and more, even these features depend on interactions with the player, and the number of instances of the character blueprint present.
The player acts as the target of the enemies and is generally responsible for the enemy characters ability to interact with the world around them.
Essentially, the player is the central hub of relationships and behaviours within the game world, examples of this include how ownership affects the scoring (touched upon elsewhere in the report), as well as how the world behaves.

regarding inheritance, all active characters: enemies, boss and players are children of the Character base class, and inherit the basic framework of the class, while their blueprints differ (significantly, when considering difference between the player and enemies), they all inherit from the same root class.

Both ranged projectiles inherit from the actor class, whereas the Hud and the damage vignette inherits from the User Widget class. Both Game modes, MainMenuGameMode, and ThirdPersonGameMode inherit from the GameModeBase.

The enemy controllers are interesting, as they use a special type of controller called the DetourCrowdAIController, which is itself a derivate of the base AI controller, with steering behaviour-like adjustments to the movement component. Both this method, and RVO avoidance were tested, both together and independently, concluding that using the controller in isolation was the best approach. Currently the max agent number is 100, though the game in its current state does not have 100 controlled agents within the world at once.

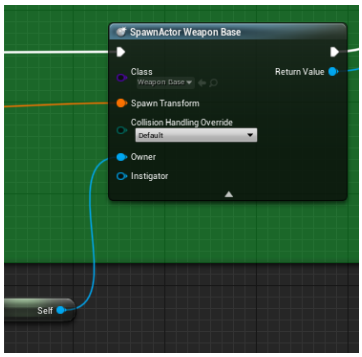
Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

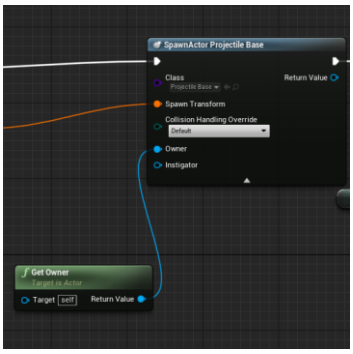
IMPLEMENTATION

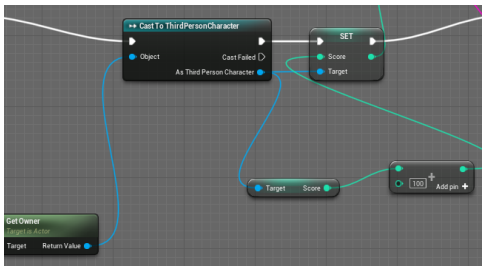
The resolution of issues pertaining to multiplayerwere some interesting areas of implementation.The inclusion of a second player required an approach to areas which would not be necessary on a single player game.

A few examples of this can be seen in the individual scores, the AI; and game-over conditions.

For the players to have accurate and separate scores, the enemy needs to register the ID of the player who fired the killing blow. This meant every bullet fired must know which gun fired it, and which player held the gun. To resolve this, the weapon registers the player holding it, as its owner upon spawning. This allows the bullets owner to also be set as the player upon spawn, which in turn enables the enemy to use the owner pin on the AnyDamage event to cast to the instigating player and increase the correct score at the appropriate time. This is also an example of the relationship between different BPs and how they handle the mechanics.

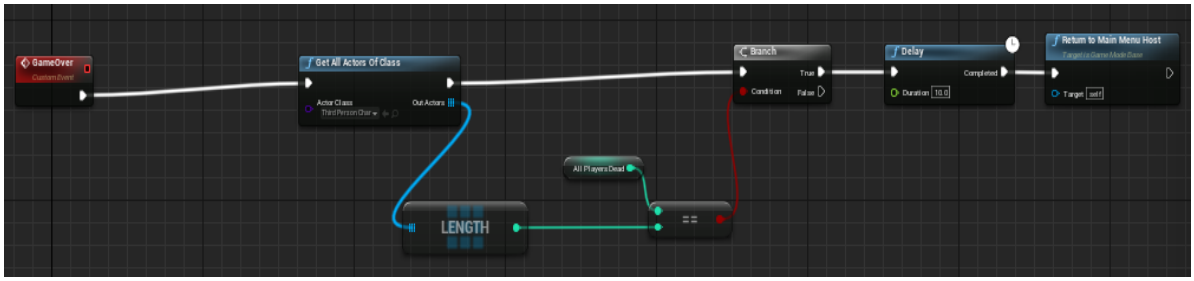




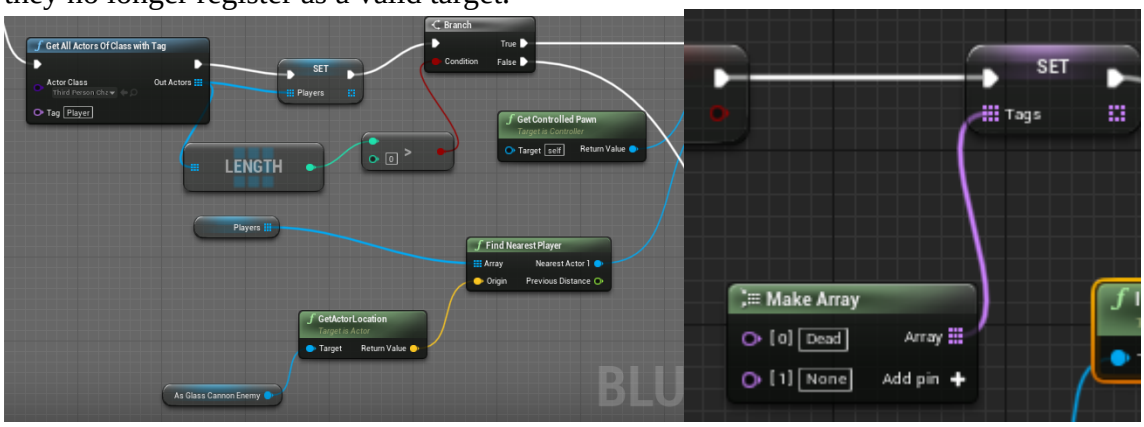


The game-over conditions needed to work for both single player and multiplayer, and therefore a variable solution was required. Upon the death of a player, shortly before un-possessing, the character blueprint casts to the game mode, increases the ‘AllPlayersDead’ variable by 1, and calls the GameOver event. This event checks the value of the variable against the length of an array of the player characters and returns to the main menu if the values are equal. Allowing the event to function for both modes.

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829



The AI needed a targeting system that would work with multiple players, while also being capable of differentiating between those dead, and alive. To do so, all players needed the tag ‘Player’. A custom event within the enemy would make an array out of actors with the right class and tag parameters, before calling a function to find the nearest player and moving toward them. This event is called periodically for the enemy to register which player was closest at that moment, and when a player dies, their tag is overwritten with a new one, so they no longer register as a valid target.



Collision is implemented using overlap events, and details settings dependant on the object in question. An example being how the destruction of projectiles is handled; upon the triggering of an overlap event, the projectiles check for a specific tag which is dependent on the instigator. If the tag is present, they apply the relevant damage. If it is not present, the projectile deletes after a brief delay.

TESTING AND EVALUATION

Testing has primarily been undertaken by the developers, with the occasional helping hand from their friends.

Originally, the plan was to use a public sharing platform to receive feedback, though delays in development resulted in a lack of available time to have this effectively performed, especially considering there would have been no guarantee on any response.

As a result, the team made use of each other.

Due to the project being group based, and each individual member having their own areas to specialise in, each member could provide feedback on are area addressed by another member. In turn, they would tweak their work in order to either fix a bug or produce an effect the team agreed to be more appropriate.

An example of this is perhaps best demonstrated by the navigation system and its application to the levels. Looking specifically at the single player level, members were able to share their understanding of both the level design, and the navigation mesh, to identify areas where potential issues could arise and fix them accordingly. The designer was able to utilise their knowledge of the level to take the AI through different paths, and return feedback to the member working on AI, regarding what worked and what issues arose.

Through this method, a myriad of bugs were resolved. Firstly, problems with specific enemies becoming stuck on staircases were identified and fixed by tweaking both the max walkable angle, and the shrinking of the agent radius, as well as the inclusion of nav proxy links to increase the available pathways through some of the narrower level sections. The congestion caused by enemies getting stuck behind one another was also identified and resolved using detour controllers.

Another area of improvement, through the same method of testing, is the player controls. Originally, upon moving left or right while aiming, the gun would snap left or right also. Through utilising a fresh set of eyes, the solution to the problem was found. It had seemed as though the problem laid within the animation itself. However, upon the other member inspecting the Blendspace, the preview showed a smoothly running animation, concluding that the issue likely arose somewhere else. As a result of testing, the problem was identified to be a checkbox within the players movement component, ‘orient rotation to movement’, which resulted in an immediate change in rotation, rather than the gradual change needed for the Blendspace.

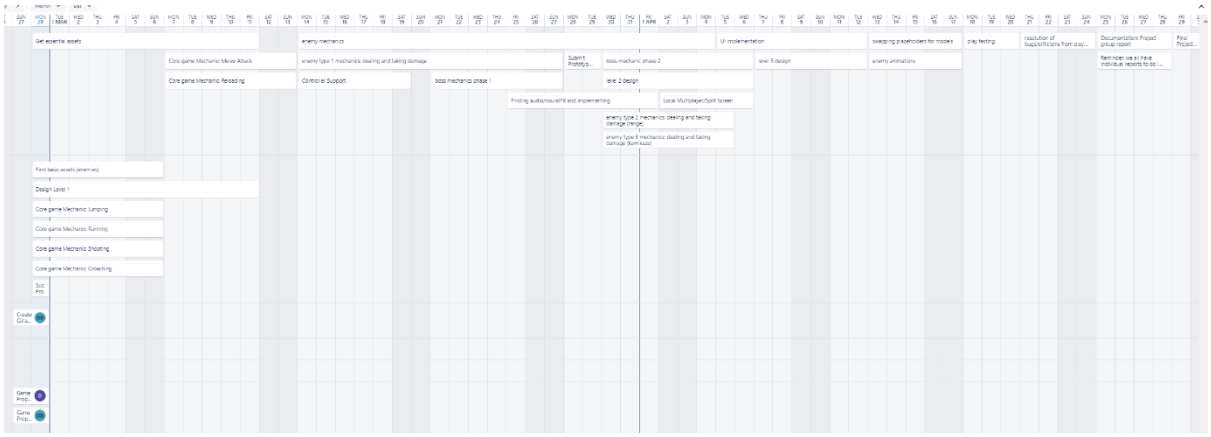
While certainly not the optimal method of testing, this approach stopped the individual members from being limited by preconceptions of their own ability, as shown by the final example, where one member believed the problem lied in their own work, though the issue was a default setting.

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

MANAGEMENT

On a planning and design level, the project was managed relatively well using a Trello board, and a designated discord server for the team to coordinate. It is of a somewhat smaller scale to the idea originally proposed; some features were cut in their entirety, such as health and ammunition pick-ups, whereas others were approached in a simpler way than originally planned, such as the AI being simplified to BPs with detour controllers, and the second boss stage being removed. Juggling both personal responsibilities and the development of the project led to a collective decision to avoid scope-creep toward the end of the assignment period and focus on fixing the features which were already present at the point of this decision. This was an entirely conscious decision, with the team deciding that having a workable submission was the most important thing, and that further development on the project to bring it to a standard worthy of presenting on their portfolios, could be undertaken over the holidays, after the submission deadline.

Most of the Must Haves from the proposal have been achieved, including local multiplayer which was identified as one during the proposal feedback. The outlier is the money system- though it was primarily not included due to the cutting of features rendering it obsolete, and the individual scoring system provides a usable framework for a COD Zombies-like system during future development.



LEGAL, SOCIAL, AND ETHICAL ISSUES

All assets present within the project have been appropriately licensed. There is a mixture of both assets freely available on the asset store, and paid assets. The enemy models, and the zombie sound FX were paid assets.

In accordance with the Unreal licensing agreement, license holders are free to use assets however they wish within a private setting. Article 3A of the Unreal EULA covers distribution, which is defined as sharing the project with any other entity, or person, outside of the license holder. It states that the license holder may 'permit end users to use, reproduce, display, and publicly perform… solely as incorporated in the Project'

Consequently, as the license holder is a member of the development team, and the assets are being used within a project they can claim as their own project, there is no breach of licensing.

The project has stayed aligned to many of the original concepts which defined its PEGI rating in the proposal, and therefore would still be PEGI 18. While the use of pharmaceuticals is not present; the violent nature of the game, in conjunction with its horrific theme would warrant targeting a mature audience.

Demon Slayers does not store any personal information on the player.

CONCLUSION

While the original scope of the game was not met in its entirety, the team are happy with their production and found it an important educational tool regarding both Unreal Engine, and cohesive development. The principle of group-based work required breadth of thought, as each member not only had to consider their own assigned tasks, but how their pieces would fit into the grander architecture, and its applicability alongside the work of others.

The project provided an insight to the depths of possibility that would come with mastery of the engine, an exciting teaser as to what one could produce upon growing their knowledge base. It was both indicative of how far the team had come this year, on an engine that was entirely new to many, and how much further the educational journey can go, which is equal parts exciting and daunting.

Development was an enjoyable, periodically frustrating process, which required the team to present their work to one another in a digestible fashion. Designing their blueprints in a way which would allow others to utilise the mechanics, without having to navigate complexities which would require them to stop their own work and further their understanding of another area. While developing understanding over a wide range of areas is not a bad thing, it would have resulted in a significant impact to productivity, and what would have been achievable in the given time.

The team has discussed, and currently plans to continue with the project. It is both a style of game they favour, and a project they feel that, with further development, could prove to be both portfolio worthy, and an enjoyable game to experience. For this reason, there are a few unfinished features still present in the project files. These are elements which, while not making it into the submitted game, will be further developed.

Unfortunately, due to the technical limitations of the computers used, the video is not as smooth as the team would like it to be, due to significant issues with frame rate while recording within the engine. This stutter is not present within the project itself.

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

ASSET LIST

Realistic starter VFX

Realistic Starter VFX Pack Vol 2 in Visual Effects - UE Marketplace (unrealengine.com)

Zombie Mutant 2

Zombie mutant 2 in Characters - UE Marketplace (unrealengine.com)

Human Mutant 3

Human mutant 3 in Characters - UE Marketplace (unrealengine.com)

Paragon: Sevarog

Paragon: Sevarog in Epic Content - UE Marketplace (unrealengine.com)

Monster_Flesh

Monster_Flesh in Characters - UE Marketplace (unrealengine.com)

Premium Zombie SFX

Premium Zombie SFX in Sound Effects - UE Marketplace (unrealengine.com)

Animation Starter Pack

Animation Starter Pack in Epic Content - UE Marketplace (unrealengine.com)

Grass Landscape Material

Grass Landscape Material Vol. I in Materials - UE Marketplace (unrealengine.com)

Infinity Blade: Fire Lands

Infinity Blade: Fire Lands in Epic Content - UE Marketplace (unrealengine.com)

Open World Demo Collection

Open World Demo Collection in Epic Content - UE Marketplace (unrealengine.com)

Landscape Pro 2.0

Daniel Bellido: K1925456
Daniel Khatkar: K1910497
Oliver Dunn: K1941829

Landscape Pro 2.0 Auto-Generated Material in Materials - UE Marketplace (unrealengine.com)

Interactive Open World Foliage

interactive Open World Foliage in Props - UE Marketplace (unrealengine.com)

Modular Neighbourhood Pack

Modular Neighborhood Pack in Environments - UE Marketplace (unrealengine.com)

Infinity Blade: Grasslands

Infinity Blade: Grass Lands in Epic Content - UE Marketplace (unrealengine.com)

Temperate Vegetation: Spruce

temperate Vegetation: Spruce Forest in Props - UE Marketplace (unrealengine.com)

Polar Sci-Fi Facility

https://www.unrealengine.com/marketplace/en-US/item/1da045dbf3a744b2aa5eb0bf20f6f32e

All sound effects from Metal Gear Solid

https://www.youtube.com/watch?v=NoS0sEhphEA

Mox Jade- Metroid Prime

https://www.youtube.com/watch?v=I15T7ptSFtU

Sepiks prime theme cover by Bulb

https://www.youtube.com/watch?v=tDCLYfUOhaQ

Swat Player Character
Mixamo

Gun
https://virtushub.com/p/resources

Reference List

3 - implementing projectiles (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/ProgrammingWithCPP/CPPTutorials/FirstPersonShooter/3/ (Accessed: April 28, 2022).

4 - adding character animation (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/CPPTutorials/FirstPersonShooter/4/ (Accessed: April 28, 2022).

Animation notifies (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/5.0/en-US/animation-notifies-in-unreal-engine/ (Accessed: April 28, 2022).

Animation rotation (2015) *Unreal Engine Forums*. Available at: https://forums.unrealengine.com/t/animation-rotation/26802 (Accessed: April 28, 2022).

Apply damage (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/Game/Damage/ApplyDamage/ (Accessed: April 28, 2022).

Aspland, M. (2020) *How to create A main menu - Unreal Engine 4 tutorial*. Youtube. Available at: https://www.youtube.com/watch?v=K1vVbwMJCTQ (Accessed: April 28, 2022).

Aspland, M. (2021a) *How to make AI turn smoothly | smooth rotation turning - unreal engine tutorial*. Youtube. Available at: https://www.youtube.com/watch?v=8zN9bQSX4fQ (Accessed: April 28, 2022).

Aspland, M. (2021b) *How to migrate content between unreal projects - Unreal Engine tutorial*. Youtube. Available at: https://www.youtube.com/watch?v=qEK-Dtk9Ank (Accessed: April 28, 2022).

Creating an Animation Montage (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimMontage/Creation/ (Accessed: April 28, 2022).

Get nearest actor to AIPawn (2017) *Unreal Engine Forums*. Available at: https://forums.unrealengine.com/t/get-nearest-actor-to-aipawn/4211/10 (Accessed: April 28, 2022).

Health bar isn't working (2015) *Unreal Engine Forums*. Available at: https://forums.unrealengine.com/t/health-bar-isnt-working/317503/4 (Accessed: April 28, 2022).

Laley, R. (2021) *Unreal engine 4 tutorial - find nearest actor of class*. Youtube. Available at: https://www.youtube.com/watch?v=aKzjNTadrAc (Accessed: April 28, 2022).

Markom3D (2019) *UE4 stop sound at location Blueprint tutorial*. Youtube. Available at: https://www.youtube.com/watch?v=uWrvr0MHIac (Accessed: April 28, 2022).

My Character Keeps falling through static mesh (2016) *Unreal Engine Forums*. Available at: https://forums.unrealengine.com/t/my-character-keeps-falling-through-static-mesh/53679/2 (Accessed: April 28, 2022).

Scale box slot (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/5.0/en-US/BlueprintAPI/Layout/ScaleBoxSlot/ (Accessed: April 28, 2022).

Set disable collision (no date) *Unrealengine.com*. Available at: https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/Physics/Components/PhysicsConstraint/SetDisableCollision/ (Accessed: April 28, 2022).

Swapping widgets in game (2016) *Unreal Engine Forums*. Available at: https://forums.unrealengine.com/t/swapping-widgets-in-game/345182/2 (Accessed: April 28, 2022).

Virtus (2016) *Introduction & setting up hunger system -#1 creating A survival horror (Unreal Engine 4)*. Youtube. Available at: https://www.youtube.com/watch?v=wJJn9igmznc&list=PLL0cLF8gjBpqGJwEe5XL5mSL8UvwwVMKu (Accessed: April 28, 2022).

Virtus (2017a) *Adding the AK-47 weapon - #8 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=YCdAX6zhyBs (Accessed: April 28, 2022).

Virtus (2017b) *Aiming down sights - #14 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=5nNGq6fS6vI (Accessed: April 28, 2022).

Virtus (2017c) *Blood splash effect - #5 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=RL-x6rHGDxs (Accessed: April 28, 2022).

Virtus (2017d) *Conservative ammo reloading - #18 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=5e13-2C_r_c (Accessed: April 28, 2022).

Virtus (2017e) *Crouching with animations - #10 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=U4HnqE_OZ7Y (Accessed: April 28, 2022).

Virtus (2017f) *Finishing the animation blueprint - #7 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=ftLBejDtlqc (Accessed: April 28, 2022).

Virtus (2017g) *Firing our AK-47 weapon - #9 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=-Q2ZEFmnnmw (Accessed: April 28, 2022).

Virtus (2017h) *Fully automatic rifle - #13 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=w2I7rPwGOuU (Accessed: April 28, 2022).

Virtus (2017i) *Player health & armor - #3 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=UneAoM979uc (Accessed: April 28, 2022).

Virtus (2017j) *Regenerating armor & damage function - #4 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=iFu2WXc8lz0 (Accessed: April 28, 2022).

Virtus (2017k) *Setting up character animations - #6 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=RPdPmys9JWY (Accessed: April 28, 2022).

Virtus (2017l) *Sprinting with animations - #11 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=aAIbKT-hQF0 (Accessed: April 28, 2022).

Virtus (2017m) *True first person camera - #2 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=0mgm16ki8zM (Accessed: April 28, 2022).

Virtus (2017n) *Using control rotation - #12 creating A first person shooter (FPS) with unreal engine 4*. Youtube. Available at: https://www.youtube.com/watch?v=AAd7g8tvheI (Accessed: April 28, 2022).