AIV : Fine anno 2024 - 2025

Lezioni Unreal Engine

1h e 53!!!!

7 mar: Unreal

12:40 – Sentinel: da Base Enemy quando ti becca spawnano i nemici

Okkio alla sentinella nel caso richiedi a Andrea!

31:01 – Sentinel: Manda allarmi ma gli allarmi stanno più avanti

[Subsystems](https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Subsystems)   
32:48 – in Unity usavi un GO ma qui usi i Subsystems: vengono spawnati all'avvio in automatico quando viene spawnata la loro “classe genitore”

Se scegli il WorldSusbsystem -> EnemyManager, NPC manager, etc. tutti questi ereditano dal World e possiamo metterci tutte le reference che ce pare quindi a tutti gli effetti   
Sono come i Manager statici che hai sempre usato (57mins anche)

Immagine che contiene testo, software, Software multimediale, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Che cambia fra i vari Subsystems? La durata:

Engine subs. sta in vita per tutto tempo che funzia l'engine

Editor subs funziona solo se usi l’editor : se non se rilasci il gioco

Qui usiamo: WorldSusbsystem (che va rifatto a ogni Uworld [o livello], perché un WorldSusbsystem **resta in vita** fintanto che un Uworld esiste o viene creato) -> dovra prendere una refer. dei nemici e il delegate per prendersi gli eventi e far fare cose ai NEMICI [Il WorldSusbsystem resta in vita per ogni UWorld (livello) che viene creato quindi ogni volta viene deinizializzato il vecchio e inizial. il new]

Quindi [Subsystems](https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Subsystems) = managers -> come parte il gioco si istanziano e ce metti i nemici, i delegate e altre cagate

Delegati

1h :16 – Dichiarare i delegati:  
Il sistema di delegati è fatto da delle macro di UE DECLARE\_DELEGATE, DECLARE\_MULTICAST\_DELEGATE etc. i delegati C++ sono uguali ai delegati C# per il funzionamento.

1h :38 – Passare valori ai delegati:  
Vedi classe: UEnemiesManagerSubsystem.cpp

1h :42 – Accedere ai delegati dai blueprint:  
(ma fallo a codice che è più no sbattone a ricordasse tutti i spaghetti che a bestemmia su c++):

Immagine che contiene testo, schermata, software, Software multimediale

Il contenuto generato dall'IA potrebbe non essere corretto.

20:06 - SUBSYSTEMS

Immagine che contiene testo, schermata, software, Software multimediale

Il contenuto generato dall'IA potrebbe non essere corretto.

Consentono di implementare logiche centralizzate: tipo Networking o sistema di salvataggio. No classi specifiche ma vengono creati, inizializzati e poi distrutti dall’engine.

22:50 - Insomma li usi dai blueprint e puoi chiamare una funzione specifica senza scriverti una classe apposita

14 mar: Unreal

1h e 55!!!!

[Approfondimento sui Subsystems](https://dev.epicgames.com/documentation/en-us/unreal-engine/programming-subsystems-in-unreal-engine)

Vedi foto sopra: ne esistono di vario tipo “padre” e “figli”. Tipo il wolrdsubsystem è “fratello” (occhio quelli nello specchio so tutti fratelli) dell’Engine subs, perché sono tutti figli del **USubsystem** (Il padre dei subsystem) e nipoti di **UObject** (Il capostipite di TUTTO) ma comunuqe non te sbatte troppo è solo pe capi a livello logico ma tanto poi fanno talmente mille cazzi tutti che ste differenze so relative vai veloce!

**Dove vivono i susbsystem (“a che servono” a grandi linee) . Questo invece è fondamentale!**

* **USubsystem:** Padre di tutti, definisce le regole.
* **UEngineSubsystem:** Vive nel **Runtime** (motore di gioco, anche fuori dall'editor).
* **UEditorSubsystem:** Vive **solo nell'Editor** (non esiste nel gioco finale). Dal 3° anno quindi ciao.
* [UGameInstanceSubsystem](https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Subsystems/UGameInstanceSubsystem)**:** Vive nel **GameInstance** (persiste tra livelli, tutta la sessione di gioco). -> per i salvataggi ad esempio
* [**UWorldSubsystem**](https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Subsystems/UWorldSubsystem)**:** Vive in un **World** (solo per la durata di un livello specifico)- >> nelle lez. Noi abbiamo usato questo perché ci serviva che la roba si distruggesse al passaggio da un livello all’altro.
* **ULocalPlayerSubsystem:** Vive in un **LocalPlayer** (per giocatore, persiste tra livelli ma è specifico per ogni giocatore).

46 :00 – Per vedere le funzioni che sono già presenti clicca ad esempio su [**UWorldSubsystem**](https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Subsystems/UWorldSubsystem)

58:51 - Spawnare un actor (enemy, player…) in scena

1h:54 – Come aggiungere e bindare un nuovo evento nei subsystems: okkio nel video non è spiegato un cazzo volutamente da lui. Sei tu che dovrai copiare la logica di SpawnEnemiesOnAlarm e poi rifarlo in pratica ctrl c e ctrl v creativo al solito.

2h:09 - **Come fare il destroy di un enemy** [ virtual void OnActorDelete(AActor\* DestroyedActor); ] **:** vedrai che è con i subs ma secondo me va bene pe tutti sta logica vabbeh

2h:12 – questa è solo una variante dedicata allo sphere trace ma per gli enemyes: è qui che devi usare l'array di Enemyies TArray<ABaseEnemy\*> del subsystem per non fare i calcoli della mira assistita su tutti gli oggetti in scena ma solo sui nemici.

2h:27 – aggiunge una skeletal mesh alle blueprint (o 30 36 mins vabbhe tanto è scialla)

NOTE PER ANDREA:

2h:27 – Qui andrea dice che non sa usare il behavior tree ma Isonni proprio al minuto 2h:27 – spiega che in realtà lo abbiamo fatto con la state machineil behav. Tree (e forse con l’AI)

21 mar: Unreal

2h:57!!!!

Chiamare allerta su nemici spawnati; salvataggi files; cambio materiali con funzioni: spari e se distrugge robba

Okkio 8:48 -> qui la funzione FsoundOnAlarm lui l’ha chiamata OnPerceived e non te lo dice …perché è un cojone !

11:30 – Quando scrive una classe in c++ devi. Progettarla in modo tale che. I valori siano. Settabili poi anche da BluePrint e prevedendo modifiche dinamiche, vedi un esempio nella classe. H. Nella funzione on High alert.

16:22 – **Reset di una funzione: HighAlert reset + TIMER (set timer by event)** : si fa con un timer rendendole è un timer e un timer delegate . Servono anche degli include: #include "Engine/TimerHandle.h"

16:22 – Aggiungere uno stato alla FSM integrandolo coi subsystems e utilizzare una funzione per far fare cose all’enemy (o all’actor di turno) stoppardi…

57:00 – Come aggiungere comportamenti negli Stati della FSM (Esempio, se sto in High alert vado ad indagare.):   
Qui fa un esempio di cosa puoi far fare a un pawn OA un actor. Quando entri nell’ enter state, o quando vuoi far fare un determinato comportamento a un pawn a un actor? In questo caso? Lui. Fa muovere l'ennemy verso Il punto da cui è scaturito l'allarme

1h:00 – Fa una disquisizione filosfoca per dire cosa fare se hai nemici diversi e fargli fare cose diverse a seconda del component che gli attacchi oppure gestire tutto con variabili

1h:09 – Il nemico colpisce il player ( Ripasso.) si riduce la vita : Life -=

1h:27 – **Salvataggi** (meno imp per esame)

Intro: ereditare da save game + data Manager (slava/carica). Al 7 mar il gioco è single player, ma se lo vuoi fare “local coop” [che sarebbe multiplayer ma in locale (assasins creed)] ti serve un local player subssystem, però siccome è complesso lui dice accanna e usamo solo il salvataggio classico.

Tutte le caratteristiche vengono salvate in una classe e ogni caratteristica è una variabile. Esempio con pseudo code

**1h50 – salvataggio con struct** (nei video i salvataggi con le S. sovrascrivono i salvat normali!)

2h09 – Ok, adesso la logica è Spawnare. solo i nemici. Che non sono stati uccisi prima del salvataggio. Dopo il caricamento.

**2h40 –** il Save game dipende da come volete gestire il salvataggio del vostro gioco. Può esserci salv automatico e il salvataggio amanuale, quindi potete anche averli entrambi. Ovviamente il salvataggio automatico avrà uno slot name e un index che gli passiamo che va da 0 A 2 (volte) E poi si sovrascrive alla terza sovrascrittura (per non sovraccaricare la memoria)

2h42 – Aggiungere salvataggi particolari con trigger : potrebbe servi per esame (ma indirettamente perché ti fa vedere le cose che devi fare normalmente per mette in scena un trigger coi subsystem ma nelle bluprint ad esempio!) ->> come mettere un autosave in scena ad esempio: perché col trigger a un certo checkpoint in una zona parte l’autosave. A 2h e 50/55 dice pure che poi puoi gesti pure tutto il checkpoint index a codice, quindi vedi se te serve -> comunque non l’hai fatto e non c’è nei BP se vuoi vederlo devi farlo rivedendo il video

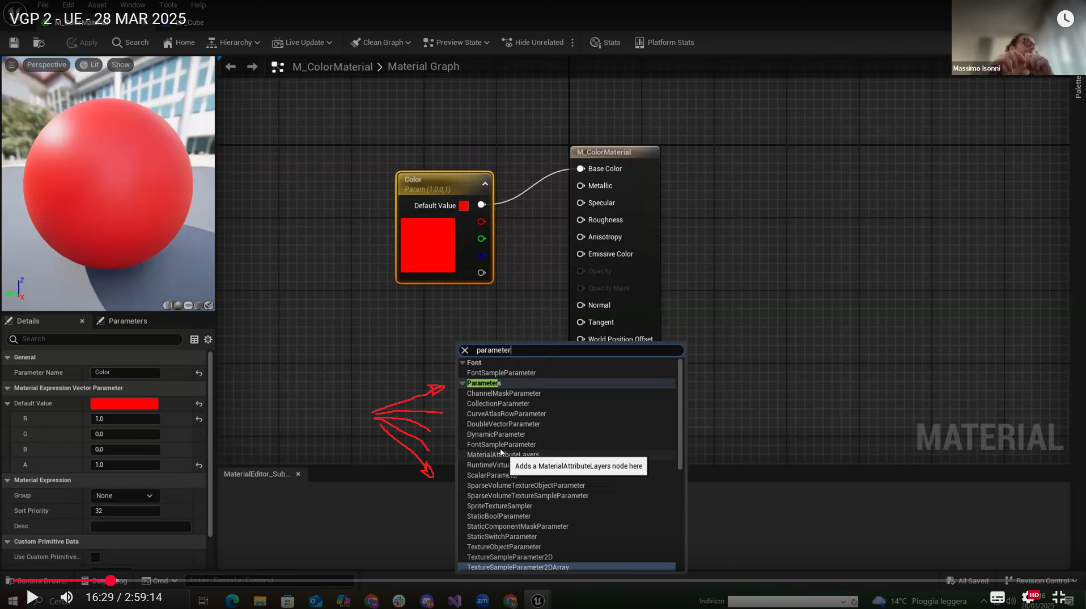
3h02 – Considerazioni sul Load: mentre per il Save è a nostra discrezione dove farlo in base a come vogliamo fare il gioco, il load ovviamente dobbiamo stare attenti alle varie cose a cui accediamo e che andiamo a settare che quando facciamo il load quelle risorse esistano e devi utilizzare una [UGameInstanceSubsystem](https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Subsystems/UGameInstanceSubsystem)

3h05 – Fare i salvataggi come un’interazione del player ( quando premi S/L, salvi o carichi il gioco) -> sta proprio nella classe player!

28 mar: Unreal

**11:20 – Materiali (fai un mat nel modo classico, vedi)**

**14:40 -** Material vs Dynamic material: il material fai la mesh ma a non puoi più cambiarlo, il dynamic cambia dinamicamente coi **parameters (vedi immagine). Il concetto è che puoi cambiare materiale a runtime.**  Questo è solo per avere un’idea in blueprint ma tu lo fai in C++ vedi punto dopo!



**22:00 -** Material vs Dynamic material: leggi prima sopra qui es. base ->in pratica pui cambià texture /colore a runtime e quindi se spari a un vetro da vetro sano je passi la text col vetro rotto a ragnatela etc.

**25:00 -** Material vs Dynamic material in C++: 2 modi tutto: in C++, oppure Blueprint e C++.

1. BP e C++: SetDynamicMaterialOnStaticMesh -> dentro a UAIV\_BP\_FunLibrary.cpp che in pratica: okkio il cambio colore a 1h e 05 non funziona riparti da quando genera il materiale al **14:40**
2. Aggiunge un materiale dynamico alla mesh e controlla se lo può fare (SetDynamicMaterialOnStaticMesh)
3. Cambia materiale co un parametro se trova il materiale bono (

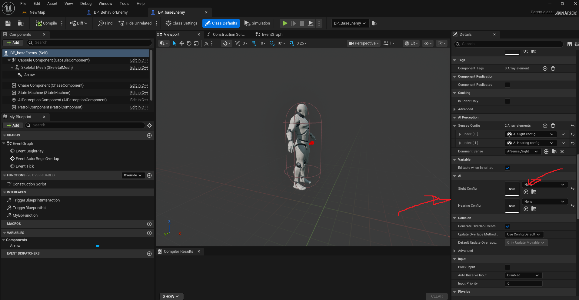
**46:32 – GUARDIE DEL PREPROCESSORE** (approfondimento nn centra coi mat). Esempio: #if WITH\_EDITOR…… ….#endif. Controlli solo nell’editor a fini debug per non appesantire il gioco finale rilasciato

**1h:05 -** okkio il cambio colore a 1h e 05 non funziona, perché non hai creato il material base quindi riparti da quando genera il materiale **14:40**

2h:10 -debug AI (con antonio) e le shrtcut pe il debug -> apostrofo e poi 5 tastierino numerico

**NON FUNZIONA IL TRACCIAMENTO DEL NEMICO. NON TE SEGUE IL LINE TRACE del BaseEnemy, lo DISEGNA, MA NON GENERA UNA “SFERA DI RILEVAMENTO!”**

**Forse dipende da sta roba sull’ai dellenemy. O da qualche impostazione sull’AI, rivedi lezione AI:**

****

4 apr: Unreal

**AI approfondimento**

**22:00 – AI approfondimento: gruppi di controllo affiliationFiilter + “TeamsID” come funziona di default (Fuoco amico, fuoco nemico.)**

**22:00 – AI approfondimento: dare un team al baseenemy**

**Behavior tree eeeeeeeeeeeee (da ora in poi BT)**

1h:22 - **FSM vs BT**: F è più efficiente ma è più difficile da gestire con + stati: + aumenti gli stati più è difficile programmare le transazioni fra stati -> se pattuglio ma se vado solo in una certa zona cambio stato etc.   
**I BT INVECE SONO:** Azioni – Condizioni (se fare o no un’azione) - Decoratori (“*se fai così allora fai questa azione in questo modo*”) -> ma tikka (fa il tick) in continuo su tutto il tree la FSM invece solo sul sing. Stato.

**1h:42 – INIZIO BT: Okkio! Il BT usa il ComplexEnemy. La FSM il BaseEnemy**

**1h:50 – AFPSComplexAIController che ha un BT component e una blackboard component**

**2h:02 – Qui fai il BT dentro all’Editor perché lo dichiari così e perché è meglio farlo logicamente in UE e non in c++ visto che viene fori no schema e poi i dettagli che definiscono tutto li passi in c++**

* **Behavior Tree → È il capo che decide cosa fare**
* **Blackboard → È il tabellone dove appuntano indizi e informazioni**
* **Tasks → Sono i detective che eseguono compiti**

1. **Task "Osserva" → Scrive nella Blackboard: CanSeePlayer = true**
2. **Blackboard-Based Condition → Legge: if (CanSeePlayer == true)**
3. **Se vero → Esegue il task "Attacca"**
4. **Se falso → Esegue il task "Cerca"**

**Selector se un ramo se blocca esegue gli altri Sequence se un nodo figlio si blocca si interrompe tutto il Behav Tree**

**Immagine che contiene schermata, testo, Software multimediale, software

Il contenuto generato dall'IA potrebbe non essere corretto. Immagine che contiene testo, schermata, diagramma, linea

Il contenuto generato dall'IA potrebbe non essere corretto.**

**2h:28 - Spiega come fare un tree su C++ …. Dovrai poi settare e prenderti le chiavi dalla blueprint**

**Note:**

Lo sparo è aggiustato lo trovi dal primo minuto al ventiduesimo di questo video. Adesso però devi ricordarti di riattribuire Di togliere. La mira? Assistita. All'interno del. Player. Per vedere anche le collisioni del line trace single by Channel. Del player. Con l'ennemi. Questo sempre se le vuoi vedere e ancora non funziona. Il tracciamento della vista. Ai Sensing

***Quando fai il Boss digli che avresti dovuto fare un nuovo nemico col BT ma che non hai avuto tempo e che quindi adesso eredita da BaseEnemy***

Libri per AI complesse:

Ai for games (di **Ian Millington)**: <https://www.amazon.it/AI-Games-Third-Ian-Millington/dp/0367670569/ref=asc_df_0367670569?mcid=5f1152d55d17350bb12d8ab2910950d6&tag=googshopit-21&linkCode=df0&hvadid=700862289033&hvpos=&hvnetw=g&hvrand=2968215689465455234&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9050591&hvtargid=pla-946782971463&psc=1&hvocijid=2968215689465455234-0367670569-&hvexpln=0>

11 apr: Unreal

**Okkio ricontrolla ma In tutti i casi tu stai settando Dei valori che sono stati PRE-impostati nel behavr Tree, dentro Unreal Engine e non il contrario! In pratica li prendi dalle dal beaver Tree “E li assegni dentro C++”.**

05:00 - **Recap Bono del Behavior treeeeeee! (con Decorator. Task. Punti. Nodi. Eccetera eccetera.)**

11:40 - BT : il nome che dai al task deve inizà sempre con UBTTask

18:15 – REFACTORING DELL’ENEMY: okkio perché lui non fa col secondo modo come hai scritto nei commenti qua sotto ma col primo -> fa una classe padre e poi base enemy la fa ereditare dalla classe padre

//Per fare una cosa pulita dovresti fare un BaseEnemy e poi derivarne due: uno con la FSM e uno con il BT //In questa implementazione non serve la FSM perché l'unica cosa che fa è tikkare la FSM //quindi la disabilitiamo: la "by-passi" in pratica.

34:42 - REFACTORING DELL’ENEMY: I manager (subs)

45:00 – Cosa fare in caso di ERROR log dei file della cartella “Intermediate” e “linker” . Cancella e rigenera soluzione ma vedi video!

48:00 – *Questo punto saltalo se stai vedendo il video fa un casino con Abstract class in c++ puro vs Abs clas in Unreal e non si capisce un cazzo e vai subito a 1h17 c’è pausa in mezzo*

1h:17 - UBTTask\_ChaseActor il chase dopo il patrol (patrol a lezione precedente)

2h:31 - UBTTask\_AttackActor

2h:35 – PER ESAME Qui dice come fare gli attacchi ma non dice come implementarli

2:38 – PER ESAME: tornare al chase **tornare a** UBTTask\_ChaseActor

2h:50 - Non hai messo gli ultimi due metodi che comunque so finti perché non fanno un cazzo dice lui a che cazzo servono sia qui: 2h:50 che qui: 2h:55 -> quando la vita ti secnde scappi

NOTE IMPORTANTI:

Come funziona il Behavior tree:

1. \*\*Behavior Tree (BT)\*\*: Decide il comportamento dell'AI in base alla logica definita (es. inseguire, attaccare, pattugliare).

2. \*\*Blackboard (BB)\*\*: Memorizza i dati necessari per il funzionamento del BT, come obiettivi o stati dell'AI. Il BT utilizza questi dati per prendere decisioni dinamiche.

3. \*\*AIController\*\*: Esegue i comandi decisi dal BT, come muovere il Pawn o interagire con l'ambiente, utilizzando i dati presenti nella BB.

4. \*\*Pawn\*\*: È l'entità fisica controllata dall'AIController, che si muove e agisce nel mondo di gioco.

In sintesi: \*\*BT → BB → AIController → Pawn\*\*.

La \*\*Blackboard\*\* funge da intermediario tra il BT e l'AIController, fornendo i dati necessari per eseguire le azioni.

\_\_\_\_\_\_\_\_\_Da qui in poi non sono appunti presi benissimo (solo il punto importante è scritto bene) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

16 mag: Unreal

Boss

DA 38: inizio reale del boss fino alla fine

39: BossPhaseComponent fasi di salute del Boss cre diverse classi c++

49: cosa sono BTservice: e//BTservice sono dei nodi il cui tick viene eseguito in continuazione monitoring della salute e Update della BB i

1h11 - UbossPhaseComponent, apply damage HandleTakeAnyDamage

1h:28 - Shoot con ->> HandleTakeAnyDamage UGameplayStatics::ApplyDamage()

1h32: UbossAttackComponent ->> proiettile ma non spiega come fallo perché già c’è in [UE projectile movement component](https://dev.epicgames.com/documentation/en-us/unreal-engine/movement-components-in-unreal-engine) dice che se fa tutta in BP void   
2h:10: UBossAttackComponent::PerformMelee()

2h18: BossCharacter.generated.h

2h24: BossCharacter.cpp + SpawnReinforcements -> boni pe I spawn random!!!!!!!!!!!!!!!!!!!!!!!!!

2h30: ABossAIController

2h42: fai un nuovo BLUEPRINT del boss in UE espandi e vedi boss nella select. 244 Apply damage

Immagine che contiene testo, schermata, Software multimediale, software

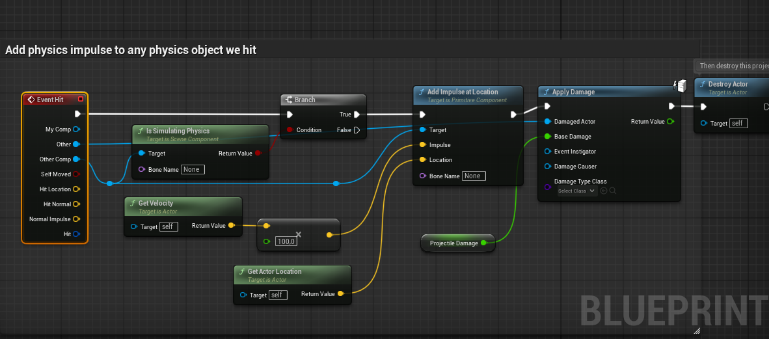
Il contenuto generato dall'IA potrebbe non essere corretto.

2h44: Apply damage che ha effetto se fai Ontakedamage sull’FPSCharacter e lo bindi metti foto. Se fa uguale al BossCharacter l’on take

Immagine che contiene schermata, testo, Software multimediale, software

Il contenuto generato dall'IA potrebbe non essere corretto.

Apri il bluprint di projeectile



E aggiungi varibles

Immagine che contiene testo, schermata, Software multimediale, Software per la grafica

Il contenuto generato dall'IA potrebbe non essere corretto.

2h50: Behvior tree in UE vedi screen finale!!!

perfo

23 mag: Unreal

5mins: Se ti servono basi di terzo anno per farti un corso come te pare su Assembly qui c’è un pizzico su Assembly/puntatori memoria

44min: UBTTask\_BossChargeAttack ->> che cazzo fa il boss in attacco. Idee per farlo non te lo dice, prova a fallo te boh

50: Cooldown e Is At Location: decorators nativi UE

1h:00 : UBTDecorator\_IsDistantFromActor

**Da 1h26 a 1h:57: prosegue col Beahviortree in blueprint: lo scopo qua è inserire task decorator e service in modo simile a quello che hai già fatto col task charge attack. tu devi passarlo in C++ ti da qualche dritta in c++ pe fa il “recharge e check delle ammo” ma devi fallo te! Quindi rivedilo ma tranquillo: in pratica devi copiare creativamente.**

**1h:51: Move to come task  
se aggiungi altri attacchi vai ad aggiungere nell’attackcomponent quello che effettivamente fa l’attacco e poi te crei er task (seeeeeeeeeeeee vabbeh ciaoooooooooooooo, ma che cazzo ho appena scritto!?!?!?!?!?!?!?!?)**

**PUNTO MOLTO MOLTO MOLTO IMPORTANTE. LOGICA E “GIRI” DELLE CLASSI:**

2h17: cosa fa la classe apply damage vs “Danno normale” col sistema delle interfacce (2 modi diversifare ) **e come come Capire Che giri Fa un metodo nelle librerie di UnReal Engine:**

In pratica qui si tratta di capire come fare a utilizzare le classi padre date da dal motore e capire come utilizzare le funzioni presenti, per esempio all'interno di actor. In modo tale da prendere solo parti dei comportamenti prestabiliti dalla classe padre/di base E utilizzarli per i tuoi scopi.

Che significa? Significa, per esempio, saper usare i delegati. E chiudere il cerchio con i delegati. In modo tale da poter ottenere determinati comportamenti specifici**.**

**ESEMPIO IN QUESTO CASO (**2h17 del video): **Usi il delegato**. Per applicare un danno all'altro nemico.

**La cosa fondamentale che devi fare è cliccare sulle classi, Sui metodi delle classi Prestabilite, ( le classi padre/base): tipo l’apply damage della classe actor.**

Un esempio che chiude il cerchio è che fai il danno, il nemico ha uno scudo riflettente e il danno torna indietro (Sembra una cagata, ma non lo è.) Infatti. E proprio la summa. Di come. Capire. Che giri. Fa un metodo nelle librerie di Real Engine (per capire questo esempio o te ricordi e devi vede il video dal minuto 2h:17 sennò nisba).

2h30: come fare una classe damage type : UDamageType::StaticClass(). **Che serve per rendere il danno più complesso!!!**

ogni volta che facciamo l apply dmage gli passiamo una classe. Noi stiamo passando una classe generica **UDamageType::StaticClass(),** questa classe in sé da sola non fa nulla, non viene usata dal sistema di UE per fare cose, è semplicemente ci mette a disposizione una classe da cui possiamo ereditare, quindi possiamo crearci. Poi la nostra per definire tante varie cose che possiamo usare nel momento in cui facciamo apply damage, **cioè per portarci dietro varie informazioni per poi calcolare il danno**.

2h57: okkio qua comincia a svalvolà

NOTE:

in pratica. Puoi copiare. Quello che hai fatto nel boss charge Attack. Anche per il ranged e per il Meli.

I Selector finiscono quando un nodo figlio ha successo, per le sequenze quando fallisce il figlio.

ABaseEnemy\* NewEnemy = GetWorld()->SpawnActor<AGenericEnemy>(ClassToSpawn, InTransform, SpawnParams); //spawna l'attore nel mondo: funzione nativa UE clicca e vedi come funzia

Dentro al subs

**1h:20 - RECAP DELLE PUNTATE PRECEDENTI! (da febbraio al 21 mar circa)**

Okkio quest è solo recap vedi video dura 5 mins. Qui numero lezioni:

1. Classi base + macro + blueprint fun library + actor
2. FPS Chaarcter + mira assist + Input e input binding movement e shoot
3. Interfacce shoot + interact + lay down base enemy
4. AI perception system + FSM
5. AI perception system + Delegate + Subsystem
6. Approfondimento: Spawn Enemies + Subsystem + Delegate
7. FSM aggiungere uno stato + Salvataggio con subs

Manca un pezzo porcodio manca dalla lezione del 28 feb riaggiungilo al project!

[2h:51 LaunchPebble -> sta nella lezione preced 28 feb se lo incontri mezzo finito copialo e via (e cancella sta riga!)]

14 marzo

Da23:13 mins

PROGETTO

Quando smadonni sul progetto ricordati che tu il Player l’hai chiamato Player e Lui PlayerCharacter…