Fakulta elektrotechniky a informatiky STU BA

Niagara VFX pre Unreal Engine 4

KEGA 030STU-4/2017

Autor: Peter Melek, Editor: Erik Kučera

OBSAH

ÚVOD

Unreal Engine má dva hlavné VFX systémy, pôvodný, zvaný Cascade a nový (2018), zvaný Niagara. Tento dokument slúži ako úvod do základov Niagara VFX s demonštratívnym príkladom. Na úvod je ale vhodné vysvetliť, prečo existujú dva VFX systémy.

Cascade je pôvodný VFX systém pre Unreal Engine. Jeho výhodou je, že obsahuje mnoho preddefinovaných správaní, ktoré medzi sebou dokážu jednoducho a efektívne komunikovať. Je vhodný najmä pre menej technicky zdatných dizajnérov, ktorí s ním môžu pracovať aj bez znalosti programovania. Problém pri Cascade ale nastáva vtedy, keď používateľ potrebuje upraviť existujúce funkcionality. Hlbšia modifikácia funkcií alebo vytvorenie nových funkcii vyžaduje nový kód a bez hlbšej optimalizácie môže byť „bottleneck“ pre pôvodný tok komunikácie modulov. Ďalšou z nevýhod Cascade je fakt, že je takmer nemožné zdieľať dáta s inými časťami Unreal Enginu a v neposlednom rade je tu problém so značnými rozdielmi simulácii na GPU a CPU.

Na základe nedostatkov Cascade vznikla filozofia pre Niagara. Hlavný rozdiel medzi systémami spočíva v tom, že Niagara umožní prístup k takmer všetkým parametrom simulácie pre ostatné časti Unreal Enginu. Premenné ako delta čas, poloha v priestore, rýchlosť pohybu, vzdialenosť od kamery a desiatky ďalších parametrov sú dostupné pre iné komponenty na čítanie a prepisovanie. Navyše je väčšina týchto parametrov voliteľná podľa potreby používateľa, čím sa zároveň odľahčuje dátový tok. Niagara sa týmto stáva veľmi flexibilným nástrojom pre efekty rôznych využití.

ZÁKLADNÁ ŠTRUKTÚRA NIAGARA VFX

Niagara pozostáva z troch základných komponentov:

- Moduly
    - Využívajú bežné dáta, popisujú správanie a je možné ich navzájom vrstviť
- Emittery
    - Sú kontajnery pre moduly
    - Sú jednoúčelové (jedna funkcionalita) ale opakovateľne použiteľné
    - Umožňujú modifikáciu parametrov z modulov
- Systémy
    - Kombinujú emittery do jedného „efektu“
    - Prepisovateľné

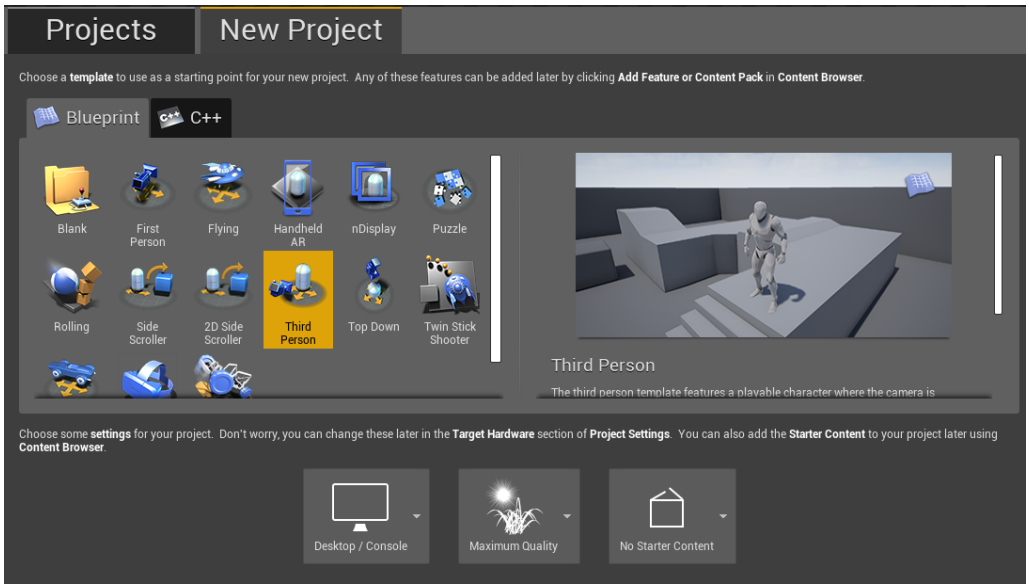
Na prvý pohľad nemusí byť evidentný rozdiel medzi systémami a emittermi. Ako bolo uvedené vyššie, systémy pozostávajú z emitterov. Systémy zároveň môžu obsahovať vlastné globálne premenné, ktoré sú dostupné všetkým emitterom v jednom systéme. Systémy zároveň definujú kadenciu efektu, teda funkcionalitu ako cyklickosť efektu alebo spawning (vznikanie efektu).

PRAKTICKÁ UKÁŽKA

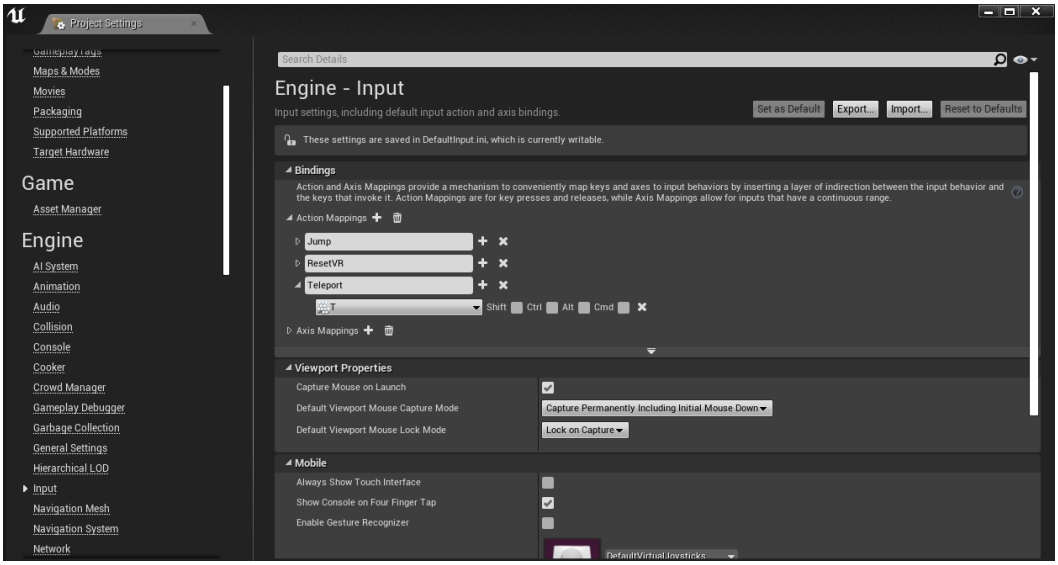
V tejto ukážke vychádzajúcej z Third Person predlohy umožníme hráčovi teleportovať sa v smere jeho pohybu a zanechať za sebou časticový efekt, ktorý vytvoríme pomocou Niagara VFX.

PRVÁ ČASŤ – TELEPORT

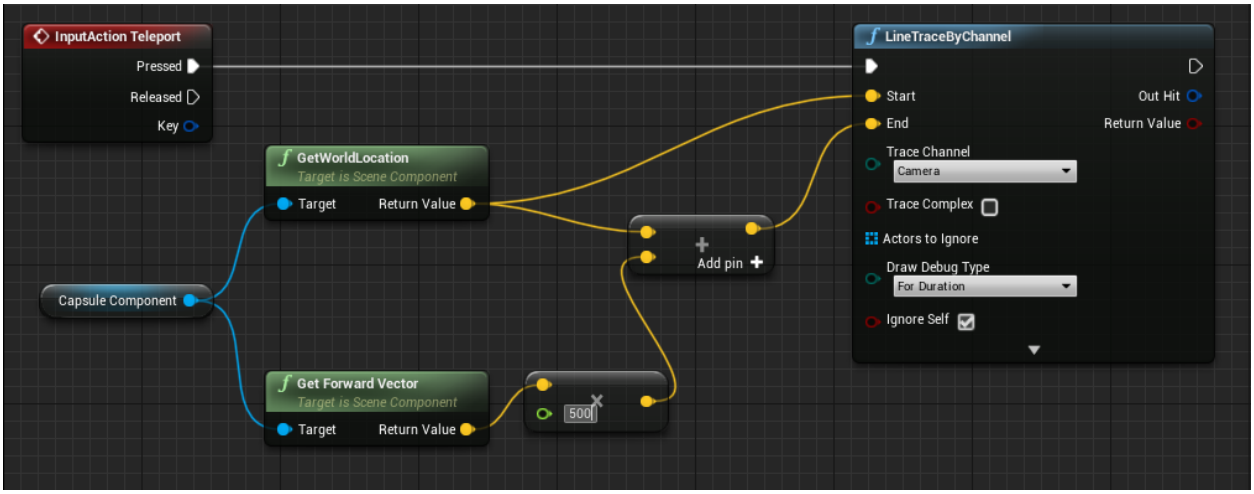
1. Otvoríme Unreal Engine, vytvoríme nový **Blueprint projekt** vychádzajúci z predlohy **Third Person.** Zvolíme vývoj pre **Desktop/Console**, **Maximum Quality** a **No Starter Content**.



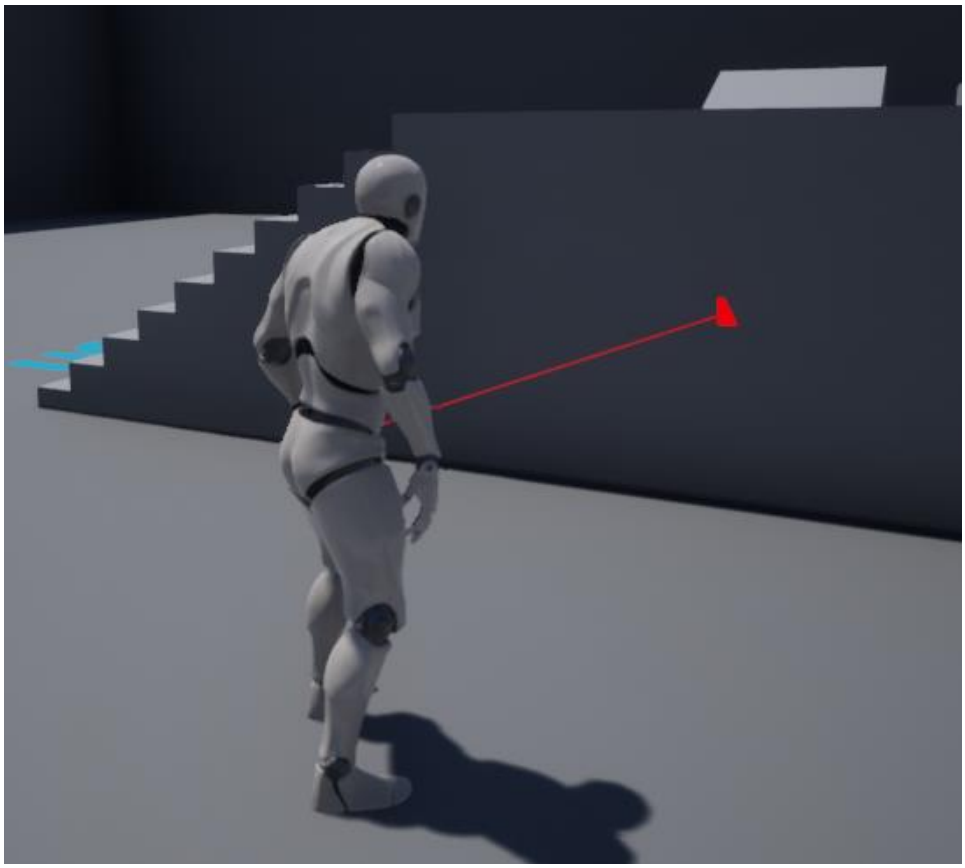
2. V prvom rade vytvoríme veľmi jednoduchú funkcionalitu pre teleport. Prejdeme do **Edit – Project Settings – Engine – Input – Bindings – Action mappings**, klikneme na plus, nazveme akciu Teleport a priradíme jej klávesu T.



3. Zavrieme okno Settings a v časti **World Outliner** klikneme na **Edit ThirdPersonCharacter**.
4. V Blueprinte sa presunieme držaním pravého tlačidla myši na voľný priestor. Opäť stlačíme pravé tlačidlo, zobrazí sa menu na pridanie akcie. Začneme písať „teleport“, alebo náš vlastný názov akcie z kroku 2 a zvolíme **Input – Action Events – Teleport**.
5. Začneme ťahať líniu **z výstupu Pressed** našej InputAction, keď pustíme ľavé tlačidlo, ukáže sa podobné menu ako v predchádzajúcom kroku. Nájdeme modul „**LineTraceByChannel**“ a pridáme ho do blueprintu. Jedná sa o základný prvok pre prácu s vektormi, ktoré budeme využívať na teleportáciu hráča. V prvku samotnom nastavíme **Trace Channel** na **Camera**. Tým sme vyriešili kolíziu s objektami (aby sa hráč neteleportoval dovnútra solidu), nakoľko kamera má preddefinované kolízie.
6. Do blueprintu z menu Components (v ľavej časti UI pre UE4) potiahneme **CapsuleComponent**, ktorý je súčasť ThirdPersonCharacter. Tento prvok budeme premiestňovať v priestore.
7. Z **CapsuleComponent** potiahneme jednu líniu a podobne ako v kroku 5 pridáme **GetWorldLocation** modul. Jeho **výstup pripojíme na Start vstup LineTraceByChannel**.
8. V tomto kroku implementujeme smer teleportácie. Máme dve základné možnosti, môže to byť v smere vektoru hráča alebo v smere vektoru kamery. Zvolíme vektor hráča, čím umožníme hráčovi teleportovať sa v jednom smere a zároveň mať voľnosť kamerou sledovať napr. čo sa deje za ním. Z existujúceho **CapsuleComponent** potiahneme ďalšiu líniu, tentokrát pridáme **Get Forward Vector**. Keby chceme pracovať s vektorom kamery, jednoducho do blueprintu pritiahneme komponent FollowCamera a využijeme jeho Forward Vector.
9. Keďže výstupom GetForwardVector je vždy jednotkový vektor, je potrebné ho prenásobiť žiadanou vzdialenosťou. Vytiahneme teda jeho výstup a vyhľadáme násobenie „*“, pridáme “**vector * float“.** Ako Float zadáme hodnotu napr. 500.
10. Výstup z násobenia podobne ako v predchádzajúcom kroku ešte musíme sčítať s počiatočnou polohou, teda pridáme „**vector + vector**“ a potiahneme to na **End vstup LineTraceByChannel**.
11. Náš blueprint by mal vyzerať ako na obrázku nižšie.

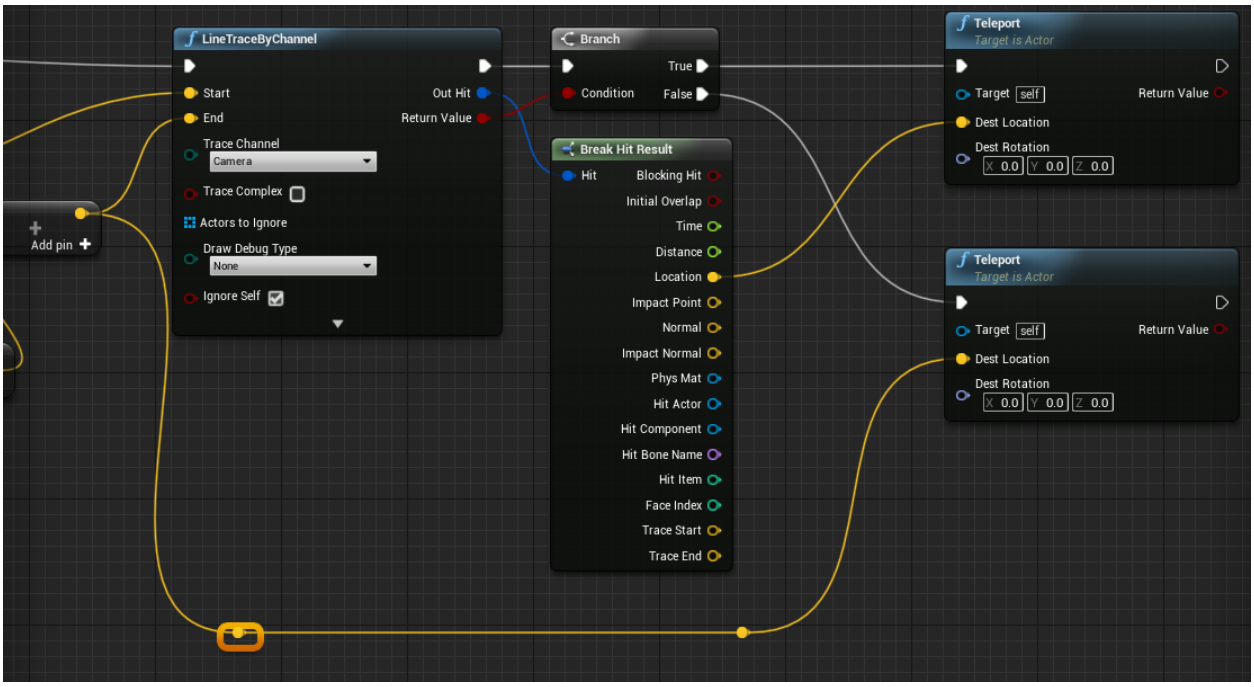


12. V **LineTraceByChannel** môžeme nastaviť Draw Debug Type na For Duration, projekt skompilovať a spustiť. Po stlačení klávesy T by mal zo stredu hráča vychádzať vektor, ktorý bude mať aktívnu kolíziu s objektami.



13. Vrátime sa späť do blueprintu, v LineTraceByChannel vypneme Draw Debug Type (nastavíme None).
14. Z **LineTraceByChannel** potiahneme z bieleho výstupu modul **Branch**, ktorému na **Condition** vložíme **Return Value z LineTraceByChannel**.
15. Z **LineTraceByChannel** výstupu **Out Hit** vytvoríme modul **Break Hit Result**.
16. Z **Branch** výstupu **True** prejdeme na **Teleport** (tentoraz funkcionalita vstavaná v engine, nie naša vstupná akcia!). Krok zopakujeme pre False výstup.
17. Na **Teleport vstup Dest Location** potiahneme **Location z Break Hit Result**.
18. Na druhý **Teleport** potiahneme do **Dest Location výstup zo súčtu vektorov z kroku 10**.

19. Na obrázku nižšie je znázornená druhá polovica Blueprintu, ktorú sme pridali od poslednej kompilácie projektu.

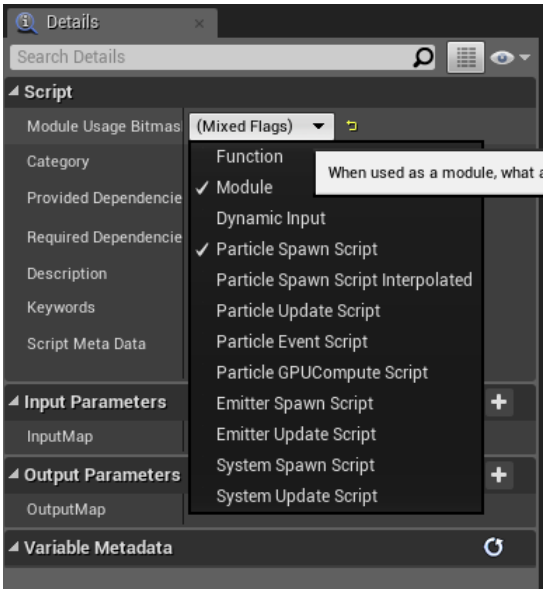


20. Projekt uložíme a skompilujeme. Keď ho spustíme, po stlačení T by sa mala postava hráča teleportovať v smere, na ktorý je natočená.

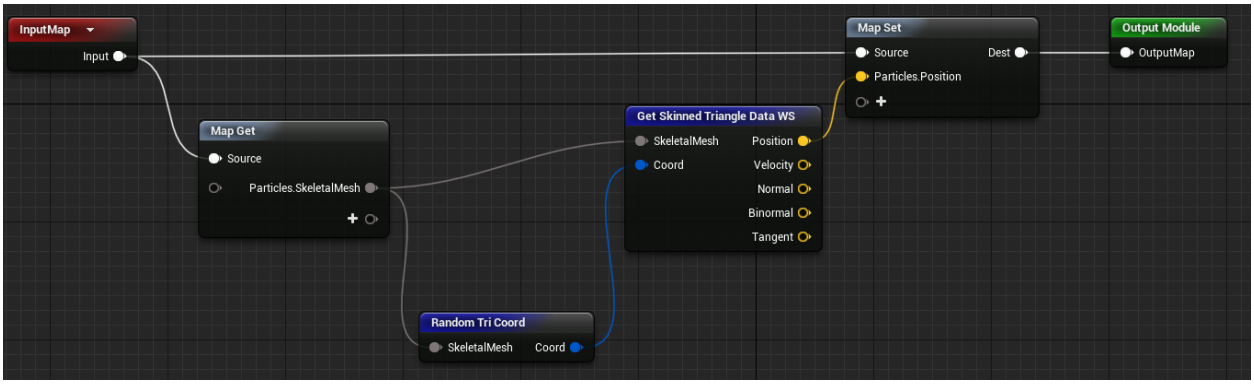
DRUHÁ ČASŤ – NIAGARA VFX

1. V Unreal Engine prejdeme **do Edit – Plugins – FX** a uistíme sa, že **Niagara je Enabled**. Pokiaľ nie je aktivovaný, bude po aktivácii treba reštart Unreal Enginu, ktorého následné spúšťanie chvíľu potrvá. Je to normálny proces, netreba sa báť, že by zamrzol alebo nereagoval.
2. V Content Browser si vytvoríme nový priečinok, nazveme ho napr. TeleportVFX.
3. V priečinku pridáme emitter: **Add New – FX – Niagara Emitter**. Založíme ho na **Fountain template** a nazveme ho *TeleportEmitter*.
4. V priečinku pridáme system: **Add New – FX – Niagara System**. Vyberieme možnosť „**Create a new system from a set of selected emitters**“ a zvolíme náš emitter, v tomto prípade nazvaný TeleportEmitter. Pre konzistenciu ho nazveme *TeleportSystem*.
5. V neposlednom rade pridáme modul: **Add New – FX – Niagara Module Script**, nazveme ho TeleportScript.
6. Dvojkikom otvorime náš TeleportEmitter. Otvori sa nám nové okno. V okne najskôr deaktivujeme pozadie efektu, ktoré môže byť rušivé. V časti **Window – Preview Scene Settings** nájdeme **Show Environment** a uistíme sa, že je možnosť **deaktivovaná**. Zavrieme tento tab.
7. V tabe Selected Emitters vidíme náš emitter a všetky jeho parametre. Využijeme fakt, že väčšina parametrov v Niagara prvkoch je voliteľná, a ikonou smetného koša **odstránime Gravity Force**, **Add Velocity a Spawn Rate**.
8. V časti Emitter Update, z ktorej sme odstraňovali Spawn Rate, klikneme na + a pridáme komponent **Spawn Burst Instantaneous**. V ňom nastavíme **Spawn Count** na **1000**.

9. V tej istej časti otvorime **Emitter Life Cycle** a nastavíme **Max Loop Count** na **1**, teda v efekte ako takom sa častice vytvoria iba raz.
10. V tabuľke **Parameters** klikneme na + pri Particles a pridáme **Skeletal Mesh**. Pomenujeme *Particles.SkeletalMesh*.
11. Novovytvorené particles z predchádzajúceho kroku potiahneme na Set Variables v časti Particle Spawn. V časti Default Mesh nastavíme mesh našej postavy, teda SK_Mannequin.
12. Vrátime sa do hlavnej časti Unreal Engine a otvorime náš *TeleportScript*.
13. V časti Module Usage ponecháme zaznačené len Module a Particle Spawn Script, ako na obrázku nižšie.



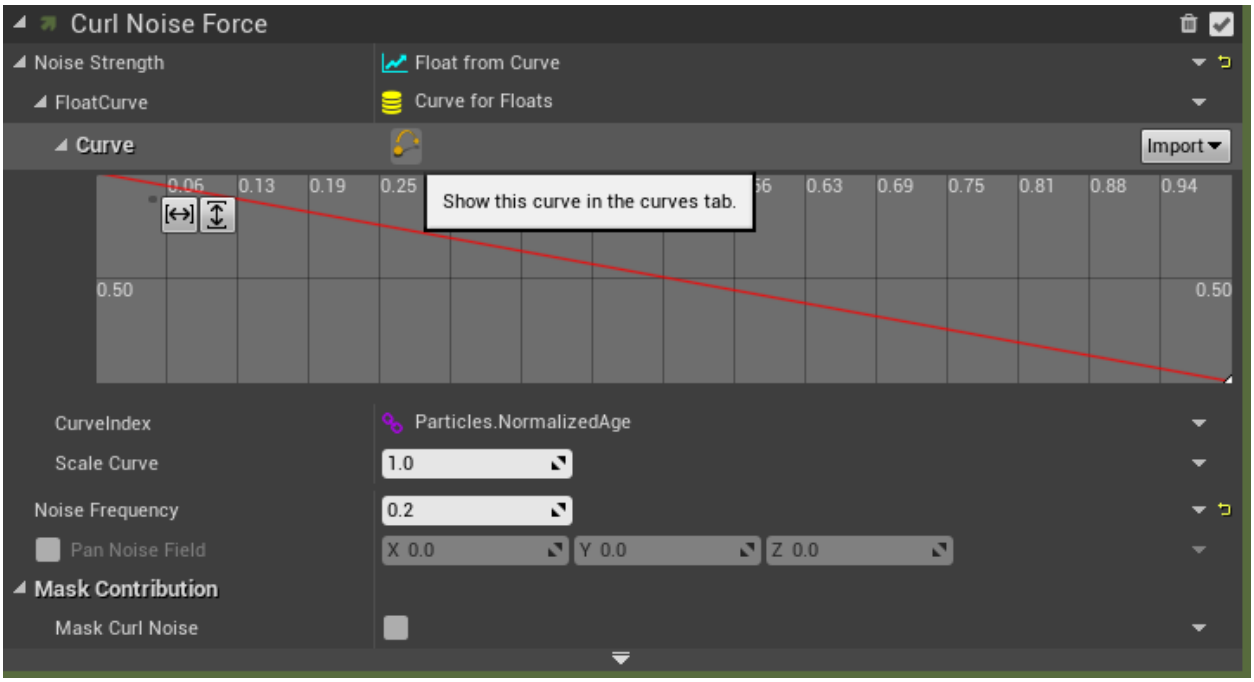
14. Pomenujeme si kategóriu modulu, napr. *PlayerTeleportVFX* (položka priamo pod Module Usage).
15. V blueprinte v module **Map Get** pridáme **výstup Particles.SkeletalMesh**.
16. Z neho vytiahneme modul **Random Tri Coord** a **Get Skinned Triangle Data WS.**
17. Z **Random Tri Coord** potiahneme **Coord** na vstup **Get Skinned Triange Data WS**.
18. V **MapSet** pridáme **vstup Particles.Position** a napojíme naň **Position** z WS.
19. Celkový Blueprint vyzerá nasledovne



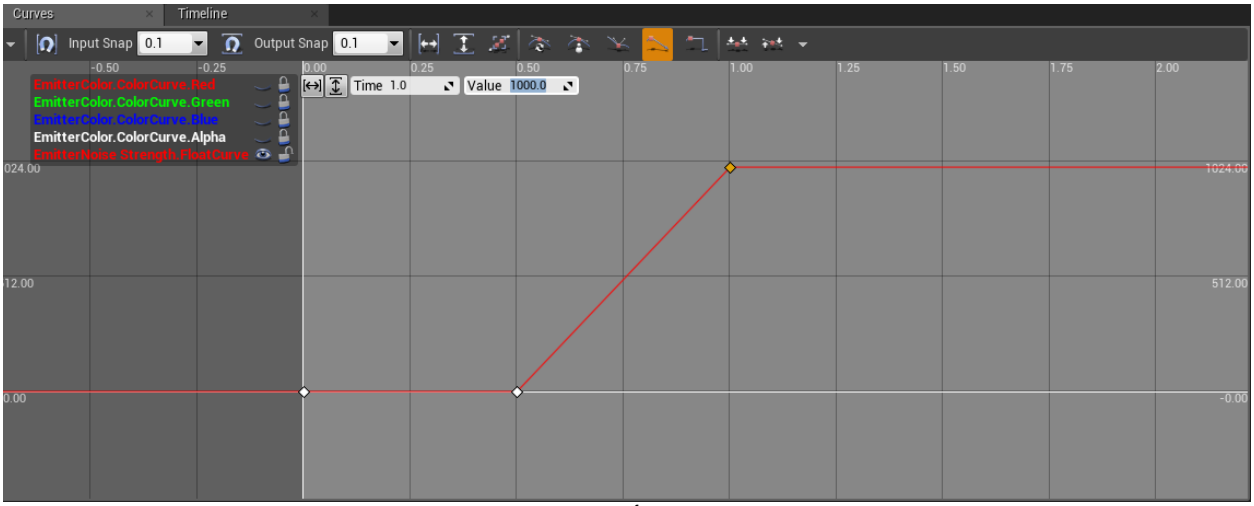
20. Vrátime sa do Emittera a v časti Particle Spawn klikneme + a nájdeme TeleportScript v našej kategórii PlayerTeleportVFX.
21. Ak sme všetko urobili správne, po uložení a kompilovaní by mali častice nadobudnúť obrys humanoida, ako na obrázku nižšie.

22. Zatiaľ náš efekt nie je príliš dynamický, nakoľko sme na začiatku odobrali všetok pohyb. Napravíme to tak, že v časti Particle Update pridáme **Curl Noise Force**. Ak sa objavia problémy s dependencies, stlačíme **Fix issue**. Parameter **Noise Strength** nastavíme na **1000** a **Noise Frequency** na **0.35**.
23. Chceme aby časticová siluéta chvíľu držala pokope a nerozletela sa hneď, preto potrebujeme oneskoriť efekt. Pri Noise Strength klikneme na dropdown šípku a pridáme parameter **Float from Curve**.

24. Zobrazíme krivku kliknutím na tlačidlo ako na obrázku.

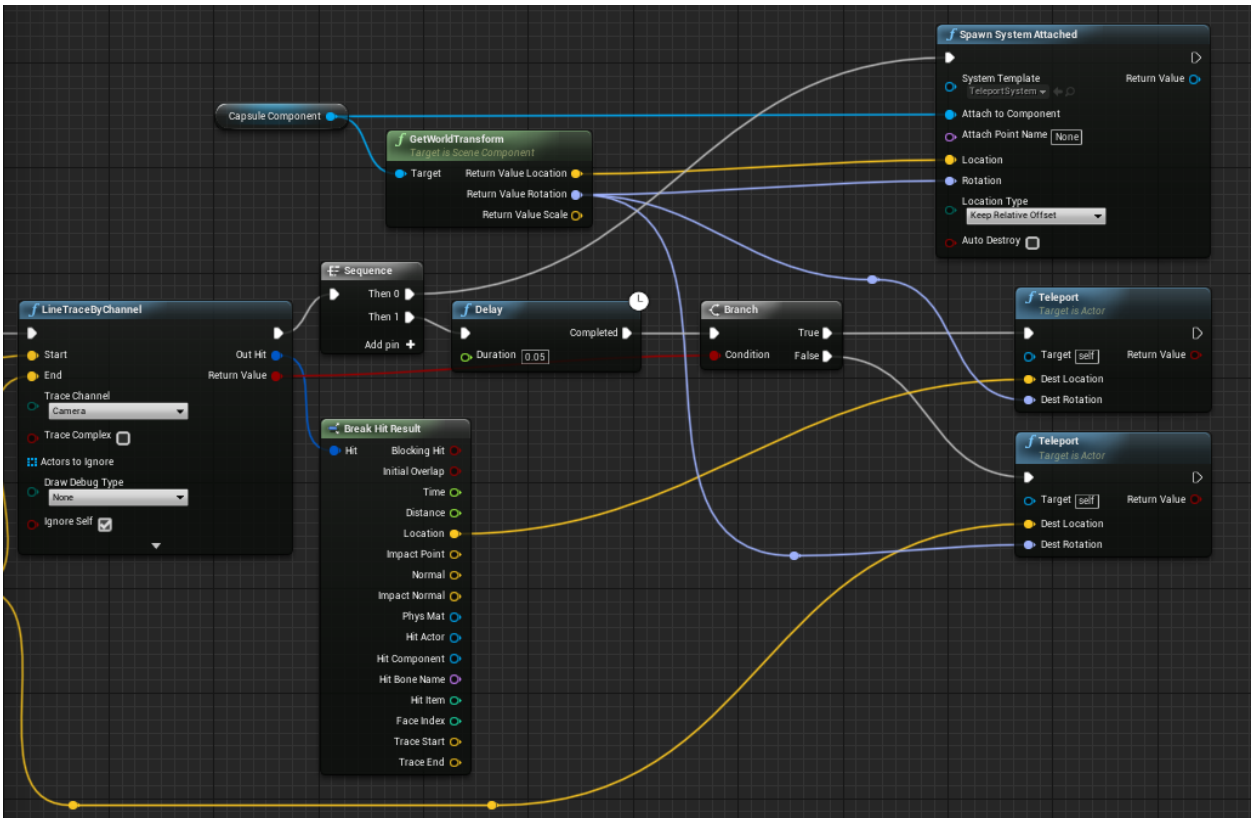


25. Ak sa na grafe nachádzajú aj iné krivky ako krivka sily, skryjeme ich stlačením ikony oka a uzamkneme ikonou zámku.
26. Klikneme do grafu pravým tlačidlom myši a zvolíme **Add Key to Emmit Noise Strength.FloatCurve**. Zobrazia sa položky Time a Value. Týmto spôsobom pridáme dva kľúčové body s hodnotami [Time: 0.5; Value: 0.0] a [Time: 1.0; Value: 1000]. Graf by mal vyzerať ako na obrázku nižšie.



27. V neposlednom rade ešte mierne predĺžime životnosť častíc, v časti Particle Spawn, **Particles.LifeTime** nastavíme **Minimum 3.0** a **Maximum 5.0**.
28. Efekt je hotový, je čas ho pridať do hry. Opäť otvoríme Edit ThirdPersonCharacter ako v prvej časti návodu pri tvorbe teleportu.
29. **Z existujúceho LineTraceByChannel** z prvej časti návodu potiahneme z výstupu modul **Sequence**.

30. **Z výstupu modulu Sequence Then 1** prejdeme na nový modul **Delay**, ktorému dáme **duration 0.05** sekundy. Jeho **výstup** pripojíme na existujúci **Branch** z prvej časti návodu. Týmto sme mierne oneskorili teleport, aby sa stihol vytvoriť efekt na pôvodnej polohe hráča a nie na novej.
31. **Z výstupu modulu Sequence Then 0** vytvoríme modul **Spawn System Attached** z kategórie Niagara. V jeho vstupe System Template zvolíme náš TeleportSystem.
32. Do plochy blueprintu potiahneme **CapsuleComponent** hráčovej postavy. (Môžeme pracovať aj z pôvodným, ktorý využíva teleport funkcia, nový je len pre prehľadnosť.)
33. Z **CapsuleComponent** vytvoríme výstupný modul **GetWorldTransform**. Pravým tlačidlom klikneme na **Return Value** a zvolíme **Split Struct Pin**.
34. **Location** a **Rotation** modulu **GetWorldTransform** napojíme na relevantné **vstupy Spawn System Attached** modulu z kroku 31.
35. **Rotation** zároveň napojíme aj na Dest Rotation teleportu, nakoľko sa v pôvodnej verzií hráč po teleportácii stále otáčal do defaultnej polohy.
36. Pravá polovica hráčovho blueprintu by mala vyzerať nasledovne



37. Všetko uložíme, skompilujeme a demonštračný projekt je pripravený na publikovanie (File – Package Project).

ZÁVER

Ako sme si mohli uvedomiť pri sledovaní praktickej ukážky, Niagara je veľmi flexibilný a prispôsobiteľný nástroj na tvorbu efektov, ktoré si môže používateľ nastaviť presne podľa svojich požiadaviek, či už tvorí efekt pre hru, vizualizáciu alebo akúkoľvek inú aplikáciu. Aj napriek tomu, že Niagara je pomerne nový nástroj, predstavený na GDC 2018 (začiatkom roka), už teraz je evidentná jeho sila a že sa jedná o nástroj, ktorý má veľmi vysoký potenciál.