# 

Táto metodika slúži na ukázanie postupu tvorby VR projektu v softvéri Unreal Engine a porovnanie rôznych spôsobov pohybu vo virtuálnom priestore.

# Ukážka metodiky tvorby VR prostredia

## Implementácia

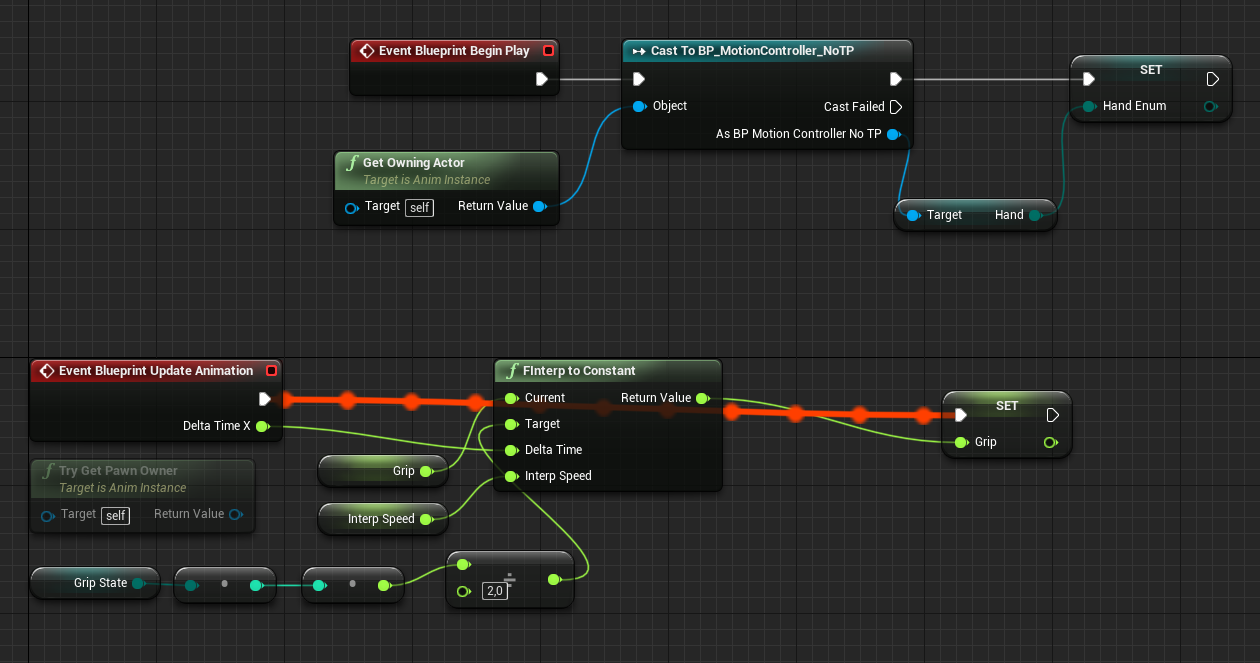
V tejto časti sa pozrieme na tvorbu samotnej aplikácie, pričom sa budeme snažiť dodržať všetky kritériá, ktoré sme si zadefinovali v predošlej časti Špecifikácie. Do implementácie spadá tvorba projektu, *mainmenu*, mapy Volleyball, mapy Lightsaber a mapy Basketball. V časti projekt rozoberieme aj tvorbu mechaník pre virtuálnu realitu, jedného z možných pohybov, uchopenia vecí a podobne.

### Projekt

Tvorba projektu začala stiahnutím a nainštalovaním programu Epic Launcher, ktorý umožňuje prácu s Unreal Enginom vďaka Unreal Editoru. Následne sme stiahli verziu Unreal Engine 4.20. Aby *engine* vedel pracovať s virtuálnou realitou, stiahli sme si program Steam. Ten umožňuje využívanie programu SteamVR, ktorý dokáže pracovať s headsetom virtuálnej reality.

Následne sme vytvorili v Unreal Editore nový projekt, ktorý využíva *blueprinty*. Tu sme si zvolili, že budeme robiť aplikáciu, ktorá bude pre virtuálnu realitu. Ďalšie nastavenia pri vytváraní nového projektu boli platforma, pre ktorú sa hra bude vyvíjať. Vybrali sme teda *Desktop*/*Console* a maximálnu kvalitu. Po týchto nastaveniach sme dali vytvoriť nový projekt.

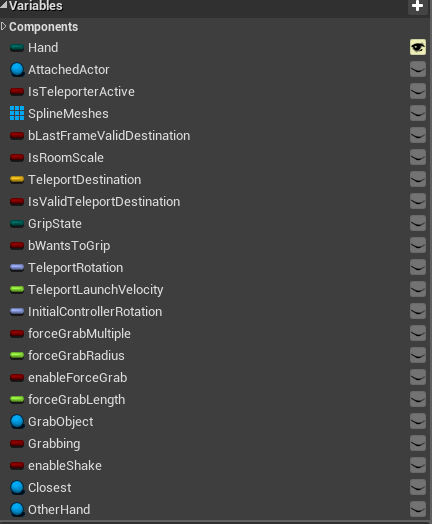
Najskôr sme začali vytvorením hráča, jeho modelu a základných mechaník. Do editoru sme si naimportovali 3D model ruky, ktorému sme predtým v programe Cinema 4D pridali kosti. Následne sme vytvorili nový materiál, ktorému sme dali farbu pleti. Tento materiál sme potom aplikovali na model ruky, aby vyzerala ako ľudská ruka.

Po úspešnom vytvorení ruky sme ju museli rozhýbať. V programe Cinema 4D sme vytvorili 3 rôzne pozície ruky. Prvá pozícia ukazuje roztvorenú ruku, ktorá signalizuje, že hráč môže uchopiť predmet. Druhá pozícia je ruka v pästi, ktorá predmet drží. Posledná pozícia je voľná ruka, ktorá sa zobrazuje za bežných okolností, ak sa nedá nič chytiť alebo hráč nič nedrží. Po úspešnom importe do projektu sme vytvorili Blend Space 1D, ktorý slúži na vytvorenie animácií medzi pozíciami, ktoré sme vytvorili. Pri vytváraní sme si vybrali, že chceme použiť *skeleton*, ktorý je aplikovaný na modely ruky. Vďaka tomu sme mohli do rozsahu jednorozmerného poľa vložiť naše 3 pozície v poradí: bežná pozícia, otvorená ruka, uchopenie predmetu. Horizontálnu os sme pomenovali ako Grip, teda uchopenie. Os sme rozdelili na 3 časti, pričom prvý bod má hodnotu 0. Druhý bod má hodnotu 0,5. Posledný bod má hodnotu 1. Ku každému bodu sme priradili animáciu v poradí, ako bolo písané vyššie. Vo výsledku sme mohli upravovať hodnotu na osi a vidieť ako ruka prirodzene mení svoj pohyb medzi pozíciami. Po tomto kroku sme vytvorili nový *animation blueprint*, znova na základe *skeletonu* ruky. Tu sme v Anim Graphe vytvorili novú premennú typu *float* Grip. Pre Grip sme vytvorili getter, ktorý zoberie momentálnu hodnotu uchopenia. Následne sa táto hodnota prenesie do Blendspaceu a na záver sa animácia zobrazí hráčovi na obrazovke. Ďalej sme si vytvorili v Event Graphe *event*, ktorý sa zapne po začatí. Ten bude prenášať do *blueprintu* MotionController premennú HandEnum, ktorá je typu EController Hand. Posledná vec, ktorá bola potrebná pre sfunkčnenie animácie bolo vytvorenie *eventu* Blueprint Update Animation. V ňom sme nastavili, aby sa po zmene nastavila hodnota premennej Grip. Vytvorili sme novú premennú GripState typu *GripEnum* a nastavili sme hodnotu *Open*. Takisto sme nastavili premennú HandEnum typu *EController Hand* a nastavili jej hodnotu *left*. Poslednú premennú sme nazvali InterpSpeed typu *float* a nastavili jej hodnotu 7. Následne sme využili interpolačnú funkciu FIntrep to Constant, ktorá použije hodnotu GripState prekonvertovanú na premennú typu *integer* vydelenú dvomi ako cieľovú pozíciu. Ako aktuálnu pozíciu využije hodnotu z premennej Grip. Na časový rozdiel sa využije delta X. Rýchlosť je hodnota premennej Interp Speed, ktorú sme nastavili na 7. Výsledok tejto matematickej funkcie nastaví hodnotu premennej Grip cez setter.

Obrázok 16 Tvorba animácie ruky

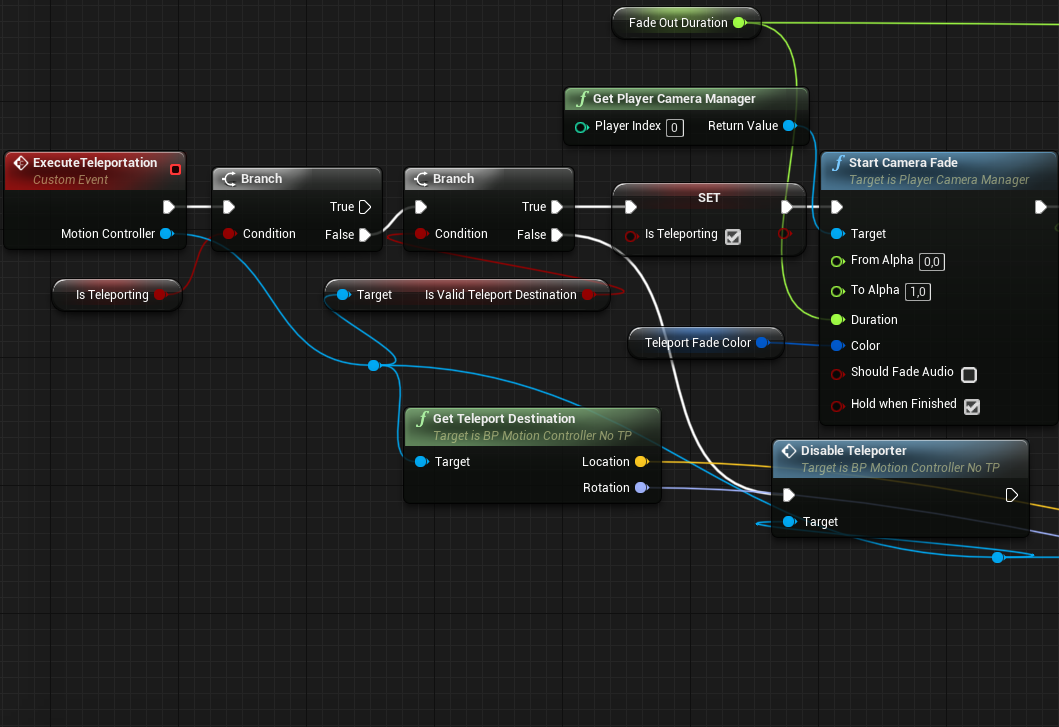
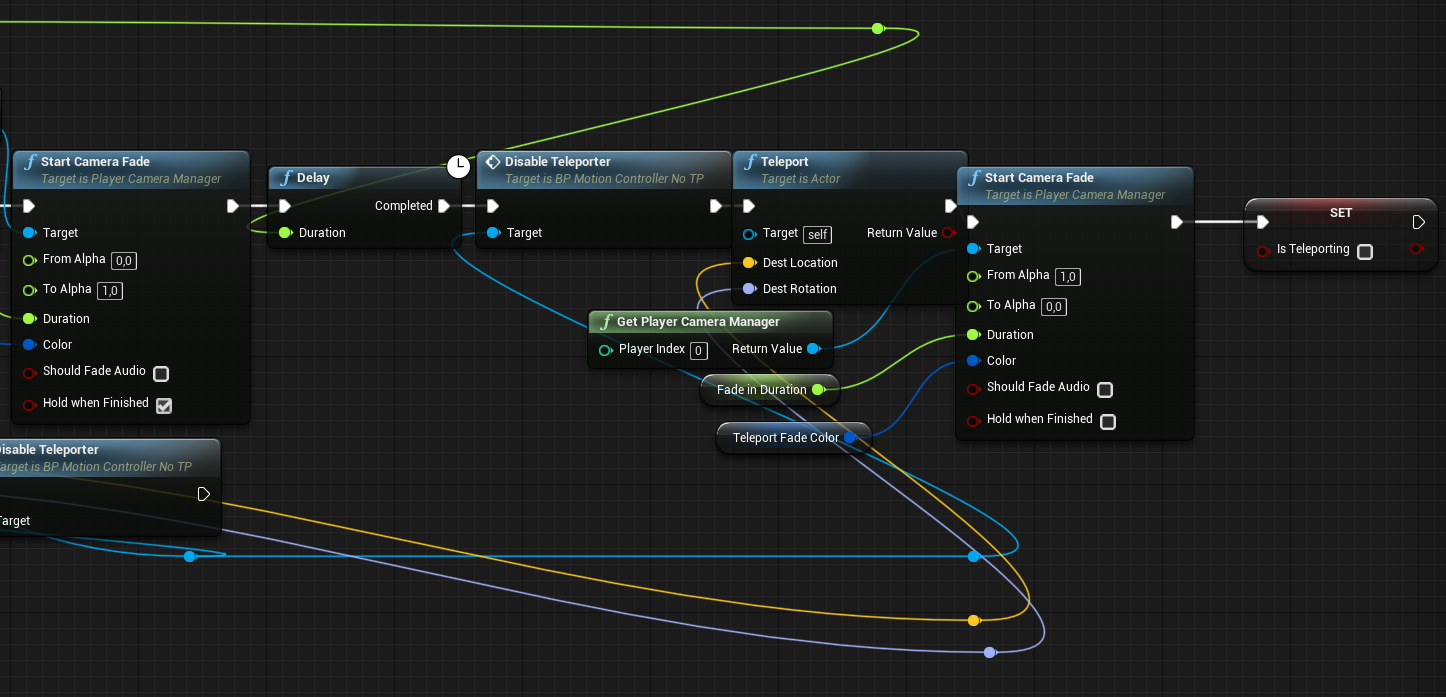
Následne bolo potrebné vytvoriť základnú fyziku pre ruku, aby mohla interagovať s prostredím. Preto sme otvorili model ruky a vytvorili v ňom kolíznu kapsulu. Jej pozíciu sme umiestnili hneď nad prvou kosťou. Vo výsledku pri zapnutí kolízie je ruka v pästi a teda aj kapsula sa nachádza len okolo dlane.

Po úspešnom vytvorení ruky, jej animácie a kolízie sme vytvorili mechaniku hráča. Preto sme vytvorili nový *blueprint* typu *MotionControllerPawn*. V ňom sme spravili premenné: LeftController typu BP *Motion Controller*, RightController typu BP *Motion Controller,* FadeOutDuration typu *float*, FadeInDuration typu *float*, IsTeleporting typu *boolean*, TeleportFadeColor typu *Linear Color*, ThumbDeadzone typu *float*, RightStickDown typu *boolean*, LeftStickDown typu *boolean*, DefaultPlayerHeight typu *float* a UseControllerRollToRatet typu *boolean*. V Event Graphe sme vytvorili *event* BeginPlay, ktorý ide do sekvencie. Tá najskôr nastaví výšku hráča vzhľadom na rôzne platformy pomocou mena *HMDDevice*. Následne sa spustí druhá vetva sekvencie, ktorá zapne novú sekvenciu. Tá umiestni hráča na mape a vytvorí z modelu ruky aj druhú ruku, ktorá sa objaví zrkadlovo. Zároveň sa cez settery nastaví každej ruke jeden ovládač, teda RightController a LeftController. Obe ruky sa pripevnia na VROrigin. Následne sme sa rozhodli urobiť najprv pohyb pomocou teleportácie. Vytvorili sme teda *event* ExecuteTeleportation. Ten vedie do vetvy, kedy pokiaľ nie je splnené IsTeleporting, ide do druhej vetvy, v ktorej kontroluje, či je platná destinácia, kam sa chce hráč teleportovať. Pokiaľ áno, IsTeleporting sa nastaví na *true* a zapne sa funkcia Start Camera Fade, ktorá využíva premennú Fade Out Duration a jej *defaultnú* hodnotu 0,1. Následne sa prenesie hráč na novú pozíciu a znova sa využije funkcia Start Camera Fade. Po tejto funkcií sa nastaví hodnota IsTeleporting na *false*. Ďalej sme vytvorili *event* InputAction TeleportLeft, ktorý po stlačení tlačidla na ovládači aktivuje *event* Teleporter pre ľavý ovládač a pre pravý vypne. Po uvoľnení tlačidla sa hráč presunie na miesto, na ktoré ukazoval. To isté sme zopakovali, ale naopak pre pravú ruku s *eventom* InputAction TeleportRight. Poslednú vec, ktorú sme nastavili v tomto *blueprinte* bol *input* pre uchopenie vecí. Vytvorili sme teda *event* pre InputAction GrabLeft a InputAction GrabRight, aby ruka po stlačení uchopila objekt a po uvoľnení objekt pustila. Obe udalosti sa vykonajú zavolaním *eventov* Grab Actor a Release Actor.

Ďalej sme vytvorili nový *blueprint* BP\_MotionController. V ňom sme si vytvorili premenné: Hand typu *EController Hand*, AttachedActor typu *actor*, IsTeleporterActive typu *boolean*, SplineMeshes typu *Spline Mesh Component Array*, bLastFrameValidDestination typu *boolean*, IsRoomScale typu *boolean*, TeleportDestination typu *vector*, IsValidTeleportDestination typu *boolean*, GripState typu *Grip Enum*, bWantsToGrip typu *boolean*, TeleportRotation typu *rotator*, TeleportLaunchVelocity typu *float* a InitialControllerRotation typu *rotator*.

Obrázok 17 Premenné v blueprinte BP\_MotionController

Ako prvé sme vytvorili event BeginPlay, ktorý zistí, kam sa môže hráč teleportovať a kam nie, na základe vymedzenej plochy a objektov, ktoré sú na mape. Následne zviditeľní TeleporterCylinder, ktorý umožní hráčovi vybrať pomocou ovládača miesto, kam by sa chcel presunúť. Po tomto sa cez switch využije model ruky, ktorý vytvorí pomocou funkcie Set World Scale 3D zrkadlovú kópiu ruky. Tej sa priradí správny Controller vďaka *blueprintu* MotionControllerPawn. Ďalej sme vytvorili event Tick. Ten v sekvencií pomocou vetiev a podmienok nastavuje pomocou setterov Grip State na *Grab*, *CanGrab* a *Open*. Rovnako Tick *event* kontroluje, či sa hráč nechce teleportovať. Ak áno, vezme sa vektor, na ktorý hráč ukazuje ovládačom a na jeho základe sa vytvorí nová lokácia, do ktorej sa hráč prenesie. Na záver sme spravili *event* OnComponentHit(ControllerMesh). V ňom sme priradili, že pokiaľ dôjde k normálovému impulzu, vyráta sa dĺžka vektoru a tá sa bude rovnať intenzite vibrácií ovládača pri náraze.



Obrázok 18 Event ExecuteTeleportation

### Mainmenu

Po úspešnom vytvorení projektu a základných mechaník hráča sme začali pracovať na prvej mape, ktorá bude slúžiť ako hlavné menu. Vytvorili sme preto novú mapu, ktorú sme nazvali MainMenu. Na tejto mape sme použili *assety* z *marketplacu*, jednalo sa hlavne o 3D modely a textúry. Najskôr sme vytvorili pozadie na mape, ktoré využíva textúru Hong Kongu. Ďalej sme vytvorili miestnosť, ktorá by mala pripomínať obývačku. Vytvorili sme balkónové dvere, cez ktoré je možné vidieť textúru nočného mesta a vytvorili BlockingVolume, aby sme nevypadli z balkóna. Ďalej sme do miestností pridali pohovku a kreslo. Rovnako sme na aplikovali BlockingVolume, aby sme do nich neprepadli. Následne sme pridali stôl, obrazy, skrine, vázy a iné dekoračné maličkosti. Na záver sme pridali stropové osvetlenie a svetlo, aby sme na mape niečo videli. Pri niektorých objektoch sme využili plugin *Apex* a vytvorili z nich objekty, ktoré je možné robiť, tzn. DestructibleActor. Pri každom sme použili Fracture Mesh a zapli Impact Damage. Damage Threshold, teda koľko *damage* potrebujú objekty na zničenie sme nastavili na 1000. Následne sme aj tieto objekty umiestnili do miestnosti. Pokiaľ hráč do nich udrie, môže ich rozbiť, prípadne s nimi hýbať. Rovnako sme vytvorili trojicu dverí: Volleyball, Lightsaber a Baskteball. Každé dvere majú na kľučke svoj vlastný *blueprint*. Napríklad mapa Volleyball má *blueprint* TP\_Volleyball. Ten obsahuje Collision Box, ktorý pri spustení eventu Pickup zapne novú mapu s názvom Volleyball. Ostatné dvere to majú rovnako, avšak každý podľa svojej mapy. Do pozadia sme pridali jazzovú hudbu, ktorá bude hrať počas spustenia.

Ďalej sme do miestnosti pridali panel, ktorý ukazuje návod, ako sa ovláda dané prostredie. Následne sme sa rozhodli využiť v menu iný druh pohybu a teda pohyb na základe pozerania kamery a stlačenia tlačidla. V *blueprinte* MotionControllerPawn sme vypli *event* InputAction Teleport pre obe ruky. Pre pohyb na základe tlačidla sme vytvorili nový *event* InputAxis MotionControllerThumbLeft\_X. Ten vezme objekt Camera, zistí jeho predný vektor. Následne sa využije funkcia Add Movement input, ktorá na základe vektoru z kamery a stlačenia tlačidla, začne posúvať hráča. To isté aplikujeme aj pre *event* InputAxis MotionControllerThumbLeft\_Y. Tým sa docieli požadovaný efekt a používateľ sa bude môcť voľne pohybovať po miestnosti.

Aby sme mohli predmety chytať a dvíhať, musíme vytvoriť nový *blueprint*. V *blueprinte* sme pridali StatichMeshComponent. Static Mesh sme menili podľa toho, aký objekt sme chceli zobrať. V Event Graphe sme vytvorili tradične *event* Tick. Ten ide do Flip Flopu. Flip Flop po zavolaní vykonáva vetvy za radom, v akom sú poradí. Po dokončení poslednej vetvy sa znova vráti k prvej vetve. Ten využíva pre vetvu A funkciu AddActorWorldOffset a na osi X nastavuje deltu o 0,1. To isté sa deje aj vo vetve B, ktorá má však hodnotu osi X -0,1. Potom sme vytvorili *event* Pickup. Ten začne simulovať fyziku a použije funkcie AttachToComponent a pripevní používateľovi predmet k rukám. Niečo podobné sa deje v *evente* Drop. Rovnako sa začne pre StaticMeshComponent simulovať fyzika, avšak sa využíva funkcia DetachFromActor.

Obrázok 19 Mainmenu

### Volleyball mapa

Najskôr sme vytvorili ostrov pomocou nástroja *landscape*. Po vytvorení súše, sme vytvorili navôkol ostrova vodnú plochu, ktorá ostrov obmýva. Na okrajoch vodnej plochy sú BlockingVolume, ktoré zabránia prepadu hráča do textúry. Následne sme pridali vyschnutú trávu vďaka nástroju *brush*. Potom sme vymodelovali v programe Cinema 4D volejbalové ihrisko. Proces tvorby zahŕňal prácu s kockami, ich naťahovanie, spájanie. Pri tvorbe siete však boli využité priamky, ktoré sa následne vyplnili. Na mape nesmie chýbať návod na obsluhu daného prostredia.

Po vymodelovaní bolo volejbalové ihrisko importované do prostredia a následne aj umiestnené na ostrov. Na model ihriska sme vložili textúru, ktorá je biela. Rovnako sme do pozadia rozmiestnili rôzne sochy a stromy, ktoré majú dotvárať atmosféru. Takisto v pozadí hrá havajská hudba z rozprávky Lilo a Stitch. Na ihrisku sa nachádza aj stolík, ktorý slúži na generovanie volejbalových lôpt. Vytvorili sme si teda nový *blueprint* na objekty, ktoré sa dajú uchopiť a nastavili sme naň model volejbalovej lopty, ktorý sme vymodelovali a natiahli naň textúru volejbalovej lopty.

Pohyb sme v tomto prostredí ponechali z hlavného menu, avšak v *blueprinte* MotionControllerPawn sme vytvorili nový *event* MotionController (R) Facebutton1. Ten zistí cez funkciu Get Current Level Name názov mapy a následne pokiaľ sa tento názov rovná hodnote Volleyball, tak na mieste, kde sa nachádza stolík vygeneruje volejbalovú loptu po stlačení tlačidla na pravom ovládači. Pokiaľ sa však hodnota rovná Basketball, vygeneruje sa na pozícií stolíku v mape Basketball basketbalovú loptu. To iste sa udeje, ale so svetelným mečom na mape Lightsaber.

Na záver sme vytvorili *blueprint* CountPoints. Ten obsahuje Collision Box a Text. Vytvorili sme v ňom *event* On Component Begin Overlap. Ten po stretnutí s iným komponentom, v tomto prípade s volejbalovou loptou, priráta do textu body + 1. Ďalej pomocou funkcie Get Overlapping Components vygeneruje pole objektov, ktoré sa s boxom zrazili, z tohto poľa vytvorí list a všetky komponenty odstráni, tzn. že volejbalová lopta po dopade priráta body a zároveň sa zničí. Posledná vec, ktorú sme urobili bolo pridanie dverí, ktoré keď používateľ otvorí, ocitne sa znova v hlavnom menu.

Obrázok 20 Mapa Volleyball

### Lightsaber mapa

Pri vytváraní mapy Lightsaber sme využili znova *assety* z *marketplacu* a teda rôzne kamene. Následne sme scénu vyskladali z rôznych svetiel, kameňov, stĺpov, mlák vody a zrúcanín. Ako pri predošlom prostredí, aj tu sme umiestnili dvere, ktoré vrátia používateľa do hlavného menu. Svetelný meč do videa sme vytvorili za použitia dvoch *blueprintov*. Prvý sme nazvali SaberBase a teda bude obsahovať svetlo meča. Čo sa týka komponentov obsahuje Box Collision s DeflectDir pre odrážanie a určenie smeru odrazu. Ďalej obsahuje svetlo meča a text, ktorý bude rátať skóre, teda koľko svetelných striel sa podarilo používateľovi odraziť. Vytvorili sme *event* TurnOn a TurnOff. Pri *evente* TurnOff sme skontrolovali cez premennú IsActive typu *boolean*, či je meč zapnutý. Ak je, vypne sa a rovnako sa vypnú aj kolízie. V opačnej funkcií ak nie je aktívny, meč sa zapne. Ďalší event, ktorý vytvoríme, je *event* On Component Begin Overlap (Collision), ktorý po úspešnom odrazení pripočíta body do komponentu Text, ktorý sa nachádza vedľa svetla meča.

Obrázok 21 Mapa Lightsaber

Ďalší *blueprint* sme nazvali HandBase. Ten obsahuje komponenty Saber a Static Mesh rukoväte svetelného meča. Čo sa týka *eventov*, obsahuje *event* Pickup, ktorý rukoväť meča pripevní k ruke. Rovnako obsahuje aj *event* Force Grab, ktorý umožňuje používateľovi pritiahnuť si k sebe meč pomocou sily. Ten využíva simulovanie fyziky a vektory hráča a meča. Pokiaľ je meč v blízkej vzdialenosti používateľa a hráč naň mieri, je možné, aby si ho k sebe pritiahol. Kvôli tejto mechanike sme museli upraviť aj BP\_MotionController. V ňom sme vytvorili nový *event* ForceGrab. Ten zistí, či je ForceGrab zapnutý a ak áno, umožní hráčovi priťahovať k sebe objekty typu ako je Saber. Následne v *blueprinte* MotionControllerPawn sme vytvorili *eventy* na zapnutie *gripu* pomocou eventov na tlačidlá ovládača. Rovnako sme nastavili, aby pri stlačení pravého tlačidla sa zapol alebo vypol meč, ak ho používateľ drží, samozrejme pre každú ruku zvlášť.

Ďalej sme importovali do hry model nepriateľa z Mixama, ktorý bude po hráčovi strieľať. Vytvorili sme *blueprint* LaserBullet. V tom sme vytvorili *event* On Component Hit (Box). Ten po vystrelení vytvorí iskry, vytvorí zvuk výstrelu a letí na pozíciu v určitom náhodnom rádiuse od hráča. *Blueprintu* nepriateľa sme vytvorili *event* Shoot, ktorý pri každom výstrele vytvorí LaserBullet, ktorý bude letieť smerom na používateľa.

Na záver sme pridali nový pohyb hráča, aby každé prostredie obsahovalo výnimočný pohyb, pri ktorom môžeme v testovaní aplikácie vyhodnotiť určité závery. Pohyb Arm Swing docielime odpojením pohybu teleportácie a pohybu nasledovania smeru, ktorým sa pozerá kamera. Potom nastavíme Tick *event*, aby vypočítaval cez funckiu Calculate Arm Swinger Movement pohyb kamery. V praxi to znamená, že pokiaľ hráč bude držať tlačidlo a nastane pohyb Controllera, kamera sa začne posúvať v smere, akým sa hráč pozerá. To umožní držať tlačidlo a hýbať rukami pre dosiahnutie reálnejšieho pohybu.

### Basketball mapa

Poslednú mapu sme vytvorili ako basketbalové ihrisko. Niektoré modely sme použili z *assetov* a iné sme vytvorili v programe Cinema 4D. V tejto mape sme využili pohyb teleportácie, ktorý sme robili na začiatku a v predošlých mapách sme ho vypli. Postavili sme halu a natiahli sme na ňu textúry. Podobne ako pri volejbale, aj tu sme využili *blueprint*, avšak sme použili model basketbalovej lopty. Ako pri ostatných mapách, aj tu sme pridali dvere, ktoré hráča prenesú do hlavného menu. Na mape sme pridali basketbalový kôš. Vytvorili sme nový *blueprint*, ktorý obsahuje komponenty Collision Sphere a Text. V *blueprinte* sme spravili *event* On Component Begin Overlap (Sphere), pričom pokiaľ lopta dopadne do koša, prirátajú sa body, ktoré sa zvýšia o 1, konvertujú sa z *integer* na *string* a nastavia sa ako hodnota pre komponent Text.

Po stlačení tlačidla na pravom ovládači sa na mape vygeneruje basketbalová lopta. Po úspešnom hode na kôš sa prirátajú body, ktoré sa nachádzajú vedľa koša. Pohyb na mape prebieha pomocou teleportácie, kedy hráč drží tlačidlo na ovládači a vyberie miesto, kam by sa chcel posunúť.

Obrázok 22 Mapa Basketball

## Finalizácia

Po úspešnom implementovaní všetkých mechaník, všetkých modelov, vytvorení všetkých máp sme skontrolovali funkčnosť všetkých komponentov aplikácie. Pre zdokonalenie zážitku sme vymenili niektoré textúry za kvalitnejšie a zlepšili osvetlenie. Následne sme nechali *vybuildovať* osvetlenie v každom prostredí na kvalitu Production.

Potom sme otvorili nastavenia projektu v Setting v Project Settings. Vytvorili sme pre projekt nový náhľadový obrázok, ktorý sme vytvorili v programe Photoshop CC, ktorý bude slúžiť ako logo aplikácie. Nastavili sme Publisher na meno Martin Bako. Projekt sme nazvali VR Interactivity. Rovnako sme v záložke Maps & Modes nastavili, aby začiatočná mapa bola MainMenu. Medzi podporované platformy sme zvolili Windows. Ako posledné sme vybrali, že chceme Shipping verziu, ktorá sa dokáže spustiť bez nutnosti vlastnenia zdrojových súborov. Vybrali sme Package Project a Windows 64-bit. Následne sme len čakali na *build* hry a na záver sme dostali spustiteľný .exe súbor, ktorý sme potom mohli testovať.