SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6238

Miješanje animacija modela čovjeka

Lovro Katalinić

Zagreb, lipanj 2019.

SADRŽAJ

1. Uvod

Završni rad prati izradu skeletalnog modela čovjeka, njegovo animiranje i omogućavanje interaktivne aplikacije animacija na stvorenom modelu. Takav model može se, primjerice, koristiti u izradi računalnih i mobilnih igara ili u simulaciji. Trodimenzionalni model tijela izrađen je unutar programske podrške Blender. Modelu se dodaju osnovne teksture. Pojedini dijelovi modela dodijeljuju se kosturu tako da prate njegove pokrete. Stvaraju se osnovne animacije poput hodanja i trčanja. Model i animacije izvoze se iz Blendera i uvoze u grafički pogon Unreal. Animacije hodanja i trčanja se, ovisno o trenutnoj brzini lika, stapaju pomoću alata kojeg nudi sam pogon. Projekt unutar Unreal-a konfigurira se tako da omogućuje osnovno kretanje lika u jednostavnom prostornom okruženju putem tipkovnice i miša.

2. Izrada skeletnog modela čovjeka

U računalnoj grafici trodimenzionalno modeliranje predstavlja proces preslikavanja matematičke reprezentacije plašta nekog objekta u tri dimenzije, pomoću nekog softvera. Modeli objekata danas se koriste unutar širokog raspona područja. Neizostavni su u industriji video igara, koriste se u medicini za vizualizaciju organa, potpora su inženjerima koji se na njih oslanjaju pri dizajniranju novih uređaja [9].

Unutar ovog poglavlja opisat će se izrada trodimenzionalnog modela čovjeka, stvaranje i aplikacija jednostavnih tekstura na model te montaža modela tijela na skeletni model (engl. *rigging*), koja će omogućiti njegovo animiranje.

2.1. Modeliranje tijela čovjeka

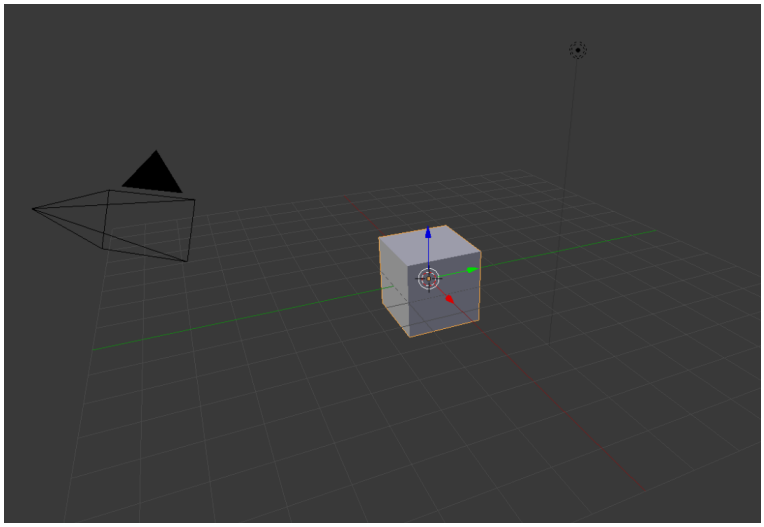
2.1.1. Postupak modeliranja u Blenderu

Do današnjeg dana razvijeni su brojni softveri za dizajniranje 3D objekata, od čega je velik broj njih besplatan za preuzimanje.

Blender je besplatan softver otvorenog koda koji pored modeliranja 3D objekata omogućuje izradu tekstura, simulacija, animacija, iscrtavanje prikaza na dvodimenzionalnom zaslonu (engl. *rendering*) te uređivanje videozapisa [2]. U sklopu završnog rada korišten je za modeliranje čovječuljka, stvaranje tekstura i animiranje.

Proces izrade započinje stvaranjem nove scene. Početna scena sastoji se od dvodimenzionalne koordinatne mreže postavljene na ravninu $z = 0$, modela kocke postavljenog u ishodište te lampe i kamere koji se koriste za projekciju scene na specifičnu ravninu (slika 2.1).

Program korisniku pruža nekoliko različitih uređivača (engl. *editors*) za prikaz i modifikaciju različitih aspekata podataka. Najkorišteniji uređivač je 3D prikaz scene. Unutar opsega završnog rada koriste se i ploča animacija (engl. *Dope Sheet*), UV uređivač (engl. *UV/Image editor*) za teksture te tekstualni uređivač za pisanje i izvršenje skripti.



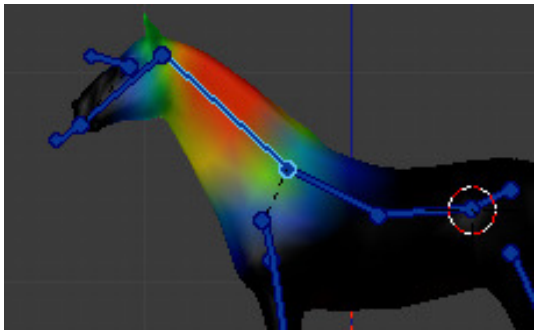
Slika 2.1: Početna scena Blendera

Tijekom modeliranja u 3D uređivaču kombinira se nekoliko različitih načina rada. Objektni način rada (engl. *Object Mode*) koristi se za stvaranje novih primitivnih oblika u sceni. Neki od primitivnih oblika koje Blender nudi su ravnina, kocka, kružnica, cilindar i torus. Pored stvaranja primitiva, ovaj način rada omogućuje jednostavne transformacije nad odabranim objektima: pomicanje, skaliranje i rotiranje.

Drugi, često korišten, način rada je uređivački način rada (engl. *Edit Mode*). U ovom načinu korisnik može uređivati manje jedinice objekta kao što su njegovi vrhovi, bridovi koji povezuju vrhove ili pak poligoni.

Od načina rada koji se koriste za bojanje, unutar rada koriste se bojanje tekstura (engl. *Texture Paint*) i težinsko bojanje (engl. *Weight Paint*). Bojanje tekstura koristi se kod oslikavanja površine tijela. U završnom radu tijelo ćemo oslikati na način da se na modelu čovjeka mogu raspoznati majica, hlače, obuća i kosa od ostatak tijela. Alternativa bojanju tekstura je bojanja vrhova (engl. *Vertex Paint*) kojim se svakom vrhu tijela pridodaje vrijednost određene boje, a poligoni su obojani na način da se interpoliraju vrijednosti boja svih vrhova kojima je on razapet. Težinsko bojanje koristi se pretežito kod povezivanja modela s njegovim kosturom. Ono prikazuje koliki učinak imaju deformacije određene kosti na sam model na vrlo intuitivan način. Dijelovi modela koje deformacije kostiju više pogađaju označavaju se crvenom bojom, dijelovi koje uopće ne pogađaju plavom, a ostatak jednom od boja spektra između ove dvije krajnosti, ovisno o intenzitetu utjecaja deformacije na tijelo[5]. Primjer utjecaja kosti vrata na tijelo konja, u ovom načinu rada, prikazan je na slici 2.2 [4].

Nakon što se model poveže s armaturom, moguće ga je animirati. Određeno stanje armature u nekom trenutku naziva se pozom. Način rada uređivanja poza (engl. *Pose*



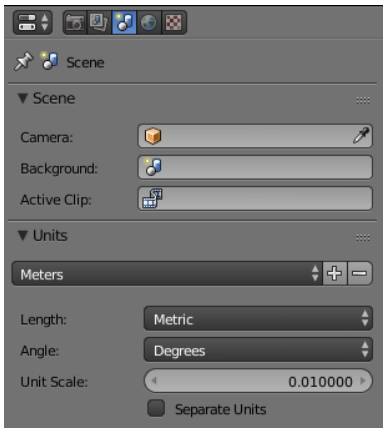
Slika 2.2: Težinsko bojanje

Mode) pruža mogućnost olakšanog manipuliranja kostima i stvaranje različitih poza. Ovakav način rada dopušta da se kosti transformiraju jednako kao i objekti. Tako je moguće kosti translatirati, rotirati ili skalirati u odnosu na početnu pozu mirovanja.

2.1.2. Modeliranje tijela čovjeka

Podešavanje scene

Kako bi model koji napravimo u Blenderu mogli prenijeti u Unreal bez dodatnih konfiguracija, potrebno je podesiti skalu globalnog sustava prije same izrade. Jedna jedinica u Blenderu preslikava se u 1 centimetar unutar Unreal-a, zbog čega je potrebno jedinicu u Blenderu skalirati na 1 cm. Također, zbog kompatibilnosti s Unreal-om potrebno je broj prikazanih slika u sekundu (engl. *frame rate*) postaviti na 30, kako se animacije ne bi poremetile.

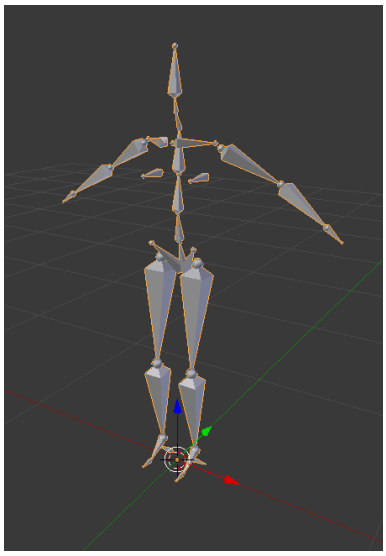


Slika 2.3: Skaliranje jedinica u Blenderu

Umetanje ljudskog kostura

Tijekom stvaranja modela čovjeka oslanjat ćemo se na raspored kostiju skeletona. Koristit ćemo dodatak *Rigify* (verzija 2.76b) za automatsku generaciju ljudskog kostura. Dodatak dopušta korisniku generiranje kostura čovjeka te kostura nekoliko životinja kao što su mačka, ptica ili vuk. Detaljnije o ovom alatu, stvaranju armature unutar Blendera te spajanju modela s kostima bit će objašnjeno u poglavlju 2.4.

Novi kostur u scenu dodajemo pomoću naredbe *Add* → *Armature* → *Human (Meta-Rig)*. Nakon stvaranja kostur je potrebno skalirati na normalnu veličinu ljudskog tijela (u ovom slučaju 100 puta). Prikaz kostura nakon skaliranja vidljiv je na slici 2.4.

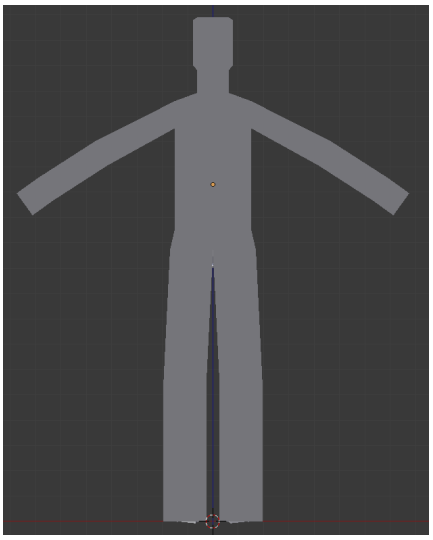


Slika 2.4: Generiranje ljudskog kostura

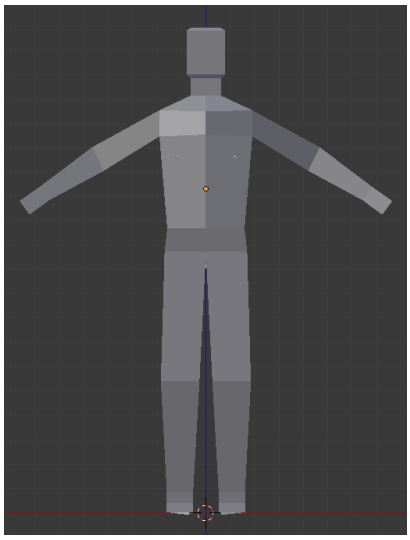
Stvaranje modela čovjeka

Stvaranje modela započinjemo dodavanjem primitiva kocke. S obzirom da su lijeva i desna strana ljudskog tijela simetrične, olakšat ćemo postupak izrade tako da oblikujemo samo jednu stranu te na nju primjenimo modifikator zrcaljenja (engl. *Mirror modifier*). S njime će se svaka promjena na modelu lijeve strane aplicirati i na desnu stranu tijela. Prvo ćemo transformirati objekt kocke tako da, gledano iz prednje perspektive, otprilike poprimi oblik ljudskog tijela. Postupak se sastoji od odabira stranica modela i korištenja alata za izvlačenje (engl. *Extrusion tool*). Alat duplicira označene poligone (ili bridove ili vrhove) i povezuje duplikat s originalnim poligonom, tako da postupak izgleda kao da izvlačimo novi poligon iz starog. Koristeći postupak izvlačenja uz skaliranje novoizvučenih poligona dolazimo do modela nalik ljudskom tijelu

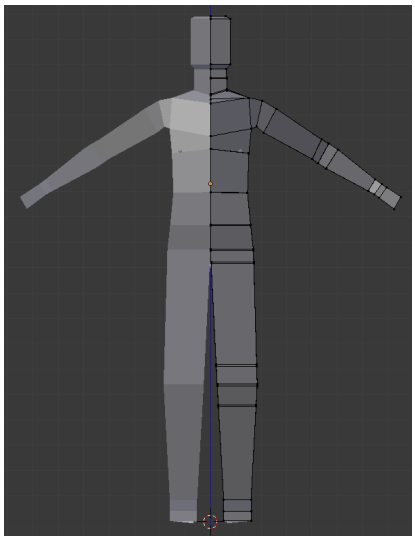
(slika 2.5a). S obzirom da nam je za realniji model potrebno više detalja, dodat ćemo bridove alatom podijele s obzirom petlju (engl. *Loop Subdivide*), koji stvara novu petlju bridova između dvije postojeće petlje. Stvorit ćemo nove bridove tako da se oni nalaze na savitljivim dijelovima tijela (na zglobovima, vratu i trupu). To će omogućiti bolje modeliranje i precizniju animaciju u kasnijim koracima. Nakon dodavanja i transformacije bridova model je poprimio oblik prikazan na slici 2.5b. Daljnjim poboljšavanjem modela čovjeka rotiranjem i skaliranjem određenih poligona dobiven je oblik na slici 2.5c. Oblik je prikazan u uređivačkom načinu rada na kojem je vidljiva podijela modela s obzirom na bridove i poligone[1].



(a) Model nastao izvlačenjem poligona



(b) Model nakon transformacija - objektni način



(c) Model nakon transformacija - uređivački način

(d) Model nakon primjene modifikatora

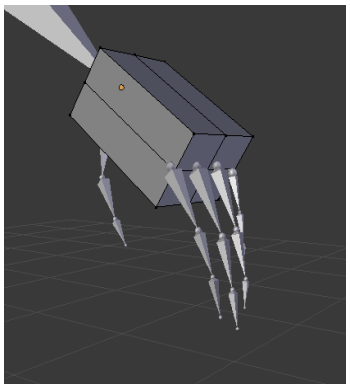
Slika 2.5: Stvaranje modela čovjeka

Stvoreni model priliči obliku ljudskog tijela, međutim poligoni od kojih je sastav-

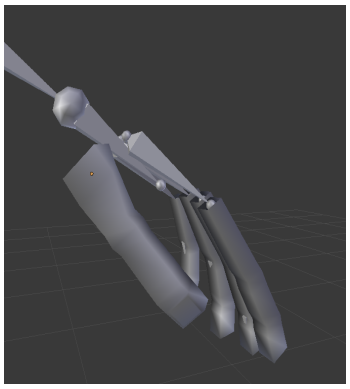
ljen preveliki su i prijelazi između njih čine se preoštrima, stoga je potrebno izgladiti površinu tijela. Površinu ćemo izgladiti tako da već postojeće poligone podijelimo na više manjih poligona. To možemo napraviti ručno, ili se poslužiti jednim od mnogih modifikatora koje nam Blender daje na raspolaganje. Modifikator podjele površine (engl. *Subdivision Surface*) izgladit će površinu upravo na način da dijeli poligone na više manjih i potom ih transformira tako da prijelazi ne budu grubi. Rezultat primjene modifikatora na model vidljiv je na slici 2.5d.

Stvaranje ljudske šake

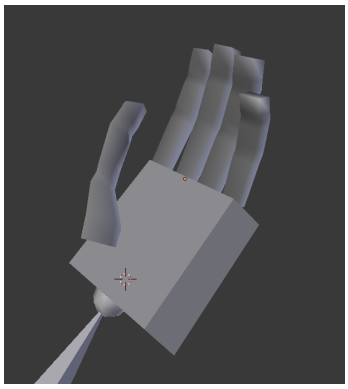
Postupak stvaranja ljudske šake malo je složeniji od dosadašnjeg postupka modeliranja. Na području kostura šake stvorit ćemo kvadar koji će predstavljati dlan i prema obliku kostiju prsta stvoriti oblik nalik na kažiprst. Ostale prste ćemo stvoriti tako da dupliciramo kažiprst i prilagodimo ga obliku armature tih prstiju. Nakon što su svi prsti stvoreni, pojedine poligone dlana i dno prstiju povezat ćemo na način da ćemo između odabranih bridova stvoriti nove poligone koji ih povezuju. Na model ljudske šake također ćemo primjeniti modifikator podjele površine i zatim ga povezati s ostatkom ljudskog tijela. Korake modeliranja ruke moguće je vidjeti na slici 2.6.



(a) Kvadar koji predstavlja dlan



(b) Modeli prstiju šake



(c) Prsti i dlan prije povezivanja

Slika 2.6: Stvaranje ljudske šake

2.2. Odmatanje teskture (UV unwrapping)

Nakon izrade modela čovjeka, dobili smo mrežu poligona koja opisuje njegovu strukturu. Idući korak je oslikavanje te teksture kako bi tijelo dobilo realniji prikaz. Korak teksturiranja sastoji se od pripreme postojeće mreže za izradu tesktura njeno oslikavanje.

2.2.1. UV mapiranje

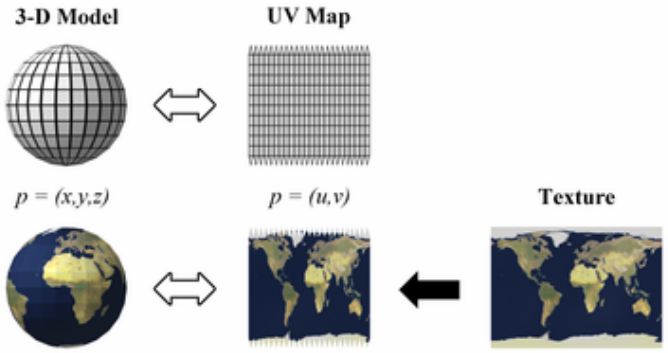
Za dodavanje tekstura na model potrebno je trodimenzionalni plašt modela (kojeg čine poligoni) preslikati na dvodimenzionalnu mrežu. Takav proces naziva se UV mapiranje. Naziv dolazi od imena koordinata ploče *U* i *V*, koje su projekcija originalnih koordinata *X*, *Y* i *Z* iz trodimenzionalnog prostora (slika 2.7).

Za svaku točku modela P moguće je pronaći odgovarajuće UV koordinate na način da se izračuna jedinični vektor $\vec{d}$ koji je razapet između točke P i ishodišne točke modela. Tada su UV koordinate iz intervala $[0, 1]$ definirane formulama:

$$u = 0.5 + \frac{\arctan2(d_z, d_x)}{2\pi}$$

$$v = 0.5 - \frac{\arcsin(d_y)}{\pi}$$

Preslikavanje na dvodimenzionalnu mrežu omogućuje teksturiranje modela slikom koju nazivamo UV mapom tesktura. Mapiranje se sastoji od tri osnovna koraka: odmatanje mreže modela, stvaranje teksture i njena primjena na model[7] [10].

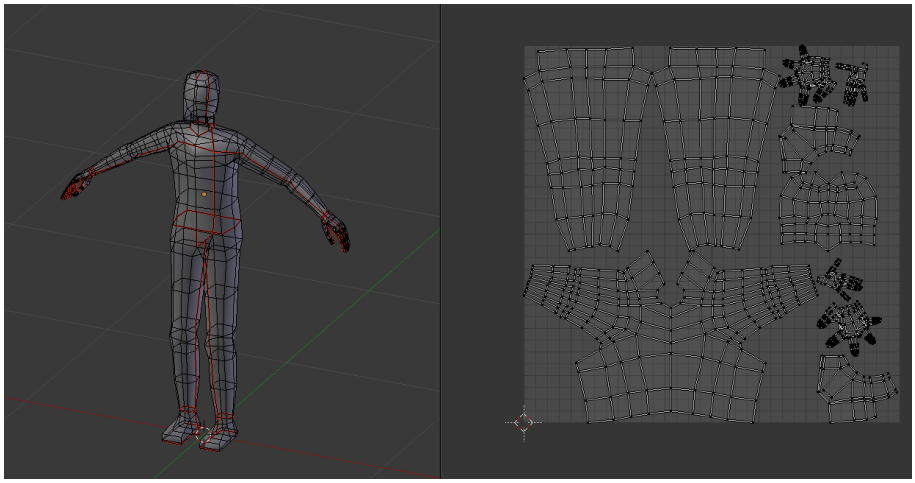


Slika 2.7: Preslikavanje tijela na dvodimenzionalnu mrežu

2.2.2. Odmatanje mreže modela

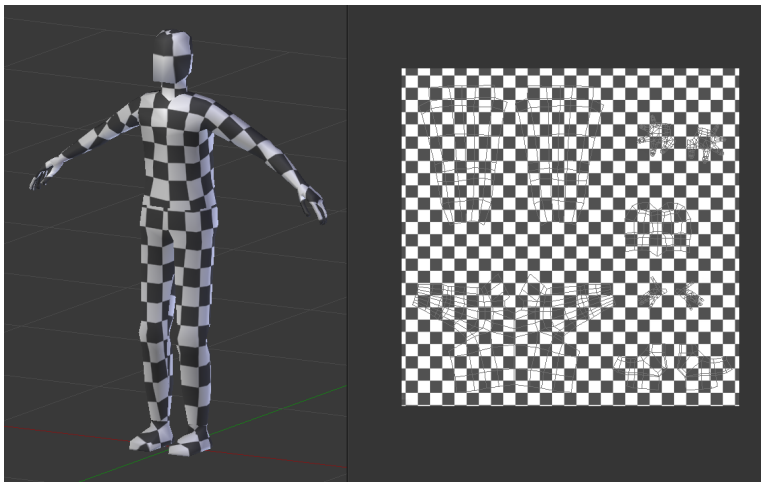
Odmatanje mreže može se raditi automatiziranim postupcima software-a u kojem modeliramo ili ručno. Najčešće se ono zbiva prvo automatski, nakon čega osoba koja modelira dotjeruje generiranu mrežu.

U konkretnom slučaju, automatizirano odmatanje proizvelo bi previše distorzije (deformacije). Zbog toga što je model sastavljen od velikog broja zaobljenih dijelova, automatizirano preslikavanje na dvije dimenzije bez dodatnih smjernica vrlo vjerojatno će prouzročiti deformaciju oblika i veličina poligona. Stoga ćemo odmotavanje



Slika 2.8: Označavanje šavova na modelu

modela čovjeka započeti označavanjem šavova (engl. *seam*) na određenim dijelovima tijela. Na lijevoj strani slike 2.8 crvenom bojom su označeni šavovi po kojima je plašt čovjeka pripremljen za preslikavanje, dok se na desnoj strani slike nalazi generirana mreža, na kojoj je moguće raspoznati pojedine dijelove ljudskoga tijela (torzo i ruke, noge, šake, glava i stopala). Označavanje odabranih bridova šavovima omogućuje generiranje mreže na način da se poligoni koji dijele brid označen kao šav razdvoje na UV mapi, što će rezultirati manjim deformacijama i olakšati bojanje različitih dijelova tijela koji se na modelu dodiruju.

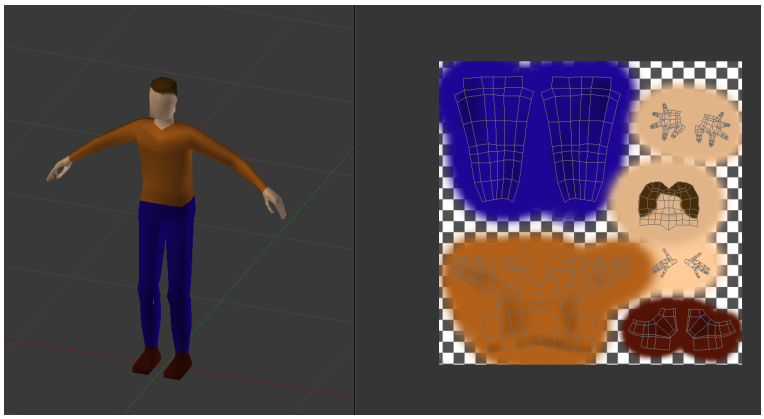


Slika 2.9: Primjena teksture šahovske ploče

Provjeru ispravnosti mreže provjerit ćemo primjenom teksture šahovske ploče na stvoreni model. U slučaju kad je uzorak ploče proporcionalno rastegnut na model, razina distorzije na mreži je minimalna, što je vidljivo na slici 2.9.

2.3. Stvaranje osnovnih tekstura

Nakon odmatanja modela, moguće je na njega primjeniti teksture. U našem slučaju, teksturu ćemo stvarati tako da određene dijelove tijela obojamo različitim bojama unutar načina rada bojanja tekstura. To možemo učiniti primjenom kista na našu trodimenzionalnu reprezentaciju ili na njegovu UV mapu. Za dijelove tijela koji zahtijevaju oštru granicu između tekstura bojat ćemo UV mapu. Prvo ćemo cijelu teksturu obojati bojom ljudske kože. Nakon toga dijelove mreže na kojima se nalaze hlače obojat ćemo u plavo, one na kojima se nalazi majica narančastom, a one na kojima se nalaze cipele smeđom bojom. Kosa ne zahtijeva oštru granicu s ostatkom lica, stoga ćemo nju naslikati direktnom primjenom kista na model.



Slika 2.10: Stvaranje osnovnih tekstura

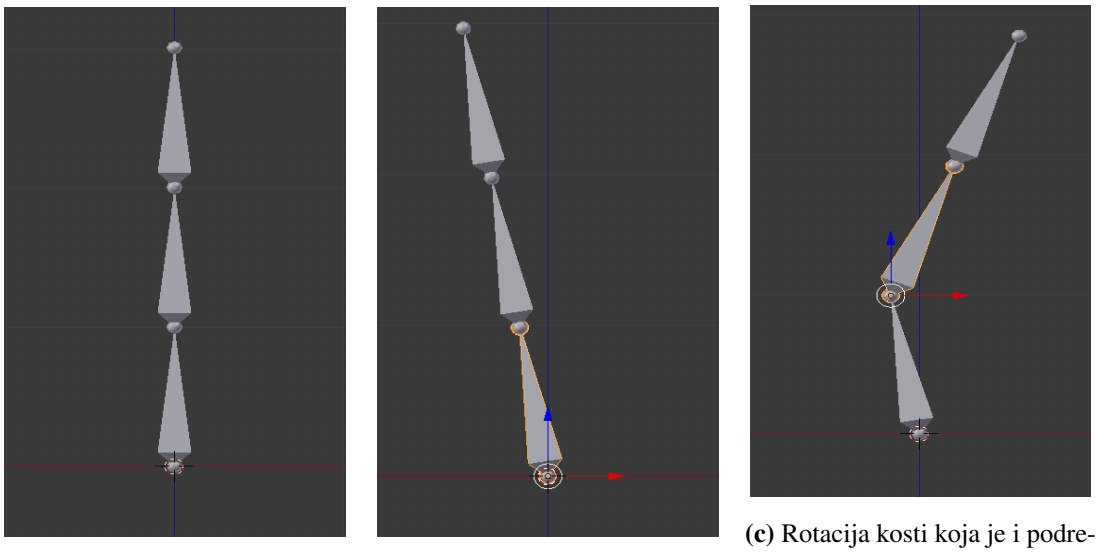
Nakon nanošenja osnovnih boja na mapu tekstura, dotjerat ćemo ih potamnjivanjem određenih dijelova kako bi postigli realniji prikaz. Nakon toga na teksturu ćemo primjeniti premazni kist (engl. *Smear brush*) koji će izgladiti prijelaz između svijetlijih i tamnijih nijansa boje. Rezultate stvaranja tekstura moguće je vidjeti na slici 2.10. Kako bi stvorenu teksturu mogli kasnije povezati s modelom i u Unreal pogonu, potrebno ju je spremiti na disk kao slikovnu datoteku.

2.4. Montaža modela tijela na skeletni model

Izraz montaža (engl. *rigging*) u svijetu modeliranja ima značenje dodijele određenog stupnja kontrole objektima. Najčešće se takva kontrola koristi za animaciju. Model montiran na skeleton sastoji se od armature koja omogućuje fleksibilnost na području zglobova te ograničenja koja kontroliraju kretnje.

2.4.1. Armatura u Blenderu

Objekti koji čine armaturu unutar Blendera su kosti. Nalik pravom kosturu, kosti se međusobno povezuju u hijerarhijsku strukturu. Deformacija nadređene kosti tako će utjecati i na stanje svih njoj podređenih kostiju. Osnovna ideja hijerahije kostiju unutar kostura prikazana je na slikama 2.11. Hijerarhija sa slike sastoji se od triju kosti kod kojih je svaka nadređena onima iznad nje (2.11a). Tako će rotacija najdonje kosti utjecati i na gornje dvije (2.11b), a rotacija srednje samo na sebe i na najgornju (2.11c).



(a) Hijerarhijski odnos kostiju

(b) Rotacija korijenske kosti

(c) Rotacija kosti koja je i podređena i nadređena

Slika 2.11: Deformacija kostiju u hijerarhijskom odnosu

U slučaju da želimo pomaknuti neku krajnju kost čovjeka (npr. stopalo) i ujedno pomaknuti ostatak noge, trebali bismo pomicati prvo natkoljenicu, pa potkoljenicu pa tek tada stopalo. U slučaju da postoji još više nadređenih kostiju koje bi pritom također morali pomicati kako bi postigli željeni efekt, ovaj postupak postaje sve složeniji. Stoga nam Blender nudi opciju inverzne kinematike koja kod transformacije kosti automatski transformira i sve roditeljske kosti.

Objekti kostiju mogu se podijeliti u dvije skupine: deformirajuće kosti koje svojom transformacijom uzrokuju deformaciju svega što je povezano s njima, te kontrolne kosti koje ne utječu direktno na sebe, već služe za kontroliranje deformacije drugih kostiju. Svaki objekt kosti, kao što je prikazano i na prethodnim slikama sastoji se od tijela kosti te početnog i završnog zgloba.

Kažemo da kost kontrolira dio tijela kada vrhovi tog dijela tijela slijede pomake kosti. Da bi se postigla kontrola, potrebno je definirati područje učinka kostiju na zadanom modelu. Učinak kosti na model već je prethodno prikazan na slici 2.2.

2.4.2. Montaža modela tijela na skeletni model

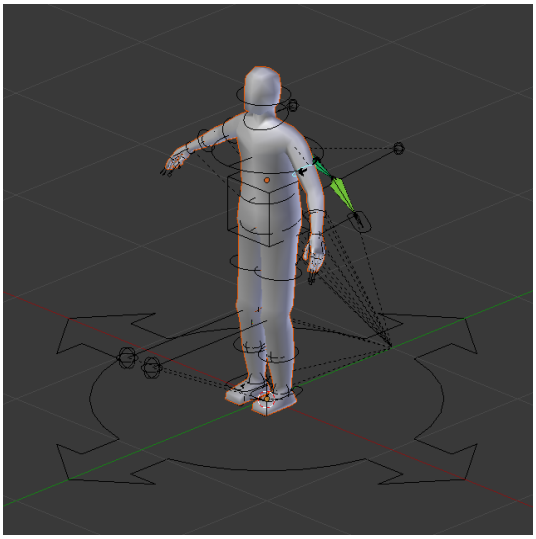
Rad se trenutno sastoji od modela čovjeka i kostura kojeg smo postavili na početku kao referencu za oblik i proporcije ljudskog tijela. Nakon što smo tijelo modelirali, potrebno je montirati model tijela na model skeletona. To će nam omogućiti da svaka kost koju pomaknemo na skeletonu ujedno na isti način pomakne i pripadni dio modela.

Koristili smo dodatak *Riggify* koji nam je izgradio takozvani *meta-rig*. Meta-rig je predložak kostura kojeg možemo modificirati u slučaju da neke kosti moraju biti veće, pod drukčijim kutem ili postavljene na nekom drugom mjestu. Nakon što smo zadovoljni s njegovim oblikom, potrebno je generirati pravu montažu (engl. *rig*). Dodatak kojeg koristimo sam će to napraviti nakon odabira naredbe *Object Data* $\rightarrow$ *Riggify Buttons* $\rightarrow$ *Generate*. Ono što smo ovime dobili su kontrolne strune (engl. *strings*) kojima skeletonom možemo upravljati na način da ga kontroliramo poput marionete. U konkretnom slučaju bilo je potrebno dobivene strune skalirati tako da se poklapaju s kosturom. Nakon generiranja prave montaže potrebno je izbrisati meta-rig i objekte prefiksa *WGT_ (widget)* koji se automatski generiraju, a za potrebe rada neće nam trebati. Bitno je napomenuti da alat neće funkcionirati ako neki dio kostura ostane nepokriven od strane modela. U slučaju da smo zaboravili modelirati ljudsku šaku, taj dio kostura ne bi bio pokriven i alat ne bi znao zaključiti na koji dio tijela utječu kosti ljudske šake.

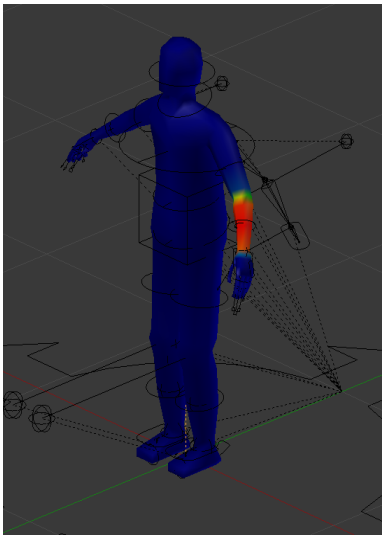
Nakon što je izgrađena montaža potrebno ju je dodatno povezati s modelom. To ćemo napraviti tako da označimo model i potom montažu i odaberemo opciju *Set Parent To* $\rightarrow$ *Armature Deforme* $\rightarrow$ *With Automatic Weights*. Ukoliko nismo zadovoljni s generiranim utjecajem kosti na model, možemo pokrenuti način rada težinskog bojanja i pojačati, odnosno smanjiti utjecaj na određenom dijelu modela.

Na slici 2.12a vidi se rezultat primjene dodatka. Rotacija skeletona transformirat će i ruku modela. Na slici su vidljive i relacije među kostima iscrtkanim linijama. Najočitije su linije koje povezuju svaku kost tijela sa korijenskom kosti. Kružnice oko tijela služe kao kontrolni objekti kojima utječemo na model. Najveća kružnica, sa strelicama koje pokazuju u četiri različita smjera, predstavlja korijensku kost i služi za pomicanje cijelog tijela na neku drugu lokaciju. Slika 2.12b pokazuje utjecaj kosti podlaktice na model putem težinskih boja.

Sada je model spreman za animiranje. Trenutno na sceni imamo model čovjeka koji je povezan sa ljudskim skeletonom. Imamo vrlo detaljan skup kontrolnih struna pomoću kojih možemo transformirati pojedine dijelove tijela.



(a) Utjecaj pomaka skeletona na tijelo



(b) Utjecaj kosti podlaktice na model

Slika 2.12: Montaža modela tijela na skeletni model

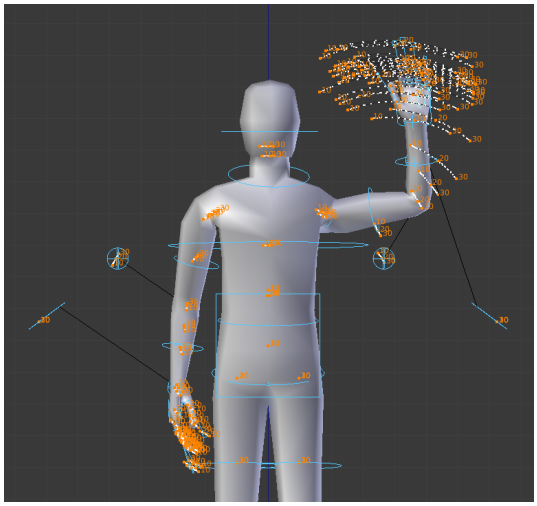
3. Izrada animiranja kretnji

Animacija predstavlja pomak ili promjenu oblika objekta tijekom određenog vremena. Vrijeme trajanja animacije t podijeljeno je na više sličica (engl. *keyframes*). Svaka sličica sadrži vrijednosti svojstava objekata u određenom trenutku. Ako definiramo početni izgled modela u trenutku $t = t_0$ i završni izgled u trenutku $t = t_1$, Blender će automatski odrediti izglede modela u svim sličicama u trenutcima $t \in \langle t_0, t_1 \rangle$ s obzirom na odabranu interpolacijsku metodu (linearna, kvadratna, ...).

Proces animiranja sastoji se od pomicanja određenih dijelova tijela u željenu pozu i spremanje svojstava umetanjem nove sličice na vremensku liniju.

Pored animiranja skeletona, moguće je vremenu promjeniti dio modela bez da transformiramo kost. Takve animacije nazivamo ključnim okvirima. To su, na primjer, promjene izraza lica ovisno o nekim parametrima.

Program pruža i prikaz putanja gibanja koja vizualizira stanje ishodišta objekta ili poziciju zglobova u svakom trenutku odabranog intervala. Prikaz putanje gibanja tijela kod animacije mahanja prikazan je na slici 3.1.

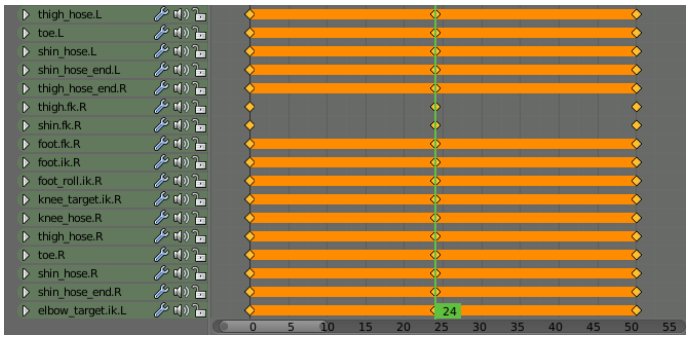


Slika 3.1: Prikaz putanje gibanja

U odjeljku 2.1.1 spomenuto je da Blender sadrži nekoliko različitih uređivača. Ure-

đivač kojeg ćemo pored 3D prikaza koristiti za izradu animacija je ploča animacija. Uređivač prikazuje sličice za sve moguće akcije nad modelom. U završnom radu akcije će predstavljati različite kosti koje je moguće transformirati. Na ploči animacija označit ćemo neki vremenski trenutak, zatim u 3D prikazu namjestiti kostur tijela u željenu pozu, i zatim tu pozu umetnuti kao sličicu koja se pojavljuje u označenom trenutku animacije. Primjer prikaza uređivača za animaciju besposlenosti (engl. *idle*) prikazan je na slici 3.2.

Iz točke do koje smo stigli izvodit će se različite animacije, stoga je potrebno dosadašnji rad u Blenderu spremiti kako bi svaku od animacija mogli raditi zasebno.

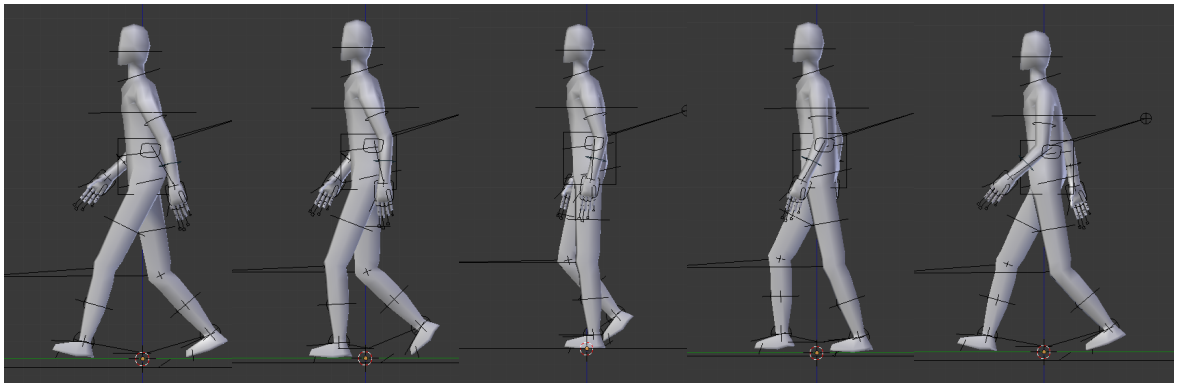


Slika 3.2: Ploča animacija

3.1. Izrada animacije hoda

Animacija hoda vremenski će trajati 50 sličica. Prva polovica predstavljat će prijelaze iz stanja raskoraka u kojem je jedna noga ispred tijela, a druga iza njega, u drugo stanje raskoraka u kojem su pozicije nogu zamjenjene. Prijelaz između ta dva stanja Blender može sam interpolirati, ali ćemo zbog realnijeg prikaza ljudskog hoda upotpuniti stanja modela s dodatne tri sličice između ove dvije krajnosti. Druga polovica animacije bit će zrcaljenje prve polovice u kojoj će se model iz poze na srednjoj sličici obrnutom animacijom vratiti u početnu pozu. Dobivene poze tijela tijekom repliciranja ljudskog hoda unutar načina rada uređivanja poza prikazane su na slici 3.3.

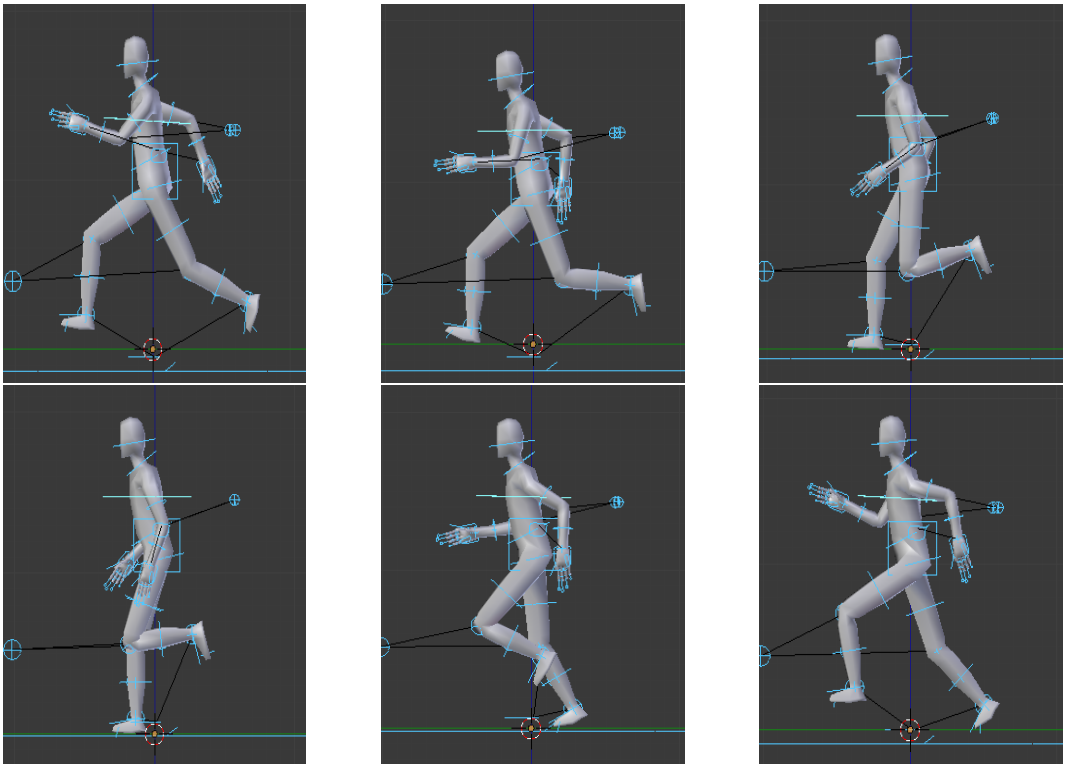
Neke od bitnih stavki animacije hoda su jednoliko pomicanje udova naprijed-natrag pazeći pritom da se suprotna ruka i suprotna noga nalaze na istoj strani u odnosu na trup, lagano naginjanje trupa prema naprijed pri raskoraku te rotacija stopala tako da ono prati podlogu.



Slika 3.3: Poze animacije hodanja

3.2. Izrada animacije trčanja

Animacija trčanja bit će izrađena na sličan način kao i animacija hoda. S obzirom da je trčanje gibanje koje je brže od hodanja, jedan ciklus animacije trajat će nešto kraće od jednog ciklusa hodanja. Također, zbog toga što je trčanje složenija animacija za koju je potrebno više detalja, unutar raspona animacije ručno ćemo odrediti više sličica nego kod hodanja. Jedan ciklus trčanja prikazan je na slikama 3.4.



Slika 3.4: Poze animacije trčanja

3.3. Izrada dodatnih animacija

Za potrebu demonstracije animacija, u sklopu završnog rada napravljene su još dvije jednostavne animacije: mahanje i besposlena poza. Besploslena poza koristit će se u slučaju da karakter stoji na mjestu i ne miče se, a animacija mahanja pokretat će se pritiskom na određenu tipku na tipkovnici. Animacija mahanja već je prikazana putanjom gibanja na slici 3.1, a animacija besposlenosti putem ploče animacija na slici 3.2.

3.4. Pohrana modela i animacija iz Blendera

Model čovjeka i animacije odvojeno ćemo pohraniti na disk. Datoteke je potrebno spremiti s ekstenzijom *fbx* koju je moguće učitati u pogon Unreal. Otvorit ćemo prethodno spremljeni montirani model, označiti model na sceni i odabrati naredbu *File* $\rightarrow$ *Export* $\rightarrow$ *FBX (.fbx)*. Naredba će otvoriti novi prozor koji se koristi za konfiguraciju spremanja datoteke. U kartici *Main* potrebno je odabrati objekte koji se pohranjuju, u ovom slučaju opcijama *Armature* i *Mesh* koje predstavljaju armaturu i tijelo čovjeka. U kartici *Geometries* potrebno je odznačiti opciju *Apply Modifiers*, a u kartiic *Armatures* označiti samo opciju *Only Deform Bones* koja će pohraniti samo deformirajuće kosti. Dodatno je potrebno u kartici *Animation* odznačiti sve opcije. Ta kartica koristit će se pri pohrani animacija. Pohrana je konfigurirana, stoga je potrebno odabrati ime datoteke i spremiti u odabrani direktorij.

Animacije ćemo spremiti svaku zasebno. Potrebno je konfigurirati pohranu na isti način kao i za spremanje modela. Dodatno se mora u kartici *Animation* označiti opcije *Baked Animation*, *Key All Bones* te *Force Start/End Keying*[3].

Idući korak u radu je učitavanje modela i animacija u Unreal, i stvaranje okruženja u kojem će model dobiti ulogu karaktera kojim je moguće upravljati putem tipkovnice.

4. Miješanje animiranih pokreta modela

4.1. Grafički pogon Unreal

Unreal Engine je grafički pogon za izradu trodimenzionalnih igara razvijen od kompanije *Epic Games* prije više od 20 godina. Prvobitno stvoren za izradu pucačkih igara u prvom licu, danas se koristi za izradu skoro svakog igračkog žanra. Najnovija verzija pogona, Unreal Engine 4, dopušta korisniku programiranje jezikom *C++* ili povezivanjem shema koje se popularno nazivaju *Blueprint*-ovi. Povezivanje shema jedna je od prednosti ovog pogona u odnosu na druge, jer omogućuje neiskusnim programerima da lakše implementiraju logiku igre. Pogon je potpuno besplatan za korištenje bez zarade. Neke od najpopularnijih igara proizvedenih u Unreal-u su PUBG, Fornite, mobilna verzija PES 2019, Forged Fantasy itd [6].

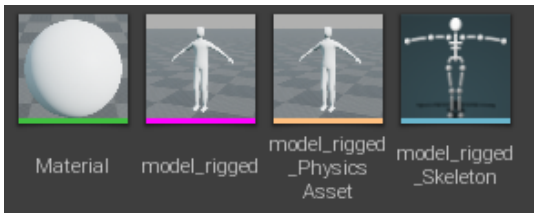
4.2. Učitavanje modela i animacija u Unreal

Pri pokretanju pogona Unreal, otvara se preglednik projekata (engl. *Project browser*) koji nam nudi odabir već postojećeg projekta ili stvaranje novog. Kod stvaranja novog projekta, mogu se odabrati različiti predlošci kao što su, na primjer, igra iz prvog lica, igra s letjelicom, igra s vozilom te igra iz trećeg lica. Za potrebe rada odabrat ćemo prazan projekt.

4.2.1. Učitavanje modela

U pogonu razlikujemo nekoliko vrsta figura (engl. *asset*), od kojih ćemo koristiti kostur, model tijela i animacije. Više modela koji se oslanjaju na isti kostur mogu dijeliti animacije. Pri učitavanju modela kostur se automatski generira. Model tijela učitat ćemo tako da povučemo spremljenu FBX datoteku iz direktorija u kojem se nalazi u

sučelje pogona. Nakon što otpustimo povučeno datoteku, otvara se ekran za konfiguraciju učitavanja. Ekran nam dopušta da mjenjamo postavke o samoj mreži modela, unesemo animacije, transformiramo objekt ili mu dodamo materijal. Zasada ćemo samo unijeti model bez dodatnih specifikacije, pri čemu će Unreal razdvojiti tijelo modela od njegovog kostura. Figure stvorene unosom datoteke moguće je vidjeti u trenutnom sadržaju projekta koji je prikazan na slici 4.1.



Slika 4.1: Učitavanje modela u Unreal

4.2.2. Učitavanje animacija

Za učitani model stvorene su četiri figure: osnovni materijal bijele boje koji se stvara automatski ukoliko ne unesemo svoj materijal s modelom, tijelo modela, fizikalna figura koja služi za definiranje fizike modela i kolizije te kostur modela.

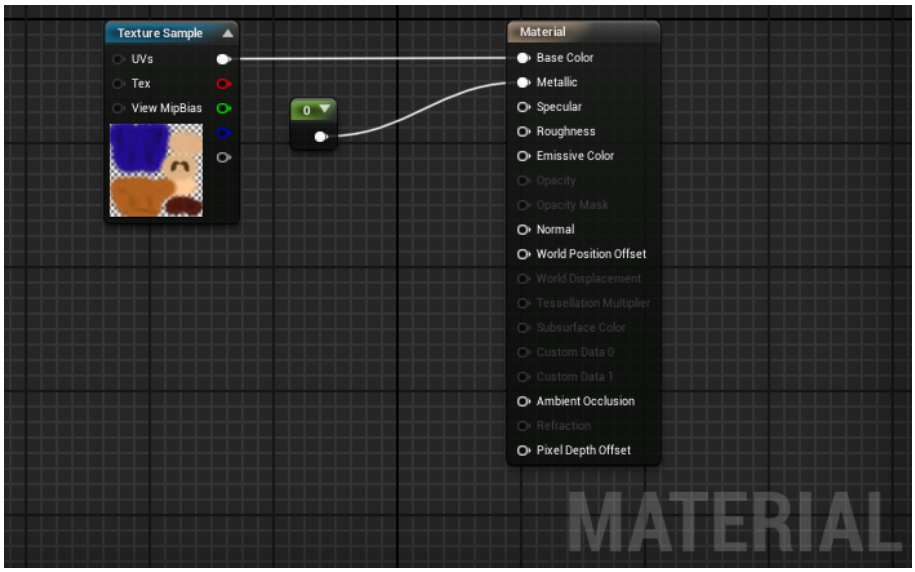
Sada je na redu učitavanje animacija. Njih ćemo učitati istom tehnikom povlačenja datoteka u sučelje Unreal-a. Konfiguracija unosa zahtijeva od nas da odaberemo kostur na koji će se animacije aplicirati. Potrebno je odabrati kostur koji je prethodno generiran. Izgled učitanih animacija moguće je vidjeti na slici 4.2.

Slika 4.2: Učitavanje animacija u Unreal

4.2.3. Podešavanje tekstura

Nakon učitavanja animacija, red je za podešavanje materijala. Sliku teksture unijet ćemo desnim klikom na preglednik sadržaja, odabirom opcije *Import Asset* te odabirom spremljene datoteke teksture.

Dvostrukim klikom na generiranu figuru materijala otvorit će se uređivač materijala. Unutar uređivača možemo vidjeti da se shema materijala sastoji od jednog čvora imena *Material*. Taj čvor ima mnogo svojstava, koja u kombinaciji određuju konačan izgled materijala. Desnim klikom na shemu moguće je dodati nove čvorove. Mi ćemo dodati čvor *Texture Sample* koji će nam omogućiti da stvorimo teksturu iz slike. Nakon što odaberemo sliku teksture, izlaz čvora spojit ćemo s osnovnom bojom materijala (engl. *Base Color*). Sada smo dobili figuru iz materijala, međutim potrebno je još postaviti vrijednost metalik boje na nulu kako se materijal ne bi sjajio dok je izložen nekakvom osvjetljenju. Izgled sheme materijala moguće je vidjeti na slici 4.3.



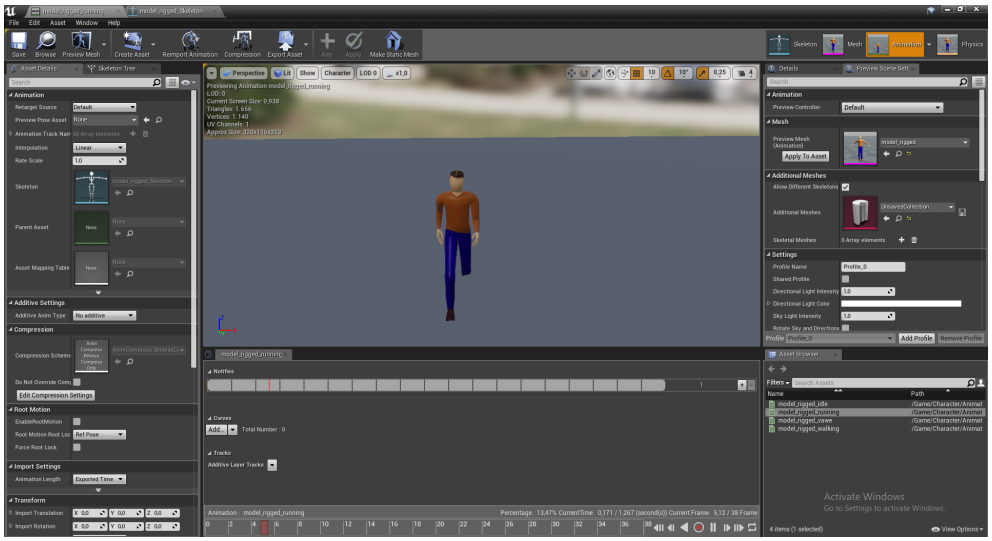
Slika 4.3: Podešavanje materijala

4.2.4. Uređivač Persona

Svaku figuru moguće je detaljnije prikazati u Unreal-ovom uređivaču *Persona* dvostrukim klikom na nju. Persona se sastoji od četiri uređivača: uređivač kostura, uređivač modela, uređivač animacija te uređivač fizike. Svaki od njih nudi različito sučelje za uređivanje i manipulaciju figura.

Uređivač skeletona prikazuje hijerarhiju kostiju. Moguće je, pored transformacije kostiju, kostima dodati takozvane utičnice (engl. *socket*) u koje se može postaviti neki drugi objekt. Primjer je utičnica na dlanu za koju je vezan model lopte. Uređivač modela koristi se za transformaciju modela, odabir materijala, promijenu odjeće i podešavanje ostalih detalja. Uređivač animacija dopušta nam odabir specifične animacije i njen prikaz na odabranom modelu. Prikaz animacije trčanja u uređivaču prikazan je

na slici 4.4.



Slika 4.4: Uređivač Persona

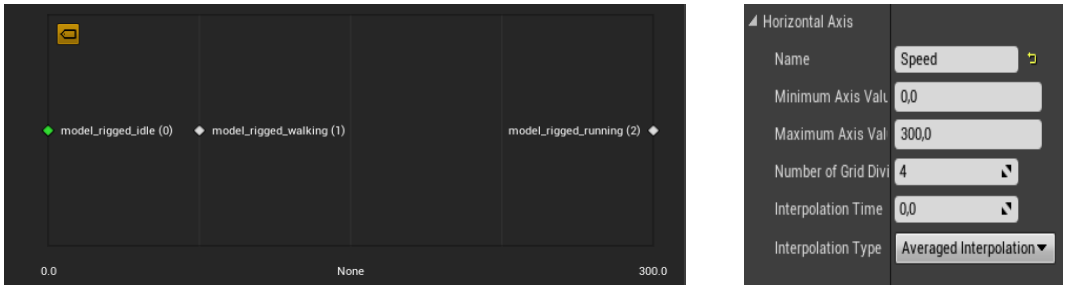
4.3. Miješanje animiranih pokreta modela

Miješanje animiranih pokreta unutar Unreal-a predstavlja kombinaciju dviju animacija s obzirom na vrijednost neke varijable. U konkretnom slučaju htijet ćemo miješati tri različite animacije: besposlenost, hodanje i trčanje. Kad je brzina kretanja jednaka nuli $v = 0$, djelovat će jedino animacija besplosenosti. Kako se brzina približava brzini $v = v_{hod}$, animacije besposlenosti i hodanja će se stapati, a kad je dosegne, djelovat će samo animacija hodanja. Daljnjim povećavanjem brzine stapat će se hodanje i trčanje sve dok brzina ne dosegne $v = v_{trcanje}$ nakon čega će djelovati jedino animacija trčanja. Ovakvo miješanje animacija naziva se jednodimenzionalno. Pored jednodimenzionalnog, moguće je miješati animacije u dvije dimenzija. Takvo miješanje ovisi o dvije varijable. Primjer su brzina trčanja i nagib terena za navedene animacije.

Stvaranje miješanja animacija započet ćemo stvaranjem nove figure. Desnim klikom miša na preglednik sadržaja otvara se izbornik u kojem je potrebno odabrati *Animation* → *Blend Space 1D*.

Uređivač miješanja animacija sastoji se od horizontalne linije koja je određena minimalnom i maksimalnom vrijedošću te određenim brojem podjela. Za početak, odredit ćemo da se raspon brzina kreće od 0 do 300 te podijeliti skalu na 5 različitih vrijednosti. Na prvu vrijednost $v_0 = 0$ postavit ćemo animaciju besposlenosti, na drugu vrijednost $v_1 = 75$ animaciju hoda, a na posljednju vrijednost $v_2 = 300$ animaciju

trčanja. Dodatno smo mogli promijeniti vrstu interpolacije. Na slici 4.5a prikazan je izgled linije miješanja nakon dodavanja animacija, a na slici 4.5b njene postavke.



(a) Linija miješanja animacija **(b)** Postavke linije miješanja

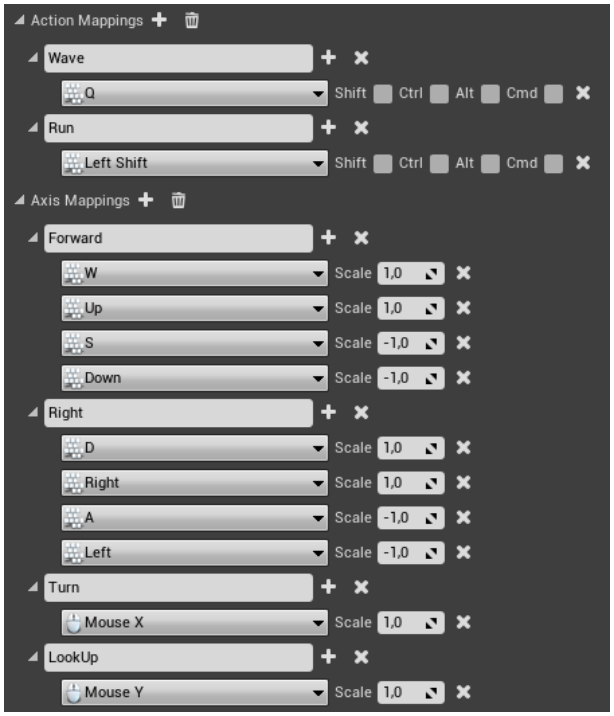
Slika 4.5: Miješanje animacija

4.4. Konfiguriranje pogona Unreal za testiranje animacija

Kako bi testirali animacije unutar pogona, potrebno je određenim tipkama na tipkovnici pridjeliti specifične akcije[8]. Spajanje ulaza s akcijama u Unreal-u određuje se u prozoru do kojeg se dolazi putanjom *Edit* → *Project Settings* → *Engine* → *Input*. Prozor nam omogućuje mapiranje osi i akcija. Definirat ćemo os kretanja unaprijed i omogućiti pokret u pozitivnom smjeru strelicom gore i tipkom *W* i pokret u negativnom smjeru strelicom dole i tipkom *S*. Drugi os je kretanje u desno, a ostvaruje se strelicom desno i tipkom *D* za pozitivan ili strelicom lijevo i tipkom *A* za negativan smjer. Postavit ćemo još i osi *X* i *Y* miša tako da kamera slijedi njegove pokrete. Pored osi, mapirat ćemo dvije akcije: mahanje i trčanje. Akciju mahanja pridružit ćemo pritisku na tipku *Q*, a akciju trčanja uključiti držanjem tipke *Shift*. Konfiguraciju ulaza moguće je vidjeti na slici 4.6.

4.5. Stvaranje sheme animacija

Idući korak predstavlja stvaranje sheme animacija, koja će kontrolirati koja će se animacija prikazivati nad modelom u nekom trenutku. U shemi animacija postoje dvije komponente koje zajedno kreiraju animaciju u određenom trenutku. Graf događaja (engl. *Event Graph*) ažurira vrijednosti nekih varijabli u slučaju nekog događaja, na primjer pritiska tipke na tipkovnici. Graf animacija (engl. *Anim Graph*) koristi te va-



Slika 4.6: Povezivanje tipkovnice i akcija

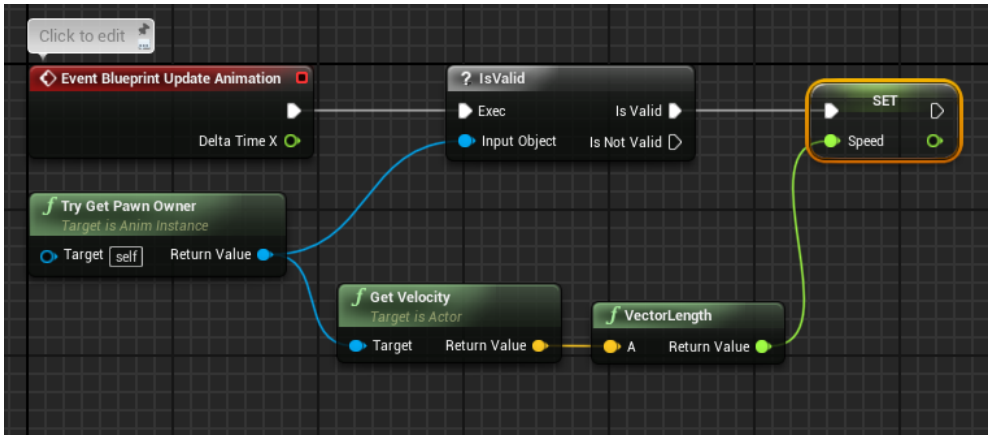
rijable i ovisno o njima prelazi čvorovima automata stanja kojeg čine animacije, miješane animacije ili druga stanja. Shemu kreiramo pritiskom desne tipke miša i odabira opcije *Animation* → *Animation Blueprint*.

4.5.1. Graf događaja

Zadani graf događaja sadrži dva čvora: prvi čvor aktivira se svakim taktom i zadužen je za osvježavanje varijabli. Drugi čvor pokušava dohvatiti objekt pijuna. Na slici 4.7 ta dva čvora nalaze se na lijevoj strani. Izvršavanje grafa događaja možemo pratiti slijedeći bijele linije između čvorova. Dakle, svaki takt aktivirat će prijelaz u čvor *isValid* koji će propustiti daljnje izvršavanje jedino u slučaju da je objekt pijuna na sceni pronađen. U slučaju da je to istina, aktivirat će čvor *setSpeed* koji varijablu *speed* postavlja na vrijednost dobivenu iz brzine kretanja pijuna. Čvor *VectorLength* koristi se za pretvorbu vektora u decimalni broj (točnije, računa se duljina vektora).

4.5.2. Graf animacija

Graf animacija podesit ćemo tako da rezultatna poza trenutnog prikaza bude jednaka trenutnom stanju automata stanja (slka 4.8a). Automat stanja sagradit ćemo tako da se iz početnog stanja odmah prijeđe u stanje u kojem je aktivna miješana animacija



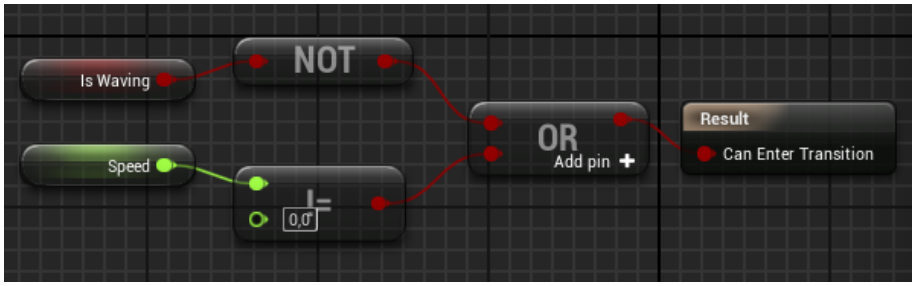
Slika 4.7: Graf događaja

besposlenosti, hoda i trčanja (slika 4.8b). To stanje interno će ovisiti o varijabli *speed* koja predstavlja brzinu kretanja. Iduće stanje predstavljat će animaciju mahanja. Prijelaz iz prvog u drugo stanje odvijat će se u slučaju da je brzina jednaka nuli i da je varijabla *isWaving*, koji će ovisiti o pritisku tipke *Q*, istinita. Obrnuti prijelaz odvijat će se ili ako se brzina poveća ili ako korisnik otpusti tipku za mahanje. Shema ovog prijelaza prikazana je na slici 4.9.

(a) Graf animacija

(b) Automat stanja animacija

Slika 4.8: Podešavanje grafa animacija



Slika 4.9: Shema prijelaza iz stanja mahanja u miješano stanje besposlenosti, hoda i trčanja

4.6. Stvaranje sheme karaktera

Razred *Blueprint* je figura koja se koristi za dodavanje funkcionalnosti na već postojeće razrede igre. Kreiranje novog razreda zahtijeva odabir jednog od postojećeg razreda čija će se svojstva naslijediti i funkcionalnost nadograditi. Osnovni razredi koje je moguće nasljediti i njihovu funkcionalnost moguće je pogledati u tablici 4.1.

Tablica 4.1: Osnovni razredi Blueprint-a

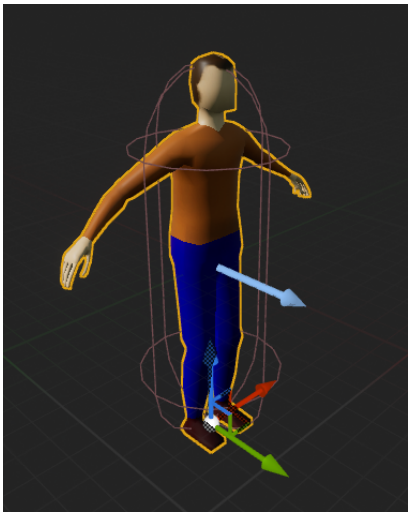
Ime razreda	Opis razreda
Actor	primjerak razreda može se postaviti ili stvoriti unutar svijeta
Pawn	primjerak razreda koji ima funkcionalnost Actora, dodatno se može posjedovati i kontrolirati
Character	primjerak razreda koji ima funkcionalnost Pawna, ima mogućnost hodanja, trčanja i skakanja
PlayerController	primjerak razreda koji ima funkcionalnost Actora, zadužen je za kontroliranje Pawna kojim igrač upravlja
Game Mode	definira pravila igre, bodovanje i ostale detalje

Za potrebe završnog rada, kreirat ćemo shemu koja nasljeđuje razred *Character*, kako bi obradili ulazne signale i pomoću njih kontrolirali pokrete karaktera. Blueprint uređivač sastoji se od tri kartice: prikaz (engl. *viewport*), skripta za konstrukciju te graf događaja.

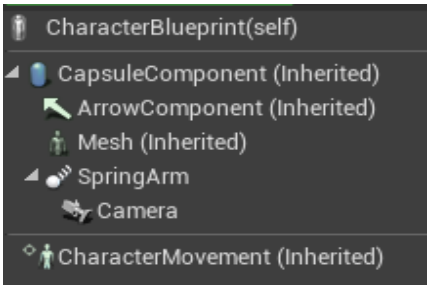
4.6.1. Prikaz sheme karaktera

Prikaz sheme prikazuje sve komponente koje sačinjavaju Blueprint. Komponente koje su već u prikazu naslijeđene su od roditeljskog razreda, a to su kapsula, strelica, mreža modela i kretnje karaktera. Kapsula predstavlja objekt zadužen za sudaranje s drugim objektima u sceni. Uz to, sprječava da objekti prolaze jedan kroz drugi. Strelice je komponenta koja pokazuje u kojem je smjeru karakter okrenut. Mreža modela je komponenta kojoj pridružujemo napravljeni model. Komponenta kretnji karaktera dostupna je samo za razrede naslijeđene iz razreda *Character*. Ona služi za konfiguraciju kretnji čovjeka i njihovih parametara (primjerice najveća moguća brzina plivanja, stupanj kontrole karaktera u zraku ili stupanj trenja pri hodanju). U komponenti kretnji ograničit ćemo maksimalnu brzinu hodanja na 300 (najveća brzina trčanja) te uključiti opciju *Orient Rotation to Movement* koja će dopustiti da rotacija putem miša kontrolira smjer kretnji karaktera.

Pri dodjeljivanju modela komponenti, potrebno je pripaziti da se nalazi unutar kapsule i da je okrenut u smjeru strelice (slika 4.10a).



(a) Prikaz sheme Blueprinta karaktera



(b) Komponente sheme karaktera

Slika 4.10: Podešavanje sheme karaktera

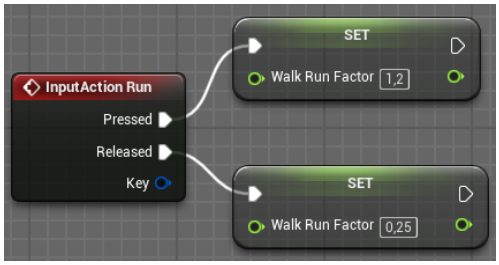
Komponenti mreže modela potrebno je u svojstvu *Animation* postaviti prethodno napravljenu shemu animacija. Osim postojećih komponenti, potrebno je shemi dodati kameru koja će služiti za iscrtavanje scene kada pokrenemo igru. Prvo ćemo dodati komponentu *Spring Arm* kojoj ćemo kao dijete dodati komponentu *Camera*. Time ćemo dobiti mogućnost da se kamera udaljava i približava karakteru ovisno o položaju drugih objekata na toj razini (engl. *level*). Pregled svih komponenti dostupan je na slici 4.10b.

4.6.2. Graf događaja

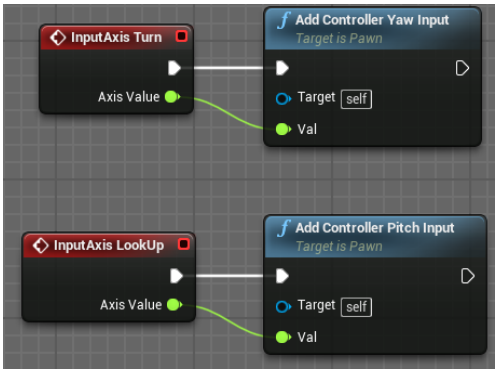
Graf događaja već smo implementirali za shemu animacija. Za karakter ćemo graf događaja implementirati tako da obrađujemo ulazne akcije koje smo postavili u poglavlju 4.4. Akcije koje je potrebno obraditi su kretanje po osima, rotacija te akcije mahanja i trčanja.

Trčanje ćemo obraditi na način da osluškujemo ulazni signal kojeg igrač šalje pritiskom na tipku *Shift*. Definirat ćemo varijabu *WalkRunFactor* koja će određivati brzinu kretanja igrača. U slučaju kad tipka nije pritisnuta, vrijednost varijable bit će 0.25, a u slučaju da je tipka pritisnuta, postavit ćemo vrijednost varijable na 1.2. Vrijednosti su dobivene empirijski. Izgled blueprinta vidljiv je na slici 4.11a.

Mahanje ćemo obraditi na sličan način. Definirat ćemo varijablu *isWaving* čija će



(a) Stvaranje blueprinta za trčanje



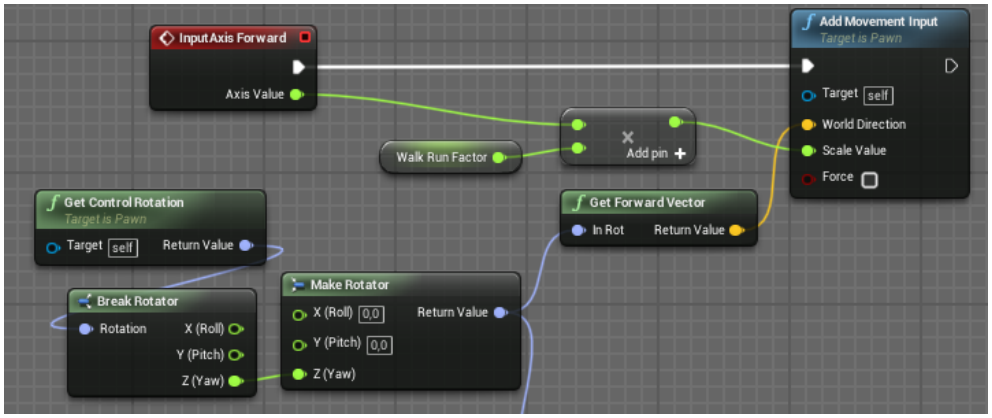
(b) Stvaranje blueprinta za rotaciju

Slika 4.11: Stvaranje blueprintova

vrijednost u slučaju da je tipka mapirana za mahanje pritisnuta biti istinita, a u suprotnom lažna. Ovu varijablu koristit ćemo u ažuriranoj verziji grafa događaja animacije.

Obrada kretanja čovječuljka nešto je složenija od obrada prethodnih animacija. Obradu započinjemo definiranjem čvora *InputAxis* koji će nam vraćati trenutnu vrijednost pomaka po zadanoj osi pri svakoj promjeni. Taj čvor spojit ćemo linijom izvršavanja (tako da svaka promjena prvog čvora uzrokuje pozivanje drugog čvora) s čvorom *AddMovementInput* koji nekom ciljnom objektu, ovisno o definiranom smjeru kretanja, zadaje vrijednost koja određuje količinu pomaka u tom smjeru. Za ciljni objekt postavit ćemo samog karaktera. Vrijednost pomaka predstavljat će vrijednost koju nam daje već spomenuti početni čvor, ali skaliranu za vrijednost *WalkRunFactor* koju smo definirali kod obrade trčanja. Na taj način veličina pomaka ovisit će o tome je li igrač pritisnuo tipku ili ne. Smjer kretanja ovisit će o trenutnoj rotaciji karaktera. Stoga je potrebno definirati novi čvor *GetControlRotation* čiji je ulaz objekt čija se rotacija traži (u ovom slučaju to je ponovno naš karakter), a izlaz varijabla koja u sebi čuva rotaciju. Potrebno je varijablu rotacije filtrirati tako da se očuva jedino vrijednost rotacije oko Z osi. Nakon toga, potrebno je objekt rotacije kojeg trenutno imamo pretvoriti u vektor koji je moguće spojiti kao ulaz u varijablu koja određuje smjer kretanja. Čvor koji transformira rotaciju u vektor smjera naziva se *GetForwardVector*. Definirali smo postupak izrade blueprinta kretanja. Potrebno je taj postupak implementirati za kretanje unaprijed i kretanje udesno. Svi ostali smjerovi definirani su u prethodna dva. Uz tekstualni opis, na slici 4.12 moguće je vidjeti izgled dijela blueprinta kretanja koji definira kretanje unaprijed.

Zadnje što je preostalo je obrada kretnji miša. Za pomicanje miša definirali smo dvije osi. Prva os definira pomicanje miša ulijevo i udesno (pomicanje po X osi koordinatnog sustava miša na podlozi), a druga os pomicanje miša unaprijed i unazad



Slika 4.12: Stvaranje blueprinta za kretanje

(pomicanje po Y osi). Želimo da pomicanje po X osi definira smjer prema kojem je okrenut karakter. Stoga ćemo čvor koji nam dojavljuje pomake miša po X osi spojiti s čvorom *AddControllerYawInput* koji će ovisno o vrijednosti pomaka miša zarotirati karaktera oko Z osi (drugi naziv te rotacije je Yaw). Za pomake po Y osi miša želimo da se pogled pomiče gore ili dolje, stoga je potrebno čvor koji nam dojavljuje pomake miša po Y osi spojiti s čvorom *AddControllerPitchInput* koji će karakter zarotirati oko Y osi miša (ta rotacija naziva se Pitch). Slika 4.11b prikazuje stvoreni blueprint.

4.7. Stvaranje igre i testiranje animacija

Za testiranje animacija i kontrole nad karakterom putem tipkovnice i miša, koju smo prethodno pokušali ostvariti, potrebno je stvoriti novi objekt razreda koji predstavlja način igre (engl. *Game Mode Base*). Stvaranjem objekta i dvostrukim klikom na njega, ulazimo u njegov uređivač, koji je vrlo sličan uređivaču za blueprint karaktera. Unutar uređivača, potreno je odabrati opciju *Class Defaults* koja će prikazati listu raznih opcija načina igre koje je moguće mijenjati. Za potrebe rada, dovoljno je postaviti vrijednost opcije *Default Pawn Class* na prethodno napravljeni blueprint karaktera.

Nakon stvaranja načina igre, potrebno je u postavkama trenutne razine (engl. *level*) postaviti stvoreni način rada kao odabrani. Potrebno je i u postavkama projekta odabrati da se pokretanjem igre pokreće i ta razina.

Pristiskom na tipku *Play* pokreće se igra. Karaktera kojeg smo stvorili vidimo iz perspektive kamere koju smo u dijelu ovog projekta podešavali, a njega možemo pomicati pritiskom na prethodno definirane tipke tipkovnice. U slučaju da stojimo u mjestu prikazuje se animacija besposlenosti i moguće je pokrenuti animaciju mahanja pritiskom na tipu *Q*. U slučaju da se krećemo i držimo tipku *Shift*, hodanje će se pretvoriti

u trčanje. Pored tipkovnice, mišem se možemo rotirati oko vlastite osi, ili pogledati u smjeru iznad ili ispod.

5. Zaključak

Završni rad opisuje kompletan postupak izrade jednostavnog modela čovjeka i njegovo animiranje. Animacije se na model apliciraju unutar jednostavne simulacije napravljene unutar grafičkog pogona Unreal Engine.

Stvaranje čovječuljka pokazalo se kao kompleksan postupak koji se sastoji od mnogo koraka. Cijeli postupak uključuje vještine modeliranja, teksturiranja, izrade animacija metodom imitacije realnih ljudskih pokreta te povezivanja stvorenog s grafičkim pogonom. Potrebno je uložiti puno vremena u detalje koji povećavaju realnost izgleda modela i animacija.

Modeliranje unutar rada koncentriralo se na oblik ljudskog tijela, a zbog složenosti nije modeliran i oblik ljudskog lica. Kod teksturiranja korištene su osnovne boje. U složenijim projektima teksturiranje može uključivati teksture koje se sastoje od uzoraka. Kod animiranja deformacije prilikom pomicanja dijelova modela nisu idealne. Miješanje animacija unutar Unreala temeljeno na interpolaciji pokazalo se kao vrlo korisno svojstvo pogona jer izglađuje prijelaz između dviju animacija.

LITERATURA

[1] Alex Young. How to create a character in blender for ue4 and unity. https://www.youtube.com/watch?v=LhWYwVPd2Oo&list=PLSfh6YsA0LhuevD_gen5_nBnZDMaoP2Tr.

[2] All3DP. Best 3d design/3d modeling software 2019. https://all3dp.com/1/best-free-3d-modeling-software-3d-cad-3d-design-software/.

[3] Ariana Alexander. Animation workflows between blender and unreal engine 4. https://www.youtube.com/watch?v=waC5tjCH2GI.

[4] Blender.org. Blender editors. https://docs.blender.org/manual/en/latest/editors/index.html,.

[5] Blender.org. Weight paint. https://docs.blender.org/manual/en/latest/sculpt_paint/painting/weight_paint/introduction.html,.

[6] Epic games. Unreal engine. https://www.unrealengine.com/en-US/.

[7] Thomas Denham. What is uv mapping and unwrapping? https://conceptartempire.com/uv-mapping-unwrapping/.

[8] Unreal engine. Unreal engine 4 3rd person tutorial. https://www.youtube.com/watch?v=hRO82u1phyw&list=PLfQ3pODBwOcaV1TdnqNWLTJ4wiUzEvXis.

[9] Wikipedia contributors. 3d modeling. https://en.wikipedia.org/wiki/3D_modeling,.

[10] Wikipedia contributors. Uv mapping. https://en.wikipedia.org/wiki/UV_mapping,.

Miješanje animacija modela čovjeka

Sažetak

Prvi dio završnog rada prati izradu modela čovjeka, teksturiranje modela, spajanje modela s armaturom te stvaranje osnovnih animacija unutar programa Blender. Drugi dio opisuje učitavanje modela i animacija u grafički pogon Unreal, stvaranje logike koja povezuje unos korisnika i animacije te miješanje animacija hoda i trčanja.

Ključne riječi: modeliranje, tekstura, armatura, animacija, Blender, Unreal

Blending Animations of the Human Model

Abstract

First part of the final work describes process of manikin modelling, texturing and rigging model. It also covers creation of basic animations for model. Second part concentrates on importing created assets into Unreal game engine and implementation of interaction between user inputs and animations. Blend space feature in game engine is also demonstrated on walk and run animations.

Keywords: modelling, texture, rigging, animation, Blender, Unreal