FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Sedm statečných - optimalizace animací
Student:	Petr Hromják
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Sedm statečných je studentská hra vzniklá v předmětu Virtuální herní světy. Cílem bakalářské práce je finalizace a optimalizace hry.

1) Shrňte stav hry na konci předmětu VHS.
2) Analyzujte stav hry, soustřeďte se na animace a vizuální chyby, proveďte uživatelské testování. Shrňte možnosti finalizace a rozšíření hry, soustřeďte se na možnosti optimalizace.
3) Navrhněte finální podobu hry. Soustřeďte se na grafické optimalizace.
4) Upravte systém animací ve hře.
5) Hru vhodným způsobem otestujte.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 11. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

7S-Anima: Úprava animací postav ve hře Western Town

Petr Hromják

Katedra softwarového inženýrství
Vedoucí práce: Ing. Radek Richtr, Ph.D.

27. června 2021

Poděkování

Rád bych poděkoval mému vedoucímu práce Ing. Radku Richtrovi, Ph.D. za skvělé vedení při vypracovávání bakalářské práce. Dále bych chtěl poděkovat Anně Doležalové za podporu při psaní práce. V neposlední řadě bych chtěl poděkovat rodině za podporu při studiu. A nakonec přátelům, kteří se podíleli na testování vyvíjeného systému.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. června 2021 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Hromják, Petr. *7S-Anima: Úprava animací postav ve hře Western Town*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato bakalářská práce se zabývá animacemi postav. V práci jsou předvedeny základní principy a postupy tvorby 3D modelu postav a jejich následné animace. Z animačních metod je blíže rozvedena kosterní animace, s čímž souvisí i tvorba kostry samotné. Pomocí kosterní animace jsou také předvedeny postupy přímé a inverzní kinematiky.

Klíčová slova 3D počítačová grafika, animace postav, klíčové snímky, kosterní animace, inverzní kinematika, přímá kinematika, procedurální animace

Abstract

This bachelor thesis is focused on character animations. The work presents the basic principles and procedures of creating 3D models of characters and their subsequent animations. Of the animation methods, skeletal animation, which is closely related to the creation of the skeleton itself, is elaborated in more detail. Forward and inverse kinematics procedures are also demonstrated using skeletal animation.

Keywords 3D computer graphics, character animation, keyframes, skeletal animation, inverse kinematics, forward kinematics, procedural animation

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Počítačová animace je jedno z velmi rychle se rozvíjejících odvětví počítačové grafiky. Hlavním hnacím motorem pro toto odvětví jsou hry a filmy. Tvůrci se snaží o stále dokonalejší a realističtější animace, což jim dovoluje zejména pokrok v oblasti hardwaru, ale i algoritmů. Zejména v počítačových hrách je tento pokrok vidět, jelikož jsou zde kladeny požadavky nejen na výsledný vzhled, ale především na rychlost výpočtu a co nejmenší zatížení procesoru. Výkonnější hardware nám dovoluje věrohodnější výsledky, ale je nutné algoritmy také optimalizovat, jelikož výpočetní výkon bude vždy omezený. Ušetřený výkon lze pak použít v jiné oblasti (například složitější systém AI) a zlepšit tak celkový dojem ze hry.

Animace není jen umělecká tvorba spojená s programováním, ale zahrnuje mnoho dalších oborů. Například pro vytvoření realistické animace srážky dvou aut je potřeba znát fyzikální vlastnosti objektů a fyzikální zákony, kterými se pohyb těchto objektů řídí. Na druhou stranu, když chceme animovat živočichy, tak je potřeba znát jejich anatomii – vlastnosti jednotlivých kloubů, svalů atd.

V rámci předmětu BI-VHS jsem měl možnost vést tým, ve kterém jsme pracovali na počítačové hře s názvem Western Town. Příběh hry je zasazen do westernového prostředí. Hlavním úkolem hráče je zachránit snoubenku a bratra, přičemž hra nabízí několik konců na základě rozhodnutí učiněných v průběhu hry.

Western Town je stále ve vývoji a v momentálním stavu jsou animace zjednodušené a nedokonalé. Proto se hodlám v rámci této práce zaměřit na problematiku animací lidských postav a všech věcí s tím spojených. Především tedy na úpravu současného modelu, způsoby animace postavy, výhody a nevýhody jednotlivých metod a jejich využití v počítačových hrách s ohledem na optimalizaci.

Kapitola 1

Cíl práce

Cílem této práce je seznámit čtenáře s problematikou počítačové animace ve 3D, vysvětlit základní pojmy a techniky tvorby lidských modelů a jejich animace. Zároveň se práce zaměřuje na porovnání jednotlivých metod animací a na vysvětlení, kdy kterou metodu použít (na základě jejích výhod a nevýhod).

Nejdříve bude provedena analýza problémových částí v projektu Western Town. Na základě této analýzy se vytvoří testy pro uživatelské testování, výsledky testování se využijí k určení nedostatků modelů a animací.

V části zaměřené na realizaci se vybrané metody využijí k úpravě modelů a animací v projektu Western Town. Výsledkem práce bude upravený model postavy, animace modelu postavy a implementace těchto modelů a animací do projektu Western Town, kde bude vytvořen systém pro částečně procedurální animaci.

Výsledná aplikace umožní s modelem postavy pohybovat a interagovat s již vytvořeným herním světem. Animace nebude mít jen hlavní postava, ale i veškeré vedlejší postavy, takže bude možné animace pozorovat i z pohledu třetí osoby. Částečně procedurální animace také zajistí realistickou reakci na okolní prostředí v reálném čase.

Část I

Teoretická část

Kapitola 2

Rešerše

V první části této kapitoly se zaměříme na způsoby reprezentace 3D modelů. Druhá část se zaměřuje na nejdůležitější typy animací a na jejich výhody a nevýhody. Nejsou zde uvedeny všechny typy animací, ale pouze ty, které jsou nejčastěji využívány v počítačové grafice. V další části bude podrobně rozebírána metoda kosterní animace. Také se zaměříme na potřebné znalosti pro vytvoření realisticky se chovající lidské kostry. Poslední část je věnována typům kinematiky.

2.1 Reprezentace 3D modelů

Počítačová 3D grafika je obor, který se zabývá zobrazováním trojrozměrných těles. Před zobrazením na obrazovku je nutné 3D scénu převést do 2D. Celému procesu se říká vykreslování.

Na rozdíl od 2D grafiky, kde jsou pouze dva rozměry (výška a šířka), ve 3D přibývá třetí rozměr a tím je hloubka. V počítačové grafice jsou tyto tři rozměry reprezentovány pomocí tří os: X, Y, Z. Přičemž to, která osa je výška, šířka, či hloubka, záleží už na způsobu reprezentace v konkrétním používaném programu.

Trojrozměrným tělesům ve 3D grafice se říká 3D modely. Podle [Žára, 2005] můžeme všechny modely rozdělit do dvou skupin, a to na objemově a hraničně reprezentované.

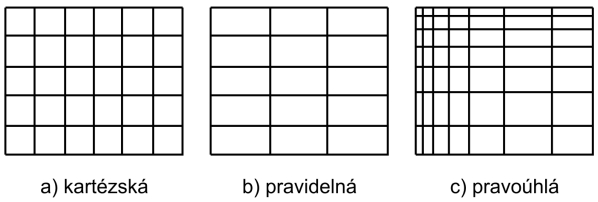
2.1.1 Objemová reprezentace těles

Objemová reprezentace těles je ze dvou zmíněných reprezentací blíže realitě, jelikož si tělesa zachovávají informace o vnitřních bodech na rozdíl od hraniční reprezentace. Jelikož jsou v této reprezentaci tělesa určena svým objemem, jednoznačně tak reprezentují prostor, který jim náleží.

Voxel

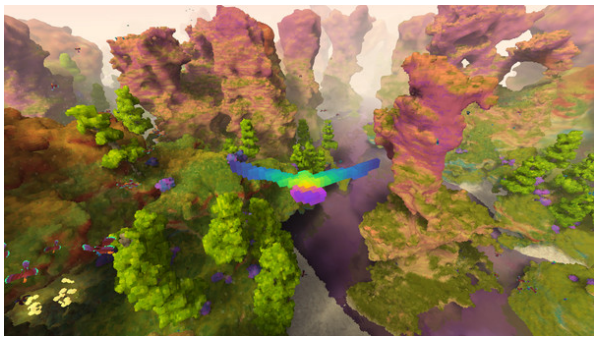
Jak se můžeme dočíst v [Phyllis, 2017], Voxel je trojrozměrným ekvivalentem dvourozměrného pixelu. Zároveň je nejmenší nedělitelnou jednotkou voxelového prostoru (objemu). V podstatě tak můžeme voxel přirovnat k atomu. Jako jsou reálné objekty tvořeny z jednotlivých atomů, tak objekty ve voxelovém prostoru (3D) jsou tvořeny z jednotlivých voxelů.

Jak se píše v [Žára, 2005], voxely mají tvar kvádru nebo krychle a jsou uspořádány do pravoúhlé mřížky. Jednotlivé buňky mřížky nemusí být stejně velké (obr. 2.1, pro vizualizaci jsou mřížky pouze ve 2D).



Obrázek 2.1: 2D pravoúhlé mřížky [Žára, 2005]

Voxely se využívají například ve zdravotnictví (CT, MRI), ale i v počítačových hrách. Příkladem takové hry je Minecraft. Nicméně voxelová grafika se využívá málo kdy a voxelová reprezentace je většinou převáděna při zobrazení na hraniční reprezentaci, stejně tomu tak je i u zmiňovaného Minecraftu. Příkladem čistě voxelové grafiky je například hra Fugl (obr. 2.2).



Obrázek 2.2: Počítačová hra Fugl [Team Fugl, 2017]

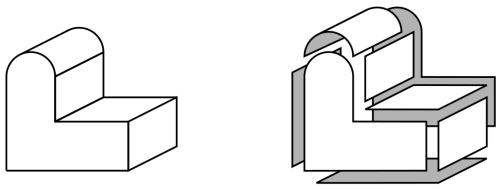
Výhodou této reprezentace je vysoká přesnost z fyzikálního hlediska. Například je velmi snadné detekovat kolize objektů, kolizi lze totiž detekovat pomocí kolize voxelů (tedy jestli se nacházejí na stejné pozici). Nevýhodou je na druhou stranu složitější konstrukce modelů a vyšší paměťová náročnost[1].

[1]Model je uložen v paměti pomocí jednotlivých voxelů. Musí se ukládat alespoň pozice a materiál všech voxelů, i těch které nejsou na povrchu objektu. Pokud mají voxely různou velikost, tak je nutné ukládat i jejich rozměry.

2.1.2 Hraniční reprezentace těles

V počítačové grafice se častěji než objemová využívá hraniční reprezentace. Těleso je reprezentováno pouze jeho povrchem. Lze si ho tak představit jako nekonečně tenkou skořápku.

Popis tělesa se tak zjednoduší [Žára, 2005] na popis hranice, tedy na popis množiny hraničních bodů (obr.2.3). Hranice tak rozděluje prostor na dvě části a to na vnějšek a vnitřek tělesa. Pokud chceme zjistit, zda je bod součástí tělesa, musíme zjistit na které straně hranice se nachází.

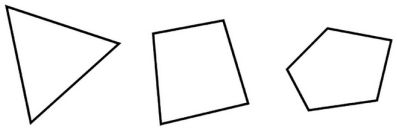


Obrázek 2.3: Popis tělesa je převeden na popis pláště [Žára, 2005]

Polygonová síť

Většina objektů v počítačové grafice je reprezentována polygonovou sítí. Polygonová síť je tvořena z jednotlivých polygonů (mnohoúhelníků). Základními stavebními prvky polygonů jsou podle [Paquette, 2013]:

- **Bod (vertex)** - Bod je bezrozměrný základní prvek 3D prostoru. Každý bod je jednoznačně určený svými souřadnicemi v 3D prostoru. Body jako takové nelze vykreslit ve finální aplikaci.
- **Hrana (edge)** - Hrana je spojnicí mezi dvěma body. Vždy spojuje přesně dva body. Stejně jako bod ani hranu nelze vykreslit ve finální aplikaci.
- **Plocha (face)** - Plocha je definována pomocí hran. Nejjednodušší plocha je ohraničena třemi hranami. Plocha je jediný vykreslitelný objekt. Je to způsobeno tím, že plochy (na rozdíl od ostatních objektů) zabírají část prostoru, která může odrážet světelné paprsky, což umožňuje, aby plochy byly viditelné pro kameru, a tudíž i vykreslitelné.



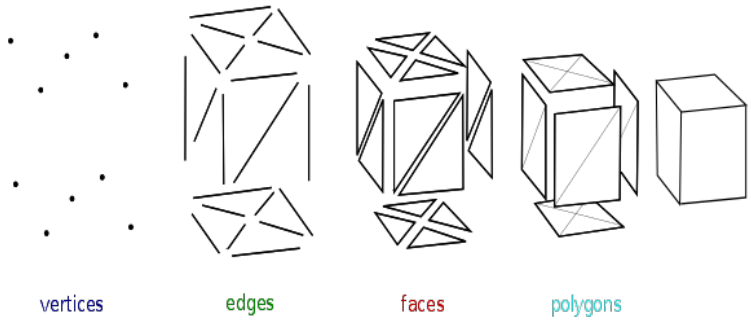
Obrázek 2.4: Tri, quad a n-gon [Chopine, 2011]

Polygony mohou být tvořeny libovolným počtem hran (nejméně 3). Polygonům se třemi hranami se říká tri (triangle), se čtyřmi quad (guadangle) a polygonům s více než čtyřmi se říká n-gony. (obr.2.4). [Chopine, 2011]

Pokud polygony sdílejí vrcholy a hrany, říkáme takové množině polygonová síť. Základním stavebním prvkem polygonové sítě je trojúhelník. Trojúhelníky se používají především díky svým vlastnostem. Žára ve své knize říká o trojúhelnících:

> „*Má mnoho dobrých vlastností ... je vždy konvexní a všechny jeho vrcholy leží v rovině. Pro jeho vyplňování existují velmi rychlé algoritmy. ... Mnoho geometrických výpočtů nad trojúhelníkem lze optimalizovat.*“ [Žára, 2005, str. 237]

V grafických procesorech (GPU) se výpočty nad trojúhelníky optimalizují a GPU se vyrábějí tak, aby zvládaly spoustu takovýchto výpočtů. Z tohoto důvodu se každý složitější polygon před vykreslováním převádí na trojúhelníky. Tomuto procesu se říká triangulace[2]. Příklad, jak je vytvořena polygonová síť jednoduchého kvádru, je vidět na obrázku 2.5[3].



Obrázek 2.5: Rozdělení objektu na jednotlivé stavební prvky polygonové sítě [Anonymous, 2021]

Pro jednodušší popis polygonových sítí je dobré si říci, jaké vlastnosti mohou mít. V [Power, 2012] se píše, že polygonové sítě mohou mít následující vlastnosti:

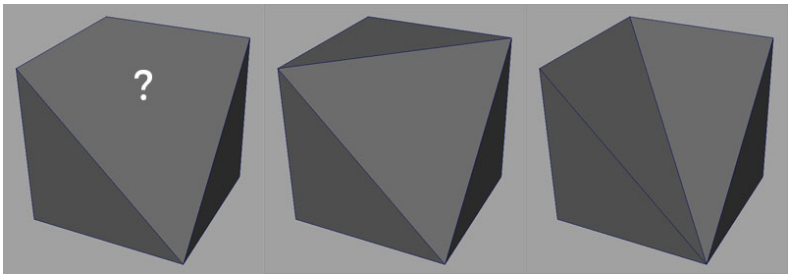
- **Pevnost** - Síť představuje pevný objekt, pokud jeho plochy obklopují kladné a konečné množství prostoru.
- **Rovinnost** - Síť je rovinná, pokud jsou všechny její polygony rovinné.

[2] O různých metodách triangulace čtyřúhelníků se můžete více dočíst například v knize *A Quadrilateral Rendering Primitive*.

[3] Je zde i vidět, jak lze převést čtyřúhelníky na trojúhelníky.

- **Propojenost** - Síť je propojená, pokud existuje nepřerušená cesta mezi dvěma libovolnými vrcholy. Pokud síť není propojená, obvykle se považuje za dva objekty.
- **Jednoduchost** - Síť je jednoduchá, pokud v ní nejsou žádné otvory.
- **Konvexnost** - Síť je konvexní, pokud čára spojující jakékoli dva body leží zcela uvnitř objektu.

Při triangulaci mohou v určitých situacích vznikat problémy s reprezentací polygonů. Stává se to především pokud polygonová síť není rovinná. Pokud modelář[4] neprovede triangulaci sám, tak je konverze provedena buď herním jádrem, nebo automaticky v grafické kartě. Neexistuje [Adobe, 2021b] jednotný postup jak triangulaci provádět. Výsledný model tak může vypadat jinak, než modelář zamýšlel (obr. 2.6). Obecně pokud je polygonová síť pevná, rovinná a propojená, tak nedochází k chybám při jejím vykreslování.



Obrázek 2.6: Výsledky triangulace nerovinného čtyřúhelníku [Adobe, 2021b]

Nevýhodou [Power, 2012] polygonové sítě je, že nedokáže vždy[5] přesně popsat povrch reálného objektu, ale snaží se ho pouze aproximovat. Další nevýhodou je složitější detekce kolizí. Na druhou stranu mezi výhody patří nižší paměťová náročnost a jednoduchá konstrukce modelů.

Topologie

Jak jsme jsme si již řekli, tak polygonová síť je tvořena množstvím vrcholů, hran a ploch, nicméně jsme si neřekli jak takovou síť vytvořit. Počtem vrcholů, jejich propojením hranami a tvorbou ploch se zabývá topologie povrchu.

Při tvorbě topologie povrchu neexistuje jediný správný postup, avšak existují určitá pravidla a přístupy, které usnadňují úpravy modelu, jeho animaci a další operace nad ním. Jelikož se zabýváme animacemi, tak způsob tvorby topologie bude zaměřený na toto téma. V případě jiného použití modelu mohou při tvorbě topologie být použita jiná pravidla a postupy.

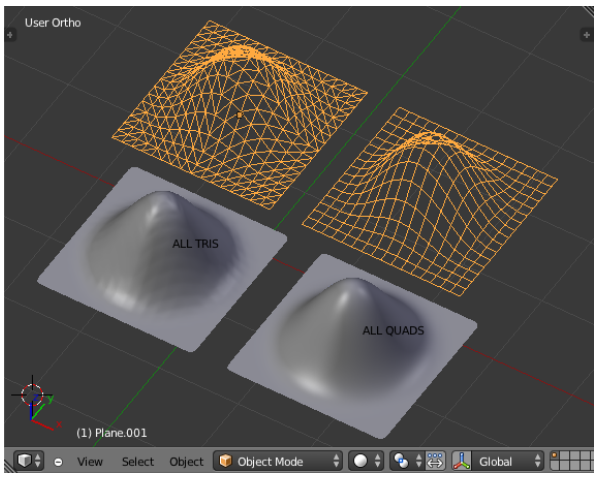
[4] Člověk, který model vytváří.

[5] Pokud objekt obsahuje pouze ostré hrany a nemá žádné křivky, tak je možné ho popsat polygonovou sítí přesně.

Čtyřúhelníky vs. trojúhelníky

První důležitou otázkou při tvorbě topologie je volba typu mnohoúhelníků, kterými bude topologie tvořena. Každý polygon s větším počtem vrcholů, než má trojúhelník, musí být před vykreslením triangulován, což může způsobit chyby při vykreslování, tak proč rovnou nepoužít trojúhelníky a vyhnout se tak tomuto problému?

Přesto se často v praxi místo trojúhelníkové sítě používá čtyřúhelníková síť. Podle autorů[6] z fóra [Haunt_House and ideasman42, 2013] je hlavním důvodem pro používání čtyřúhelníkové sítě možnost používat postupy a především nástroje, které nelze na trojúhelníkové sítě použít. Mezi takové operace patří využívání edge loops[7], jejich přidávání a odebírání, provádění dalších operací nad edge loops a face loops[8] a podle [Žára, 2005] algoritmy na dělení povrchů obvykle nad takovou sítí dosahují lepších výsledků (obr. 2.7)



Obrázek 2.7: Dělení povrchu tvořeného trojúhelníkovou sítí (vlevo), čtyřúhelníkovou sítí (vpravo) [Haunt_House and ideasman42, 2013]

Langerovy linie

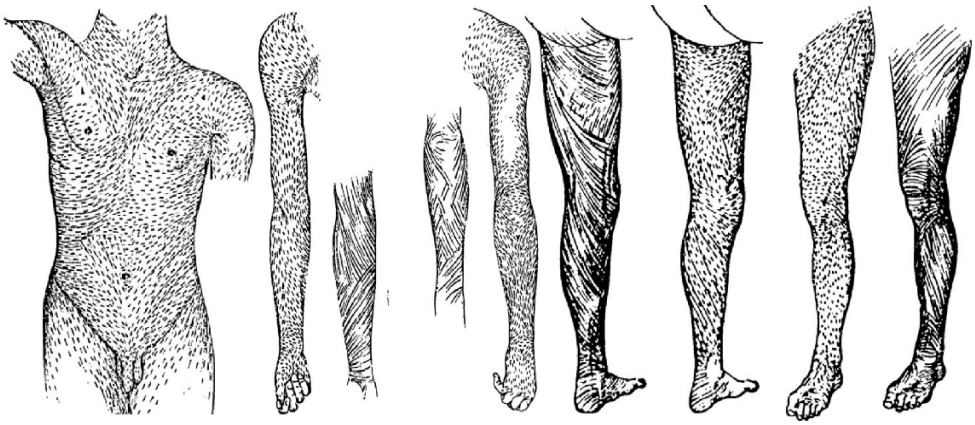
Další otázkou, kterou je potřeba vyřešit, je umístění hran v modelu. Jedním z možných řešení je využít Langerových linií. Tyto linie byly poprvé objeveny v roce 1861 rakouským anatomem Karlem Langerem (1819-1887). Jedná se o topologické linie nakreslené na mapě lidského těla. Podle [Agache, 2016] odpovídají přirozené orientaci kolagenových vláken v pokožce a jsou obecně

[6] Fórum `blender.stackexchange.com` je zaměřeno na program Blender, ale obecně se zabývá jakýmikoliv dotazy k 3D grafice. Oba citovaní autoři, Haunt_House i ideasman42, často odpovídají na různé dotazy v rámci fóra a jsou v komunitě uznávaní.

[7] Edge loop je souvislá linie hran.

[8] Face loop je smyčka tvořená plochami mezi dvěma rovnoběžnými edge loop.

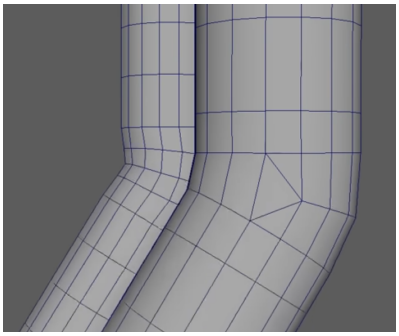
rovnoběžné s orientací podkladových svalových vláken. Langerovy linie nám tak dávají velmi konkrétní představu o tom jak by výsledná topologie měla vypadat, což můžeme vidět na obrázku 2.8.



Obrázek 2.8: Langerovy linie [Agache, 2016]

Topologie v místě ohybu

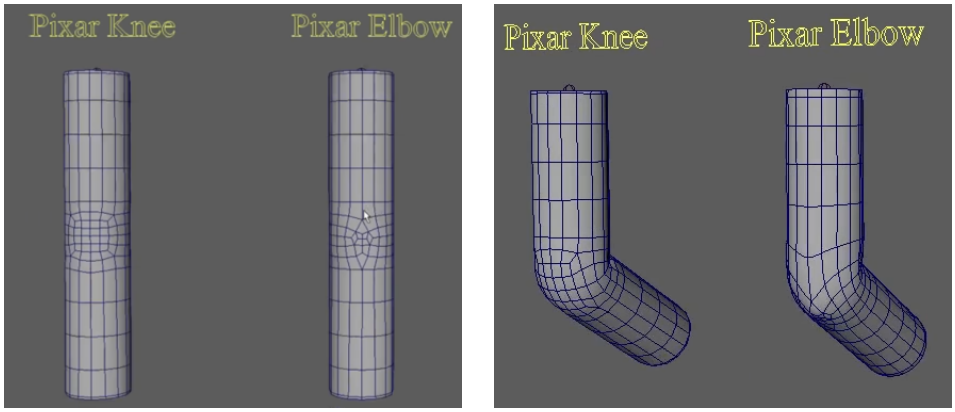
Langerovy linie vyřeší pouze hrany ve směru pnutí kůže, ale je také potřeba přidat hrany v místech ohybu. Dle [CG Futures, 2018] stačí přidat jeden edge loop do místa ohybu, avšak takový ohyb nezachovává objem, a tak se přidávají alespoň dva podpůrné edge loops. Jeden před ohybem a jeden za ohybem (obr. 2.9 vlevo). Běžnou praxí je spojování hran ve vnitřní straně ohybu, což vytváří plynulejší přechod v místě ohybu (obr. 2.9 vpravo).



Obrázek 2.9: Ohyb podpořený více edge loopy (vlevo), spojení hran ve vnitřní straně ohybu (vpravo) [Diehard, 2020]

[Diehard, 2020] doporučuje použít speciální topologii (obr. 2.10) přední strany ohybu. Pixar[9] topologie pro koleno a loket realisticky simuluje pevnou kost, která by se normálně nacházela pod kůží a svaly.

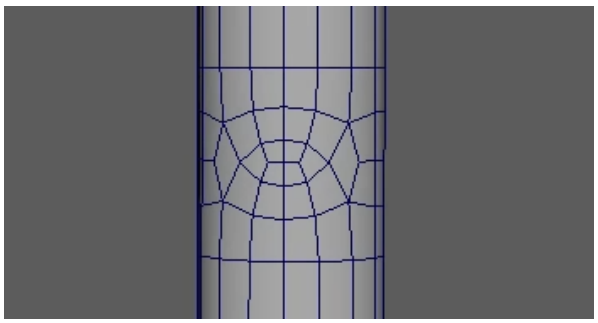
[9]Topologie se tak nazývá, jelikož jí vymyslelo a používá studio Pixar.



(a) narovnaný (b) v ohybu

Obrázek 2.10: Pixar topologie kolena (vlevo) a lokte (vpravo) [Diehard, 2020]

Navíc [Diehard, 2020] vytvořil vlastní dvou-smyčkovou topologii[10] (obr. 2.11). Z vlastností, které jeho dvou smyčková topologie má, vyplývá, že dovoluje velmi realisticky simulovat patelu (čéšku) nacházející se v koleni. Umožňuje totiž tibie (kosti holenní) táhnout vrchní vertexy simulující patelu dolů a naopak femuru (kosti stehenní) táhnout vertexy spodní, což dokáže simulovat pohyb pately bez změny tvaru a objemu.[11]



Obrázek 2.11: Dvou-smyčková topologie [Diehard, 2020]

2.2 Počítačová animace

Animace je optický klam, který se snaží navodit iluzi pohybu. Navození iluze pohybu, je docíleno zobrazováním sekvence po sobě jdoucích obrazů. Aby

[10] Dvou-smyčkovou topologii vytvořil experimentováním a z poznatků získaných z Pixar topologie.

[11] Zde je terminologie kostí použita v kontextu k lidské biologii, nikoliv v kontextu počítačové animace. O kostech v počítačové animaci a jejich ovlivňování povrchové sítě se dozvíte více v sekci 2.3

byla navozena iluze pohybu je nutné, aby se obrazy zobrazovaly několikrát za sekundu. Počet zobrazených obrazů za sekundu se označuje FPS. Lidské oko začíná vnímat obraz jako plynulý [Sito, 2013] od 24FPS. Nicméně v současné době se animace zobrazují i v řádově vyšších hodnotách FPS a to především kvůli plynulosti. Často se používá hodnota 60FPS. V některých počítačových hrách dosahují hodnoty FPS i řádu stovek.[12]

První animace vznikaly jako trikové filmy, které byly promítány v divadlech. Velké zpravodajské firmy platily za to, aby se natáčely filmy s postavami z novinových komiksů, za účelem zvýšení prodeje novin. Felix the Cat, který se poprvé objevil v krátkém filmu v roce 1919, byl první animovanou postavou, která nepochází z tištěného komiksu. Nevinné kouzlo antropomorfních kreslených zvířat rostlo v popularitě.[Sito, 2013]

S každým pokrokem v technice přišel i pokrok v animaci. S příchodem televizorů se zvýšila i poptávka po animovaných filmech. Studia zvládala poptávku především díky metodě snímkové animace (více v sekci 2.2.1). Velkým průkopníkem v animaci byl Walt Disney (1901–1966). Vliv [Selby, 2013], který mělo a stále má na diváky Walt Disney Studios, je obrovský a dalekosáhlý. Walt Disney ovlivnil svou tvorbou populární kulturu nejen své doby, ale i minimálně několika následujících desetiletí.

Dalším obrovským skokem v animačních technologiích byl rozvoj počítačů, který umožnil předchozí metody zautomatizovat, a dovolil také vznik dalších nových metod, jako je například kosterní animace (více v sekci 2.3).

Animace se nevyužívají jen v zábavním průmyslu, ale jsou důležitou součástí digitálních simulací [Parent, 2012], které se používají k tréninku pilotů, SWAT jednotek, ale i třeba operátorů nukleárních reaktorů. Animace tak nachází uplatnění v širokém spektru oborů a lze je použít kdekoliv, kde je potřeba věrohodně simulovat chování rozličných pohybujících se objektů.

2.2.1 Klíčové snímky

Metoda klíčových snímků (keyframing) pochází z dílen Walta Disneyho. Úkolem hlavních animátorů bylo vymyslet postavičky a jejich pohyb. Ale jelikož malování celé sekvence je zdlouhavé a rutinní, tak hlavní animátor vytvořil jen důležité (klíčové) snímky a zbývající (mezi-snímky) malovali animátoři určení pro tuto práci. [Žára, 2005]

V dnešní počítačové grafice se také využívá metoda klíčových snímků. Animátor vytváří klíčové snímky, ale mezi-snímky již dělat nemusí. Mezi-snímky jsou interpolovány [Parent, 2012] pomocí křivek z klíčových snímků. V klíčových snímcích nemusí být uložena pouze pozice objektu. Běžně lze v klíčových

[12] Příkladem takových her je třeba Counter Strike: Global Offensive nebo League of Legends. V tomto případě již nejde o snahu zlepšit plynulost animací. Jedná se o velmi akční hry a cílem vysokých hodnot FPS je zobrazit akce, které při nižších FPS zobrazit nelze, jelikož probíhají příliš rychle.

snímcích ukládat [Microsoft, 2011] barvu, rychlost objektu, natočení, textury [Richtr and Haindl, 2018, Richtr and Haindl, 2015, Haindl and Richtr, 2013] aj.

V rámci metody klíčových snímků rozlišujeme další metody, jak animaci provádět. Každá z metod se liší tím, jaké celky jsou v ní animovány.

- **Per-vertex animation** - Přináší největší volnost pro animátora. Animovány jsou jednotlivé body polygonové sítě. Pomocí této techniky lze vytvořit libovolné animace.

  Velkou nevýhodou je časová náročnost [Hughes, 2014] a pracnost při vytváření klíčových snímků. Další nevýhodou je datová náročnost. Animační data [Jeppsson, 2000] jsou obrovská i při komprimaci. Zároveň neexistuje jednoduchý způsob, jak prolínat mezi animacemi a nebo je kombinovat. Jedinou podstatnou výhodou této metody je nízká zátěž procesoru [Jeppsson, 2000] a jednoduchost implementace animace do herního jádra.

- **Morph target animation** - Jedná se v podstatě o variaci per-vertex animace. Animátor předpřipraví kolekce poloh [Engel, 2018] a výsledná animace je kombinací předpřipravených poloh. Dnes se tato metoda používá především pro animaci obličeje.

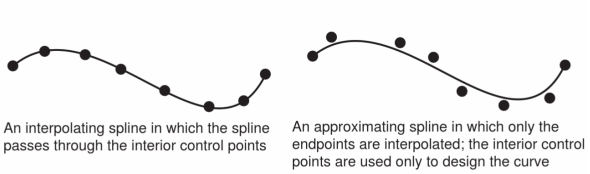
  Morph target animation řeší nemožnost kombinace animací u per-vertex metody. Zároveň snižuje datovou náročnost [Engel, 2018], jelikož mezisnímky se dopočítávají průběžně při zobrazování. Z toho důvodu se zvyšuje zátěž procesoru. Složitost implementace je stále nízká.

- **Kosterní animace** - V této metodě se celá animace provádí za pomocí kostry. Animátor nepohybuje jednotlivými vertexy, místo toho pohybuje kostmi, které zařizují pohyb vertexů. Celkově je tak snadnější a rychlejší animovat postavy (více v sekci 2.3).

2.2.2 Animační křivky

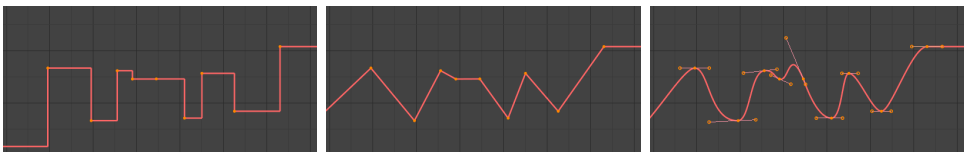
Hodnoty z klíčových snímků popisují animační křivku. Svislá osa představuje hodnotu a vodorovná čas měřený ve snímcích. Animační křivka potom představuje vývoj hodnoty v čase. Důležitou otázkou pro animátora je, jak si bude klíčové hodnoty vykládat. Podle [Parent, 2012] záleží na tom, jestli představují konkrétní body na křivce (interpolace), nebo jestli se jedná o přibližné hodnoty a nepředstavují tak nutně body křivky (aproximace). Rozdíl je vidět na obrázku 2.12.

V manuálech modelovací programů jako je Cinema4D [Maxon, 2020], Maya [Autodesk.Help, 2019] a Blender [Blender Documentation Team, 2021] se můžeme dočíst, že v praxi se běžně používají 3 typy interpolace: konstantní, lineární a Bézierova. Zmíněné typy interpolace se používají především pro jejich



Obrázek 2.12: Porovnání interpolačních a aproximačních křivek [Parent, 2012]

jednoduché ovládání a také pomocí nich lze poměrně snadno docílit všech požadovaných křivek. Porovnání křivek jednotlivých způsobů interpolace je vidět na obr. 2.13.



(a) Konstantní interpolace (b) Lineární interpolace (c) Bézierova interpolace

Obrázek 2.13: Animační křivky [Blender Documentation Team, 2021]

Konstantní

Podle [Blender Documentation Team, 2021] se u konstantní interpolace v podstatě neprovádí žádná interpolace. Pro mezi-snímky [Mukundan, 2012] se využívají hodnoty předchozího klíčového snímku. Takovýto způsob interpolace pro animaci postav je zcela nevhodný, jelikož taková animace by skokově měnila stavy definované klíčovými snímky. Konstantní interpolaci lze použít např. pro vytvoření blikající žárovky.

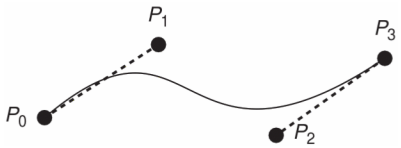
Lineární

Tato interpolace [Blender Documentation Team, 2021] spojuje hodnoty klíčových snímků úsečkou. Hodnoty v mezi-snímcích se pak hledají na této spojnici. Podle [Microsoft, 2011], při lineární interpolaci postupuje animace konstantní rychlostí, ale pouze v mezi-snímcích. Objekty animované tímto způsobem nezrychlují ani nezpomalují. Svou rychlost mění skokově. Takové chování není realistické a pro animaci postav se nehodí.

Bézierova

Při Bézierově interpolaci se používají Bézierovy kubiky. Každá Bézierova kubika [Parent, 2012] je tvořena čtyřmi body: P_0, P_1, P_2 a P_3, přičemž začíná v bodě P_0 a končí v bodě P_3 . Tyto dva body jsou při animaci tvořeny klíčovými snímky. Průběh křivky [Parent, 2012] je řízen pomocí vnějších bodů P_1 a P_2, které tvoří řídící vektory (obr. 2.14) . Při animacích chceme docílit toho,

aby přechody mezi segmenty křivky byly plynulé, čehož lze dosáhnout, pokud je koncový řídící vektor segmentu rovnoběžný s počátečním řídícím vektorem segmentu následujícího. Tato interpolace je nejvhodnější pro animace, jelikož nejvíce odpovídá reálnému chování objektů.



Obrázek 2.14: Segment Bézierovy kubiky [Parent, 2012]

2.3 Kosterní animace

V 3D počítačové grafice často potřebujeme animovat složité objekty, ať už se jedná o postavy, stroje, nebo zvířata. Takovéto modely dost často obsahují mnoho pohyblivých částí, které animátor potřebuje zanimovat. Per-vertex animace by byla zdlouhavá a náročná, ale ani morph target animace by nebyla o moc jednoduší[13]. Z toho důvodu se v takovém případě používá kosterní animace.

Abychom pochopili, jak funguje kosterní animace, tak se podíváme na její vznik. Předchůdcem kosterní animace byla rigidní hierarchická animace. V tomto přístupu je postava modelována jako soustava pevných částí. Humanoidní postavu lze typicky rozložit na pánev, trup, paže, předloktí, ruce, stehna, lýtka, chodidla a hlavu. Rigidní části jsou vzájemně omezeny jejich hierarchií, jak tomu je u kostí savců, které jsou spojeny v kloubech. To umožňuje postavě přirozený pohyb. Například když se pohne paže, předloktí a ruka ji automaticky následují. [Gregory and Lemarchand, 2014]

Velkým problémem tohoto přístupu jsou vznikající trhliny v místě ohybu, jelikož povrchy jednotlivých částí na sebe nejsou žádným způsobem napojeny. Z toho důvodu došlo ke kombinaci přístupu rigidní hierarchické animace a per-vertex animace, čímž vznikla kosterní animace.

2.3.1 Kosterní soustava

V kosterní animaci [Gregory and Lemarchand, 2014] se využívá principů rigidní hierarchické animace k vytvoření kostry. Kostra je tak tvořena jednotlivými rigidními kostmi a klouby, ve kterých lze s kostmi rotovat.

Podle [Žára, 2005] lze kostru definovat jako strom. Jeho uzly odpovídají kloubům a hrany kostem. V každé kostře je jeden kořenový kloub. Kořen je nejvyšší kloub v kloubové soustavě. Nemá žádného předchůdce. Ostatní klouby

[13] I když se historicky obě dvě metody pro zmíněné animace používaly.

jsou jeho potomci. Každý kloub může mít jen jednoho rodiče, ale libovolný počet potomků. Kloub [Parent, 2012], který nemá žádné potomky, se nazývá koncový efektor.

Počet kloubů [Žára, 2005], který strom obsahuje, určuje počet stupňů volnosti celé struktury a tak i míru pohyblivosti. Obecně lidské tělo obsahuje 230-360 kloubů[14]. Při tvorbě kostry je tedy nutné brát v úvahu účel a potřebnou pohyblivost modelu. V praxi se volí mnohem menší počet kloubů. Obvykle stačí počet kloubů v řádu desítek, jelikož není potřeba vytvářet tak detailní animace.

Důvodem, proč je kosterní animace tak výhodná z hlediska datové úspornosti, je podle [Žára, 2005] to, že v každém uzlu je uložena pouze homogenní matice, která popisuje afinní transformaci od jednoho kloubu k druhému. Díky tomu stačí klíčový snímek uložit pouze jako několik matic, na rozdíl od ukládání polohy všech vertexů.

2.3.2 Stupně volnosti

Počet stupňů volnosti určuje, kolika nezávislými způsoby se může objekt pohybovat. V mechanice či robotice se můžeme setkat s termínem „šest stupňů volnosti“. 3D objekt, jehož pohyb není nijak omezen, má tři stupně volnosti díky změně pozice (posun na osách x, y a z) a tři stupně volnosti mu dává rotace (rotace kolem os x, y a z). [Gregory and Lemarchand, 2014]

Nicméně klouby lidské kostry nemají tolik stupňů volnosti. V rámci lidského těla existuje šest různých typů kloubů (obr. 2.15). Klouby v lidském těle tak mají omezení pouze na určitý počet stupňů volnosti. Od každého typu je vybrán jeden zástupce (z biologického hlediska některé klouby vypadají výrazně odlišně, proto se objevují v seznamu, i když mají stejné stupně volnosti):

- Tři rotační stupně volnosti:

  Kyčelní kloub (rotace kolem os x, y a z)
- Dva rotační stupně volnosti:

  Zápěstí (rotace kolem os x a y)

  Karpometakarpální kloub palce (rotace kolem os x a y)

[14] Vědecké články se nedovedou na konkrétním počtu shodnout z několika důvodů. Číslo je nepřesné, jelikož počet kloubů se v průběhu života mění a dalším důvodem je, že neexistuje jednotná definice kloubu.

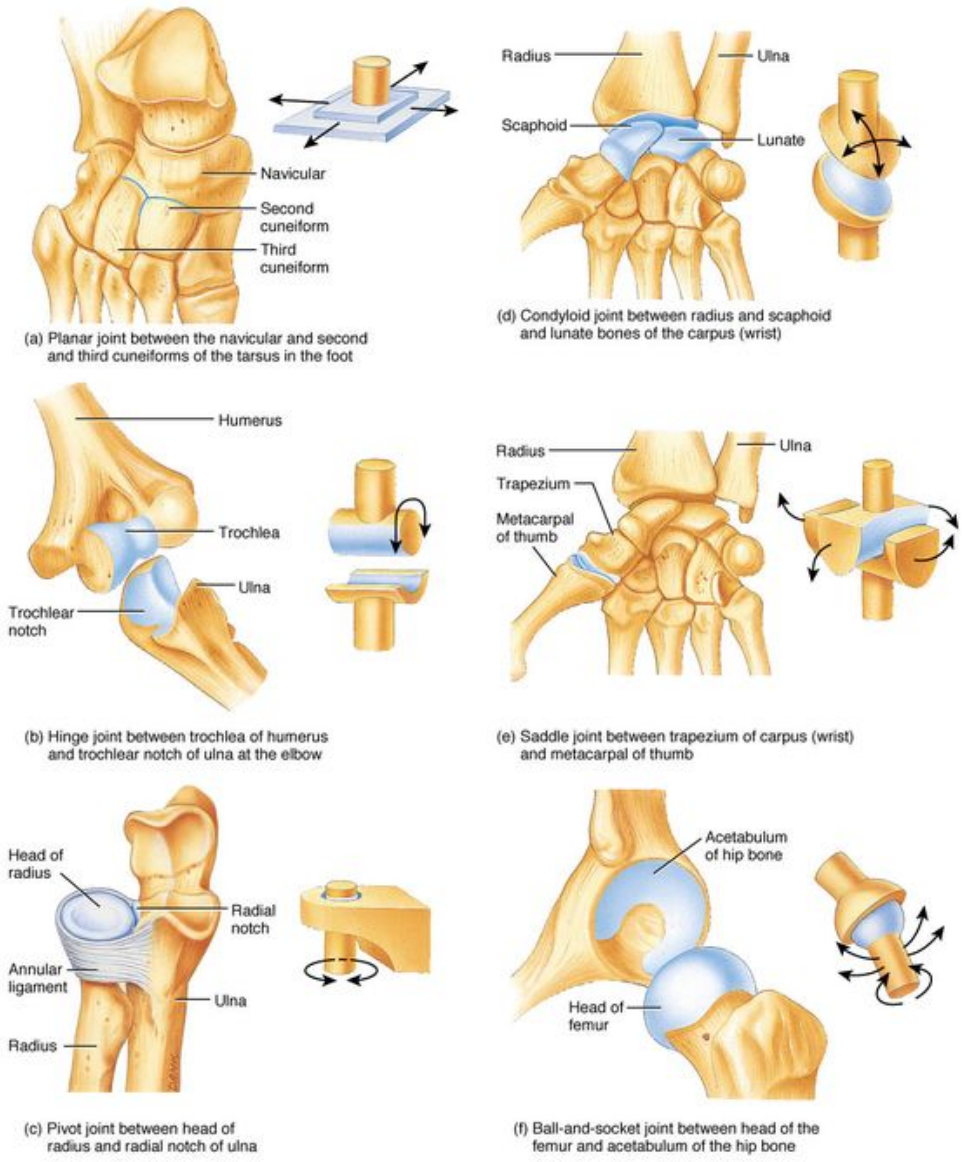
- Jeden rotační stupeň volnosti:

  Loketní kloub, mezi kostmi ulna a humerus (rotace kolem osy x)

  Loketní kloub, mezi kostmi radius a humerus (rotace kolem osy z)

- Dva poziční stupně volnosti:

  Klouby mezi kostmi v krajině zánártní (posun po osách z a y)



Obrázek 2.15: Typy kloubů v lidském těle [LearnBones, 2021]

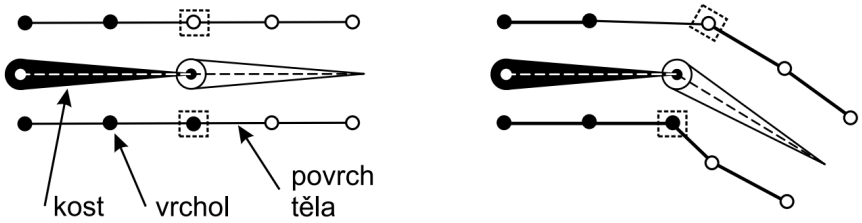
V rámci 3D grafiky se vychází z biologické stavby lidského těla, tudíž se klouby také omezují ve stupních volnosti. Avšak omezení na stupně volnosti není dostatečné. Například ramenní kloub není schopný rotace o 360° kolem osy z. Je tedy nutné omezit i rozsah možných hodnot u jednotlivých stupňů volnosti. Hlavním důvodem této praktiky je snaha zamezit při animaci tvorbě nerealistických poloh. Metody stupňů volnosti pak využívá i inverzní kinematika, o které se dozvíte v sekci 2.4.2.

2.3.3 Skinning

Už víme, jak vytvořit kostru a jak ji animovat, avšak kostra není vidět, tak jak vlastně animovat výsledný model? Abychom mohli pomocí kostry animovat výsledný model, je nutné propojit jednotlivé vertexy modelu s kostmi. Tomuto procesu se říká skinning.

Skinning [Jeppsson, 2000, Žára, 2005, Gregory and Lemarchand, 2014] lze provést dvěma různými způsoby. Prvním z nich je základní kosterní animace [Žára, 2005], která přiřazuje každý vrchol sítě reprezentující pokožku právě jednomu kloubu. Vertexy jsou potom transformovány stejně jako je transformován kloub, ke kterému jsou přiřazeny.

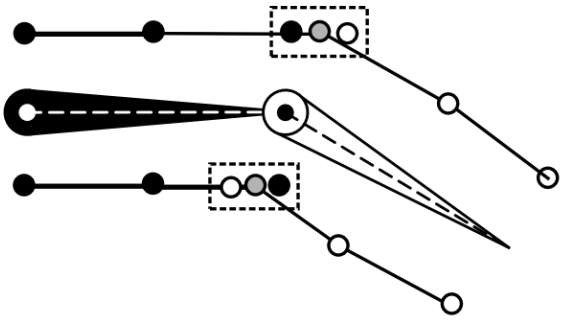
Základní kosterní animace [Žára, 2005] je vhodná pro modely tvořené více pevnými částmi, kde je každá část přiřazena jednomu kloubu. Při tomto použití v podstatě dostáváme rigidní hierarchickou animaci popsanou na začátku této sekce (2.3), akorát je navíc vytvořena kostra. Podle [Žára, 2005, Jeppsson, 2000] můžeme tuto metodu použít také na deformovatelný model, avšak oba autoři se schodují, že výsledky nejsou o moc lepší. Povrch modelu není hladký a deformace se projeví jen na plochách, jejichž vertexy jsou přiřazeny k rozdílným kloubům (obr. 2.16).



Obrázek 2.16: Základní skeletální animace aplikovaná na deformovatelnou pokožku. Vlevo dvojice kostí v referenční poloze, vpravo po rotaci kloubu. Vrcholy pokožky přiřazené první kosti jsou označeny černě. Na vrcholech zvýrazněných rámečky je vidět nepřirozená deformace pokožky způsobená pevným přiřazením k jednotlivým kostem [Žára, 2005]

Druhým způsobem je míchání vrcholů [Gregory and Lemarchand, 2014, Žára, 2005, Jeppsson, 2000], které na rozdíl od základní kosterní animace povoluje vazbu vertexu na libovolný počet kloubů. Výsledná poloha [Žára, 2005]

vrcholu se spočítá jako konvexní kombinace transformací tohoto vrcholu příslušnými klouby. Každý kloub, ke kterému je vertex přiřazen, se tak podílí na výsledné poloze kloubu. Míra [Gregory and Lemarchand, 2014], jakou se podílí transformace kloubu na výsledné poloze vrcholu, se označuje jako váha. Součet vah jednotlivých kloubů pro daný vertex musí být vždy roven jedné. Proces míchání vrcholů je znázorněn na obrázku 2.17.



Obrázek 2.17: Deformace mícháním vrcholů. V označené oblasti jsou nakresleny černě a bíle polohy vrcholů po transformaci způsobené první a druhou kostí. Šedé kroužky představují výsledné vrcholy vzniklé mícháním [Žára, 2005]

[Jeppsson, 2000] poukazuje na to, že i tato metoda má své nevýhody. Za první považuje vyšší výpočetní náročnost oproti základní kosterní animaci, jelikož na výpočtu se podílí více kloubů. Druhou nevýhodu vidí v tom, že některé modely mohou vytvářet neočekávané deformace v extrémních pozicích v případě, že v místě ohybu je použit malý počet vertexů.

První nevýhoda není tak velkým problémem vzhledem k tomu, že vertexy ovlivňuje více kloubů především v oblastech ohybu (tudíž převážná část zůstává ovlivňována pouze jedním kloubem). Navíc vertexy, které jsou ovlivňovány více klouby, většinou ovlivňují pouze dva klouby (pouze vzácně se jedná o větší počet). Druhou nevýhodu má za úkol řešit správná topologie (sekce 2.1.2).

2.3.4 Kombinování animací

V počátcích počítačových her bylo pro vytváření animací potřeba velké množství animátorů, jelikož neexistovaly jednoduché metody pro kombinovaní animací. Pokud byla do hry vyžadována animace bežící a mávající postavy, tak animátor musel tuto animaci vytvořit, aniž by mohl použít již existující animace. Stejně tomu bylo i pokud bylo potřeba vytvořit přechod mezi dvěma animacemi (například běh a chůze). Navíc v takovém případě musela být postava ve speciální části animace, aby vůbec k přechodu mohlo dojít.

Animace vytvořené kosterní animací lze kombinovat díky podobné metodě jakou jsou vertexy napojeny na klouby. Při kombinaci animací se v každé animaci nastaví váha [Jeppsson, 2000] kloubu, která určuje jeho důležitost v animaci. Tímto způsobem lze kombinovat 2 a více animací.

Například u animace běhu jsou nejdůležitější nohy a pas, z toho důvodu jim bude nastavena nejvyšší váha. Zbytku kloubů pak bude nastavena váha minimální (až nulová). U mávání ruky budou mít nejvyšší důležitost klouby ruky, kterou se mává, a zbylé klouby budou mít váhu menší.

Při kombinování animací tak váha kloubu [Jeppsson, 2000] rozhoduje, jak bude kloub výslednou animaci ovlivňovat. V případě, že stejný kloub má ve více animacích nenulovou váhu, je nutné mezi maticemi v nich uložených interpolovat.

Příkladem pak mohou být animace ukazovaní pravou rukou doleva a ukazovaní pravou rukou doprava. Pokud tyto dvě animace budeme kombinovat, tak zjistíme, že v obou animacích mají vysokou váhu stejné klouby. Interpolací[15] lze pak vytvořit animaci ukazovaní rukou v libovolném úhlu mezi extrémy.

2.4 Kinematika

V rámci práce jsme už stihli zavítat do matematiky, geometrie, biologie, medicíny a nyní se podíváme i na fyziku. Mechanika je obor fyziky, který se zabývá pohybem těles. Lze jï rozdělit na dynamiku a kinematiku. Dynamika se zabývá pohybem z hlediska působení sil. Naproti tomu kinematika se zabývá pouze druhy pohybu a jejich klasifikací.

My se zaměříme na kinematiku, která nám vysvětlí, jak správně pohybovat s vytvořenou kostrou modelu. Ve spojitosti s kinematikou [Žára, 2005] je potřeba si vysvětlit několik pojmů[16]:

- **Spojení** - Otočné spojení mezi dvěma pevnými segmenty. V kostře se nazývá kloub.
- **Segmentová struktura** - Posloupnost pevných částí, které jsou mezi sebou spojeny spojením (kloubem), tvoří segmentovou strukturu. V každém spojení je možno s oběma segmenty otáčet. Segmentová struktura bývá na jednom konci pevně zakotvená. Je-li její druhý konec volný, říká se takové struktuře otevřená segmentová struktura.
- **Koncový efektor** - Bod, který je na volném konci otevřené segmentové struktury, se nazývá koncový efektor.

[15] Konkrétními způsoby interpolace se v této práci zabývat nebudeme. Pokud vás téma zajímá, tak se můžete dočíst více v [Gregory and Lemarchand, 2014, str. 575], nebo v [Jeppsson, 2000, str. 17].

[16] Některé se v rámci práce již objevily, ale pro jistotu je ještě zopakujeme.

- **Stavový vektor** - Okamžitý stav segmentové struktury může být jednoznačně popsán tzv. stavovým vektorem. Délka stavového vektoru je stejná jako je počet stupňů volnosti segmentové struktury. Stavový vektor budeme označovat:

$$\Theta = (\theta_0, \theta_1, ..., \theta_n), \tag{2.1}$$

2.1: kde θ_i jsou možné parametry modelu [Žára, 2005]

Cílem kinematiky [Parent, 2012, Žára, 2005, Mukundan, 2012] je určit polohu koncového efektoru a hodnoty stavového vektoru segmentové struktury. Existují dva různé přístupy, jak tento problém řešit – přímá kinematika a inverzní kinematika.

2.4.1 Přímá kinematika

Přímá kinematika (FK z anglického Forward Kinematics) [Mukundan, 2012, Parent, 2012, Žára, 2005] se dívá na problém určení polohy koncového efektoru postupným určením jednotlivých parametrů stavového vektoru pro všechny segmenty stavové struktury. Proces si můžeme představit na vytvoření klíčového snímku animace mávání. Nejprve se stanoví úhel natočení v rameni, potom v lokti a následně v zápěstí. Výsledkem bude poloha ruky (koncového efektoru). Pokud změníme natočení libovolného kloubu, změní se pozice všech jeho potomků, tedy všech kloubů po něm následujících včetně koncového efektoru. Z toho důvodu se prochází strom segmentové struktury vždy od kořene a postupně se nastavují úhly všech kloubů, což je i hlavní nevýhodou přímé kinematiky. Formálně [Žára, 2005] lze zmíněný postup zapsat jako

$$X = f(\Theta). \tag{2.2}$$

2.2: Poloha koncového efektoru X je určena pomocí transformace f, která je sestavena na základě nové hodnoty stavového vektoru Θ, v němž byly promítnuty všechny změny v jednotlivých stupních volnosti animované struktury [Žára, 2005]

Algoritmy využívající FK jsou oproti inverzní kinematice jednodušší na implementaci. Nicméně [Parent, 2012, Žára, 2005] se shodují na tom, že umístění koncového efektoru do konkrétní polohy je pro animátora neintuitivní a někdy i téměř nemožné. Z tohoto důvodu se často využívá inverzní kinematika.

2.4.2 Inverzní kinematika

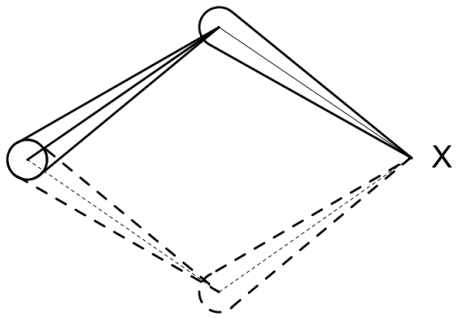
Inverzní kinematika (IK z anglického Inverse Kinematics) [Mukundan, 2012, Parent, 2012, Žára, 2005] používá k výpočtu postup opačný. Poloha koncového

efektoru je dopředu známá a cílem je nalézt hodnoty stavového vektoru Θ. [Žára, 2005] uvádí jako příklad stanovení úhlů natočení kloubů při jízdě na kole. Koncový efektor (chodidlo) je známý a algoritmus inverzní kinematiky určí stavový vektor. Formálně můžeme tento krok zapsat jako:

$$\Theta = f^{-1}(X). \tag{2.3}$$

2.3: Poloha koncového efektoru X je známa, pomocí inverzní funkce f^{-1} určíme stavový vektor Θ [Žára, 2005]

Zásadním problémem inverzní kinematiky je nalezení inverzní funkce f^{-1}. Obecně nemusí taková funkce existovat. Nejjednodušším příkladem je příliš vzdálený koncový efektor. Po struktuře je vyžadováno natažení delší, než je celková délka segmentů. Nebo naopak může být řešení více jako na obrázku 2.18, kde je koncový efektor zvolen tak, že existují dvě různá řešení, jak ho dosáhnout. [Mukundan, 2012, Žára, 2005, Parent, 2012]



Obrázek 2.18: Nejednoznačné řešení pro danou strukturu [Žára, 2005]

Nejednoznačnost řešení se dá částečně vyřešit omezením [Parent, 2012, Mukundan, 2012, Žára, 2005] stavového prostoru. Otáčení kloubů můžeme omezit na konkrétní rozsah hodnot. Sice tímto způsobem nezamezíme všem nejednoznačným řešením, ale alespoň odstraníme jejich podstatnou část.

U struktur s nízkým počtem segmentů [Mukundan, 2012] lze použít analytický výpočet, který je nejrychlejším známým řešením. Výhodou analytického řešení je, že najde všechna řešení. Avšak u rozsáhlejších struktur se tato výhoda stává nevýhodou, jelikož vzhledem k počtu řešení se problém začne velmi rychle rozšiřovat.

Pro složitější struktury se tak používá iterativní metoda, kdy se koncový efektor z počáteční pozice postupně přibližuje cíli. Způsobů výpočtu iterativní metody je více, my si v této práci ukážeme výpočet za pomoci inverze jakobiánu.

Inverze jakobiánu

Jakobián je matice parciálních derivací o rozměrech $m \times n$, která mapuje změny $\triangle\Theta$ na změny $\triangle X$, kde m je dimenze X (v prostoru je obvykle $m = 3$) a n je dimenze Θ (n závisí na počtu stupňů volnosti soustavy). Jakobián $\mathbf{J}$ závisí na konkretním stavu segmentové struktury. Pro jakobián platí

$$\triangle X = \mathbf{J}(\Theta)\triangle\Theta. \tag{2.4}$$

2.4: Výpočet změny polohy koncového efektoru [Žára, 2005]

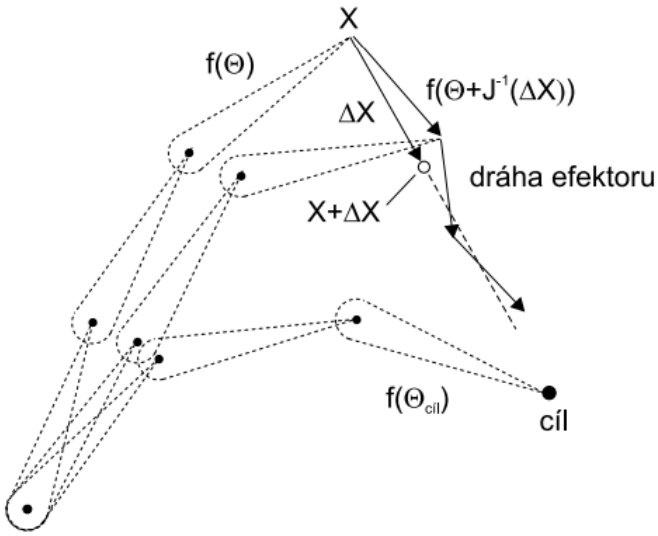
Přičemž jakobián můžeme zapsat jako:

$$\mathbf{J}_{m\times n} = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_n} \\ \vdots & \ddots & \cdots & \vdots \\ \frac{\partial f_m}{\partial \theta_1} & \cdots & \cdots & \frac{\partial f_m}{\partial \theta_n} \end{bmatrix}$$

Pokud dokážeme vypočítat inverzi jakobiánu, tak lze původní rovnici 2.3 po úpravě převést do tvaru:

$$\triangle\Theta = \mathbf{J}^{-1}(\Theta)\triangle X. \tag{2.5}$$

2.5: Pomocí inverzního jakobiánu $\mathbf{J}^{-1}$ určíme změnu stavového vektoru $\triangle\Theta$ [Žára, 2005]



Obrázek 2.19: Postupná iterace při použití pseudoinverze [Žára, 2005]

Nyní jen stačí dosadit do rovnice 2.5. Iterativní výpočet, o kterém jsme mluvili na konci sekce 2.4.2, je charakterizován algoritmem 1. Jeho fungování je zobrazeno na obrázku 2.19.

Algoritmus 1: Výpočet inverzní kinematiky pomocí inverze jakobiánu [Žára, 2005]

```
/* stavový vektor hierarchie Θ
   poloha koncového efektoru v daném stavu, tj. X = f(Θ)
   cílová poloha koncového efektoru X_goal                    */
Vstup: Θ
       X
       X_goal
while X ≠ X_goal do
    J(Θ) ;                                  // vypočítej jakobián f(Θ)
    J^-1(Θ) ;                                    // invertuj jakobián
    △X = k · (X_goal − X), 0 < k < 1;  // zvol malý pohyb směrem k cíli
    Θ += J^-1(Θ)△X;                  // odhadni změnu stavového vektoru
    X = f(Θ);              // efektor do polohy odpovídající změně stavu
end
/* stavový vektor hierarchie Θ s koncovým efektorem v cílové pozici */
Výstup: Θ
```

V algoritmu narazíme na dva zásadní problémy. Prvním problémem je tvar matice **J**. Matice je nejčastěji obdélníková [Mukundan, 2012, Parent, 2012] a její inverze tudíž neexistuje. Problém lze snadno vyřešit použitím metody pseudoinverze obdélníkové matice [Mukundan, 2012, Parent, 2012].

Druhý problém spočívá v závislosti jakobiánu $\mathbf{J}(\Theta)$ na aktuálním stavu hierarchie. Tedy výpočet má smysl jen pro malé $\triangle\Theta$ a tudíž i malé $\triangle X$. Z toho důvodu se omezuje maximální velikost $\triangle\Theta$ [Parent, 2012, Žára, 2005] zmenšením $\triangle X$ na polovinu, což se provádí tak dlouho, dokud $\triangle\Theta$ není dostatečně malé. Zmenšování $\triangle X$ způsobí, že posuneme koncový efektor pouze o část požadované vzdálenosti. Z toho důvodu je nutné výpočet opakovat tak dlouho, dokud se nedostaneme do cílového bodu.

Kapitola 3

Projekt Western Town

První část této kapitoly se zaměřuje na projekt Western Town. V druhé části se seznámíme s jednotlivými členy týmu a s jejich prací na projektu. Na projektu spolupracovalo sedm studentů FIT ČVUT a je plánována spolupráce i nad rámec studia.

3.1 Vznik

Projekt vznikl v rámci předmětu Virtuální herní světy. Cílem předmětu je vytvořit herní svět s veškerými náležitostmi. Jedná se o velice obsáhlý předmět, který využívá předchozích znalostí z předmětů grafického zaměření a dává jim novou náplň. Zároveň však přináší novou látku jako jsou základy teorie herního designu a principy psaní dialogů a postav. Získané znalosti jsou poté uplatněny v hlavním výstupu předmětu, kterým je velice obsáhlá semestrální práce.

3.2 Členové týmu

V rámci prvních přednášek bylo nastíněno o jak obsáhlý projekt se jedná, a tak byla snaha identifikovat hlavní role, které členové týmu budou zastávat. Celkově tak bylo vymezeno šest nejpodstatnějších rolí:

- Vedení týmu
- Grafika
- Programování
- Hudba a zvuk
- Příběh a postavy
- Herní mechaniky

Smyslem takového rozdělení bylo, aby se každý z členů věnoval menšímu okruhu úkolů a byl co nejvíce zachován jednotný styl v rámci zaměření. Rozdělení mělo další výhodu, a tou byla jednoduchá správa jednotlivých sekcí, jelikož za ní vždy zodpovídal konkrétní člen. Nicméně se našla i nevýhoda, a tím byl problém, že některá zaměření byla úkolově obsáhlejší a náročnější než jiná, a tak každý z členů týmu dostal sekundární roli. Kvůli obsáhlosti některých témat tak byl počet členů týmu nastaven na sedm.

Následně na vymezené pozice proběhl nábor podle zkušeností, které byly pro danou pozici potřeba (tab. 3.1). Sám jsem se stal vedoucím týmu a jako sekundární roli jsem si zvolil grafiku, jelikož se o ni hodně zajímám a už mám zkušenosti s tvorbou modelů, materiálů a animací. Na místo hlavní grafičky byla vybrána Anna Doležalová, jelikož se zabývá tvorbou modelů postav. Jako sekundární roli si vybrala příběh a postavy, jelikož s tvorbou modelů postav souvisí tvorba jejich charakteru. Tomáš Selmeci byl vybrán na základě předchozí spolupráce v rámci GameJamu[17], kde v našem týmu dělal taktéž programátora. Jako vedlejší role mu byly přiděleny herní mechaniky, především z důvodu znalosti oboru a nutnosti následného naprogramování. Hlavním tvůrcem příběhu se stala Kristína Václavová, díky předchozím zkušenostem s tvorbou krátkých povídek. Zároveň jako vedlejší role jí byla přidělena grafika. Ježek Kryštof se kvalifikoval na pozici hudba a zvuk na základě předchozích zkušeností s programy na vytváření hudby. Jako vedlejší mu byly vybrány herní mechaniky. Vladislav Vertskhayzer byl vybrán pro tvorbu herních mechanik na základě jeho předchozího zájmu v oboru. Zároveň se přihlásil na pozici příběh a postavy. A posledním členem týmu se stal Andrey Cherkasov, který byl vybrán na pozici hudba a zvuk především díky tomu, že umí hrát na piano a skládat hudbu. Jako sekundární roli si také vybral herní mechaniky.

Jméno	Hlavní pozice	Vedlejší pozice
Petr Hromják	Vedení týmu	Grafika
Anna Doležalová	Grafika	Příběh a dialogy
Tomáš Selmeci	Programátor	Herní mechaniky
Kristína Václavová	Příběh a dialogy	Grafika
Ježek Kryštof	Zvuk	Herní mechaniky
Vladislav Vertskhayzer	Herní mechaniky	Příběh a dialogy
Cherkasov, Andrey	Zvuk	Herní mechaniky

Tabulka 3.1: Členové týmu a jejich zaměření

V rámci týmu bylo rozhodnuto, že si veškeré podklady vytvoříme vlastní, abychom si vyzkoušeli tvorbu hry od úplného začátku. Díky tomuto přístupu

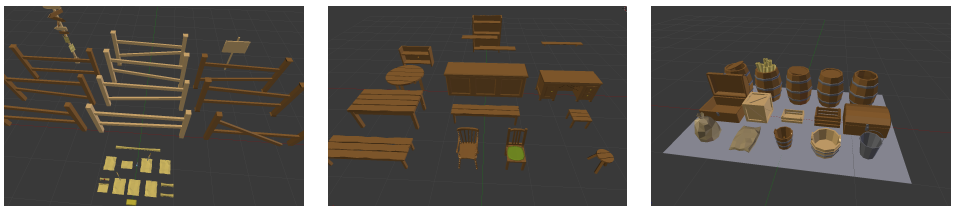
[17]GameJam je 24 hodin dlouhá programovací akce s cílem vytvořit hru dle zadání, jež bude vyhlášeno na začátku. Akci pořádá FIT.

si mohl každý člen vyzkoušet, jak probíhá tvorba hry v rámci oboru, který ho zajímá.

3.2.1 Grafika

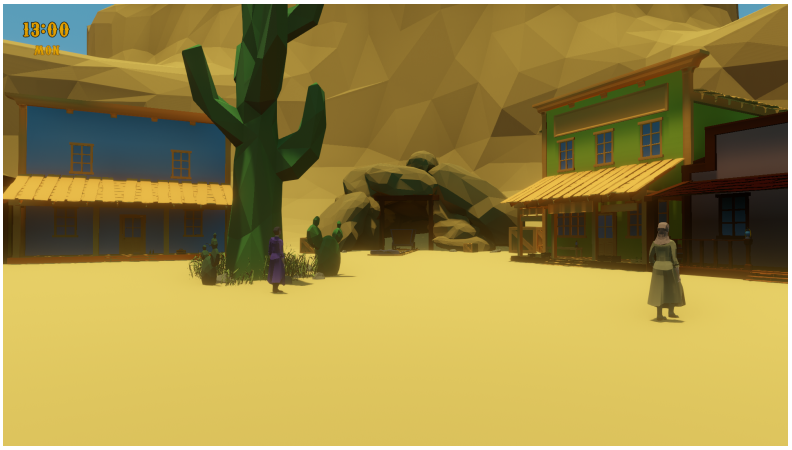
Grafici si pro tvorbu modelů vybrali modelovací program Blender. Blender má všechny důležité funkcionality potřebné pro modelování ale i pro animování. Dalším důvodem pro výběr tohoto programu bylo také to, že jeho užívání je bezplatné. Nicméně hlavním důvodem pro výběr programu Blender byly předchozí zkušenosti členů týmu.

Z počátku (tedy v rámci předmětu VHS) byly vytvořeny balíčky modelů. Vzniklo tak několik balíčků (příklady jsou vidět na obr. 3.1), ze kterých byly následně sestaveny složitější scény (obr. 3.2, 3.3, 3.4). Veškeré modely byly vytvořeny ve stylu low-poly s jednoduchými materiály.



(a) vedlejší objekty (b) nábytek (c) úložný prostor

Obrázek 3.1: Balíčky modelů pro projekt Western Town



Obrázek 3.2: Venkovní scéna

Obrázek 3.3: Saloon

Obrázek 3.4: Obchod

V rámci předmětu se bohužel nestihly dodělat animace, a tak byly vytvořeny pomocí webového programu Mixamo [Adobe, 2021a]. Přestože se jedná o velice propracovaný program, tak animace nedopadly podle očekávání. Především z tohoto důvodu se jejich úpravě věnuje tato práce. (Více o Mixamo a dosavadních animacích se dozvíte v sekci 4.1.)

3.2.2 Programování

Hráč byl nejdříve vytvořen jako kamera z pohledu první osoby a postupně se kolem této struktury začaly přidávat funkcionality. Pohyb hráče bere údaje z horizontálního a vertikálního vstupu Unity (tudíž nezáleží na zařízení ovládajícím hru). Interakce hráče s okolím je zprostředkována pomocí skriptu na

výběr objektů pomocí raycast[18]. Hráč má také vytvořený systém životů, který reaguje na to, když se zraní, a také řeší případ smrti. Zároveň je pro hráče vytvořen stavový automat, který řídí animace a přechody mezi nimi.

Chování NPC[19] je řízeno pomocí jednoduchého stavového automatu. Postavy mají vytvořenou svojí denní rutinu, která je závislá na dni a je řízena fuzzy logikou. Jelikož jsou venkovní a vnitřní části budov odděleny, bylo taky nutné vytvořit systém, který bude správně NPC přesouvat mezi scénami.

Pro hru bylo také potřeba vytvořit dialogový systém, který je provázán se systémem úkolů. V dialogu je tak možné úkoly přijímat, odevzdávat, nebo v nich i jinak postupovat. Zároveň byla do dialogu přidána možnost přijímat a předávat předměty v inventáři, což umožňuje vytvářet složitější úkoly, ale také lze přes dialogový systém řešit obchodování s ostatními postavami.

Ve hře je dále vytvořen systém loot boxů[20]. Každý objekt, ze kterého lze získat předměty (náboje, peníze atp.) má vlastní tabulku, která určuje, jaký předmět hráč získá a v jakém množství.

Události ve hře jsou řízeny pomocí manažerů. Například Danger Manager má na starosti míru nebezpečí ve městě. Pokud nebezpečí překročí určitou mez, Danger Manager na to zareaguje přepnutím stavu měšťanů a ti začnou reagovat na nebezpečí. Tímto způsobem jsou řešeny i další události ve hře.

3.2.3 Hudba a zvuk

Před tvorbou hudby se zvukaři dohodli, že budou používat autentický výběr nástrojů, jako je banjo, foukací harmonika, flétna, či kytara (která byla ovšem z technických důvodů elektrická a ne akustická). Jak výběr nástrojů, tak samotné skladby jsou inspirované soundtracky k filmům od italského skladatele Ennio Morricone, skladbami z jiných westernových her jako Call of Juarez a hudbou skladatelů jako Nick Cave, či The Monkeywrench.

Každá skladba ve hře má svůj účel a má za úkol navodit určitou atmosféru. Například skladba v menu má navodit pocit začínajícího příběhu, a proto je hravá a má rychlejší tempo než třeba ambientní skladby hrající během noci, které jsou klidné a tiché. V salónu hraje skladba na klavír, kterou zvukaři společně složili. Skladba je krátká a bezstarostná na rozdíl od ostatních skladeb. Opakování skladby salón částečně odděluje od okolního světa a ukazuje jej jako jakési neměnné útočiště.

Při připravování zvuků do hry zvukaři zjistili, že je občas důležitější subjektivní vjem zvuku než jeho autenticita. Například pro zvuk střelby z revolveru použili nahrávku střelby z brokovnice Beretta 12, a paradoxně pro zvuk střelby z brokovnice použili nahrávku revolveru Remington New Army 1858. Některé

[18] Zjišťování okolních objektů pomocí vyslání paprsku.
[19] Ostatní nehratelné postavy.
[20] Z určitých objektů lze získávat předměty.

zvuky nahrávali vlastnoručně. Například zvuk kopání do dřevěných přepravek byl nahráván kopáním do dřevěné bedny. Než byl nalezen vhodný zvuk, vyzkoušel Kryštof mnoho bot a povrchů na kterých bedna stála. Obdobně natáčel i zvuk kopání do skleněných lahví, kdy ovšem musel lahev zavěsit na provaze, aby nahrávka neobsahovala zvuk pohybu lahve po dřevěné podlaze.

3.2.4 Příběh a postavy

Příběh je hráči představen v úvodním rozhovoru s šerifem a dále se rozvíjí na základě výběru dialogových možností či konkrétních akcí v jednotlivých hlavních úkolech, které pozitivně nebo negativně ovlivňují hráčovu karmu. Hra končí bojovou scénou, ve které se hráč snaží osvobodit svého bratra a snoubenku ze zajetí banditů. Díky rozvětvení příběhu na základě karma systému vzniklo sedm různých konců hry. Hlavním důvodem pro sedm konců byla symbolika. Tým má sedm členů a sedmičku má i ve svém názvu.

1. Hráč v úvodním dialogu odmítne spolupracovat se šerifem a zůstane vězněm.
2. V poslední bitvě přežije bratr spolu se šerifem.
3. V poslední bitvě přežije snoubenka spolu se šerifem.
4. Zachráněni jsou bratr, snoubenka, šerif a přežije bandita z předchozího úkolu.
5. Šerif zemře, zachráněn je bratr.
6. Šerif zemře, zachráněna je snoubenka.
7. Šerif zemře, zachráněni jsou bratr i snoubenka, a navíc i kriminálník z vězení.

Součástí příběhu je i několik vedlejších postav, které se podílí na zadávání úkolů a umožňují hráči obchodovat, dozvídat se více o městě, ostatních měšťanech a příběhu hry. Jednou z nejinformovanějších osob je barman ze salónu, který se při své práci od zákazníků dozvídá všemožné zprávy o současném dění a je ochotný je mezi řečí předávat dál. Téměř veškeré obchodování ve městě zařizuje místní hokynář. Výborně se vyzná ve zbraních, hráči je schopen jich několik poskytnout a případně mu i prodat další náboje. Příliš informací ale neposkytuje, zajímá se spíše sám o sebe a svůj obchod. Nejdůležitější osobou je ale samotný šerif. Ví o každé drobnosti, co se v okolí stane, a snaží se své město chránit před neustálými útoky banditů. Hráči za jeho spolupráci postupně odhaluje, co se stalo jeho nejbližším, upozorňuje jej na přicházející útoky banditů a přiděluje úkoly.

3.2.5 Herní mechaniky

Návrh herních mechanik zahrnoval mnoho kategorií ovlivňujících různé části hry – od vzhledu a funkčnosti uživatelského rozhraní, přes principy herní ekonomiky a skrytého karma systému, až po návrh boje na základě průzkumu dobových zbraní. V rámci předmětu BI-VHS nebyly všechny zamýšlené funkcionality implementovány. Z časových důvodů musely některé navrhované koncepty být odloženy k pozdějšímu zpracování.

Pojetí a implementace herního času umožnily vznik dalších mechanik, jakými jsou například příchod měšťanů na nedělní mši či odchod z práce domů na polední pauzu. Tato mechanika umožňuje hráči nepozorovaný vnik do nezabezpečených budov, jelikož měšťané občas zapomenou zamknout. Zároveň na části dne (poledne, stmívání apod.) upozorňuje zvuk kostelního zvonu. S každým herním dnem se hráči navíc otevírá možnost plnění dalšího hlavního úkolu.

Kapitola 4

Analýza současného řešení

V první části této kapitoly se podíváme na současné řešení animací v projektu Western Town. Zároveň si popíšeme nástroj Mixamo a jeho výhody a nevýhody. V druhé kapitole se zaměříme na chybně vypočítané animace, oblasti ve scéně, kde může docházet k problémům, ale i na špatně vytvořené kolizní modely. U všech problému také zjistíme, proč k nim dochází.

4.1 Zhodnocení použitého řešení

Veškeré animace ve hře byly vytvořeny za pomoci programu Mixamo. Modeláři stačí vymodelovat humanoidní[21] postavu a Mixamo pro ní vytvoří kostru. Nnásledně si může modelář vybrat z předpřipravených animací, které si už jen stáhne a nahraje do svého herního jádra.

Výhodou je rychlost a velká databáze animací, ze které lze vybírat. Nevýhodou je, že v případě špatně vytvořené kostry nemůže modelář kostru změnit a napravit tak chybné animace. Obdobně pak nemůže opravit automatický skinning, kterým je model napojen na kostru. Obě tyto operace provádí Mixamo, ale nelze mu kostru předpřipravit. Vstupem programu je pouze model postavy.

Ve hře je použita pouze přímá kinematika, která nedovoluje přizpůsobit animace okolí. Kvůli tomu pak některé animace nepůsobí příliš věrohodně a zároveň se postavy zaseknou na každé drobné překážce. Dalším problémem je, že takové animace nelze využít k tvorbě procedurálních animací.

[21] Taková postava musí mít hlavu, torso, dvě nohy a dvě ruce. Končetiny se musí nacházet na obvyklých místech.

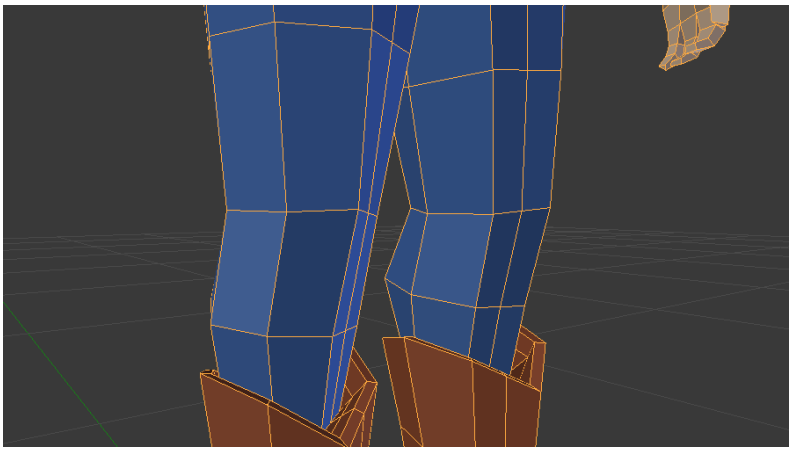
4.2 Problémová místa

Bohužel ve hře Western Town došlo k několika chybám v animacích. Za některé může použitý software Mixamo a za některé může použití přímé kinematiky místo kinematiky inverzní. Celkově tak můžeme tyto chyby rozdělit do pěti kategorií:

- Model postavy
- Kostra
- Skinning
- Animace
- Procedurální animace

4.2.1 Model postavy

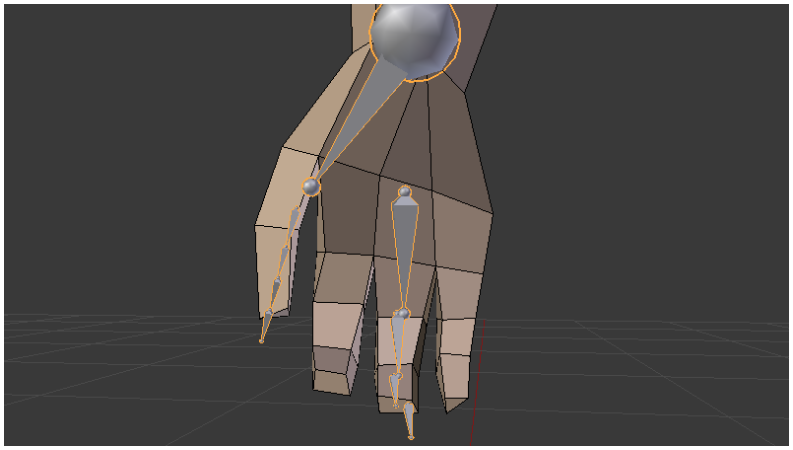
Pokud by model postavy nebyl animován, tak je topologie naprosto v pořádku. Model neobsahuje díry, normály jsou správně otočené, nebyly použity n-gony, atd. Nicméně pokud se podíváme na správnost topologie z hlediska animací, tak disponuje zásadním nedostatkem. V místech kloubů se sice nachází edge loops (obr. 4.1), ale jedná se pouze o nejjednodušší typ (sekce 2.1.2).



Obrázek 4.1: Nevhodná topologie kolen pro animace

4.2.2 Kostra

Při otevření souboru s animací si hned lze všimnout, že kostra nebyla správně vygenerována. Když odhlédneme od různých možností počtu kostí, tak je vidět, že ruka má vytvořené kosti pouze pro dva prsty (obr. 4.2). Z názvů kostí je patrné, že Mixamo detekovalo pouze ukazováček a palec. Navíc některé umístění kloubů není příliš vhodně zvoleno.



Obrázek 4.2: Špatně vygenerovaná kostra

4.2.3 Skinning

V některých případech si můžeme všimnout, že Mixamo nedokázalo některé z vertexů správně přiřadit kostem. Buď nejsou ke kostem přiřazeny vůbec, nebo mají špatně nastavenou váhu. Výrazně je tento problém vidět na složitějším oblečení. Nejvíce si toho můžeme všimnout na sukních měšťanek (obr. 4.3).

Obrázek 4.3: Špatně provedený skinning

4.2.4 Animace

V rámci hry bylo použito pár nevhodných animací. Jedním z výrazných příkladů je použití stejné animace pro chůzi dopředu, dozadu a do obou stran. Dalším příkladem je nehodící se poloha rukou při držení zbraně (obr. 4.4). Navíc je použita vždy stejná poloha rukou bez ohledu na zbraň.

Obrázek 4.4: Nevhodné držení zbraně

4.2.5 Procedurální animace

Další podstatnou chybou animací je jejich neschopnost reagovat na okolí. A tak postava nebere v potaz, kde stojí, a nijak tomu nepřizpůsobuje svůj postoj. Proto se může například stát, že při chůzi po schodech nepokládá chodidla na jednotlivé schody. Zároveň se také může stát, že pokud se postava zastaví na okraji objektu (např. schodu), tak jedno z chodidel se chová jako by na objektu stálo (obr. 4.5), i když pod ním objekt není (stojí „ve vzduchu“).

Obrázek 4.5: Animace nereaguje na okolí (modrá - objekt, na kterém postava stojí; červená - chodidlo „ve vzduchu“)

Kapitola 5

Uživatelské testování

V první části této kapitoly budou navrženy testy pro uživatelské testování na základě zjištěných nedostatků v projektu. Ve druhé části bude navržena forma a způsob testování. V poslední části budou zhodnoceny výsledky testování.

5.1 Navržení testů

Při navrhování testů pro uživatelské testování budeme vycházet z sekce 4.2, kde byly chyby rozděleny do pěti kategorií. Bohužel z povahy chyb v kategoriích model postavy, kostra a skinning není možné pro běžného uživatele takovéto chyby ve hře rozeznat, nebo správně zařadit. Z toho důvodu z hlediska uživatele budeme mít pouze tři kategorie:

- Animace hlavní postavy
- Animace ostatních postav
- Interakce s okolím

V každé kategorii byl vytvořen jeden test. Zadání testů je obecné, jelikož se nesnažíme testovat konkrétní funkcionality ve hře, ale celkový estetický dojem z animací postav.

5.2 Průběh testování

Vzhledem k současné pandemii COVID-19[22] proběhne testování kompletně online formou. Testování bude zprostředkováno pomocí dotazníku, který provede uživatele jednotlivými testy. Zároveň do dotazníku uživatelé vyplní výsledky testování. V průběhu testování budou uživatelé v hovoru s moderátorem testu, který bude pomáhat s průběhem testu v případě problémů.

[22]Nemoc se v ČR poprvé vyskytla 1.března 2020. Bohužel opatření proti této nemoci stále ovlivňují každodenní život občanů.

Každému zjištěnému problému bude přidělena závažnost na základě jeho četnosti výskytu a dopadu na požitek ze hry. Přidělení závažnosti bude probíhat na základě tabulky 5.1.

	Nízká četnost	Střední četnost	Vysoká četnost
Zanedbatelný dopad	Zanedbatelná	Zanedbatelná	Nízká
Nízký dopad	Zanedbatelná	Nízká	Střední
Střední dopad	Nízká	Střední	Vysoká
Vysoký dopad	Střední	Vysoká	Velmi vysoká
Velmi vysoký dopad	Vysoká	Velmi vysoká	Velmi vysoká

Tabulka 5.1: Rozdělení závažnosti problému

5.3 Výsledky

Po skončení testování byly výsledky zpracovány a vyhodnoceny. Na základě vyhodnocení bylo zjištěno několik velmi závažných nedostatků. Seznam konkrétních nedostatků s jejich závažností je uveden v tabulce 5.2.

Popis problému	Závažnost
Nemožnost vycházet schody	Velmi vysoká
Nelze překonávat drobné překážky	Vysoká
Animace držení zbraně je zvláštní	Vysoká
NPC nepřecházejí správně mezi animacemi	Vysoká
Animace NPC nevhodně reagují na okolí	Střední
Nevhodné kolizní modely některých objektů	Střední
Chybí animace výstřelu	Střední
Nevhodná animace pití piva	Střední
Animace sezení posouvá postavy do objektu na kterém sedí	Střední
Hlavní postava nemůže sprintovat	Nízká
Při konverzaci se může postava otáčet	Zanedbatelná

Tabulka 5.2: Problémy v projektu s jejich závažností

Kapitola 6

Návrh úprav

V této kapitole budou navrženy úpravy k nápravě nejzávažnějších chyb, které byly určeny v sekci 5.3. První část se bude zabývat chybami v topologii. V druhé části se zaměříme na úpravu kostry. Zároveň v této části bude navržen i skinning, jelikož blízce navazuje na tvorbu kostry. Ve třetí části bude shrnuto, které animace je potřeba vytvořit. Poslední část se bude věnovat návrhu systému procedurálních animací a jejich kombinací.

6.1 Topologie

Topologie bude upravena, pokud bude při animaci vytvářet chyby. Pokud k chybám nebude docházet, není nutné topologii upravovat.

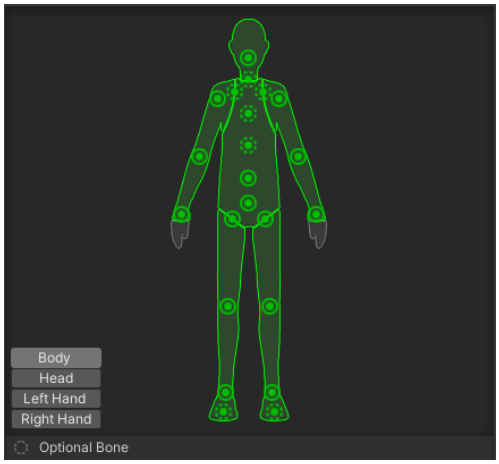
6.2 Kostra a skinnig

Vzhledem k použitému řešení je potřeba vytvořit novou kostru. Musí se tedy rozhodnout, jak detailní kostra bude, potažmo jak detailní pohyb bude umožňovat. Při tvorbě kostry je také potřeba vzít v potaz její další použití.

Hra je tvořena v Unity, a to podporuje práci s kostrou a vytváření systému inverzní kinematiky. Nicméně podpora kostry je omezená. Pro humanoidy je kostra omezena[23] na konkrétní počet kostí, ale i na jejich umístění, přičemž některé kosti nejsou nutně vyžadovány. Schéma podporované kostry v Unity můžete vidět na obrázku 6.1.

Se skinningem velmi pomůže Blender, jelikož dokáže automaticky vypočítat váhy vertexů v rámci jednotlivých kloubů na základě jejich vzdálenosti od kloubu. Výsledkem je napojení modelu na kostru pomocí metody míchání vrcholů.

[23] V případě použití odlišné kostry je nutné doprogramovat nový animátor.



Obrázek 6.1: Schéma kostry podporované Unity, kosti značené přerušovaným kolečkem nejsou povinné

Jedinou nevýhodou automatického dopočítání vah je to, že předpokládá deformovatelnost modelu. U modelu postavy se předpokládá, že bude deformovatelný, ale některé prvky oblečení postavy mohou být nedeformovatelné[24]. Pokud chceme, aby se některá část modelu nedeformovala, je nutné zkontrolovat nastavené váhy a zajistit, že tato část modelu je ovlivňována pouze jednou kostí.

6.3 Animace

Kvůli úpravě kostry bude nutné vytvořit všechny animace kompletně znovu. Vzhledem k tomu, že se jedná o zdlouhavý proces a v projektu je zapotřebí velké množství animací, nebude možné všechny animace dodělat v rámci bakalářské práce. Prioritizovány tak budou animace potřebné k otestování systému procedurálních animací.

Při vytváření animací bude využit systém inverzní kinematiky v Blenderu. Animace tak budou tvořeny kombinací přímé a inverzní kinematiky. Inverzní kinematika bude použita na vytvoření základní struktury animace a přímá kinematika na doladění detailů. Výsledné animace Blender převede na jednotlivé klíčové snímky, takže výsledkem budou animace přímé kinematiky.

U některých animací bude vyžadována cykličnost, jelikož je chceme přehrávat stále dokola. Cykličnosti dosáhneme tak, že nastavíme poslední snímek animace tak, aby plynule navazoval na první snímek animace. Získáme tak plynulý přechod mezi posledním a prvním snímkem při opakovaném přehrávání.

[24] Bráno pouze z hlediska animace postavy. Například kovový odznak šerifa se při pohybu šerifa nesmí ohýbat.

Animace budou vytvořeny in-place, jelikož o pohyb postavy se bude starat Unity nikoliv animace samotná. Pro animace to znamená, že se postava nebude hýbat z místa. Například při animaci chůze bude postava chodit na místě.

6.4 Systém procedurálních animací

Následně bude vše importováno do Unity. V Unity dojde k propojení kostry a modelu a bude vytvořen systém procedurálních animací, který bude rozhodovat, která animace se má kdy přehrát a jak bude animace poupravena, aby správně reagovala na okolí. Systém bude tvořen celkem čtyřmi částmi.

6.4.1 Character Controller

První důležitou částí bude komponenta Unity, **Character Controller**, který bude řídit pohyb postavy. Lze pomocí ní nastavit kolizní obálku postavy, maximální úhel povrchu, po kterém je postava schopna jít, a také maximální výšku překážky, na kterou lze vystoupit.

6.4.2 PlayerMovement skript

Druhou částí bude **PlayerMovement** skript, který zpracuje vstupy z klávesnice a myši a převede je na pohyb postavy ve hře. Zároveň tento skript bude řešit gravitaci postavy. Skript tak bude posílat příkazy k pohybu **Character Controlleru** a také bude nastavovat stav postavy v **Animatoru**. **PlayerMovement** není součástí Unity a je potřeba ho napsat.

6.4.3 Animator

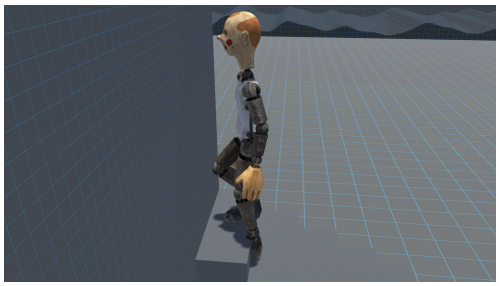
Třetí částí bude zmíněný **Animator**, který bude přehrávat animace na základě nastavených stavů. **Animator** je komponenta Unity a jedná se o stavový automat, jehož stavy jsou animace. Při přechodu mezi stavy se nastavují přechodové animace. Zároveň **Animator** dovoluje animace míchat.

6.4.4 IKFootControll skript

Poslední čtvrtou částí bude **IKFootControll** skript, který bude mít na starosti úpravu pozice nohou na základě vypočtených kolizí s objekty, po kterých postava zrovna jde. Vypočtenou pozici předá **Animatoru**, který ji zpracuje pomocí inverzní kinematiky.

Výsledný skript bude vycházet z algoritmu popsaného v bakalářské práci Oldřicha Linharta. V algoritmu [Linhart, 2020] se nejdříve vysílají dva paprsky z pozice kotníku (jeden kolmo k zemi a druhý ve směru normály) a poté je z normály a místa dopadu paprsků vypočítána pozice a rotace koncového efektoru.

Jedním z problémů tohoto algoritmu je, že výpočet pozice nohy provádí na základě paprsku z kotníku. Algoritmus neřeší, jak vypadá země pod špičkou nohy, ale ani uprostřed nohy. Což je i vidět na obrázku 6.2.



Obrázek 6.2: Zjišťování pozice v kotníku [Linhart, 2020]

Dále pak v konkrétních případech může být problémem vysílání paprsků, jelikož se zjišťuje pozice pomocí nekonečně malého bodu, ale podrážka boty má určitou velikost. Tento problém však v původní aplikaci nelze nasimulovat, jelikož neobsahuje objekty, na kterých by bylo možné problém ukázat.

Také si můžeme všimnout, že veškeré změny pozic se provádí skokově. Boky postavy se tak posunou z původní pozice do nové bez postupného přesouvání. Skok můžeme vidět mezi obrázky 6.3a a 6.3b. Tyto problémy se bude snažit algoritmus vyřešit.

(a) před pohybem

(b) po pohybu

Obrázek 6.3: Dva snímky následující po sobě zobrazující skokovou změnu pozice nohou i boků [Linhart, 2020]

Část II

Realizace

Kapitola 7

Implementace

Tato kapitola se zaměřuje na popis postupu při nápravě konkrétních problémů v projektu, které byly zjištěny v sekci 5.3. Nápravy budou provedeny za použití Blenderu a Unity, avšak popsané postupy lze použít i v jiných programech. V první části se zaměříme na úpravu kostry a provedeme skinning modelu. Druhá část se bude zabývat tvorbou animací. Animace budou vytvořeny kombinací inverzní a přímé kinematiky. Třetí část se bude věnovat úpravě topologie. Ve čtvrté části bude navržen systém procedurálních animací. Tato část bude hodně zaměřena na detekce kolizí nohou se zemí. Poslední část se bude věnovat testovací aplikaci.

7.1 Vytvoření kostry

Členění kostry jsme zvolili podle podpory Unity. Kostra je tak celkově tvořena z 49 kostí. Kořen kostry je zvolen v těžišti modelu, přičemž celková struktura kostry je zobrazena na obrázku 7.1a. Umístění kloubů je zvoleno tak, aby topologie v místě kloubu podporovala ohyb.

7 kostí je nepárových a slouží k ovládání páteře, krku a hlavy. 42 kostí je párových. K ovládání jedné nohy slouží 4 kosti. Ruce jsou ovládány celkem 17 kostmi. Počet kostí u rukou je tak vysoký především kvůli ovládání jednotlivých prstů[25], jelikož každý prst je ovládán 3 kostmi. Umístění jednotlivých kloubů je vidět na obrázku 7.1b, můžeme si všimnout vysoké koncentrace kostí v oblasti rukou.

Kostra je s modelem propojena pomocí automatického výpočtu vah. Animování pomocí kostry probíhá správně až na deformaci pevných objektů. Z toho důvodu byly u těchto objektů opraveny váhy. Jednalo se především o klobouk a šerifovu hvězdu.

[25] Animovaná postava má pouze 4 prsty.

(a) struktura kostry

(b) kostra na modelu

(c) systém IK na modelu

Obrázek 7.1: Kostra

Pro lepší animování postavy je ke kostře vytvořen systém ovládání za pomoci IK (obr. 7.1c). Ke kostře jsou přidány koncové efektory, které jsou použity pro výpočet polohy rukou a chodidel. Pro zamezení více možných výsledků při výpočtu IK jsou přidány navíc kontrolní body, které specifikují, na kterou stranu se mají klouby ohýbat, jelikož v Blenderu nelze pro IK použít metodu omezení kloubů. Systém IK na modelu je tvořen deformačními kostmi (šedá a světle žlutá), koncovými efektory (červené krychle) a kontrolními body (žluté koule).

7.2 Vytvoření animací

Ke kostře vytvořený systém inverzní kinematiky velmi zjednodušuje vytváření animací. Většina animací je vytvořena pomocí inverzní kinematiky a detaily jsou dotvořeny pomocí přímé kinematiky. Přímou kinematikou bylo upraveno například natočení v kotnících a zápěstí, také poloha a natočení prstů na rukou bylo vytvořeno pomocí přímé kinematiky.

Konkrétní vlastnosti animací jsou popsány v tabulce 7.1. Můžeme z ní vyčíst počty snímků, použitou techniku a zdali je animace cyklická. Všechny vytvořené animace jsou in-place.

Animace	Počet snímků	Technika	Cyklická
Standing still	21	FK	Ano
Idle	251	IK + FK	Ano
Walking	72	IK + FK	Ano
Aiming	41	IK + FK	Ano

Tabulka 7.1: Vlastnosti animací

7.3 Úpravy topologie

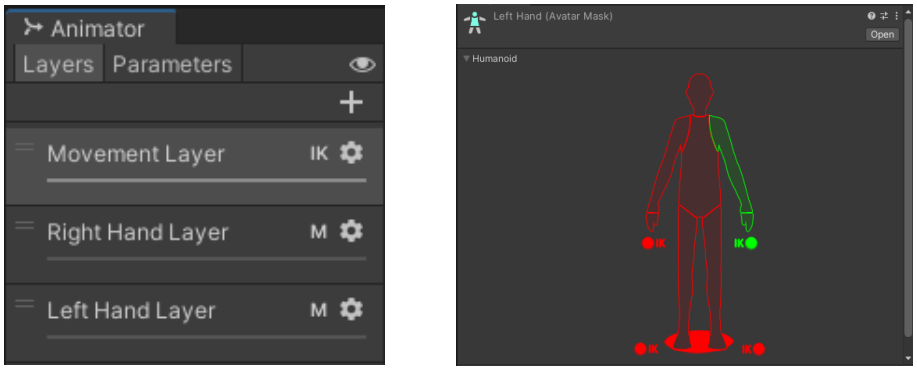
Topologie modelu se i přes své nedostatky chová při animování předvídatelně a nedochází k chybám. Z toho důvodu není nutné topologii nijak upravovat a bude použita stávající.

7.4 Systém procedurálních animací

Při popisu sytému procedurálních animací se zaměříme především na **Animator** a **IKFootControll** skript, jelikož jejich fungování je komplexnější a zbylé dvě části systému jsou triviální.

7.4.1 Animator

Výsledný animátor obsahuje tři vrstvy animací (obr. 7.2a). Vrstva **Movement Layer** je základní animační vrstva a zbylé vrstvy přepisují transformace, které provádí základní vrstva. Přepisování je prováděno na základě váhy vrstvy a přepisovány jsou pouze klouby zahrnuté v animační masce dané vrstvy (příklad takové masky je na obrázku 7.2b).



(a) vrstvy animací

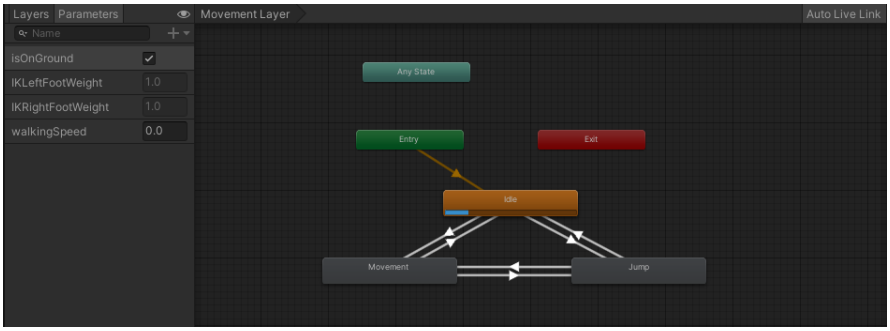
(b) animační maska levé ruky

Obrázek 7.2: Části animátoru

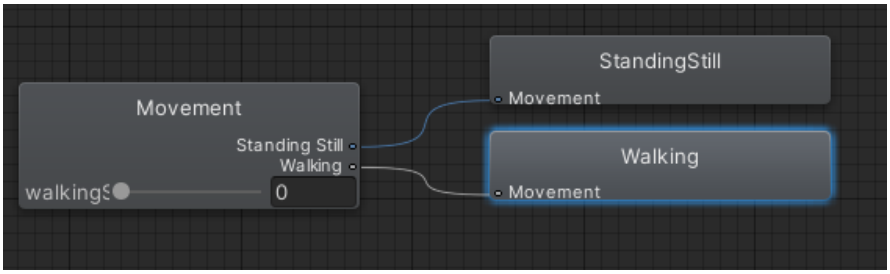
Základní vrstva je složitější, proto se jí budeme věnovat podrobněji. Tato vrstva obsahuje celkem tři animace (obr. 7.3). Avšak animace **Movement** je ve skutečnosti míchací strom, ve kterém dochází k míchání dvou animací (obr. 7.4) na základě rychlosti pohybu postavy. Mícháním animace stojící postavy a chodící postavy je možné vytvořit animace s libovolnou rychlostí[26] chůze. Důležitou poznámkou je, že animace **Walking** obsahuje křivky **IKRightFootWeight** a **IKLeftFootWeight**, které v konkrétním čase říkají v jaké výšce

[26]Pokud se do stromu přidá i animace běhu, je možné vytvářet animace libovolně rychle se pohybující postavy.

v rámci animace se nachází pravá a levá noha (0 značí minimální výšku a 1 maximální).



Obrázek 7.3: Hlavní animační vrstva řídící animace pohybu postavy



Obrázek 7.4: Míchací strom vytvářející animaci s různou rychlostí chůze

7.4.2 IKFootControll skript

Tento skript má za úkol upravovat výšku, ve které se nacházejí nohy a boky, na základě podkladu, po kterém se chodí. Skript lze rozdělit na tři fáze[27]:

1. Výpočet pozice nohou
2. Určení pozice boků na základě pozice nohou a posunutí boků
3. Nastavení inverzní kinematiky nohou

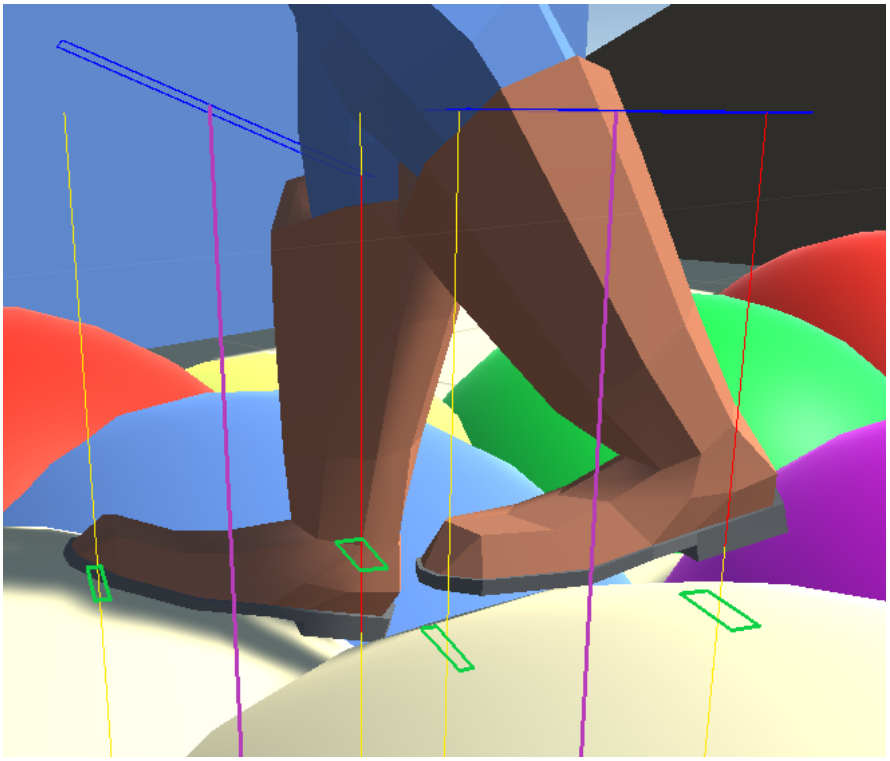
Výpočet pozice nohou

Výpočet pozice nohou probíhá v několika krocích (algoritmus 2). Jednotlivé kroky jsou také vizualizovány na obrázku 7.5.

1. Nejdříve jsou vyslány paprsky z kotníku a prstů směrem k zemi. Dráha je naznačena žlutými linkami a velikost paprsků je zobrazena jako zelené kvádry[28].

[27] Přičemž poslední dvě fáze lze provést v libovolném pořadí, jelikož nejsou na sobě závislé.
[28] Výška kvádrů je velmi malá, tudíž se mohou jevit jako obdélníky.

2. Ze získaných kolizí paprsků je vypočteno natočení chodidla vůči podkladu.
3. Následně je vyslán nový paprsek ze středu nohy. Jeho dráha je naznačena fialovou linkou. Získané natočení je využito k natočení modrého kvádru, který představuje velikost paprsku.
4. Výsledná kolize je použita k získání výšky koncového efektoru. Avšak jedná se o pozici ve středu nohy a je nutné ji přesunout zpět do kotníku.



Obrázek 7.5: Vizualizace algoritmu vypočítávajícího pozici nohou

Algoritmus 2: Výpočet pozice nohy

```
/* pozice kotníku Vector3 anklePosition
   pozice prstů u nohy Vector3 toesPosition
   rotace kotníku podle osy X z předchozího snímku float OARotationX */
```

Vstup : Vector3 anklePosition
Vector3 toesPosition
float OARotationX

```
// 1. Z polohy kotníku i z polohy prstů jsou vyslány boxcasty směrem
   k zemi
```

RaycastHit ankleHit;
RaycastHit toesHit;
BoxCast(anklePosition, ankleHit);
BoxCast(toesPosition, toesHit);

```
// Výpočet vektorů od kotníku k prstům
```

Vector3 ankleToToes = transform.parent.rotation * Vector3.forward * footLength;
Vector3 ankleToToesRotated = toesHit.position - ankleHit.position;

```
// 2. Z vektorů dopočítáno natočení kotníku k povrchu
```

Quaternion ankleRotation = FromToRotation(ankleToToes, ankleToToesRotated);

```
// Získáno natočení koncového efektoru s plynulým přechodem díky
   lineární interpolaci
```

ankleIKRotation = ankleRotation * transform.parent.rotation;
ankleIKRotation = Quaternion.Lerp(OFRotation, ankleIKRotation, footIKSpeed); OARotationX = ankleIKRotation.eulerAngles.x;

```
// 3. Vypočítán střed nohy a vyslán boxcast směrem k zemi natočený na
   základě získané rotace kotníku
```

Vector3 footCastPosition = (anklePosition + toesPosition) / 2;
RaycastHit footHit;
BoxCast(footCastPosition,footHit);

```
// 4. Získán koncový efektor
```

ankleIKPosition = footCastPosition - heelToToesRotated / 2;
ankleIKPosition.y -= footHit.distance + distanceToGround;

```
/* pozice koncového efektoru Vector3 ankleIKPosition
   rotace koncového efektoru Quaternion ankleIKRotation
   uložení rotace kotníku podle osy X pro další výpočet float
   OARotationX                                                    */
```

Výstup: Vector3 ankleIKPosition
Quaternion ankleIKRotation
float OARotationX

Výpočet pozice boků

Zbylé fáze jsou výrazně jednodušší než získávání pozic nohou. Způsob výpočtu pozice boků je podrobněji popsán v algoritmu 3. Pozice boků je určena na základě pozice nohy nacházející se níže. Následně jsou boky posunuty směrem do vypočtené pozice pomocí lineární interpolace, díky čemuž je změna pozice boků plynulejší a realističtější.

Algoritmus 3: Určení pozice boků na základě pozice nohou a posunutí boků

```
/* pozice boků na ose Y z předchozího snímku float OHPositionY
   pozice koncového efektoru levé nohy Vector3 LFIKPosition
   pozice koncového efektoru pravé nohy Vector3 RFIKPosition        */
```

Vstup : float OHPositionY
Vector3 LFIKPosition
Vector3 RFIKPosition

```
// Zjištění posunu boků od běžné pozice
float offset = Min(LFIKPosition.y, RFIKPosition.y) -
 normalHipsPosition;

// Výpočet pozice boků a posun do této pozice o malý krok za pomoci
   lineární interpolace
Vector3 NHPosition = bodyPosition + Vector3.up * offset;
NHPosition.y = Lerp(OHPositionY, NHPosition.y, hipsIKSpeed);

// Použití nové pozice boků a uložení výšky boků
bodyPosition = NHPosition;
OHPositionY = NHPosition.y;

/* uložení pozice boků na ose Y pro další výpočet float OHPositionY  */
```

Výstup: float OHPositionY

Nastavení inverzní kinematiky

Poslední fází je nastavení inverzní kinematiky nohou popsané algoritmem 4. Nejdříve je nastavena váha inverzní kinematiky při míchání animací. Následně je provedena lineární interpolace mezi současnou pozicí koncového efektoru a vypočtenou pozicí koncového efektoru. Následně je nastavena nová pozice a rotace koncového efektoru.

Algoritmus 4: Nastavení inverzní kinematiky nohou

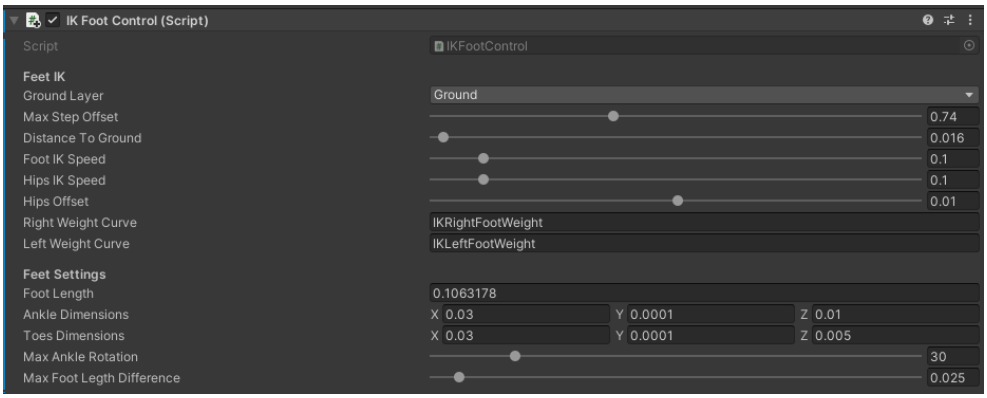
```
/* koncový efektor kotníku AvatarIKGoal ankle
   pozice koncového efektoru Vector3 IKPosition
   rotace koncového efektoru Quaternion IKRotation
   název váhové funkce string weightCurve
   pozice koncového efektoru na ose Y z předchozího snímku float
   OHPositionY                                                    */
```

Vstup : AvatarIKGoal ankle
Vector3 IKPosition
Quaternion IKRotation
string weightCurve
float OAPositionY

```
// Nastavení jak velký podíl na pozici a rotaci nohou má inverzní
   kinematika
SetIKPositionWeight(ankle, 1);
SetIKRotationWeight(ankle, GetFloat(weightCurve));

// Získání současné pozice koncového efektoru
Vector3 anklePosition = GetIKPosition(ankle);

// Posun mezi vypočtenou pozicí a současnou pozicí pomocí lineární
   interpolace, rotaci bylo nutné interpolovat dříve, proto se zde
   neprovádí
float y = Lerp(OAPositionY, IKPosition.y, footIKSpeed);
anklePosition.y += y;
OAPositionY = y;

// Použití získané pozice a rotace
SetIKRotation(ankle, IKRotation);
SetIKPosition(ankle, anklePosition);

/* uložení pozice koncového efektoru Y pro další výpočet float
   OAPositionY                                                    */
```

Výstup: float OAPositionY

Výsledný skript obsahuje několik nastavitelných hodnot (obr. 7.6). Skript je napsaný tak, aby ho bylo možné použít u libovolné humanoidní postavy, proto také obsahuje tolik nastavitelných hodnot. Konkrétní popis, co která proměnná představuje, je v tabulce 7.2.



Obrázek 7.6: Přehled nastavení IKFootControll skriptu

Název	Datový typ	Popis	Rozsah
Ground Layer	LayerMask	Vrstva s níž koliduje boxCast	x
Max Step Offset	float	Vzdálenost mezi maximální a minimální výškou, ve které se může noha nacházet (polovina této hodnoty určuje maximální výšku, do které může noha vykročit)	0 – 2
Distance To Ground	float	Vzdálenost mezi kotníkem (kostí) a zemí	0 – 1
Foot IK Speed	float	Rychlost změny pozice a rotace nohou	0 – 1
Hips IK Speed	float	Rychlost změny pozice boků	0 – 1
Hips Offset	float	Hodnota upravuje původní výšku boků	-1 – 1
Right Weight Curve	string	Název váhové křivky inverzní kinematiky pro pravou nohu (název se musí shodovat s názvem v animátoru)	x

Tabulka 7.2: Přehled nastavitelných hodnot IKFootControll skriptu

Název	Datový typ	Popis	Rozsah
Left Weight Curve	string	Název váhové křivky inverzní kinematiky pro levou nohu (název se musí shodovat s názvem v animátoru)	x
Ankle Dimensions	Vector3	Rozměry boxCastu z kotníku	x
Toes Dimensions	Vector3	Rozměry boxCastu z prstů	x
Max Ankle Rotation	float	Maximální rotace kotníku na ose X ve stupních	0 - 180
Max Foot Legth Difference	float	Maximální rozdíl ve vzdálenosti mezi vypočtenými polohami prstů a kotníku	0 - 0.5

Tabulka 7.2: Přehled nastavitelných hodnot IKFootControll skriptu (pokračování)

7.5 Testovací aplikace

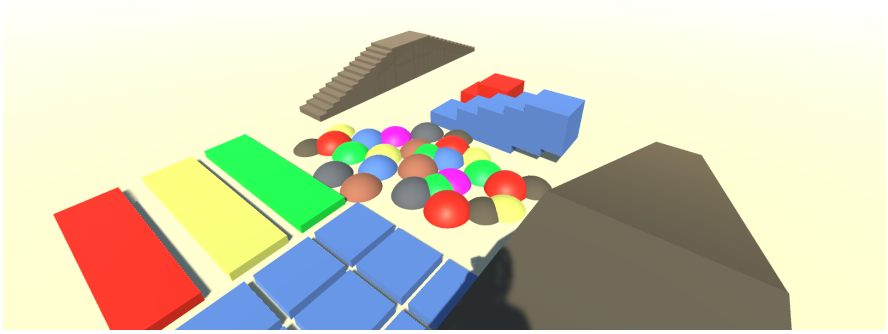
Implementaci výsledného systému do hry se bohužel nepodařilo dokončit v očekávaném termínu. Hlavní důvodem byla nutnost předělat systém pohybu a animací, jelikož původní systém nefungoval správně. Zároveň včasnému dokončení brání složitost projektu Western Town a zakomponování nového systému pohybu a animací je časově náročnější, než se původně předpokládalo. Z toho důvodu vznikla testovací aplikace, ve které je možné nový systém testovat a zhodnotit jeho chování.

Po spuštění aplikace se uživatel nachází v jednoduchém menu, ze kterého lze zobrazit ovládání aplikace, spustit testovací scénu a také aplikaci ukončit. Do tohoto menu se lze ze spuštěné testovací scény dostat pomocí klávesy escape. Konkrétní popis všech ovládacích prvků je v tabulce 7.3, nebo přímo v aplikaci.

Akce	Ovládání
Chůze	šipky / W, A, S, D
Skok	mezerník
Rozhlížení nahoru a dolů	pohyb myší dopředu a dozadu
Rozhlížení doprava a doleva	pohyb myší doprava a doleva
Míření pravou rukou	pravé tlačítko myši
Míření levou rukou	levé tlačítko myši
První pohled / pohled na nohy	F1
Návrat do menu	ESC

Tabulka 7.3: Ovládání aplikace

Testovací scéna (obr. 7.7) je velmi zjednodušená, ale výběrem objektů se snaží nasimulovat co nejvíce různých situací, se kterými se může postava ve hře setkat. Zajímavými objekty jsou tři vyvýšené plošiny, na kterých lze testovat animace různě rychlé chůze (červená - chůze starce, žlutá - normální chůze, zelená - rychlá chůze).



Obrázek 7.7: Scéna testovací aplikace

V testovací aplikaci je možné skákat, ale postava nemá animaci skoku. Možnost skákání je povolena čistě pro testovací účely, cílový projekt skákání nemá implementováno, ale zvažujeme jeho implementování do budoucna.

7.6 Zhodnocení systému procedurálních animací

Podstatným rozdílem od Linhartova algoritmu je schopnost správně přizpůsobit nohu více než jednomu objektu, jak můžeme vidět na obrázku 7.8. Dalším rozdílným přístupem je použití kombinace několika boxcastů místo raycastu, díky čemuž dává systém lepší výsledky (obr. 7.9). Díky použití lineárních interpolací jsou animace také plynulejší než v původním algoritmu.

(a) s inverzní kinematikou

(b) bez inverzní kinematiky

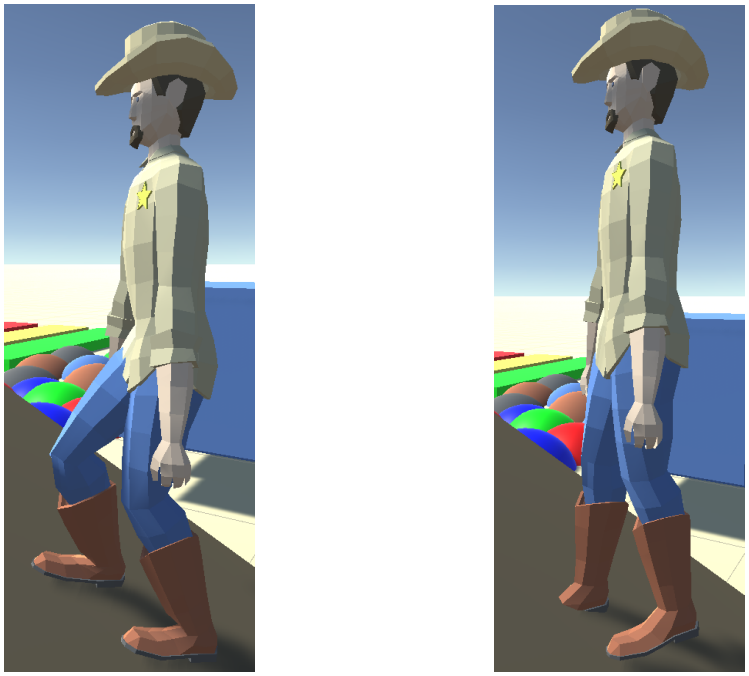
Obrázek 7.8: IK různé objekty

(a) raycast [Linhart, 2020]

(b) boxcast

(c) bez inverzní kinematiky

Obrázek 7.9: IK detekce raycast vs. boxcast



(a) s inverzní kinematikou (b) bez inverzní kinematiky

Obrázek 7.10: IK na plošině

Kapitola 8

Testování

První část této kapitoly se bude zabývat navržením testů ke zjištění, jestli nový systém je lepší než stávající. V druhé části bude popsáno jak bude testování probíhat. V poslední části bude provedeno vyhodnocení výsledků uživatelského testování.

8.1 Navržení testů

Vzhledem k napraveným chybám můžeme nyní testy rozdělit do dvou kategorií:

- Animace hlavní postavy
- Interakce s okolím

Stejně jako v prvním testování nejsou testy konkrétní jelikož nám jde o celkový estetický dojem. Nicméně v první kategorii je po uživateli požadováno přehrání všech vytvořených animací a jejich zhodnocení. Včetně různých rychlostí chůze.

V druhé kategorii je po uživatelích požadováno vyzkoušení všech překážek. Důraz je kladen na zhodnocení chůze po schodech, plošině a dalších nerovných površích.

8.2 Průběh testování

Bohužel situace se od prvního testování nezměnila a tak testování znovu proběhne kompletně online formou. Uživatele budou testy provedení za pomoci dotazníku, do kterého na konci testování vyplní výsledky testů. Uživatelé budou v průběhu testu v hovoru s moderátorem testu, kdyby při testování nastali nějaké problémy.

8.3 Výsledky

Testování ukázalo, že uživatelé jsou velmi spokojeni s novým systémem procedurálních animací. Z výsledků je patrné, že překážky se nyní překonávají snáze a očekávatelněji.

Avšak u animací byl nalezen problém. V určitých situacích inverzní kinematika způsobuje při klidové animaci, že postava tře nohami o sebe. Po bližším přezkoumání bylo zjištěno, že kolena se ohýbají k sobě a tento problém bude ještě dále řešen. V momentálním stavu není jasné, zdali za to může systém inverzní kinematiky, nebo animace samotná.

Závěr

V této práci byl proveden průzkum možných přístupů k reprezentaci 3D modelů a jejich animování. Způsoby animace byly porovnány, a nakonec bylo zjištěno, že nejlepší ze současných přístupů pro animaci postav je animace pomocí kosterní soustavy. Dále byly řešeny dva rozdílné pohledy na samotný pohyb postavy, a to inverzní a přímá kinematika. Bylo zjištěno, že v rámci animace nelze ani jeden ze zmíněných principů označit za lepší a je nutné volit přístupy na základě jejich použití.

Hra Western Town na konci předmětu BI-VHS používala pouze přímou kinematiku, která nedovoluje přizpůsobit animace okolí a nedovoluje také překonávat některé drobné překážky. Pokud odhlédneme od použití čistě přímé kinematiky, tak animace lze ve většině případů považovat správné, nicméně u některých dochází k chybám a jejich náprava není snadná z důvodu použitého přístupu.

V rámci analýzy současného řešení bylo provedeno uživatelské testování, které potvrdilo zjištěné nedostatky. Výsledky testování byly dále využity k prioritizaci řešení zjištěných problémů. Při vytváření prioritizovaného seznamu oprav bylo přihlédnuto k posloupnosti jednotlivých kroků pracovního postupu.

Následně byly opraveny nejzásadnější zjištěné problémy. Pro model postavy tak byla vytvořena kosterní soustava, která umožňuje libovolně model pózovat. Zároveň byl ke kostře vytvořen systém pro inverzní kinematiku. V rámci práce byly opraveny nejdůležitější animace, konkrétně tedy: animace chůze, idle animace, animace stání a animace míření pravou a levou rukou. Při vytváření animací byla použita metoda klíčových snímků a jednotlivé snímky byly vytvořeny pomocí kosterní soustavy. Následně byl model s animacemi importován do nového testovacího projektu v Unity. V testovací aplikaci byl vytvořen systém procedurálních animací založený na principu inverzní kinematiky. Animace chůze se tak nyní již přizpůsobuje okolním objektům a také dovoluje překonávat výrazně vyšší překážky bez problémů.

Na závěr práce bylo provedeno ještě jedno uživatelské testování se záměrem zjistit, jak provedené změny přispěly k celkovému dojmu ze hry. Výsledky provedeného testování potvrdily, že změny zlepšují estetický požitek ze hry.

Jelikož nebyl kompletně zpracován celý seznam oprav, tak je plánováno do budoucna veškeré nenapravené problémy napravit. Dále také budou opraveny ostatní animace. A vzhledem k plánovanému rozšiřování hry bude nutné přidat další animace. V rámci optimalizace rychlosti výpočtu kolizí v systému inverzní kinematiky budou také zjednodušeny kolizní modely, u kterých tak doposud nebylo učiněno.

Literatura

[Adobe, 2021a] Adobe (2021a). Mixamo [software]. [cit. 2021-4-14]. Dostupné z: https://www.mixamo.com/#/.

[Adobe, 2021b] Adobe (2021b). Triangulating before baking – substance bakers. [cit. 2021-3-18]. Dostupné z: https://docs.substance3d.com/bake/triangulating-before-baking-159451841.html.

[Agache, 2016] Agache, P. (2016). *Langer's Lines*, pages 1–6. Springer International Publishing, Cham. Dostupné z: https://doi.org/10.1007/978-3-319-26594-0_154-1.

[Anonymous, 2021] Anonymous (2021). Polygon mesh. [cit. 2021-3-18]. Dostupné z: https://en.wikipedia.org/wiki/Polygon_mesh.

[Autodesk.Help, 2019] Autodesk.Help (2019). Graph editor curves menu. [cit. 2021-3-22]. Dostupné z: http://help.autodesk.com/view/MAYAUL/2019/ENU/?guid=GUID-619CAE9C-107D-4504-A769-227DDC847153.

[Blender Documentation Team, 2021] Blender Documentation Team (2021). F-curves. [cit. 2021-3-22]. Dostupné z: https://docs.blender.org/manual/en/latest/editors/graph_editor/fcurves/introduction.html.

[CG Futures, 2018] CG Futures (2018). How to create exact and technical topology workflow for the blizzard animators. [cit. 2021-4-10]. Dostupné z: https://www.youtube.com/watch?v=ZiYEO49B768.

[Chopine, 2011] Chopine, A. (2011). Chapter 3 - polygons: How 2d becomes 3d. In Chopine, A., editor, *3D Art Essentials*, pages 21–44. Focal Press, Boston. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780240814711100037.

[Diehard, 2020] Diehard, D. (2020). Intro to 3d art pixar style topology part 2 patches. Dostupné z: https://www.youtube.com/watch?v=X5Jksu5Z8AA.

[Engel, 2018] Engel, W. (2019;2018;). *GPU Pro 360 Guide to Mobile Devices*. A K Peters/CRC Press, Milton.

[Gregory and Lemarchand, 2014] Gregory, J. and Lemarchand, R. (2014). *Game engine architecture*. CRC Press, Boca Raton, 2nd edition.

[Haindl and Richtr, 2013] Haindl, M. and Richtr, R. (2013). Dynamic texture enlargement. In *Proceedings of the 29th Spring Conference on Computer Graphics*, pages 5–12.

[Haunt_House and ideasman42, 2013] Haunt_House and ideasman42 (2013). Why should triangle meshes be avoided for character animation? [cit. 2021-4-9]. Dostupné z: https://blender.stackexchange.com/questions/2931/why-should-triangle-meshes-be-avoided-for-character-animation.

[Hughes, 2014] Hughes, J. F. (2014). *Computer graphics: principles and practice*. Addison-Wesley, Upper Saddle River, 3rd edition.

[Jeppsson, 2000] Jeppsson, D. (2000). Realtime character animation blending using weighted skeleton hierarchies. [cit. 2021-3-20]. Dostupné z: https://fileadmin.cs.lth.se/graphics/theses/reports/jeppsson_masterthesis.pdf.

[LearnBones, 2021] LearnBones (2021). Type of joints in human body. [cit. 2021-4-22]. Dostupné z: https://cz.pinterest.com/pin/342977327844727029/.

[Linhart, 2020] Linhart, O. (2020). Animace humanoidního 3D modelu. Bakalářská práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií.

[Maxon, 2020] Maxon (2020). Keys area. [cit. 2021-3-22]. Dostupné z: https://help.maxon.net/r23/en-us/#html/10608.html.

[Microsoft, 2011] Microsoft (2011). Keyframe animtion. [cit. 2021-3-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/cc189038(VS.95).aspx.

[Mukundan, 2012] Mukundan, R. (2012). *Advanced Methods in Computer Graphics: With Examples in OpenGL*. Springer London, Limited, London, 2012 edition.

[Paquette, 2013] Paquette, A. (2013). *An Introduction to Computer Graphics for Artists*. Springer, London. Dostupné z: https://doi.org/10.1007/978-3-319-17885-1_1259.

[Parent, 2012] Parent, R. (2012). *Computer Animation: Algorithms and Techniques*. Elsevier Science & Technology, San Francisco.

[Phyllis, 2017] Phyllis, A. (2017). *Spatial Data Transfer Standard (SDTS)*, pages 1981–1992. Springer International Publishing, Cham. Dostupné z: `https://doi.org/10.1007/978-3-319-17885-1_1259`.

[Power, 2012] Power, K. (2012). 3d object representations. [cit. 2021-3-14]. Dostupné z: `http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/meshes/polygon_meshes.html`.

[Richtr and Haindl, 2015] Richtr, R. and Haindl, M. (2015). Dynamic texture editing. In *SCCG*, pages 133–140.

[Richtr and Haindl, 2018] Richtr, R. and Haindl, M. (2018). Dynamic texture similarity criterion. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 904–909. IEEE.

[Selby, 2013] Selby, A. (2013). *Animation*. Laurence King Publishing, London, 1 edition.

[Sito, 2013] Sito, T. (2013). *Moving Innovation: A History of Computer Animation*. The MIT Press, Cambridge.

[Team Fugl, 2017] Team Fugl (2017). Fugl: Meditative bird flying game. [cit. 2021-3-13]. Dostupné z: `https://store.steampowered.com/app/643810/Fugl/`.

[Žára, 2005] Žára, J. (2005). *Moderní počítačová grafika*. Computer Press, Brno, 2. přeprac. a rozš. vyd. edition.

Příloha A

Seznam použitých zkratek

2D Dvou-dimenzionální

3D Tří-dimenzionální

AI Umělá inteligence, z anglického artificial intelligence

BI-VHS Virtuální herní světy

CT tomografie, z anglického computed tomography

ČVUT České vysoké učení technické v Praze

FIT Fakulta informačních technologií

FK přímá kinematika, z anglického forward kinematics

FPS snímky za sekundu, z anglického frames per second

GPU grafický procesor, z anglického graphics processing unit

IK inverzní kinematika, z anglického inverse kinematics

MRI magnetická rezonance, z anglického magnetic resonance imaging

SWAT speciální zásahová jednotka, z anglického Special Weapons and Tactics

NPC nehratelná postava, z anglického non-playable character

Příloha B

Obsah přiloženého flash disku

- `readme.txt` ..... stručný popis obsahu flash disku
- `exe` ..... adresář se spustitelnou testovací aplikací
- `src` ..... adresář se zdrojovými soubory
  - `impl` ..... zdrojové soubory implementace
    - `unity` ..... zdrojové soubory unit
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF