Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Simulace stárnutí pro aplikace virtuální reality

Plzeň 2014 Jiří Janeček

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 25. června 2014

Jiří Janeček

Abstract

Increasingly, it happens that some advertising, entertainment and forensic applications require the creation of interactive simulations of aging of human head in a 3D environment. Aging has a common solution in the 2D photos, some procedures are applicable to 3D but many of new problems arise too.

This work deals with the particular wrinkle modeling techniques due to aging and also it solves the aging of the hair as a minor issue. Formation of wrinkles involves techniques modifying the object surface in order to increase the amount of details by using of existing hardware.

The result of this work is an application enables the simulation in real-time. The proposed models and algorithms of aging are described and tested in order to find and verify the possible limitations of the procedures used.

Abstrakt

Stále častěji se stává, že některé reklamní, zábavní či kriminalistické aplikace vyžadují tvorbu interaktivních simulací stárnutí lidské hlavy ve 3D prostředí. Stárnutí je běžně řešeno na 2D fotografiích, některé postupy jsou použitelné i pro 3D, ale vyvstává také řada nových problémů.

Tato práce se zabývá zejména technikami modelování vrásek vlivem stárnutí a dále je okrajově řešeno i stárnutí vlasů. Tvorba vrásek zahrnuje postupy měnící povrch objektu tak, aby se zvýšilo množství detailů s využitím možností současného hardwaru.

Výsledkem práce je aplikace, umožňující provádět simulaci v reálném čase. Použité modely a algoritmy stárnutí jsou popsány a otestovány s cílem nalézt a ověřit možná omezení použitých postupů.

Obsah

1 Úvod

Na lidské hlavě je pozorovatelných mnoho znaků stárnutí. Bezpochyby nejdůležitějším jevem je stárnutí kůže a vlasů. Vrásky jsou nejznámějším a nejviditelnějším projevem stárnutí kůže. Není to však jediný znak. Můžeme sledovat také změny barvy pokožky, které jsou zejména spojeny s úbytkem některých látek nalézajících se pod kůží. Prakticky podobné projevy lze pozorovat také u vlasů. V obou případech se stárnutím ztrácí barevnost a pružnost.

Při modelování se lze vydat dvěma cestami. Generaci stavů hlavy ve vybraných klíčových fázích můžeme vytvořit algoritmem využívající matematický model, nebo přenechat práci umělci. První přístup nabízí automatické zpracování, zatímco druhý přístup poskytuje větší míru uživatelské kontroly nad výsledkem.

Tento druhý pohled vede cestou fyzikálně motivovaného modelu. Například pro vrásky je možné sestavit zjednodušující model svalů jako oblasti vlivu na kůži s parametry směru, rozměrů a stlačení kůže. Parametr stlačení by měl být přímo úměrný vychýlení od normální výšky kůže ve vybraném bodě. Posledním aspektem je vhodná volba profilu vrásky (často funkční předpis), který po periodickém navázání působí co nejpřirozeněji. Profil může být tedy zvolen jako empiricky stanovená funkce, či jako ručně nakreslená křivka (například zadáním uzlových bodů). Aplikací modelu (provedeme fyzikální simulaci) dostaneme výsledek, který lze přímo zobrazovat (přímá generace časového vývoje), nebo můžeme získané výsledky dále vzájemně kombinovat, například interpolací.

Výhodou programu simulujícího fyzikální vlastnosti obličeje je možnost úpravy parametrů jednotlivých součástí, tak aby vyhovovaly individuálním případům. Tento proces může být také automatický a může nabídnout rozumné nastavení. Každý obličej však vyžaduje individuální nastavení.

Aplikovatelnost této problematiky je poměrně široká a nebylo by snadné obsáhnout všechny oblasti, zejména proto, že požadavky se mohou lišit. V této práci se předpokládá použitelnost výsledků pro kriminalistickou praxi, zábavní průmysl a nebo také reklamní průmysl (pro farmaceutické společnosti).

2 Cíle

Cílem práce je navrhnout a implementovat nástroj na simulaci efektu stárnutí na 3D modelu. Program by měl být hlavně modelovacím nástrojem umožňujícím zejména nastavení parametrů souvisejících s tvorbou vrásek, změnou barvy a rozmístění vlasů. Uživatel může nastavit potřebné parametry obličeje a změnou časové roviny sleduje jeho změny.

V poslední fázi projektu se předpokládá integrace vytvořeného programového řešení do existujícího projektu na kterém pracuje Ing. Petr Martínek.

2.1 Globální modifikace tvaru

Tvar lebky se deformuje jednak vývojem podle genetické informace, tak vlivem gravitace. Článek (Ramanathan et al., 2009) se zmiňuje o modelu nazývaném *cardiodal strain model*, který vyjadřuje vývoj tvaru lebky v čase funkčním vztahem. Byl vysloven předpoklad, že lidská hlava se chová jako pružná nádoba plněná kapalinou. Tlak vyvinutý kapalinou a gravitační síla se po delším čase projeví nevratnou deformací.

Ke změně tvaru hlavy lze použít i standardní techniky morfování, například síťový warping v prostoru. Pokud je možné získat záznam měření o změně polohy vybraných bodů na lidské hlavě, například na základě antropometrických studiích, lze stanovit posuny příslušných řídících bodů mřížky.

2.2 Lokální změny tvaru

Další oblast stárnutí je projevem změny svalové hmoty. To jsou takové změny na tvaru tváře, kde nerozhoduje celková deformace hlavy a současně se neuvažují projevy stárnutí kůže.

- Některé části obličeje nabudou a jiné se zase zploští (zhubnou).
- Ztráta pružnosti kůže. Tento jev nastává typicky tam, kde se pohybují svaly – oblasti vzniku vrásek.

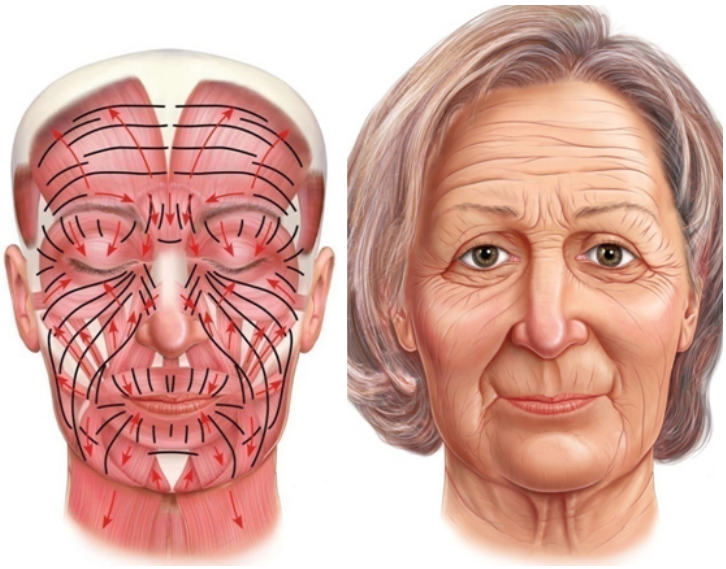
3 Simulace kůže

Kůže jednou z částí těla, na které se jednoznačně projevuje vliv stárnutí. Kůže je mnohovrstevnatý (kompozitní) materiál, který plní řadu funkcí nezbytných pro naše přežití. Vlastnosti vrstvené struktury se projevují zejména v optickém chování (například článek (Igarashi et al., 2005)) a je nutno podotknout, že simulace barvy, odrazivosti a lámavosti světla je velmi složitý úkol, který z pohledu stárnutí není tak klíčový, takže se tento dokument bude tomuto tématu věnovat jen velmi okrajově. Klíčovou částí této práce bude hledání vhodného modelu na modelování vrásek, kde se ve většině případů vystačí se zjednodušením představy kůže jako jeden vazko-elastický materiál, pevně uchycený k relativně pevné, nepohyblivé části těla, např. kost nebo sval.

3.1 Fyzikální vlastnosti kůže

Mechanické vlastnosti materiálu jsou mimo jiné ovlivněny obsahem některých chemických látek podporujících například pružnost. Tato ztráta pružnosti způsobená mechanickým opotřebením a úbytkem podpůrných chemických látek, zapříčiňuje zejména permanentní výskyt vrásek. Vznik vrásek, bez uvažování věku, je podmíněn opakovaným stlačováním a protahováním kůže. Čas je pouze ukazatel jaký je podíl permanentních vrásek. Protože je předpokládán statický model hlavy, uvažují se jen vrásky vzniklé vlivem stáří. Pohybové aspekty částí tváře vniklé stahem svalů jen nepřímo ovlivňují vrásky stárnutí, pokud použitý model je tzv. svalově orientovaný. Na obrázku 3.1 je příklad rozmístění svalů a vrásek na lidském obličeji. Vrásky často vznikají deformací kůže *kolmo na směr smrštění*, popř. protažení svalů.

Z lékařských poznatků je známo, že v průběhu času kůže ztratí pružnost a protáhne se podle toho, jaká je její tloušťka (za předpokladu, že je kůže všude stejného složení) = tloušťka kůže se může lokálně nastavit podle černobílé textury. Tloušťka kůže může ovlivňovat místa vzniku vrásek, frekvenci zvrásnění kůže (tlustá kůže bude ohýbána s menší frekvencí, tenká kůže bude mít jemné vrásky), odolnost proti již zmíněnému vlivu gravitace, změnu optických parametrů kůže (jiná barva, pohltivost, průhlednost).



Obrázek 3.1: Ukázka rozmístění obličejových svalů a s nimi spojené vrásky (vlevo). Obrázek vpravo ukazuje výsledek zobrazení odpovídajících vrásek.

3.1.1 Optické vlastnosti kůže

Existuje řada metod ke stínování povrchu kůže a řada referencí k nalezení v souvislosti s vývojem herních enginů. Zde se setkávají jednak metody velmi realistické (např. NVIDIA Human Head) a metody které jsou aktivně používány např. v řadě herních enginů jako jsou CryEngine (společnost Crytek), Frostbite od DICE a Source engine od spol. Valve. Postupy v článku (Sander et al., 2004) jsou příkladem řešení problému vrásek pro herní aplikace.

V herních aplikací se často vystačí s některými aproximacemi optického chování materiálu, které je jinak ve svém fyzikálním základu velmi těžko řešitelné, pokud je potřeba provádět výpočty v reálném čase. Například kůže se chová jako polopropustný materiál. Světlo přicházející skrz objekt je uvnitř objemu rozptýleno a opouští objekt v jiném směru než původní směr, ve kterém světlo do objektu vstupovalo. Popis vlastností z hlediska fyzikální reality je například uveden v článku (Igarashi et al., 2005). Příklad aproximací vhodných právě pro herní účely popisuje (Green, 2004).

Barva kůže se mezi věkovými epochami liší, např. starší osoby mají často tvář více bledou než mladší osoby. Simulace změny barvy v černobílé škále, lze realizovat stejně jako u barvy, vzhledem k tomu, že percepční schopnost v černobílé škále je silnější než v barevné škále.

3.2 Simulace vrásek

Nejprve je třeba zjistit, kde vrásky nejvíce vznikají, například od krajů úst, okraj očí, čelo, atd. Rostoucí věk umocňuje vliv označených oblastí na výšku vrásek (při představě vrásky jako signálu můžeme značit jako amplitudu). Dále je třeba rozhodnout o tvaru vrásek. Druhý krok musí vypočítat výšky vrásek ve všech bodech objektu, které respektují místa označená pro tvorbu vrásek. S informací o výšce vrásky, lze rekonstruovat povrch a ten zobrazovat (oblast vizualizace vrásek).

Některé metody vytvoří abstrakci celého povrchu kůže jako soustavu hmotných bodů propojených pružinami, kde následuje výpočet obdobný metodám konečných prvků. Častou vlastností soustav bodů je, že posun jednoho bodu se musí projevit vychýlením sousedních bodů, což vynucuje přepočtení výsledku i na sousedech. Uvedený postup a další metody jsou zmíněny v (Nealen et al., 2006). Výhoda této metody bude ve velké realističnosti a velké míry automatičnosti. Nevýhodou je větší výpočetní náročnost.

Velice odlišný přístup, s velkým podílem uživatelského zásahu popisuje článek (Bando et al., 2002). Zde se předpokládá přímé zadání vrásky (brázda s nejnižší výškou) jako křivky dané řídícími body. Řešeným problémem je, jak informaci o tvaru brázdy převést na prostorový model. Tento postup je méně výpočetně náročný a zároveň vyžaduje vyšší podíl uživatelského zásahu.

Další skupina postupů simulace vrásek využívá funkčního předpisu konstruovaného tak, aby se co nejlépe odpovídal vizuální předloze a zároveň využíval některé fyzikální vlastnosti. Příkladem uvedeného postupu může být model popsaný v (Zhang – Sim, 2005). Zde se předpokládá zadání polohy svalů (vrcholy modelu) a jejich vlastností, z nichž lze odvodit odpovídající oblasti vlivu tvorby vrásek.

3.2.1 Modelování tvaru vrásek

Existují dva druhy vrásek: *velké* a *jemné*. Jemné vrásky jsou například póry a strukturování kůže, které pozorujeme například jako otisky prstů. Velké vrásky jsou projevem pohybu svalů a změnou objemu podkožní hmoty.

V herním průmyslu je často oblíbený postup se vstupem hotových vrásek v textuře, kde se může jednat o vrásky velké, tak ji jemné. Tento přístup

je řešen například v (Oat, 2007). Textury se prolínají podle vyznačených oblastí obličeje v oddělených vrstvách. Provádí se alfa blending (prolínání) mezi všemi vrstvami s příslušnými maskami pro každou oblast. Článek (Reis et al., 2008) navíc používá masky na zobrazování vrásek na animovaném objektu podle aktivace geometrických částí a souvisejících vrásek.

Článek (Zhang – Sim, 2005) velmi podrobně popisuje model svalů určený k modelování vrásek vyjadřujících výraz obličeje. Model je primárně určen pro animaci, ovšem lze jeho část použít i ke generaci statického objektu. Zde se modelují tři typy svalů, které jsou charakteristické pro každou část obličeje. Například velmi odlišně se chovají svaly kolem očí a jinak se zase projevují svaly na čele. Předpokládáme, že souvislost svalů a vrásek je spolu úzce spojena a v tomto kontextu se spíše mluví o typu svalu než o typu vrásek. Směr vrásek je vždy odvozen podle modelu svalů reprezentovaným dvěma úchytnými body a velikostí oblasti vlivu.

Konvolucí dvou signálů můžeme přidat další signál představující vrásky (vrásky můžeme chápat jako šum z pohledu zpracování signálů). Uvedený postup je inspirován článkem (He et al., 2006). Teoreticky je možné navrhnout profil vrásky pomocí křivky (spline, bezier, apod.), kterou lze konvolucí přenést na libovolnou trojúhelníkovou plochu. Poté je nutné křivku diskretizovat a v diskrétní podobě provést konvoluci na povrchu trojúhelníkové sítě. Kernel vrásky bude dvojrozměrná funkce, kterou vytvoříme například vytažením profilu na plochu. Neměl by být problém míchat libovolné množství signálů (představujících vrásky), nicméně nesmíme zapomenout na interference (může dojít k násobení amplitudy nebo ke kompletní destrukci kmitů), protože často se na tváři setkávají různé typy vrásek s různým směrem, amplitudou a fází. Nelze opomenout, že metoda je velmi vzdálená fyzikální podstatě vrásek a tudíž nelze s touto metodou počítat pro opravdu věrné simulace.

Vrásky na trojúhelníkové síti s vychýlením vrcholů směru normály jsou dalším používaným postupem. Výhoda realizace simulace vrásek na děleném povrchu je hlavně v její nezávislosti na metodě vykreslování. Nevýhodou je bezpochyby větší výpočetní a programátorská náročnost. Opět zde jde řídit rozmístění vrásek podle předem namalované textury nebo generací deformace přímo na síti. Pro trojúhelníkovou síť se právě často počítá stlačení ve vybraných vrcholech a amplituda vrásek potom odpovídá stlačení. Příkladem rozboru tohoto přístupu nabízí článek (P. Volino, 1999).

3.2.2 Rendering vrásek

Jedním možným postupem kreslení vrásek je přímé kreslení trojúhelníkové sítě a druhým postupem je skupina technik, vycházejících ze simulace nerovností, které vychýlením normály k povrchu objektu přidají objektu další detaily. Například technika zmíněná v článku (Blinn, 1978) je jedna z prvních prací v této oblasti. Často jsou k nalezení pod názvy Bump-mapping a Normal Mapping. Dochází zde simulaci stínování hrbolatého povrchu, ovšem beze změny geometrie. Tuto vlastnost odstraňují další techniky navazující na tuto práci. Vznikly techniky jako, Parallax Mapping, Displacement mapping a další. Článek (Szirmay-Kalos – Umenhoffer, 2008) velmi široce popisuje většinu technik používaných v této oblasti, zejména výše uvedené.

Trojúhelníková síť je také používaná k reprezentaci vrásek. Výhodou trojúhelníkové sítě je skutečnost, že se jedná už dlouhá léta o zavedený model reprezentace objektů. Pro případ vrásek je potřeba zvážit, jak vytvořit dodatečné detaily na existující síti. Protože vráska je často velmi jemný detail potřebujeme zjemnit síť a dále pokud možno tzv. *adaptivně*. Jemnost vrásek určuje počet úrovní dělení. Adaptivitou dělení sítě lze ušetřit počet trojúhelníků se subjektivně stejným vzhledem. Pro případ hlavy je totiž běžné, že plocha využitá k tvorbě vrásek bude odhadem výrazně menší než celá plocha hlavy. Zejména zadní část hlavy a místa pokrytá vlasy zaobírají víc jak polovinu celkové plochy a nevznikají zde žádné vrásky.

Příkladem adaptivního dělení trojúhelníkové sítě s interpolačními vlastnostmi je metoda *Incremental Adaptive Subdivision*. Dělící schémata jsou volitelná a některá z nich jsou uvedena v (P. Volino, 1999), (Pakdel – Samavati, 2007) a (Pharr – Humphreys, 2010).

Postup dělení trojúhelníkové sítě popisuje například článek (Pakdel – Samavati, 2007) se schématy Loop a Butterfly s důrazem na přizpůsobitelnost na uživatelem zvolený výběr bodů. Zde jsou popsány dvě techniky adaptivní triangulace. Jeden přístup se nazývá *Červeno-zelená* triangulace (*Reg-green triangulation*) a jeho výhoda je zejména v malém počtu generovaných trojúhelníků nezbytných pro vytvoření korektního přechodu mezi různými úrovněmi dělení. Druhý postup nazývaný *Inkrementální* je v porovnání z první technikou rychlejší výpočetně a snadnější na implementaci. Za uvedené výhody se platí větším počtem generovaných trojúhelníků než je nezbytně nutné na vytvoření přechodu mezi úrovněmi.

4 Stárnutí vlasů

Stárnutí vlasů je oblast popisující jak změny geometrické (mizení vlasů), tak změny optické v průběhu času. Podobně jako u vrásek lze generovat masku, podle které se vlasy rozmístí. V případě vlasů se ale můžou oblasti měnit, což může být závislé na jedinci. Pak musí existovat způsob jak zmíněný vývoj ve všech časech reprezentovat tak, aby rekonstrukce pro zvolený čas byla nejméně výpočetně náročná. Opět můžeme rozdělit simulaci vlasů na dva celky: simulace rozmístění vlasů a jejich vykreslování (problémy stínování, odrazivosti a propustnosti světla).

Rozmístění každého vlasu na ploše trojúhelníku lze realizovat systémovým náhodným generátorem čísel v barycentrických souřadnicích. Vzhledem k tomu, že není hustota vlasů po celé ploše konstantní, je potřeba vzít v úvahu například texturu v odstínech šedi, kterou by měl vytvořit uživatel ručním namalováním. Textura obsahující hustotu vlasů (relativní hustota mezi místy s vlasy a místy bez vlasů) je užitečná tam, kde dochází ke změně hodnot. Textura mapovaná na trojúhelník může poskytnout více informací o rozložení vlasů, protože lze podle ní měnit distribuci generátoru tak, aby blíže odpovídal textuře.

Řídnutí vlasů s vyšším věkem lze modelovat dvěma způsoby. První jednodušší je vytvoření dvou okrajových stavů, které se potom budou interpolovat v průběhu věku. Případně může být podle potřeby vytvořeno více klíčových fází. Druhý postup vede na algoritmy zpracování obrazu: eroze a další morfologické operace. Zde musí být zadán počáteční stav, který bude iterativně zpracován (opakované zúžení oblasti) až do zadané úrovně, např. až objekt ztratí všechny vlasy. Je zřejmé, že není k dispozici vývoj spojitý v čase, jinak užitečný pro intuitivní práci. Tento problém jde například odstranit dodatečnou interpolací mezi iteracemi jako bylo uvedeno pro první postup.

Základní postup zobrazování vlasů spočívá v generaci lomených křivek (po částech lineární křivka s C^0 spojitostí) pro každý vlas, které později pro potřeby kreslení můžeme jemněji dělit podle některé kubické spline křivky. Často jasnou volbou bývají *Catmull-Rom* křivky, jako zvláštní případ Hermitovské formy kubických křivek, které jsou užitečné zejména pro umělecké potřeby. Parametrizace křivky nazývaná *centripetal*, je podle zdroje (Yuksel et al., 2009a) nejvhodnější k reprezentaci vlasů zejména proto, že se oproti chordálové a rovnoměrné parametrizaci chová velmi stabilně a spolehlivě. Parametrizace *centripetal* zejména potlačuje sebe-protínání a některé další

charakteristiky křivek. Problém parametrizace spočívá ve volbě kroku parametru t křivky $C(t)$ v segmentu mezi řídícími body P_i a P_{i+1}. Hodnoty parametru můžeme vyjádřit jako $t_{i+1} = t_i + |P_{i+1} - P_i|^\alpha$, kde $0 <= \alpha <= 1$, pokud je $\alpha = 0$ pak jde o pravidelnou parametrizaci, $\alpha = 1$ chordálovou a nakonec $\alpha = 1/2$ tzv. centripetal parametrizaci.

Vlasy jako takové se musí přizpůsobit tvaru hlavy, tudíž bude nutné spočítat kolize s objektem hlavy, tak aby vlasy vlivem gravitace pokryly hlavu nebo splývaly po jejím povrchu. Dále je možné kontrolovat tuhost vlasu, pokud budeme měnit velikosti tečných vektorů (určitá forma vah). Tloušťka každého vlasu se (lineárně) mění od kořene ke konečku, takže se vlas jinak deformuje vlivem gravitace. Příliš tenké vlasy se na konci mohou dokonce kroutit, což je značně netriviální úloha a zároveň pro potřeby této práce není užitečná.

Velmi vybočuje metoda nazývaná tzv. *fur rendering*, která je užitečná zejména k vizualizaci srsti. Možná je to příliš náročná metoda pro potřeby vykreslování v reálném čase, další studium ukáže jaké efektivní algoritmy existují. Uvedená technika mimo náročnost výpočtu omezuje také délku vlasů, proto může být užitečnější na vousy než na vlasy.

Další známá technika pracuje spíše na jedné geometrii (viz další odstavec) reprezentující celou pokrývku; jednotlivé vlasy se zobrazí texturou s alfa maskou na simulaci průhlednosti (masku lze syntetizovat z dvojrozměrné šumové funkce).

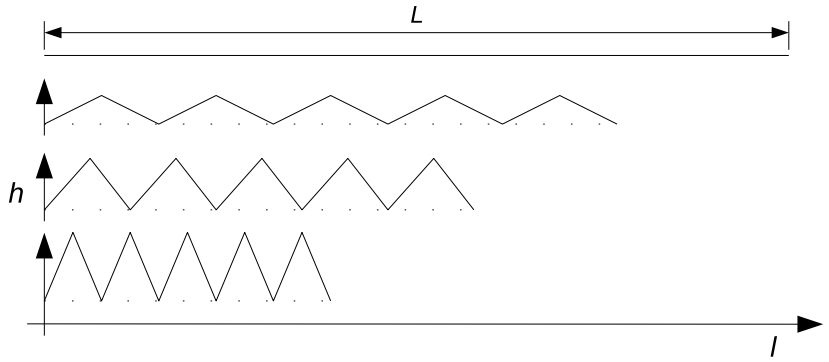
Poslední otázkou v oblasti stárnutí vlasů, je volba vhodného modelu stínování vlasů. Přestože jsou vlasy víceméně lesklé, nejsou obecně izotropní materiál, tj. neodráží světlo stejně z každého úhlu příchozího paprsku světla, navíc není možné určit normálu k samotnému vlasu, proto je potřeba jiný postup než je Phongův model. Dále pro vlasy platí, že jsou částečně průhledné, takže větší vrstva vlasů propustí méně světla k pozorovateli a naopak. Možné reference jsou (Marschner et al., 2003), (Green, 2004) lze použít i na stínování kůže, (Kajiya – Kay, 1989) a další články které zmíněné modely rozšiřují nebo se zaměřují na efektivní řazení vlasů pro alpha blending, např. (Zinke et al., 2008) a (Yuksel et al., 2009b).

5 Řešení generování vrásek

Tato kapitola popisuje postupy použité v této práci ke generaci vrásek. Nejprve je popsán model, ze kterého je získán tvar vrásek použitelný pro zobrazování. Před samotným popisem modelů simulace vrásek je třeba rozdělit vrásky na dvě skupiny, které budou v rámci simulace nezávislé. V první řadě se jedná o *velké vrásky* (*large-scale wrinkles*), vznikající nejprve pohybem svalstva a později s věkem se stávají permanentní. Velké vrásky patří mezi prvky lidského těla v makroskopické škále. Tato práce se zaměří na model, který se zakládá na vektorovém popisu svalového vlákna. Podle umístění svalu a oblasti jeho vlivu lze počítat volitelný tvar vrásek. Z vektorového popisu pak proběhne rastrový výstup do textury, jakožto nezávislé reprezentace na vlastnostech trojúhelníkové sítě použitého objektu.

Jemné vrásky (*fine-scale wrinkles*), přesněji jejich tvar, jsou nezávislé na kontrakci vlivem svalů a jsou ovlivněny pouze genetickou informací, opotřebením, zraněním a dalšími vnějšími vlivy působící na kůži po celý život člověka. Protože uvedené vnější vlivy jsou čistě individuální záležitost budou dále zanedbány. Jako předloha tvaru jemných vrásek bude sloužit textura, kterou může případný umělec nebo expert na biologickou antropologii přizpůsobit podle svých představ.

První úkol v simulaci velkých vrásek je stanovení modelu kontrakce délky kožní pokrývky. Protože není výhodné zabývat se přesnými modely kontrakce, uvažující kůži jako kontinuum, bude předpokládáno, že se kůže deformuje jako diskrétní soustava tuhých přímek.



Obrázek 5.1: Obrázek aproximace komprese povrchu kůže po částech lineární lomenou čárou.

Tuto myšlenku ilustruje obrázek 5.1. Výpočetně je tento model jednoduchý a zároveň poskytuje dobrou návaznost na aplikaci s trojúhelníkovou sítí.

Pro velké vrásky lze stanovit přibližný (lineární) vztah závislosti výšky vrásky h podle stlačení kůže μ následujícím způsobem. Protože jsou přímky lineární, platí pro výšku vrásky h, délky ve výchozím stavu L a délky L_s po stlačení, následující zákon o pravoúhlém trojúhelníku (Pythagorova věta):

$$L^2 = L_s^2 + h^2$$

Definice *stlačení*, označím μ, lze jej zapsat takto:

$$\mu = \frac{L}{L_s}$$

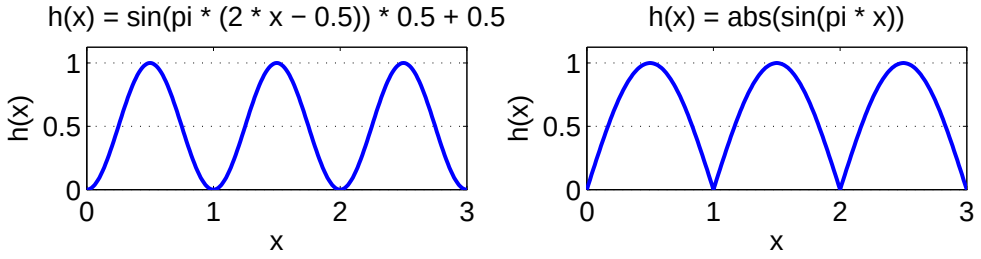
kde se předpokládá, že platí $L \geq L_s$, tj. délky jsou pouze zkracovány. Po dosazení do Pythagorovy věty, lze získat vztah pro výpočet výšky vrásek h v závislosti na délce L_s a stlačení μ.

$$h = L_s \cdot \sqrt{\mu^2 - 1} \tag{5.1}$$

5.1 Velké vrásky pomocí textur

Jak bylo předesláno dříve, na uživateli jako expertovi je přenechána zodpovědnost rozmístění vrásek. Tvar vrásek je generován v normalizované podobě, tj. s výškou od 0 do 1. Skutečná výška vrásek je určena v geometrickém prostoru až při vizualizaci. Dále popsaný model je omezen pevným tvarem oblasti vlivu. Další tvarová nezávislost na modelované geometrii svalů je kompenzovatelná maskou, která umožňuje vytvořit nepravidelnou oblast tvorby vrásek. Vstupem celého sytému bude tedy vektorový popis vrásek a maska jejich výskytu. Stejně tak platí i pro jemné vrásky, kde další maska má stejný účel.

Ať už se budou vrásky kreslit jakkoliv, postačí simulovat stárnutí změnou hrubosti povrchu (prohloubení vrásek), jenž lze řešit lineární interpolací mezi stavem v mládí (žádné nebo minimální vrásky) a ve stavu stáří (maximální intenzita).



Obrázek 5.2: Ukázky vybraných profilů vrásek podle funkcí 5.2 a 5.3.

5.1.1 Profil vrásky

Další popis modelování vrásek se odvíjí od myšlenky, že lze stanovit směr vrásky a v tomto směru lze opakovat jeden vzor (křivku) podobající se průřezu vrásky.

S funkčním předpisem se pracuje lépe než s neznámou křivkou. Navrhuji následující vyjádření tvaru vrásky, která se podobá reálnému průřezu:

$$h(x) = \sin(\pi \cdot (2 \cdot x - 0.5)) \cdot 0.5 + 0.5 \tag{5.2}$$

$$h(x) = |\sin(\pi \cdot x)| \tag{5.3}$$

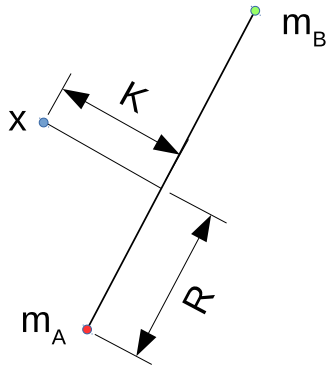
Užitečné vlastnosti má funkce 5.3, která nemá žádné dodatečné parametry, tudíž je snadno nasaditelná bez nutnosti hledání optimálních parametrů pro danou situaci. Dále je zachováno pravidlo, že vráska má ostrou brázdu a hladkou bouli.

5.1.2 Svalově orientovaný model

Model svalového vlákna ve formě úsečky ovlivňuje svým smrštěním zvolené okolí kůže. Podobný přístup jako ukazuje článek (Zhang – Sim, 2005). Zde je základem modelování vrásek poloha dvou bodů $\mathbf{m}_A$ a $\mathbf{m}_B$ definujících *hlavní svalové vlákno*. Příklad jednoho vlákna je k vidění na obrázku 5.3 na straně 13. Vedle bodů $\mathbf{m}_A$ a $\mathbf{m}_B$ je možné nastavovat také počet vrásek na rozsahu vlákna, relativní šířku vzhledem k délce vlákna W a další parametry.

Následující vztahy vedou na výpočet normalizované výšky vrásky $h(\mathbf{x})$ v bodě $\mathbf{x}$ vůči jednomu svalovému vláknu. Funkce $h(\mathbf{x})$ je závislá na pomocné funkci $h(t, d)$ stejného jména, ale jiných parametrů:

$$h(\mathbf{x}) = h(R_\perp(\mathbf{x}), K_\perp(\mathbf{x}))$$



Obrázek 5.3: Ilustrace modelu svalového vlákna ovlivňující zadané okolí pokožky.

tím se provede převod kartézských souřadnic $\mathbf{x}$ na dvojici parametrů t a d:

$$h(t,d) = A \cdot At(d) \cdot |\sin(\pi \cdot \omega_0 \cdot \omega(d) \cdot t)| \,. \tag{5.4}$$

Zde je vidět, že jsem zvolil jako profil funkce 5.3 uvedené dříve v kapitole 5.1.1 na straně 12. Konstanta ω_0 představuje základní frekvenci vrásek ležících na hlavním svalovém vlákně a její velikost je počet vrásek zadaný uživatelem.

Funkce $At(d)$ řídí útlum vrásek podle kolmé vzdálenosti od hlavního vlákna a dále změnu frekvence ve funkci $\omega(d)$.

$$At(d) = 1 - \min\left(\frac{d}{W \cdot \|\mathbf{m}_B - \mathbf{m}_A\|}, 1\right) \tag{5.5}$$

Symbol $\|$ označuje euklidovskou normu v dvojrozměrném prostoru.

Konstanta W je normalizovaná (relativní vůči délce svalového vlákna) šířka oblasti vlivu vrásky a je volena uživatelem, přičemž musí náležet intervalu hodnot $\langle W_{min}, 1\rangle$. Konstanta W_{min} je číslo blízké nule stanovené v závislosti na strojové přesnosti aritmetiky v pohyblivé řádové čárce, např. pro datový typ `float` jsem stanovil hodnotu 10^{-6}. Uvedená podmínka tedy nedovoluje šířku nulové délky což má svůj praktický smysl.

Funkce $\omega(d)$ ovlivňuje změnu úhlové frekvence vrásek podle vzdálenosti od hlavního vlákna. Výpočet $\omega(d)$ je odvozen z hodnot funkce $At(d)$ s tím, že je aplikován nelineární modifikace $S(x)$, aby se předešlo nespojitému přechodu vln na svalovém vlákně.

$$\omega(d) = f_{min} + (1 - f_{min})S(At(d)) \tag{5.6}$$

kde f_{min} je volitelné číslo z intervalu $\langle 0, 1\rangle$ a vyjadřuje relativní frekvenci vůči základní frekvenci. Je vidět, že funkce $\omega(d)$ musí splňovat podmínku $\omega(0) = 1$ a $\omega(0.5 \cdot \|\mathbf{m}_B - \mathbf{m}_A\|) = f_{min}$. Funkce $S(x)$ může být teoreticky libovolná hladká funkce s nulovou derivací v bodě $x = 0$. Požadavek na nulovou derivaci je zde kvůli nutnosti spojitého navázání na svalové vlákno, tj. spojité navázání mezi poloprostory vymezenými právě přímkou procházející svalovým vláknem. Pro realizaci výsledků jsem zvolil tuto funkci:

$$S(x) = 3x^2 - 2x^3 = x^2 \cdot (3 - 2 \cdot x) \tag{5.7}$$

kde musí být $x \in \langle 0, 1\rangle$. V případě že je $S(x)$ závislá na $At(d)$, je uvedená podmínka zaručeně splněna. Funkce $S(x)$ byla také použita na vyhlazení průběhu funkce $h(t, d)$, vypočtené podle vztahu 5.4. Bez funkce $S(x)$ by vráska měla dobře patrný ostrý vrchol na přímce hlavního svalového vlákna. Z důvodu přehlednosti tohoto textu tuto skutečnost uvádím jen jako poznámku.

Poslední otázkou je, jak spočítat t a d, resp. $R_\perp(\mathbf{x})$ a $K_\perp(\mathbf{x})$.

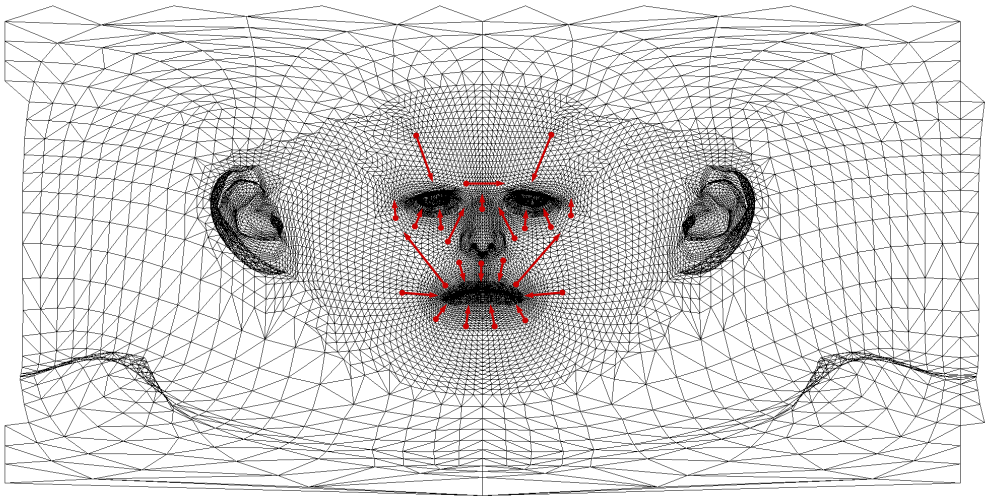
$$t = R_\perp(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{m}_A\|}{\|\mathbf{m}_B - \mathbf{m}_A\|}$$
$$d = K_\perp(\mathbf{x}) = (\mathbf{x} - \mathbf{m}_A) - t \cdot \frac{\mathbf{m}_B - \mathbf{m}_A}{\|\mathbf{m}_B - \mathbf{m}_A\|}$$

Protože výpočet t může vést na hodnoty mimo rozsah $\langle 0, 1\rangle$, přičemž není žádoucí, aby takové hodnoty existovaly, protože už z podstaty svalového vlákna ani nemají smysl. Je tedy potřeba omezit t do prostoru mezi body $\mathbf{m}_A$ a $\mathbf{m}_B$. To se provede oříznutím t na interval $\langle 0, 1\rangle$, což má za následek, že průvodiče bodů $\mathbf{x}$ přestanou být kolmé na svalové vlákno a začnou vlastně tvořit radiální systém (vrásky začínají být kružnice).

Aby šlo omezit zmíněný jev, zavedl jsem omezení úhlu α mezi směrem svalového vlákna $(\mathbf{m}_A - \mathbf{m}_B)$ a průvodičem $\mathbf{x}$[1]. Průvodič bodu $\mathbf{x}$ je vektor $\mathbf{x} - \mathbf{m}_B$. Výsledkem je vztah pro útlum vrásky, podobný funkci $At(d)$ s tím, že nyní bude tato funkce záviset na α. Omezení úhlu α zadává uživatel jako α_{max}. Musí platit, že $\alpha_{max} \in \langle \pi/2, \pi\rangle$. Jiné úhly nejsou v tomto kontextu relevantní.

Příklad rozmístění svalových vláken lze vidět na obrázku 5.4 (strana 15), kde na rozvinuté trojúhelníkové síti v rovině lze vidět vyznačené body $\mathbf{m_A}$ (označené bodem), $\mathbf{m_B}$ (označené šipkou). Pořadí těchto bodů je potřeba brát

[1]Průvodič bodu $\mathbf{x}$ pro oříznuté hodnoty $t > 1$, je vztažen k bodu $\mathbf{m}_B$.



Obrázek 5.4: Rozvinutí trojúhelníkové sítě referenčního modelu hlavy do roviny.

v úvahu během návrhu rozmístění. Platí, že vrásky tohoto začínají v bodě $\mathbf{m_A}$, odtud dál do bodu $\mathbf{m_B}$ se vrásky ohýbají, pokud je f_{min} voleno blíže nule.

5.1.3 Tvorba vrásek v textuře

Cílem následujícího kroku je uložit výsledky funkce $h(t, d)$ do textury v podobě výškové mapy, která je pak dál převedena do normálové mapy (viz podkapitola 7.1.1) a nakonec použita v zobrazovacím řetězci (viz další kapitoly, zejména podkapitola 7.1.2).

Každá vráska se kreslí samostatně do vyhrazené oblasti. Vyplnění oblasti se pak provádí semínkovým plněním „do šířky“ s použitím fronty. Je třeba počítat, že se oblasti vrásek mohou překrývat, takže je nutné stanovit postup, jak kombinovat dopředu neznámé množství vrásek nezávisle na pořadí.

Pokud se prochází pixely textury do šířky podle zvoleného okolí, je nutné uchovávat v každém pixelu informaci o jeho předchozím navštívení. Nadále budu používat pojmenování *obsazení* pixelu pro ty pixely, které byly dříve navštívené. K tomuto účelu bude sloužit textura obsazení (occupancy texture). Jedná se o matici, stejných rozměrů jako výstupní textura, obsahující celočíselné prvky. Zmíněné prvky mohou nabývat hodnot 0, když není pixel

Obrázek 5.5: Výsledek vrásek ve výškové mapě podle zadání svalů z obr. 5.4. Bílá značí výšku vrásky 0 a černá výšku 1.

obsazen, 1 když je pixel obsazen vráskou 1, 2 když je obsazen vráskou 2, atd. Navštívení souseda aktuálního pixelu je možné právě když, je jeho hodnota obsazení různá od čísla aktuálně zpracovávané vrásky. Uvedený postup zaručuje že algoritmus navštíví všechny relevantní pixely právě jednou. Díky tomu není třeba před každou iterací nulovat celou mapu obsazení.

5.1.4 Algoritmus kreslení vrásek

První krok navrženého algoritmu je vypočtení počáteční pozice (počáteční pixel), tzv. semínko. Tento pixel se zpracuje, vytvoří se záznam, že byl již zpracován a pak je přidán do fronty, kde budou ukládány další pixely. Potom se opakuje krok podobný prvnímu dokud není vyprázdněna fronta. Aby algoritmus vůbec skončil a neprocházel všechny pixely textury, je přidání do fronty podmíněno jednak hodnotou v textuře obsazení tak, amplitudou vrásky (kde je vráska viditelná). Délka fronty pro semínkové plnění je teoreticky omezena na $O(s + v)$, kde s je šířka textury a v je výška textury.

Proměnná O je textura obsazení, H je výstupní výšková mapa a $\Omega(\mathbf{x})$ je pevně stanovené okolí pixelu (4-okolí) se souřadnicí $\mathbf{x}$. Proměnná i je číslo aktuálně zpracovávané vrásky s hodnotami $i \geq 1$ a určuje jaké hodnoty bude vracet funkce $h(\mathbf{p})$ počítající výšku vrásky podle svalového modelu. Proměnná $\mathbf{p}$ je dvourozměrný bod s pozicí pixelu.

vstup : model svalů jako funkce $h(\mathbf{p})$
výstup: výšková mapa H s vráskami
inicializace H a O na nulové hodnoty;
začíná se v pixelu $\mathbf{p}$ uprostřed svalového vlákna;
pro pixel $\mathbf{p}$ výpočet $h := h(\mathbf{p})$;
přečti starou výšku $h_{old} := H(\mathbf{p})$;
zapiš výšku $H(\mathbf{p}) := h_{old} + h$;
zapiš $O(\mathbf{p}) := i$;
přidej do fronty pixel $\mathbf{p}$;
while *je fronta neprázdná* **do**
  vyber pixel $\mathbf{p}$ z fronty;
  for *každého souseda* $\mathbf{q} \in \Omega(\mathbf{p})$ **do**
    if $O(\mathbf{q}) = i$ **then**
      pokračuj dalšími sousedy;
    end
    pro pixel $\mathbf{q}$ výpočet $h := h(\mathbf{q})$ a At;
    if $At > 0$ **then**
      přečti starou výšku $h_{old} := H(\mathbf{q})$;
      zapiš výšku $H(\mathbf{q}) := h_{old} + h$;
      zapiš $O(\mathbf{q}) := i.$;
      přidej do fronty $\mathbf{q}$;
    end
  end
end

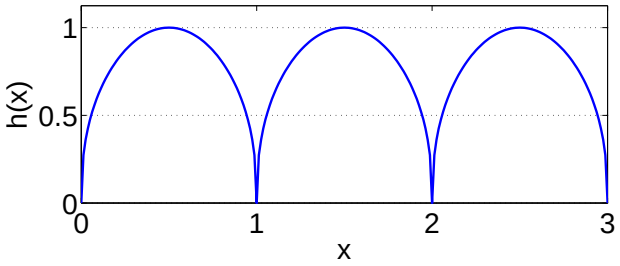
Algoritmus 1: Pseudokód vykreslení jedné vrásky s číslem i do výškové mapy H.

5.1.5 Míchání vrásek

Vrásky jsou v textuře H akumulovány součtem. Tím je zaručeno smíchání nezávislé na pořadí vykreslování. Po dokončení kreslení je vyhledána maximální výška h_{max}, kterou jsou normovány všechny výšky pixelů $H(\mathbf{p})$. Posledním kroku se provede modifikace funkcí $g(x) = x^{\gamma}$ aby bylo dosaženo zvýraznění nižších vrásek.

$$H(\mathbf{p}) := \left(\frac{H(\mathbf{p})}{h_{max}}\right)^{\gamma}$$

Důsledkem je lepší tvar profilu vrásky, protože se funkce $H(x)$ se zploští na vrcholu boule. Ukázka profilu po zmíněné modifikace je na obrázku 5.6. Exponent γ byl zvolen na hodnotu $0,4$.

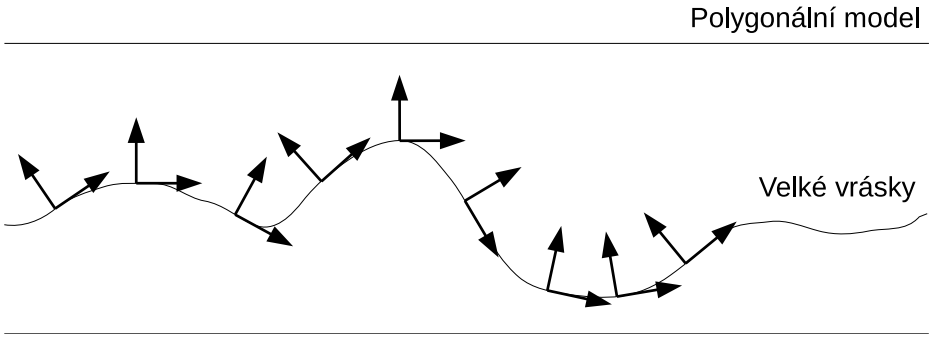


Obrázek 5.6: Výsledek ukazující zploštění profilu vrásky po použití funkce $g(x)$.

5.2 Jemné vrásky

Pro realizaci jemných vrásek lze aplikovat stejné postupy jako u velkých, nicméně je třeba brát v úvahu, že již není třeba tak precizně reprezentovat příslušnou geometrii. Pro potřeby opravdu jemných detailů postačí simulace osvětlení, protože člověk není často schopen, v takovém měřítku a běžné vzdálenosti pohledu, tyto detaily rozeznat. Použitá metoda bude tedy Normal Mapping (zkráceně NM). NM umožňuje simulovat osvětlení velmi jemného hrbolatého povrchu a dále je nezávislý na typu trojúhelníkové sítě. Jak se počítá NM je podrobněji popsáno v kapitole 7.1.1, implementační část.

Aby šlo provést mapování normálové mapy pro jemné vrásky je potřeba vyjádřit transformaci do prostoru této textury. To ovšem znamená, že se



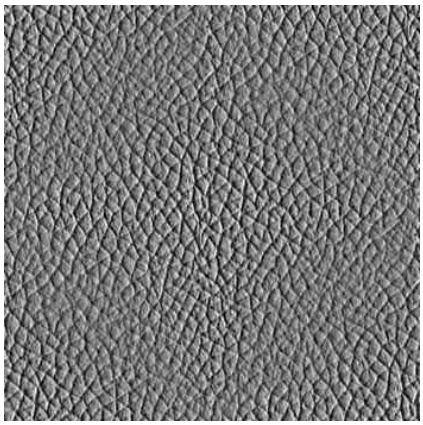
Obrázek 5.7: Ilustrace myšlenky výpočtu tečného prostoru pro jemné vrásky v tečném prostoru pro velké vrásky.

vytvoří další tečný prostor v tečném prostoru velkých vrásek. Uvedenou představu naznačuje obrázek 5.7. Pomocí tečného prostoru je možné provést transformaci vektorů, potřebných pro výpočet stínování

Efekt stárnutí v čase je realizován lineárním mícháním mezi zmíněnými normálovými vektory detailní textury a normálou povrchu na který je mapován. Normálou povrchu se pro případ jemných vrásek jedná o normálu k povrchu velkých vrásek. Nulový věk znamená žádné jemné vrásky a věk 1 se zobrazením jemných vrásek v plné intenzitě.

Jemné vrásky mohou vyžadovat trochu jiný postup mapování textury než pro velké vrásky, protože na velmi jemné detaily jedna textura nemusí stačit. Řešením může být opakování jedné textury jako dlaždice, s tím že je potřeba napojit textury bez viditelného přechodu mezi nimi. Jako vstup celého procesu jemných vrásek je považována výšková mapa, později převedena na normálovou mapu, kterou lze jako dlaždici opakovat. Příkladem budiž textura na obrázku 5.8. Ručně vytvořenou výškovou mapu lze pomocí řady existujících rastrových editorů předzpracovat tak, aby při napojování nevznikaly ostré přechody.

Souřadnice textury se vypočtou nové, pomocí *inverzního sférického mapování* $f : [x, y, z]^T \rightarrow [\phi, \theta]^T$, tj. převodem z prostorových kartézských souřadnic do polárních souřadnic. Opakování se vytvoří vynásobením existujících souřadnic volitelným celým číslem (pro obě osy nezávisle podle volby uživatele). Pokud se takto převede každý vrchol trojúhelníkové sítě, musí se počítat se zkreslením textury při napojení dvou opačných konců. V tomto místě totiž přechází texturovací souřadnice z čísla 1 do 0. Nicméně praxe nás nutí tuto záležitost explicitně ošetřit zduplikováním vrcholů a uprave-



Obrázek 5.8: Příklad opakovatelné textury s jemnými vráskami.

ním souřadnic tak, aby při přechodu přes 1 měly sousední vrcholy hodnotu z okolí 1, resp. při přechodu přes 0 byly souřadnice z okolí 0. Zde postačilo stanovit podmínku: pokud je rozdíl souřadnic sousedních vrcholů (sdílejících hranu) větší než $0,5$, přičti nebo odečti 1 v závislosti na určité podmínce. Důležité je zmínit, že testy na existenci napojení textury se provádějí pro každý trojúhelník a vždy je v něm upraven právě jeden vrchol.

Aby šlo mapovat zmíněnou normálovou mapu textury 5.8 je potřeba vypočítat posun texturovací souřadnice odpovídající bodu na velkých vráskách. V tomto bodě si vystačím s jednoduchým odhadem. Texturovací souřadnice na velké vrásce je vynásobena počtem opakování dlaždic v příslušné ose a přičtena k aktuální souřadnici pro jemné vrásky.

6 Řešení stárnutí vlasů

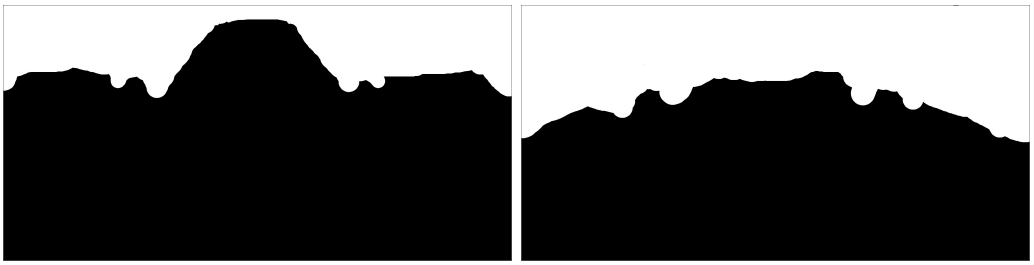
Neodmyslitelnou součástí stárnutí hlavy jsou i procesy stárnutí vlasů. V této části bude simulován zejména úbytek vlasů a změna jejich barvy (šedivění). Úbytkem vlasů je myšlena změna vnější oblasti růstu na jinou oblast, zpravidla menší (označme jako vnitřní). Změna barvy bude simulována postupným posunem výchozí barvy (uživatelská volba) k barvě velmi nízké sytosti, většího jasu a nezměněného odstínu.

6.1 Simulace úbytku vlasů

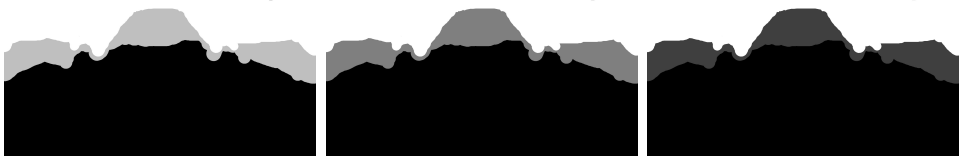
Jako vstupní informace jsou dvě oblasti (v aplikaci se jedná o textury) označené jako vnější (počáteční stav) a vnitřní (koncový stav). Bílá barva značí povolení k růstu vlasů a černá naopak růst zakazuje. Příklad těchto textur je k vidění na obrázku 6.1 Úkolem je, podle těchto masek, rozmístit po povrchu hlavy vlasy se zadanou hustotou na jednotku plochy a dále podle zadaného věku z intervalu $\langle 0, 1 \rangle$ vytvořit mezistav dvou uvedených stavů.

Pokud máme informaci o růstu vlasů v textuře, tak prvním nejednodušším řešením je provést alfa míchání mezi zmíněnými stavy. Tento postup se na první pohled nejeví jako vhodný postup, protože oblast přechodu se bude měnit všude stejně a tudíž ani nenastane požadovaný ústup vlasů. Obrázek 6.2 ukazuje jak by tato operace dopadla pro míchání 75%, 50% a 25%.

Diametrálně jiné možnosti řešení nabízí metody založené na *Rozšiřování oblasti*. Na tento postup se dá také dívat jako na zvláštní filtraci s maskou



Obrázek 6.1: Ukázka dvou textur, vnitřní (vlevo) a vnější (vpravo) oblasti.



Obrázek 6.2: Obrázek ilustrující situaci při alfa míchání 75%, 50% a 25%.

porovnávající hodnoty sousedních pixelů s aktuálním pixelem. Pro tento postup stačí projít každý pixel textury a podle masky zkontrolovat jestli sousedi nemají zadanou barvu. Pokud ji mají, obarví se aktuální pixel novou barvou jejíž hodnota bude o jedničku vyšší. Po opakované aplikaci zmíněného procesu lze postupně původní vnitřní oblast rozšířit na vnější oblast, kde se dál už nesmí pokračovat. Opakování procesu je možné zastavit až nenastane žádná změna v textuře. Jako *počáteční stav* se považuje uživatelsky zvolená vnitřní oblast, která se metodou rozšiřování o jednu generaci dostane až na hranici vnější oblasti, kde algoritmus končí.

Pro poněkud formálnější popis algoritmu popsaného v předchozím odstavci je potřeba zavést tento operátor:

$$A(i,j) \oplus M_r = \bigoplus_{\substack{k,l \ \in \ \Omega \\ k \neq 1 \ \wedge \ l \neq 1}} (A(i,j) = M_r(k,l) + 1).$$

Pixely $A(i,j)$ jsou celočíselné hodnoty větší nebo rovny nule. Jedná se zmíněnou operaci porovnání hodnoty centrálního pixelu $A(i,j)$ s pixely okolí Ω (zvoleno 8-okolí) podle masky M_r. Vrácena je hodnota $True$ nebo $False$. Operace $\bigoplus$ je logický **OR** pro všechna $k,l \in \{0,1,2\}$ z okolí Ω různých od jedné. Maska M_r je indexována podle aktuální generace označené r a její prvky jsou definovány takto:

r - 1	r - 1	r - 1
r - 1	r	r - 1
r - 1	r - 1	r - 1

K obarvení pixelu $A(i,j)$ dojde ve chvíli, kdy je splněna tato podmínka:

$$(A(i,j) \oplus M_r) \ \wedge (B(i,j) > 0) \implies A(i,j) = r \quad \forall i,j \in A$$

kde $B(i,j)$ je textura vnější oblasti s hodnotami výšky v rozsahu $\langle 0,1 \rangle$.

Každé rozšíření v jedné úrovni má své číslo a tato hodnota může být chápána jako vzdálenost od původní oblasti. Takovou informaci lze využít ke

Obrázek 6.3: Přechodová textura vytvořená z oblastí na obrázku 6.1.

kódování čísla generace na úrovně šedi z intervalu $(0; 1)$, která budou říkat, jak bude vypadat hranice v čase z uvedeného intervalu. Výsledné skalární pole je *přechodová textura*, protože umožňuje zachytit téměř spojitý přechod mezi počátečním a koncovým stavem oblasti. Čas (věk) je předpokládán právě jako jednotkový, kde pro 0 je čas nejnižší a 1 je čas nejvyšší. K získání celé oblasti v konkrétním čase, je potřeba provést prahování *přechodové textury*. Přechodovou texturu lze použít i pro přímou generaci instancí vlasů, tzn. explicitní generaci masky pro příslušný věk není potřeba.

Algoritmus stárnutí vlasů, ke kterému je směřováno, má oddělitelné tři fáze. První byla popsána v této kapitole. Druhý bod je krátce shrnut v následující kapitole, kde je kladen důraz na řešení kreslení vlasů, tj. třetí fáze.

1. Generace *přechodové textury* ze zadané *vnitřní* a *vnější* oblasti.
2. Generace instancí vlasů podle *přechodové textury*, seřazení podle věku sestupně.
3. Vykreslení vlasů v zadaném věku, kde věk implikuje počet instancí.

Distribuce vlasů je náhodná[1] a je podmíněna hodnotami *přechodové textury* a zvoleným věkem. Z věku je také odvozována variace šedivění každého vlasu jakožto další projev stárnutí vlasů. Generace vlasů probíhá pro každý trojúhelník, kde počet nových instancí je dán plochou trojúhelníka. Při zpracování každé nové instance jsou vytvořeny náhodné barycentrické souřadnice,

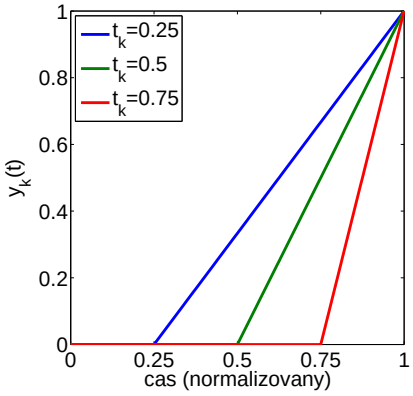
[1]Náhodnost je dána vlastnostmi generátoru v rámci standardní knihovny C a počáteční hodnotou (random seed). Konkrétně u vytvořené aplikace byla hodnota seed volena 0.

které jsou dále s výhodou použity k interpolaci dalších atributů (normála, pozice a texturovací souřadnice).

6.2 Variace barevnosti

Přenos atributů pozice a směru vlasů není jediná možná aplikace. Nabízí se zde možnost přenášet i jiné atributy jako je barva apod., o čemž pojednává i tato kapitola. Opět je zde potřeba vyjádřit vývoj některé vlastnosti vlasů v čase a v této menší kapitole to bude právě barva vlasů. Konkrétně jejich jas, který je z pohledu preferované šedé škály jediným vyjadřovacím prostředkem.

Jako aproximace blednutí vlasů je zvolena interpolace mezi základní barvou (počáteční stav) a šedou barvou (uživatelem zvolený koncový stav) pro každý vlas s rozdílným průběhem. Změna barvy probíhá v RGB, kde všechny



Obrázek 6.4: Průběh funkce $y_k(t)$ pro $t_k = \{0,25; 0,5; 0,75\}$, vyjadřující přechod mezi počáteční barvou vlasů a koncovým stavem.

složky barvy jsou interpolovány podle hodnoty jedné transformační funkce (viz obrázek 6.4). Barva vlasů vlivem stárnutí je funkce jedné proměnné $y_k(t)$ a má následující tvar:

$$y_k(t) = \max\left(\frac{t_k - t}{t_k - 1}, 0\right),$$

kde k je index vlasu, t je čas z intervalu $\langle 0, 1\rangle$. Parametr t_k, pro každý vlas, je stanoven náhodně a je ovlivněn věkovou hranicí, do které je možné vlas vidět.

Pro vyjádření přechodu mezi okrajovými stavy (C_{base} a C_{gray}) byla zvolena lineární interpolace. Výsledný výpočet barvy každého vlasu $C_k(t)$ je dán vztahem:

$$C_k(t) = (1 - y_k(t)) \cdot C_{base} + y_k(t) \cdot C_{gray}$$

Barva každého vlasu, před svým zmizením, přejde na zvolenou šedou C_{gray}. Volitelně, lze jako interpolační funkci použít vztah 5.7 na straně 14.

6.3 Kreslení vlasů

Pro potřeby této práce postačí reprezentace každého vlasu jako přímky rozmístěnými po povrchu hlavy s hlavním směrem podle normály objektu v požadovaném bodě s mírným náhodným vychýlením. Přímky jsou akceptovatelné jen pro účely této práce, jinak je možné realizovat libovolnou metodu kreslení vlasů uvedenou v teoretické části, např. každý vlas může být reprezentován jako trojúhelníkový plát s aplikovanou texturou celého trsu vlasů a příslušnou alfa maskou.

Základní barva C_{usr} je uživatelem volena a určuje hlavně odstín vlasů. O stínování se stará následující formule:

$$C_{vlasu} = (K_{diff} \cdot T_{diff} + K_{spec} \cdot T_{spec}) \cdot C_{usr} + K_{amb},$$

přičemž hodnoty T_{diff} a T_{spec} jsou dány tímto vztahem:

$$\begin{aligned} T_{diff} &= \sqrt{1 - dot(\mathbf{L}, \mathbf{T})^2} \\ T_{spec} &= \left(\sqrt{1 - dot(\mathbf{H}, \mathbf{T})^2}\right)^{K_{shininess}} . \end{aligned}$$

Vektor $\mathbf{L}$ je směr ke světlu, $\mathbf{H}$ je půl-vektor známý z *Phon-Blinn* osvětlovacího modelu a $\mathbf{T}$ je směr vlasového vlákna. Uvedený postup vychází z myšlenek publikovaných v (Kajiya – Kay, 1989).

7 Implementace

7.1 Velké vrásky

Od této kapitoly budou následovat postupy, týkající se vizualizace vrásek. Půjde zejména o to, jak výškovou mapu s vráskou převést na normálovou mapu, kterou lze použít k deformaci povrchu a výpočtu stínování. Na začátek popíšu jednu základní techniku simulace hrbolatého povrchu nazývanou *Normálových map*, která bude v dalších kapitolách rozšířena tak, že bude možné zobrazovat jak velké vrásky, tak jemné vrásky zároveň.

Vrásky jsou také maskovány jednou texturou tak, aby uživatel mohl kontrolovat, kde se mají vrásky objevit a zároveň lze tímto lokálně ovlivnit intenzitu vrásek. Maska je textura v šedých odstínech. Forma textury volena proto, že je možné provést aplikaci masky až v pixel shaderu při výpočtu stínování. Přístup k masce v pixel shaderu má jednu zásadní výhodu. Lze pomocí ní podmíněně vynechat některé sekvence výpočtu, které jsou velmi náročné v místech, kde vrásky nejsou vidět.

7.1.1 Normal-mapping

Technika normálových map (NM), někdy se také setkáme s pojmenováním *tangent space bump mapping*, sloužící k přidání detailů stínování do objektů pomocí *normálové mapy*, je už dlouhá léta známa. NM simuluje osvětlení hrbolatého povrchu s jemnějšími detaily než poskytuje trojůhelníková síť. Původní motivace k realizaci NM byla ve snaze reprezentovat jemné detaily modelu s menším počtem trojúhelníků. Tato technika byla postupně rozšířena na úroveň téměř skutečného hrbolatého povrchu se vzájemným stíněním (viz kapitola 7.1.2 na straně 29).

Původní implementace NM většinu práce s vrcholů počítaly na CPU a poté připravená data vrcholů kopírovaly do paměti GPU. Tento proces je i dnes velmi zdlouhavý a je třeba se mu vyhnout. S příchodem *OpenGL 2.0* s podporou shaderů se situace výrazně změnila. Nyní už není třeba při každém snímku kopírovat rozsáhlá data na grafickou kartu. Stačí pouze nastavit parametry osvětlení pro celý objekt a v rámci výpočtu shaderu lze parametry osvětlení transformovat do tečného prostoru, kde potom můžeme provést

výpočet stínování. Nehledě na technické prostředky, NM potřebuje dvě základní informace nutné pro jeho realizaci. První je normálová mapa a druhý je tečný prostor.

Normálová mapa je textura, která uchovává normálové vektory v každém pixelu. Normálové vektory geometrického objektu jsou nahrazeny normálami uloženými ve zmíněné textuře. Tím docílíme přidání detailů ke geometrii objektu.

Není příliš praktické vytvářet přímo normálovou mapu a vhodnější je zadání povrchu výškovou mapou (skalární 2D pole ve čtvercové mřížce) s rozsahem od 0 do 1, kde 0 znamená minimální výšku a 1 znamená maximální výšku. Normála se generuje pomocí konečných diferencí jako aproximace derivace v diskrétních datech. Označím si výšku na pozici $[x, y]$ jako $h_{x,y}$. Předpokládám, že souřadný systém má osu x orientovánu zleva doprava a osu y orientovánu od shora dolů. Tečný vektor $\mathbf{t}$, popř. $\mathbf{b}$ se spočte pomocí diference ve směru x, popř. y bez normalizace:

$$\mathbf{t} = [1, 0, h_{x+1,y} - h_{x-1,y}]^T = [1, 0, \Delta h_x]^T$$
$$\mathbf{b} = [0, 1, h_{x,y+1} - h_{x,y-1}]^T = [0, 1, \Delta h_y]^T$$

Normálový vektor je $\mathbf{n}$ po zjednodušení dán vztahem:

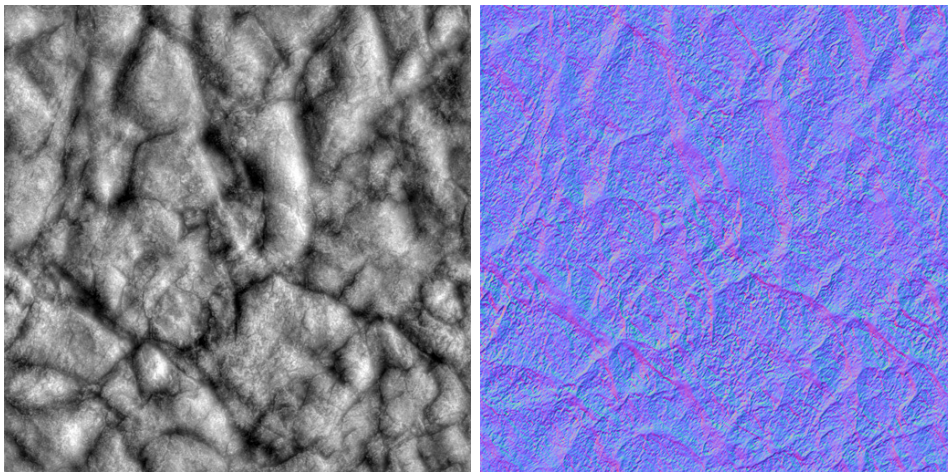
$$\mathbf{n} = \mathbf{t} \times \mathbf{b} = [-\Delta h_x, -\Delta h_y, 1]^T \tag{7.1}$$

Platí podmínka, že na vnitřních pixelech se počítají centrální diference a na krajních pixelech použiji jednostranné diference podle toho, jaké okolní pixely jsou k dispozici. Ještě třeba poznamenat, že je nutné normálové vektory normalizovat před zápisem do výstupní textury.

Nyní je potřeba zakódovat normálové vektory každého pixelu jako barvy RGB(A), což obecně obnáší převod rozsahu hodnot z intervalu $\langle -1, 1\rangle$ na interval $\langle 0, 1\rangle$. Výsledek kódování vypadá například jak ukazuje obrázek 7.1. Vzhledem k tomu, že souřadnice normály $\mathbf{n}$, označené jako n_x a n_y, jsou libovolné, tj. z rozsahu $\langle 0, 1\rangle$ a n_z je vždy z $\langle 0, 1\rangle$, je nutné upravit původní vektor $\mathbf{n}$ takto:

```
1 vec4 normalRGBA = vec4(n.xy * 0.5 + 0.5, n.z, h);
```

kde jako čtvrtá složka uvedené čtveřice je výška v příslušném bodě. Ukázka je napsána v jazyce GLSL jako konstrukce nového 4-dimenzionálního vektoru `vec4`. Obdobným způsobem probíhá rozbalení do původní podoby, jen stačí vynásobit dvěma a odečíst jedničku u n_x a n_y.



Obrázek 7.1: Ukázka výškové mapy (vlevo) jako vstupu a normálové mapy (vpravo) jako výstup.

Jak bylo už naznačeno výše, základním trikem realizace NM je změna souřadného systému tak, aby šlo osvětlení počítat v tzv. tečném prostoru. Jedná se o přístup, kdy pro každý vrchol objektu spočteme tečnou rovinu a příslušnou normálu.

Při konstrukci bázových vektorů tečného prostoru platí, že osa z míří vzhůru (ve směru normály). Báze lze uspořádat v matici, kde jsou v následujícím pořadí uloženy po řádcích:

$$\begin{bmatrix} \mathbf{t} \\ \mathbf{b} \\ \mathbf{n} \end{bmatrix}$$

Libovolný vektor relevantní vůči aktuálnímu vrcholu objektu s bázovými vektory $\mathbf{t}$, $\mathbf{b}$, $\mathbf{n}$ se transformuje takto:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t_x & t_y & t_z \\ b_x & b_y & b_z \\ n_x & n_y & n_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Lengyel (2011) ukazuje, že existuje závislost mezi tečnými vektory ($\mathbf{t}$, $\mathbf{b}$), texturovacími souřadnicemi ($[u_0, v_0]^T$, $[u_1, v_1]^T$, $[u_2, v_2]^T$) a pozicemi vrcholů ($\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$).

$$\begin{bmatrix} t_x & t_y & t_z \\ b_x & b_y & b_z \end{bmatrix} = \frac{1}{u_1 v_2 - u_2 v_1} \begin{bmatrix} v_2 & -v_1 \\ -u_2 & u_1 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{1x} & \mathbf{Q}_{1y} & \mathbf{Q}_{1z} \\ \mathbf{Q}_{2x} & \mathbf{Q}_{2y} & \mathbf{Q}_{2z} \end{bmatrix} \tag{7.2}$$

kde platí $\mathbf{Q}_1 = \mathbf{P}_1 - \mathbf{P}_0$, $\mathbf{Q}_2 = \mathbf{P}_2 - \mathbf{P}_0$ a dále přeznačení:

$$[u_1, v_1]^T = [u_1 - u_0, v_1 - v_0]^T$$
$$[u_2, v_2]^T = [u_2 - u_0, v_2 - v_0]^T.$$

Uvedené vztahy se takto počítají pro každý trojúhelník a jsou průměrovány mezi sousedy, stejně jako se počítají i normálové vektory pro Goraudovo stínování.

7.1.2 Parallax Occlusion Mapping

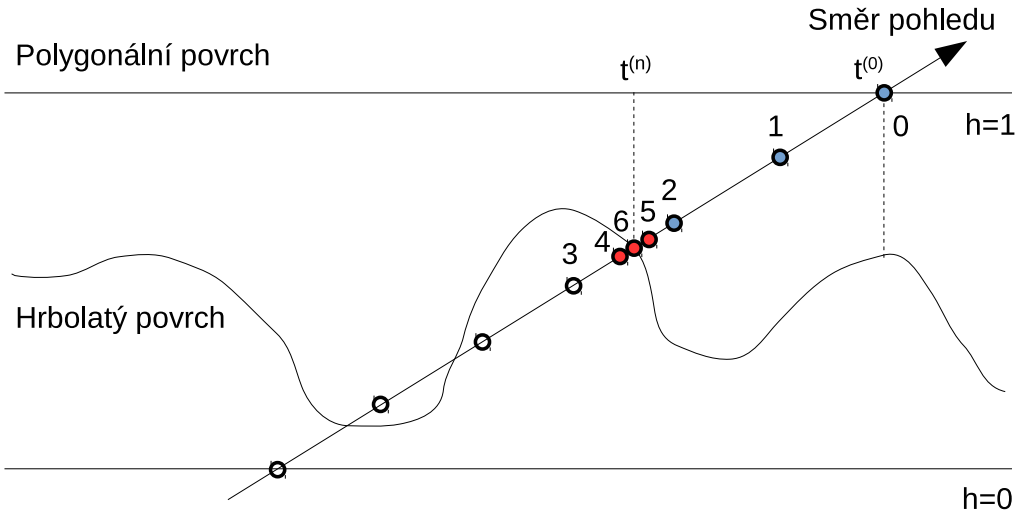
Obecně všechny techniky označované jako *Parallax Mapping* (zkráceně PM), jsou technikou simulující efekt paralaxy při změně polohy pozorovatele vůči sledovanému povrchu. Stejnojmenná technika tuto skutečnost aproximuje změnou texturovacích souřadnic podle výškové mapy, která způsobuje zdánlivé deformování povrchu.

Pokročilejší technikou je *Parallax Occlusion Mapping* (zkráceně POM). Jedná se o případ techniky nazývané souhrnně *relief mapping*, která je shrnuta v článku (Tatarchuk, 2006). Tato metoda je odvozena z algoritmu sledování paprsku, hledající průsečík *paprsku pohledu* s výškovou mapou (neboli povrch z ní vytvořený). Jako výšková mapa je použita textura, vytvořená postupem z předchozích kapitol 5.1.2 a 5.1.3, se skalární hodnotou z intervalu $\langle 0, 1 \rangle$.

Předpokládá se použití pro nepříliš hluboké vrásky, jinak metoda může vykazovat viditelné artefakty. Vzhledem tomu, že vrásky se nemusí dělat nepříliš vysoké, pak je POM preferováno, protože je méně náročnější na zobrazení detailů než rozsáhlý polygonální model.

Paprsek pohledu je dán vektorem od bodu povrchu *objektu* k pozici *pozorovatele* a počátkem, který je přímo zkoumaným bodem na povrchu objektu. Pozice na povrchu je reprezentována jako texturovací souřadnice, protože sledování paprsku probíhá v prostoru textury (tangent space). Podobně jako Normal mapping pracoval v tečném prostoru právě zpracovávaného bodu, tak i POM používá stejnou transformaci z kapitoly 7.1.1. Při výpočtu osvětlení je samozřejmé, že všechny další vektory, potřebné na tento výpočet, musí být ve stejném prostoru a tím je prostor textury.

Ilustrace postupu hledání průsečíku je na obrázku 7.2. Jedná se o iterační algoritmus obdobný těm, které se používají v numerické matematice



Obrázek 7.2: Příklad hledání průsečíku ve dvou krocích: lineárně (vynačeno modře) a binárně (vyznačeno červeně). Pohled je řezem roviny kolmé na polygonální povrch a paprsek pohledu jí prochází.

na hledání kořenů nelineární rovnice. Počátečním bodem je texturovací souřadnice $t^{(0)}$ získaná např. standardním mapováním podle vstupního modelu. Výstupem je jiná souřadnice $t^{(n)}$ odpovídající průsečíku s výškovou mapou. V každém kroku iterace se testuje, jestli je paprsek pod úrovní výškové mapy. Pokud tomu tak není, posouvá se souřadnice proti směru pohledu. Ve chvíli, kdy se dostaneme do opačné situace, tj. je jisté, že objekt byl v tomto segmentu zaručeně protnut, je potřeba rozhodnout jak v algoritmu pokračovat, resp. nepokračovat. Zde tedy rozlišuji *dvě fáze*:

Lineární vyhledávání – vždy konec iterace.

Binární vyhledávání – zkrátit krok iterace na polovinu a posun *proti směru*, pokud je paprsek *nad* úrovní výškové mapy (odečtení souřadnice), nebo *po směru* pohledu (přičtení souřadnice), pokud jsme *pod* úrovní výškové mapy.

Jak už obrázek 7.2 naznačil nejdříve se provádí lineární vyhledávání a ve chvíli, kdy máme nalezený segment kde průsečík existuje, spouští se binární vyhledávání, které celou aproximaci zpřesní. Předpokládá se, že je povrch v tečném prostoru lokálně rovinný, tudíž každé vyhledávání garantuje, že vždy nalezneme průsečík. Povrch simulovaný tímto výpočtem z výškové mapy tedy nikdy nepřesáhne siluetu polygonálního modelu.

Počet iterací pro obě fáze je libovolný (tedy větší než jedna), jen je třeba brát v úvahu nesporný vliv na přesnost výsledku. Vyšší počet iterací umožňuje zobrazovat větší hloubku vrásek. Jaká je konkrétní hranice subjektivní kvality, je podrobněji rozebráno v části testování.

Realizace algoritmů byla provedena pomocí GLSL, tj. celý výpočet je dedikován na GPU. Pokud je možné dynamicky měnit texturu, je simulace POM zvrásněného povrchu s proměnlivou výškou v čase flexibilnější než přístup dělení trojúhelníkové sítě, protože lze měnit tvar vrásek dynamicky v reálném čase (tj. bez předzpracování). Dále je výhoda v nezávislosti na hustotě dělení trojúhelníkové sítě.

V tomto dokumentu (část příloh) je k nalezení výpis zmíněného algoritmu A.1 na straně 53.

Existují i další postupy řešení uvedeného problému zlepšující rychlost konvergence, řešící rychlé přeskakování prázdných míst a dále využívají mipmapping jako nástroj na urychlení samotného vzorkování výškové mapy. Jedná se například o metody *Iterative Parallax Mapping*, *Steep Parallax Mapping*, *Interval Mapping* a *Cone Step Mapping*. V této práci tyto metody nebudou rozebírány, pouze je zde uvádím jako referenci na další postupy.

7.2 Jemné vrásky

Kreslení jemných vrásek společně s POM pro velké vrásky vyžaduje další výpočet tečného prostoru přímo na GPU a transformaci všech potřebných vektorů do tohoto prostoru.

Dá se říct, že se hledá tečný prostor v jiném tečném prostoru. Rekonstrukce tečných vektorů z normály je obdobná postupu z kapitoly 7.1.1 jen se jedná o opačný proces. Předtím byly k dispozici vektory tangenty a binormály a nyní je naopak k dispozici jen normála získaná konverzí z výškové mapy, např. podle obrázku 5.8 na straně 20.

Následuje úryvek kódu zapsaném v GLSL, kde je počítána rekonstrukce tečného prostoru pomocí normály. Vstupem tohoto procesu je normálová mapa velkých vrásek `largeWriTex` a texturovací souřadnice `texCoord0` odpovídající bodu na velké vrásce.

Listing 7.1: Rekonstrukce tečných vektorů ve směrech x a y.

```
const vec3 unpackScale = vec3(2.0, 2.0, 1.0);
const vec3 unpackOffset = vec3(-1.0, -1.0, 0.0);
vec3 normal = texture2D(largeWriTex, texCoord0).rgb;
normal = normal * unpackScale + unpackOffset;
vec3 tangent = normalize(vec3(normal.z, 0, -normal.x));
vec3 binormal = normalize(vec3(0, normal.z, -normal.y));
```

Funkce `texture2D` vrátí barvu odpovídající zadané textuře a souřadnici. Barva je převedena na normálu, změnou rozsahu hodnot konstantami `unpackScale` a `unpackOffset`, a uložena do proměnné `normal`. Složky proměnné `normal` jsou použity ke konstrukci další bázových vektorů `tangent` a `binormal`. Vypočtené vektory jsou normalizovány funkcí `normalize`.

7.3 Vlasy

Vlasy jsou generovány přímo podle přechodové textury, kde se provádí obdobné prahování podle věku, jako by se pracovalo s texturou. Masku růstu vlasů pro příslušný věk není potřeba vytvářet. Protože každý vlas je reprezentován jako přímka, je potřeba uchovávat zejména pozici a směr vlasu, atd. Tyto vlastnosti se sdružují ve struktuře, která má tyto položky:

```
struct HairInstanceAttrib
{
    QVector3D p;
    QVector3D n;
    float age;
    float param;
};
```

Obsahem uvedené struktury je pozice vlasu `p`, jeho směr `n`, věk `age` a další rezervovaný parametr `param`. Každá instance je uložena jako pole uvedených struktur a v této podobě je lze uložit i na GPU. Jak toho lze využít, je popsáno v následující kapitole.

7.3.1 Kreslení vlasů jako více instancí

Během procesu zpracování grafické informace probíhá mezi CPU a GPU komunikace obdobná jako v síťovém provozu, tj. komunikace server-klient, kde klientem je proces běžící na CPU a server je určitá softwarová vrstva zajišťující práci s GPU na hardwarové úrovni a tedy v jistém smyslu představuje GPU samotné. Platí tedy paradigma, že čím více se omezí interakce zmíněných elementů, pak lze dosáhnout kratšího času výpočtu.

Běžným postupem kreslení několika stejných objektů je vytvoření vertex bufferu pro jeden objekt a následné opakované kreslení v cyklu s nastavením parametrů pro každou instanci onoho objektu. V ukázce 7.2 figuruje pole `ageCounts`, které obsahuje počet instancí pro zvolený věk `currAgeInt` (celočíselná reprezentace věku). Taktéž se dá prohlásit, že `ageCounts` je taktéž kumulativní histogram četnosti vlasů podle věku. Pole `hairs` představuje uspořádanou množinu struktur typu `HairInstanceAttrib` z úvodu této kapitoly.

Listing 7.2: Ukázka naivního postupu kreslení vlasů v OpenGL.

```
15 vb->bind();
16 int nbHairs = ageCounts[currAgeInt];
17 for (int i = 0; i < nbHairs; ++i)
18 {
19     hairShader->setUniformValue("uHairPosition", hairs[i].p
       );
20     hairShader->setUniformValue("uHairNormal", hairs[i].n);
21     glDrawArrays(GL_LINES, 0, 2);
22 }
23 vb->release();
```

Tento přístup je velmi snadný, variabilní a dále je pochopitelně široce podporován napříč grafickými kartami, nicméně je zde jedna slabina. Kreslení každé instance se rovná jednomu požadavku na server a každý požadavek je vždy spojen s nějakým tím zdržením. Tento postup vyžaduje velmi vysoký podíl času stráveného na vzájemné komunikaci klienta a serveru a tudíž se pro větší počet instancí stává nevýhodným.

V knihovně OpenGL existuje funkce, která dovoluje vykreslit několik instancí jednoho objektu (v tomto případě přímka) s různými parametry, jako například pozice a barva, pouze jedním voláním funkce, která je uvedena v úryvku kódu 7.3, tak jak byla použita ve vytvořené aplikaci.

Listing 7.3: Ukázka volání příkazu OpenGL ke kreslení vlasů.

```
24 vb->bind();
25 instDataTex->bind(0);
26 glDrawArraysInstanced(GL_LINES, 0, 2, ageCounts[currAgeInt
       ]);
27 instDataTex->release(0);
28 vb->release();
```

Uvedený úryvek říká, že se má kreslit z aktuálně připojeného vertex bufferu `vb` několik přímek o dvou vrcholech s použitím informací uložených v textuře `instDataTex`. Jediný limitující požadavek na grafickou kartu je nutnost podpory rozšíření pod názvem `ARB_draw_instanced` nebo podpora verze OpenGL 3.1.

Aby vše mohlo fungovat, tak jak bylo výše popsáno je potřeba položky pole instancí seřadit podle věku (sestupně), tzn. na začátku pole budou nejstarší vlasy (některé nemizí vůbec) a na konci budou ty vlasy, které budou mizet nejdřív. Následná volba počtu instancí ke kreslení zahrne prvky od začátku pole až po zmíněnou hranici. Výsledkem je výběrové zobrazení vlasů odpovídajících zadanému věku a to vše lze realizovat jedním OpenGL příkazem.

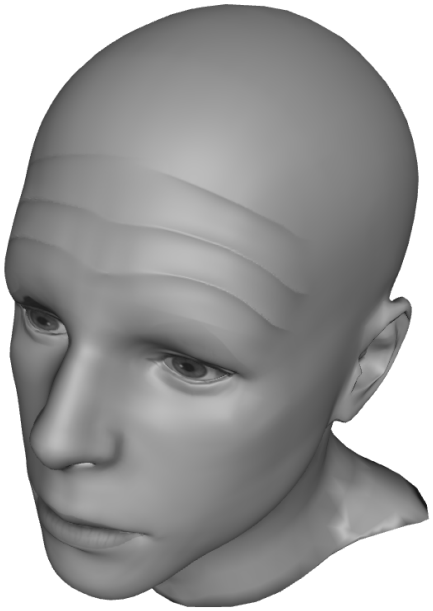
Implementační poznámkou budiž proces přenosu informací o instancích (pole `hairs`) na paměť GPU. Je potřeba uložit instance do textury (normála a pozice, věk, popř. další parametry). Při ukládání dat v textuře je k dispozici nějakých $W \times H$ pixelů, kde každý z nich může nést 4 datové položky (RGBA) typu `float`. Při vypočtení celkové velikosti struktury `HairInstanceAttrib` dostaneme celkovou velikost 32 bajtů. Na realizaci uvedeného postupu je potřeba podpory textur s 32-bit. `float` typem na jeden kanál, tj. každý pixel může nést až 4 proměnné typu `float` o celkové velikosti 16 bajtů. Každá instance tedy zabere 2 pixely v textuře. Zmíněný interní datový typ uložení pixelů v textuře je v OpenGL označován konstantou `GL_RGBA32F`.

Nakonec zbývá vypočítat vhodné rozměry textury, dostatečně velké, pro požadovaný počet instancí. Předpokládá se, že rozměry textury bližší čtvercovému poměru stran jsou rychleji zpracovávány. Dále pro větší využití přidělitelné paměti na grafickém HW bude snaha vypočítat rozměry co nejblíže poměru stran 1:1. Počet sloupců `width` se vypočte odhadem jako `(int)ceil( sqrt(nbHairs * 2))`. Počet řádků `height` je odvozen, tak aby platilo že `width * height >= nbHairs * 2`.

8 Výsledky

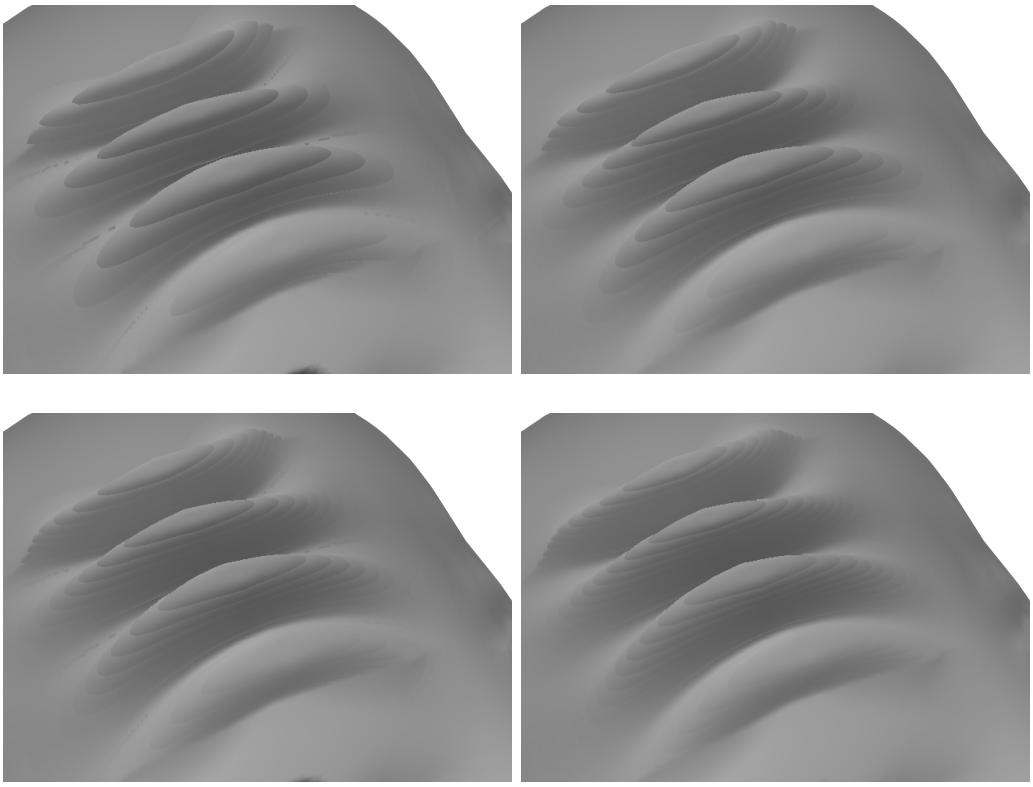
Veškeré testování bylo realizováno na jednom modelu hlavy. Pro zobrazení výsledků vrásek byl použit popisový textový soubor s vráskami.

Výsledek renderingu velkých vrásek na čele je na obrázku 8.1. K získání uvedeného tvaru vrásek bylo potřeba dvou svalů s větší relativní šířkou a jen mírně vyšší hodnotou minimální frekvence (`OmegaLow`). Tento příklad ukazuje, že existují situace, kdy uživatel může přizpůsobit tvar vrásek tak, aby vypadaly realisticky.



Obrázek 8.1: Příklad vizualizace čelních vrásek za pomocí dvou svalů.

Pro testování kvality zobrazování velkých vrásek byly porovnány tři varianty algoritmů. První varianta používá pouze kroky lineární, tj. lineární prohledávání pravidelně děleného rozsahu parametru paprsku. Druhá varianta pracuje pouze s binárním vyhledáváním, jindy také nazývaném metoda bisekce. Poslední možností je kombinace dvou předešlých s tím, že počet iterací je rozdělen v poměru 1:1. Pokud je používána kombinace obou typů vyhledávání mluví se právě o metodě POM.



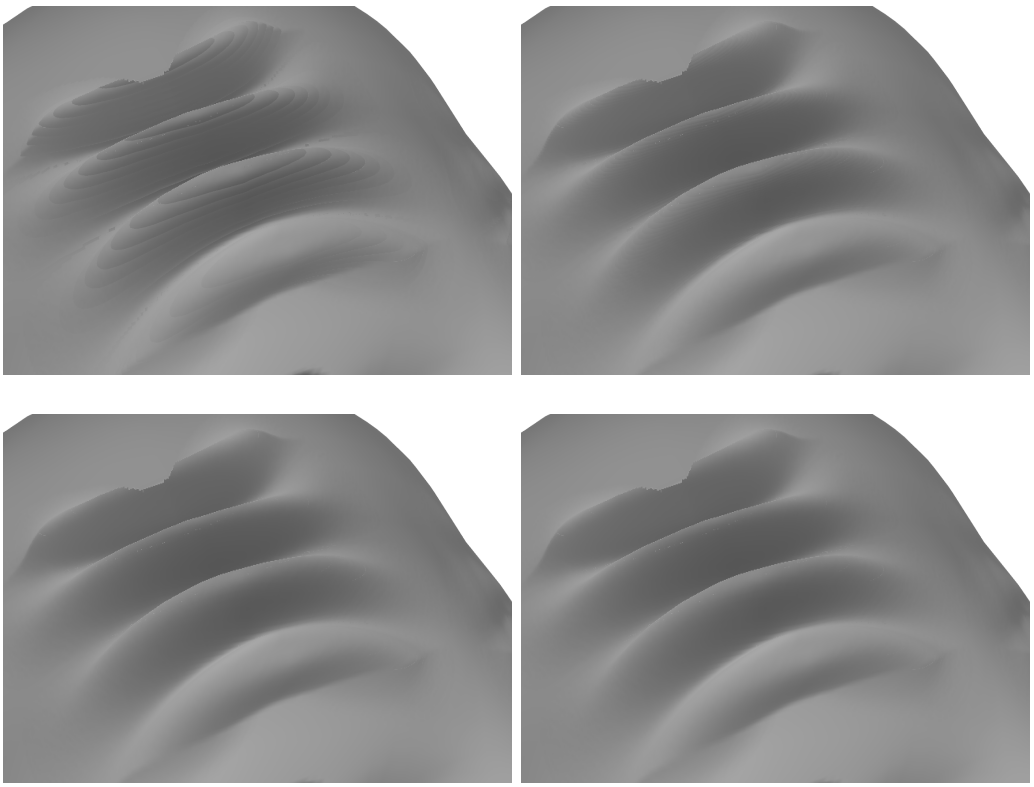
Obrázek 8.2: Výsledky vrásek při lineárním vyhledávání pro počty iterací 4 (vlevo nahoře), 6 (vpravo nahoře), 8 (vlevo dole) a 10 (vpravo dole).

Metoda POM byla otestována s důrazem na přesnost výpočtu aproximovaného povrchu a výpočetního času. V kapitole 7.1.2 bylo naznačeno, že metoda POM je metodou iterační hledající přibližné řešení. Proto v tomto kontextu je dobré vzít v úvahu jak metoda rychle konverguje a jestli jsou splněny podmínky konvergence.

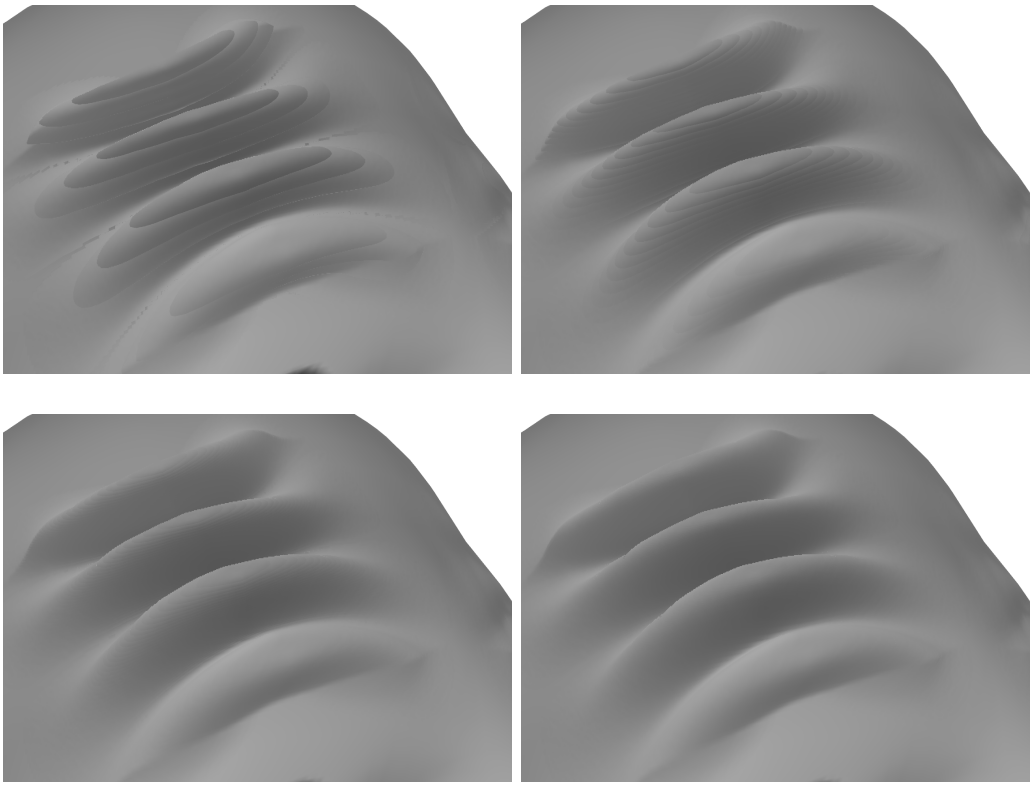
POM má omezenou hloubku vrásky v závislosti na kvalitě, kterou ještě považujeme za akceptovatelnou. Hloubka vrásek sice jde zlepšit zvýšením počtu iterací, na druhou stranu se však ztrácí výkonnost, tj. s vyšším počtem iterací klesá FPS.

Obrázek 8.2 představuje čtyři varianty počtu iterací pro samostatné lineární vyhledávání, dále obrázek 8.3 tutéž formu pro binární vyhledávání a samozřejmě nejde opomenout i kombinovaná metoda na obrázku 8.4.

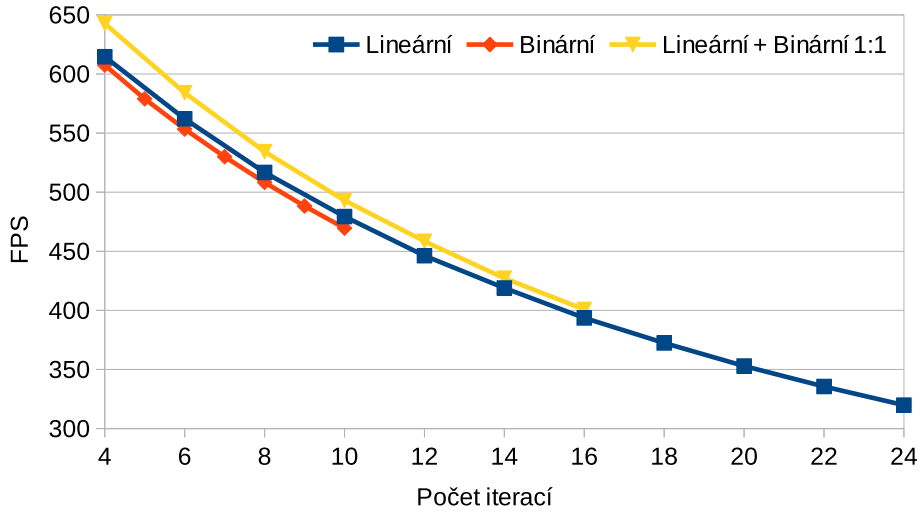
Obrázek 8.5 obsahuje záznam měření FPS na třech variantách POM, které uživatel může volit, se závislostí na zvoleném počtu iterací.



Obrázek 8.3: Výsledky vrásek při binárním vyhledávání pro počty iterací 4 (vlevo nahoře), 6 (vpravo nahoře), 8 (vlevo dole) a 10 (vpravo dole).



Obrázek 8.4: Výsledky vrásek při kombinovaném vyhledávání pro počty iterací 4 (vlevo nahoře), 6 (vpravo nahoře), 8 (vlevo dole) a 10 (vpravo dole), kde počet je rozdělen mezi obě metody v poměru 1:1.

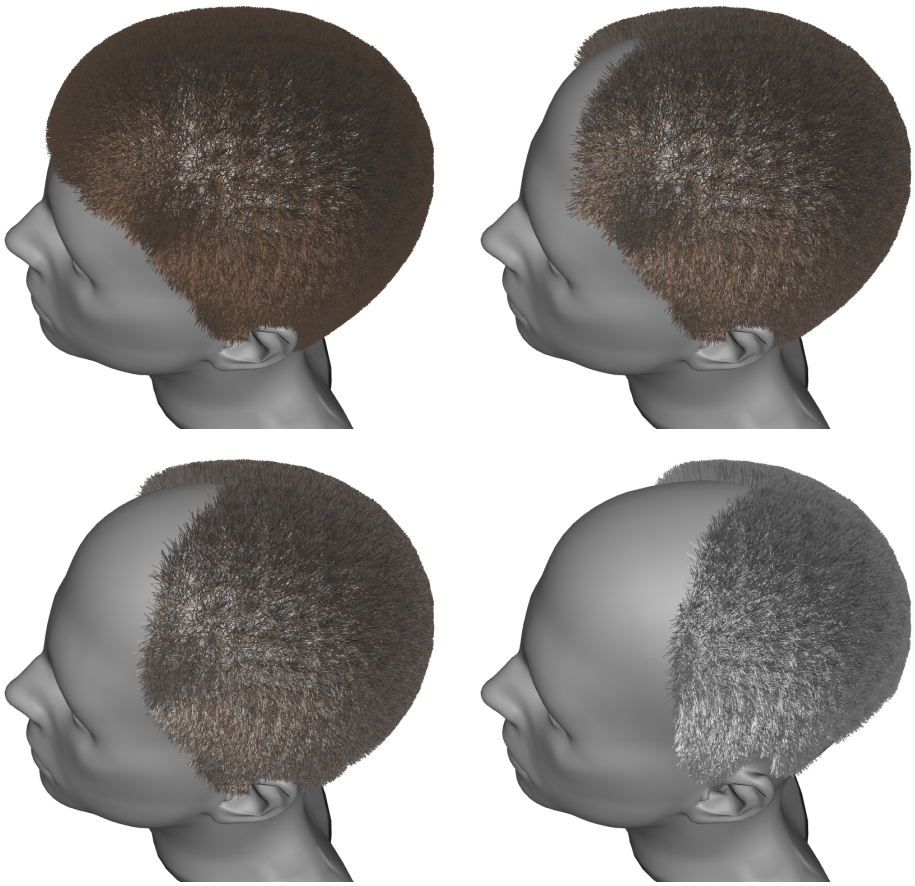


Obrázek 8.5: Graf velikosti FPS závislý na počtu iterací pro tři varianty algoritmu POM.

Na grafu 8.5 lze pozorovat, že lineární a binární metoda jsou přibližně stejně rychlé a kombinovaná metoda dokonce přináší nezanedbatelné urychlení, i když je potřeba stejné množství iterací. To, že křivky metod nekončí na stejném počtu iterací, je způsobeno dosažením stejné kvality. Od toho bodu už více iterací nepřináší viditelné vylepšení výsledku v daném pohledu a hloubce vrásek.

Binární vyhledávání konverguje nejrychleji ze všech uvedených možností. Je tu bohužel jedna nepříjemná zkutečnost, která zabraňuje tuto metodu nasazovat samostatně. Není totiž zaručeno, že binární vyhledávání nalezne vždy ten nejbližší průsečík. Uvedené tvrzení potvrzuje i obrázek 8.3 s viditelným zubem na horní vrásce.

Doporučený minimální počet lineárních iterací je alespoň 8 a pro binární je potřeba alespoň 4, kde už zpravidla nejsou žádné artefakty viditelné nezávisle na směru pohledu. Toto nastavení je nejrychlejší varianta s přijatelnou kvalitou výsledku. Je možné použít i více iterací, nicméně musí se počítat s poklesem FPS a s nevelkým nárůstem kvality. Z měření vyplynulo ještě další pozorování, že je důležitější mít zajištěnou konvergenci použitím více lineární kroků (stále jen minimální nutné) s tím, že binární vyhledávání je stále nezbytné k rychlému zpřesnění výsledku. Počet lineárních iterací by měl být větší než počet binárních iterací.



Obrázek 8.6: Ukázka úbytku vlasů ve čtyřech věkových fázích v pořadí: vlevo nahoře, vpravo nahoře, vlevo dole a vpravo dole.

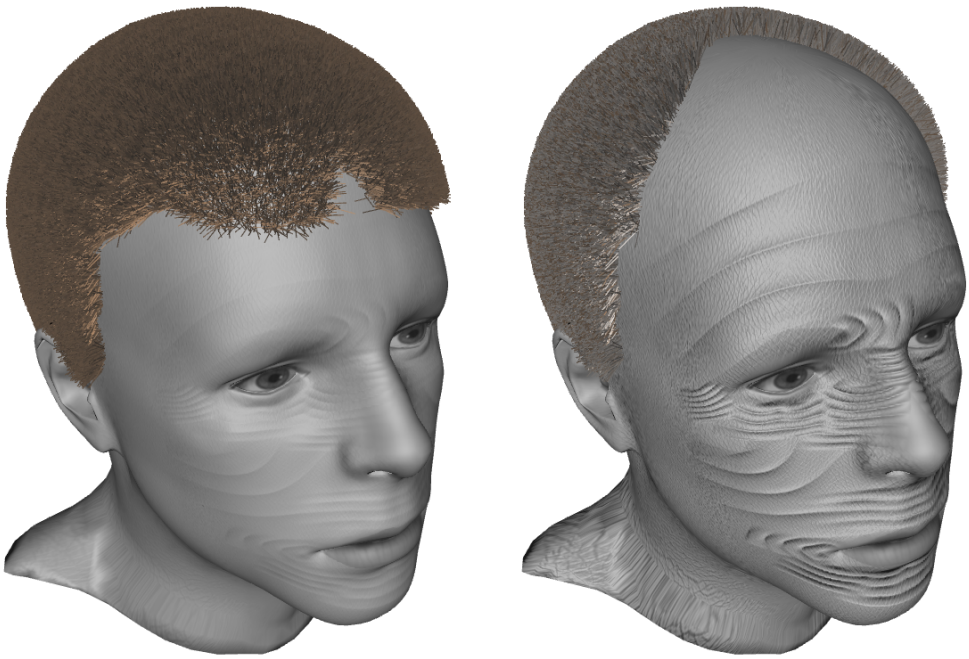
Testování vrásek bylo prováděno pro jednu velikost vrásek a ta odpovídala volbě parametru `dispScale` o hodnotě `0.005`. Tato hodnota je uživateli skryta tím, že dotyčný zadává relativní hodnotu vůči zvolenému maximu. Z pohledu GUI, kde zadává právě tato relativní výška vrásek, se jedná o poloviční intenzitu (minimum je `0.0` a maximum je `1.0`). Uvedená proměnná je nezbytným argumentem funkce hledající průsečík s výškovou mapou a její použití je zachyceno v ukázce kódu v kapitole A.4 na straně 53.

Obrázek 8.6 demonstruje, jak funguje úbytek vlasů na modelu lidské hlavy. Snímky byly pořízeny ve čtyřech fázích, přičemž první snímek je dán vnější oblastí a poslední snímek je dán vnitřní oblastí. Generace vlasů se správně chová i v případě, kdy uživatel zadá vnitřní oblast sahající mimo vnější oblast nebo alespoň některou část. Uvedená oblasti se v simulaci

úbytku vlasů projeví tak, že vlasy v této oblasti neubudou. V pixelech kde je podmínka, vnitřní oblast uvnitř vnější, splněna se simulace bude chovat běžným způsobem. Realističnost renderingu vlasů na obrázku 8.6 viditelně zlepšuje osvětlovací model s lesklými odlesky.

Ještě je třeba zmínit omezení počtu vlasů, které lze generovat. Maximální počet vlasů je nutně spojen s maximální velikostí textury dostupné na konkrétní grafické kartě. Na dnešním hardwaru je běžně dostupná maximální velikost textur kolem 8192×8192, což umožňuje kreslit přibližně 20 milionů vlasů, což je více než dostačující pro účely této práce a možná i dalších projektů. V rámci testování této práce byly generovány řádově stovky tisíc vlasů a víc než dostatečně postačovaly na pokrývku hlavy.

Na obrázku 8.7 je příklad celkového stárnutí hlavy, zahrnující velké vrásky, jemné vrásky a stárnutí vlasů. Realističnost scény je ovlivněna zejména vstupními daty. Například oblasti výskytu vrásek by měly být voleny citlivěji, s ohledem na některé partie obličeje. Tato vlastnost je spojena s kreslením masky velkých vrásek. Dále jako příklad lze uvést úbytek vlasů. Zde nevyšel ideálně koncový stav vlasů. Lepší volbou textury s vnitřní oblastí růstu vlasů může přinést lepší výsledky.



Obrázek 8.7: Simulace stárnutí hlavy ve dvou fázích: mládí (vlevo) a stáří (vpravo).

9 Závěr

Tato práce měla za cíl nalézt dostupné metody simulace stárnutí, z nich vybrat některou vhodnou metodu, tu implementovat a otestovat na dodaných datech. Simulace byla rozdělena na dvě oblasti: simulace vrásek a simulace stárnutí vlasů. Uvedené projevy stárnutí byly vybrány jako nejdůležitější vizuální prvky zejména pro kriminalistickou praxi, zábavní nebo reklamní průmysl.

Vytvořený model vrásek není ideálním řešením pro celý obličej. Pro dosažení větší realističnosti se musí rozmístit větší množství malých svalů. Zde je možnost dále rozšířit program o další typy vrásek, které zejména pracují v jiném souřadném systému, např. pro vrásky kolem očí se hodí model založený na sférických, popř. eliptických souřadnicích. Vektorově orientovaný model je výhodný v tom, že je nezávislý na cílovém rozlišení textury a může mít přiřazeno libovolné množství parametrů.

Velmi potěšující výsledky vykazuje POM, kde se ukázalo, že tato metoda poskytuje více než dostačující hloubku vrásek pro lidskou hlavu. Počet iterací metody POM ovlivňuje rychlost vykreslování (v testování sledováno FPS) a nejlepších výsledků se dosahuje kombinací lineární fáze vyhledávání a binární fáze vyhledávání. Poměr počtu iterací pro lineární/binární vyhledávání je nevhodnější volit 1:1 a vyšší, tj. počet kroků lineárního vyhledávání by měl být vyšší. Obecně pro použitelné výsledky by neměl být počet iterací menší než 4.

Generace vlasů má poměrně větší výpočetní nároky (potažmo časovou složitost) než například generace vrásek, přesto je třeba zdůraznit, že potřebné výpočty simulace stárnutí vlasů se provádějí jako předzpracování, tj. nezpomalují samotné vykreslování. Algoritmus pracuje na principu iterativního rozšiřování vnitřní oblasti až po hranici danou vnější oblastí. To, že se práce provádí v rastrovém prostředí není omezen tvar obou oblastí a dokonce obě oblasti nemusí tvořit jednu souvislou plochu.

V případě úbytku vlasů, bylo z pohledu grafické karty dosaženo vysoké efektivity kreslení. Zobrazení vlasů ve zvoleném věku vyžaduje po GPU pouze nastavení věku, poté se provede zpracování geometrie a rasterizace vlasů jedním OpenGL voláním. Implementace stínování pomocí shaderů v jazyce GLSL dále umožňuje tvořit základní i komplikované modely osvětlení vlasů, které mohou podpořit realističnost simulace stárnutí. V této práci jsem si

vyzkoušel jeden jednodušší model stínování. Budoucí zavedení jiné metody kreslení vlasů může být snadné pokud bude zachováno kreslení vlasů (nebo jednoho trsu vlasů) jako instancí s různým směrem a polohou.

Program lze používat jako dll knihovnu umožňující propojení s existujícím projektem Ing. Petra Martínka.

10 Seznam zkratek

AOI Area of Influence, oblast vlivu

CPU Central Processing Unit

DLL Dynamic-link library, dynamicky linkovaná knihovna

FPS Frames Per Second, počet snímků za sekundu

GLSL OpenGL Shading Language

GPU Graphic Processing Unit

GUI Graphical User Interface

HW Hardware

NM Normal Mapping

OpenGL Open Graphics Library

POM Parallax Occlusion Mapping

RGB(A) Red, Green, Blue – barevný model

Literatura

Bando, Y. – Kuratate, T. – Nishita, T. A Simple Method for Modeling Wrinkles on Human Skin. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, PG '02, s. 166–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1784-6.

Blinn, J. F. Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics*. 1978, 12, 3, s. 286–292.

Boissieux, L. et al. Simulation of Skin Aging and Wrinkles with Cosmetics Insight. In Magnenat-Thalmann, N. – Thalmann, D. – Arnaldi, B. (Ed.) *Computer Animation and Simulation 2000*, Eurographics. Springer Vienna, 2000. s. 15–27. ISBN 978-3-211-83549-4.

Golovinskiy, A. et al. A Statistical Model for Synthesis of Detailed Facial Geometry. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, s. 1025–1034, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6.

Green, S. Real-time approximations to subsurface scattering. *GPU Gems*. 2004, s. 263–278.

Haro, A. – Guenter, B. – Essa, I. Real-time, photo-realistic, physically based rendering of fine scale human skin structure. In *Rendering Techniques 2001*. Springer, 2001. s. 53–62.

He, Q. – Tong, M. – Liu, Y. Face Modeling and Wrinkle Simulation Using Convolution Surface. In Perales, F. – Fisher, R. (Ed.) *Articulated Motion and Deformable Objects*, 4069 / *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006. s. 244–251. ISBN 978-3-540-36031-5.

Igarashi, T. – Nishino, K. – Nayar, S. K. The appearance of human skin. 2005.

Kajiya, J. T. – Kay, T. L. Rendering fur with three dimensional textures. In *ACM Siggraph Computer Graphics*, 23, s. 271–280. ACM, 1989.

Lengyel, E. *Mathematics for 3D Game Programming and Computer Graphics*. Game development series. Course Technology/Cengage Learning, 2011. ISBN 9781435458864.

Marschner, S. R. et al. Light scattering from human hair fibers. In *ACM Transactions on Graphics (TOG)*, 22, s. 780–791. ACM, 2003.

Nealen, A. et al. Physically based deformable models in computer graphics. In *Computer Graphics Forum*, 25, s. 809–836. Wiley Online Library, 2006.

Nehab, D. ANSI C Library for PLY file format input and output, 2013. Dostupné z: <http://www.impa.br/ diego/software/rply>.

Oat, C. Animated wrinkle maps. In *ACM SIGGRAPH 2007 courses*, s. 33–37. ACM, 2007.

P. Volino, N. M. T. Fast Geometrical Wrinkles on Animated Surfaces. *Proc. VSCG '99*. 1999.

Pakdel, H. – Samavati, F. F. Incremental Subdivision for Triangle Meshes. *Int. J. Comput. Sci. Eng.* July 2007, 3, 1, s. 80–92. ISSN 1742-7185.

Pharr, M. – Humphreys, G. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., 2nd edition, 2010. ISBN 0123750792, 9780123750792.

Ramanathan, N. – Chellappa, R. – Biswas, S. Computational Methods for Modeling Facial Aging: A Survey. *J. Vis. Lang. Comput.* June 2009, 20, 3, s. 131–144. ISSN 1045-926X.

Reis, C. D. G. – Bagatelo, H. – Martino, J. Real-time simulation of wrinkles. In *Proc. WSCG*, s. 109–116, 2008.

Sander, P. V. – Gosselin, D. – Mitchell, J. L. Real-time skin rendering on graphics hardware. In *the proceedings of SIGGRAPH*, 2004.

Szirmay-Kalos, L. – Umenhoffer, T. Displacement Mapping on the GPU - State of the Art. *Comput. Graph. Forum*. 2008, 27, 6, s. 1567–1592.

Tatarchuk, N. Dynamic parallax occlusion mapping with approximate soft shadows. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, s. 63–69. ACM, 2006.

Yuksel, C. – Schaefer, S. – Keyser, J. On the Parameterization of Catmull-Rom Curves. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, SPM '09, s. 47–53, New York, NY, USA, 2009a. ACM. ISBN 978-1-60558-711-0.

Yuksel, C. – Schaefer, S. – Keyser, J. Hair meshes. In *ACM Transactions on Graphics (TOG)*, 28, s. 166. ACM, 2009b.

Žára, J. et al. *Moderní počítačová grafika*. Computer Press, 2004. ISBN 9788025104545.

Zhang, Y. – Sim, T. Realistic and Efficient Wrinkle Simulation Using an Anatomy-based Face Model with Adaptive Refinement. In *Proceedings of the Computer Graphics International 2005*, CGI '05, s. 3–10, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9330-9.

Zinke, A. et al. Dual scattering approximation for fast multiple scattering in hair. In *ACM Transactions on Graphics (TOG)*, 27, s. 32. ACM, 2008.

A Přílohy

A.1 Uživatelská příručka

K realizaci programu byly použity dvě knihovny: Qt SDK verze 5.2.1 od společnosti Digia a knihovna *RPly* ze stránky (Nehab, 2013). Implementace vizualizační části práce byla vytvořena za pomocí rozhraní OpenGL a jazyka GLSL. Pro správný běh programu je potřeba podpora verze OpenGL 3.1 a vyšší.

A.1.1 Překlad a spuštění

K překladu je potřeba distribuce Qt alespoň 5.2, nižší verze nebyly testovány. Na přiloženém DVD je připravená verze 5.3 používající překladač MinGW. Existují dvě cesty, jak spustit překlad. První možnost se provádí prostřednictvím příkazové řádky:

```
qmake
make
```

Druhou možností je překládat v prostředí Qt Creator, kde stačí otevřít soubor `DIP_2014_Janecek_AgingSim.pro` umístěný v kořenové složce projektu. Při otevření projektu budete dotázáni na volbu adresářů, určených pro výsledek překladu. Výsledek překladu je pak k nalezení buď ve zvolených složkách, nebo v adresáři `bin` (v kořenovém adresáři projektu).

A.2 Rozhraní zásuvného modulu

Vytvořený program nabízí i DLL rozhraní v nativním prostředí. Pro integraci do aplikace napsané C# a běžící pod *.NET*, jsem vytvořil ukázkovou aplikace v uvedeném jazyce, řešící převod dat z managed do unmanaged prostředí. Ukázková aplikace se jmenuje `SampleAgingSimClientCS` a je k nalezení přímo v kořenovém adresáři projektu. Třída `AgingSimLibraryCS` obsahuje metodu, kterou lze spustit okno stárnutí s předanými daty.

```
29 public static int runAgingSimWindow(AgingInputMesh mesh);
```

Struktura `AgingInputMesh` je definována takto:

```
30 public struct AgingInputMesh
31 {
32     public VertexPTN[] vertices;
33     public uint[] indices;
34 }
```

Každý vrchol `VertexPTN` obsahuje pozici, texturovací souřadnici a normálu plochy k objektu (vertex normal) vztaženou k pozici vrcholu.

```
35 public struct VertexPTN
36 {
37     float px;
38     float py;
39     float pz;
40     float tcx;
41     float tcy;
42     float nx;
43     float ny;
44     float nz;
45 }
```

A.3 Formát definice vrásek

Formát nabízí hlavní dva druhy příkazů. První je znak `!`, který vytváří novou vrásku. Druhá skupina je definice parametrů aktivní vrásky. Pořadí parametrů nemusí být dodrženo. Povinné parametry jsou `ma`, `mb` a `n`. Volitelné parametry jsou `RelWidth`, `MaxAngle` a `OmegaLow`. Následující tabulka obsahuje výčet parametrů a jejich vlastnosti.

Název	**Formát, rozsah**	**Výchozí**	**Popis**
`ma`	`float;float`	neznámá	První bod vlákna
`mb`	`float;float`	neznámá	Druhý bod vlákna
`n`	`integer`, vetší než 1	neznámá	Počet vrásek
`RelWidth`	`float`, `(0, 1>`	`0.5`	Rel. šířka vůči vláknu
`MaxAngle`	`float`, `<1, 2>`	`0.25`	Max. odchylka od vlákna
`OmegaLow`	`float`, `<0, 1>`	`0.5`	Min. frekvence

Parametr `MaxAngle` má povolený rozsah hodnot `<1,2>` a ten odpovídá vzájemnému úhlu svalového vlákna a průvodiče bodu kdekoliv na vrásce v rozsahu $\langle \pi/2, \pi \rangle$.

Komentáře jsou jednořádkové, jsou uvozeny znakem `#` a jsou ukončeny novou řádkou. Jsou podporovány i prázdné řádky a mezery před, mezi a po příkazu.

Následující ukázka souboru představuje zadání dvou svalů formujících vrásky na čele pro testovaný model.

```
# Toto je komentar kde lze psat cokoliv.
# Celo vlevo
!
ma=0.423;0.727
mb=0.439;0.640
n=3
relWidth=1.0
maxAngle=1.5
omegaLow=0.6

# Celo vpravo
!
ma=0.556;0.727
mb=0.537;0.638
n=3
relWidth=1.0
maxAngle=1.5
omegaLow=0.6
```

A.4 Paralax Oclussion Mapping shader

Zde je uveden kód funkce počítající průsečík s výškovou mapou. Návratová hodnota funkce `traceOccl` je rovna rozdílu výchozí texturovací souřadnice a souřadnice s nalezeným průsečíkem. Hodnota `uLinearStepSize` by měla být spočtena na klientovi jako `1.0 / uLinearIterCount`, kde `uLinearIterCount` a `uBinaryIterCount` musí být celá čísla větší než 1. Úryvek je vyňat ze souboru `wrinkle.glsl` v adresáři `shaders`.

Listing A.1: Funkce hledající průsečík paprsku pohledu a výškové mapy zapsaná v jazyce GLSL.

```
46 uniform int uLinearIterCount;
47 uniform int uBinaryIterCount;
48 uniform float uLinearStepSize;
49
50 vec2 traceOccl(in sampler2D tex, in vec2 texCoordsIn, in
       vec3 view, in float dispScale)
51 {
52     float h = 0.0;
53     vec3 uvStep = vec3(view.xy * uLinearStepSize *
           dispScale, uLinearStepSize);
54     vec3 uv = vec3(texCoordsIn, 1.0);
55
56     // 1. krok: linearni vyhledavani
57     uv -= uvStep;
58     for (int i = 0; i < uLinearIterCount; i++)
59     {
60         h = texture2D(tex, uv.xy).a;
61         // Ekvivaletni s if (h >= uv.z) break;
62         uv -= uvStep * step(h, uv.z);
63     }
64
65     // 2. krok: binarni vyhledavani
66     for (int i = 0; i < uBinaryIterCount; i++)
67     {
68         h = texture2D(tex, uv.xy).a;
69         uvStep *= 0.5;
70         // Ekvivaletni s if (h >= uv.z) uv += uvStep; else
               uv -= uvStep;
71         uv += uvStep * sign(h - uv.z);
72     }
73
74     return uv.xy - texCoordsIn;
75 }
```

A.5 Obsah CD

AgingSimApp Zdrojové kódy nativní aplikace stárnutí.

AgingSimLibrary Zdrojové soubory knihovny obsahují hlavní část řešení práce. Překladem tohoto projektu se vytvoří **AgingSimLibrary.dll** v adresáři `bin`.

bin Spustitelné soubory nativní aplikace.

data Ukázkové soubory sloužící jako vstupní data – model hlavy, textury a popisové soubory vrásek.

doc Dokumentace k práci.

SampleAgingSimClientCS Ukázková aplikace s managed rozhraním napsaném v jazyce C# pro nativní knihovnu **AgingSimLibrary.dll**. K dispozici je i projekt pro Microsoft Visual Studio 2013. Tento projekt je třeba překládat samostatně.

DIP__2014__Janecek__AgingSim.pro Projekt pro překlad celého projektu v Qt prostředí. Výstupem je přeložená nativní knihovna tak nativní aplikace s umístěním v adresáři `bin`.