VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VOLUMETRICKÉ EFEKTY AKCELEROVANÉ NA GPU

VOLUMETRIC EFECTS ACCELERATED ON GPU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ KUBOVČÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ STARKA**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/20001/2016/xkubov02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Kubovčík Tomáš, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Volumetrické efekty akcelerované na GPU**
Volumetric Efects accelerated on GPU
Kategorie: Počítačová grafika

Pokyny:

1. Nastudujte volumetrické efekty jako hoření, dým.
2. Nastudujte knihovny pro GPU akceleraci.
3. Implementujte vybraný volumetrický efekt pomocí vybrané knihovny.
4. Vytvořte demonstrační aplikaci.
5. Vytvořte video k aplikaci.

Literatura:

- Po dohodě s vedoucím

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Starka Tomáš, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá simuláciou a vykresľovaním volumetrických efektov založených na toku tekutín, najmä efektu ohňa a dymu. Výpočty sú akcelerované na grafickej karte použitím moderného grafického API s cieľom dosiahnutia realistických vizuálnych výstupov ako aj fyzicky korektných výpočtov. Implementované volumetrické efekty sú sprístupnené vo forme dynamickej knižnice, umožňujúcej rozšírenie stávajúcich aplikácií o dané efekty.

Abstract

This thesis deals with simulation and rendering of fluid based volumetric effects, especially effect of fire and smoke. Computations are accelerated on graphics card using modern graphics API with motivation to achieve realistic visual results as well as physically correct calculations. Implemented volumetric effects are distributed as dynamic library which allows addition of these effects to existing applications.

Kľúčové slová

volumetrické efekty, simulácia tekutín, Navier-Stokesove rovnice, oheň, dym, advekcia, compute shader, GPU, diskretizácia, osvetlenie, tiene, mriežka, voxel, 3D textúra, raymarching, volumetrický scattering, vyžarovanie, prekážky.

Keywords

volumetric effects, fluid simulation, Navier-Stokes equations, fire, smoke, advection, compute shader, GPU, discretization, lighting, shadows, grid, voxel, 3D texture, raymarching, volumetric scattering, radiation, obstacles.

Citácia

KUBOVČÍK, Tomáš. *Volumetrické efekty akcelerované na GPU*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Starka Tomáš.

Volumetrické efekty akcelerované na GPU

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Tomáša Starku a uviedol som všetky literárne pramene a publikácie z ktorých som čerpal.

.....
Tomáš Kubovčík
23. mája 2017

Poďakovanie

Ďakujem vedúcemu práce Ing. Tomášovi Starkovi za rady pri riešení práce, Radovanovi Švachulovi, Miroslavovi Mudroňovi a Petrovi Molnárovi za poskytnutie hardvéru pre vykonanie testov aplikácie, rodine a kamarátom za podporu.

Obsah

Kapitola 1

Úvod

Volumetrické efekty sú neoddeliteľnou súčasťou počítačových hier a filmov, či už sa jedná o rôzne CGI(*Computer-generated imagery*) efekty ohňa, explózie, časticové systémy dymu, atmosférické javy či rôzne formy kvapalín. Mnohé z týchto efektov sa z fyzikálneho hľadiska chovajú ako tekutiny, čím sa otvára možnosť tieto efekty simulovať vhodnými postupmi ako sú tzv. *computation fluid dynamics*. Tieto techniky sú dostupné už niekoľko rokov, no pre svoju komplexnosť a náročnosť na výpočtový výkon bolo ich použitie v real-time aplikáciách nereálne. Vďaka stále platiacemu Moorovmu zákonu však každým rokom výrobcovia grafických adaptérov produkujú výkonnejšie čipy, ktoré tieto techniky v súčasnej dobe dokážu vhodne modifikovať s cieľom dosiahnutia veľmi realistických výsledkov v reálnom čase.

V rámci tejto práce sa zoznámime z históriou techník zobrazovania a volumetrických efektov (kapitola 5), či už sa jedná o ad-hoc riešenia ako billboarding, jeho modifikácie alebo komplexnejšie techniky ako časticové systémy či voxelové techniky a ich prípadné kombinácie.

Nakoľko sa táto práca zaoberá najmä efektami ohňa a dymu, ktoré sú z fyzikálneho hľadiska tekutiny, pozrieme sa na základ všetkých súčasných pokročilých techník vykreslovania týchto volumetrických efektov - simulátor tekutín. V kapitole 3.1 preto rozoberieme Navier-Stokesove rovnice, na ktorých je väčšina týchto simulátorov postavená. Pozrieme sa na potenciálne možnosti riešenia týchto rovníc a položíme teoretický základ pre možnú implementáciu takéhoto simulátoru. V neposlednej rade ľahko nazrieme do konkrétnych problémov, ktorým pri implementácii simulátoru programátor musí čeliť.

Kapitola 2

Tekutiny

Nakoľko cieľom tejto práce je predstavenie volumetrických efektov využívajúcich simuláciu tekutín, je dôležité porozumieť, čo to vlastne tekutiny sú. V tejto kapitole definujeme tekutinu tak ako ju vníma človek a popíšeme model pohybu tekutiny. Popis fyzikálnych vlastností tekutín, je mimo rozsah tejto práce a preto budú uvedené iba fyzikálne zákonitosti potrebné pre pochopenie uvádzanej problematiky. Pre hlbšie znalosti fyzikálnych princípov tektutín, je vhodné siahnuť po odbornej literatúre [1].

Tekutiny nás obklopujú takmer všade, či už sa jedná o jednoduché fyzikálne javy ako horiaci kus papiera, dym stúpajúci z cigarety, či masívne astrofyzikálne javy ako sústavy oblakov či vznikajúce hurikány. Typickými predstaviteľmi tekutín z bežného života sú napríklad

- oheň
- dym
- hmla
- oblaky
- lokálne a globálne vetry
- kvapaliny

Je zrejmé, že povaha mnohých z uvedených fenoménov sú v mnohých veciach rozdielne, z čoho vyplýva obtiažnosť generického simulovania týchto javov. Na druhej strane, niektoré z daných javov sú si v mnohom podobné, napríklad oheň a dym majú rovnakú povahu ako aj vietor a hmla.

2.1 Oheň

Oheň je jedným z fenoménov, ktoré sú typickým predstaviteľom volumetrických efektov. Vhodným neformálnym popisom tohto javu, môžme považovať definíciu voľne dostupného anglického slovníka[1], ktorý ho popisuje nasledovne:

"Oheň je prudká, trvalá chemická premena exotermickou oxidáciou horľavej látky, ktorá uvoľňuje teplo a svetlo a je sprevádzaná plameňmi."

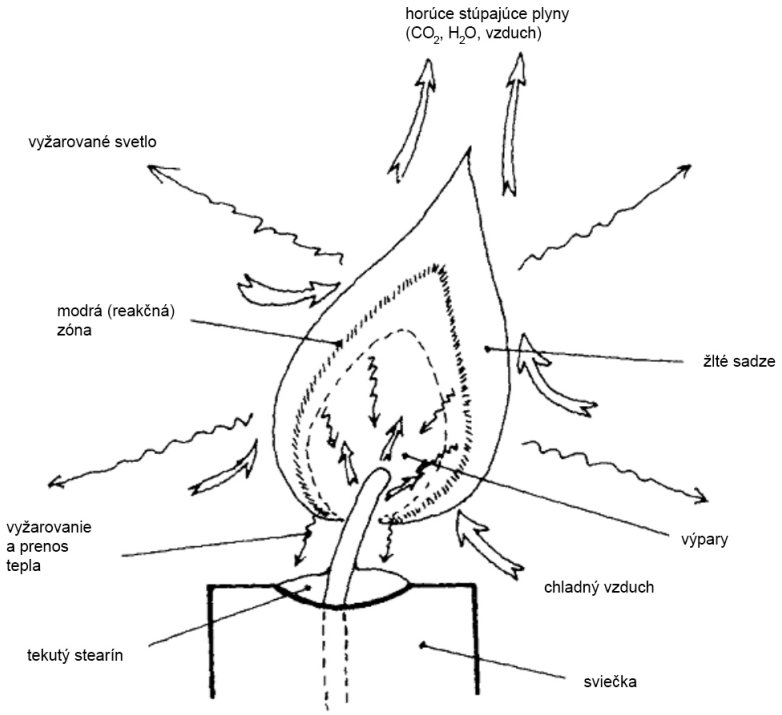
[1] http://www.thefreedictionary.com/fire

2.1.1 Fyzika ohňa

Na oheň sa však môžme pozerať z pohľadu rozličných prírodných vied. Z chemického hľadiska oheň predstavuje chemickú reakciu, ktorej hlavnými reaktantmi sú oxidant a plynové výpary.
Z mechanického hľadiska sa jedná o prenos tepla. To spočíva v difúzii, prúdení a vyžarovaní. Difúzia a prúdenie zabezpečujú najmä šírenie ohňa, pričom vyžarovanie čierneho telesa (angl. *black body radiation*) dodáva vizuálny vzhľad, ktorí si asociujeme s ohňom, teda typickú žlto-oranžovú farbu.
Napokon z fyzického hľadiska sa jedná o prenos hmoty, ktorý vďaka vztlaku a tokom spôsobuje charakteristický tvar a pohyb ohňa. Okrem toho dochádza k molekulárnej difúzii kvôli rozdielnym zloženiam plynov v rôznych častiach plameňov, čím vznikajú ďalšie chemické reakcie.

2.1.2 Vyžarovanie čierneho telesa

Počas chemickej reakcie horenia (spaľovania) vznikajú rôzne horúce plynové prvky. Vyžarovanie čierneho telesa vydané týmito plynovými prvkami, najmä uhlíkovými sadzami spôsobuje spomínaný vizuálny dojem ohňa a plameňov. Počas reakcie vzniká množstvo tepla, ktoré spúšťa ďalšie procesy spaľovania a podniecuje vznik ďalších plynových prvkov. Teplota klesá so stúpajúcou vzdialenosťou od centra reakčnej oblasti, čo vedie k dekrementácii vyžarovania absolútne čierneho telesa až do stavu, kedy oheň prestáva byť pre ľudské oko viditeľný. Znížením teploty vzniknutých plynových prvkov na určitú úroveň spôsobí to, že sa javia ako dym a sadze.



Obrázok 2.1: Fyzikálne javy v plameni sviečky.

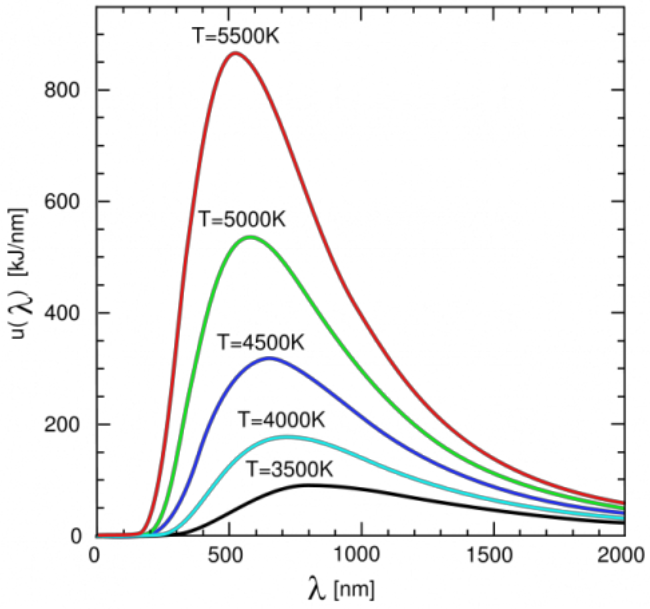
Absolútne čiernymi telesami, ktoré vydávajú žiarenie čierneho telesa sú v prípade fenoménu ohňa častice sadzí, ktoré vznikajú pred a po procese horenia. Planckova rovnica 2.1 vyjadruje intenzitu určitej vlnovej dĺžky λ vyžarovanej absolútne čiernym telesom.

$$B_\lambda(T) = \frac{2\pi hc^2}{\lambda^5(e^{\frac{hc}{\lambda kT}} - 1)} \tag{2.1}$$

kde

- T je teplota
- h je Planckova konštanta
- c je rýchlosť svetla
- k je Boltzmannova konštanta

Na obrázku 2.2 je zobrazená charakteristika spektra vyžarovania absolútne čiernym telesom ako funkcia vlnovej dĺžky. Je zrejmé, že vydaná energia prudko stúpa s teplotou. Viditeľné spektrum sa rozprestiera medzi 400 a 700 nm, pričom vyžarovanie absolútne čierneho telesa má vrchol pri pravej hranici tohto spektra, čo odpovedá práve červenej, oranžovej a žltej farbe.



Obrázok 2.2: Spektrum vyžarovania absolútne čierneho telesa ako funkcia vlnovej dĺžky.

Kapitola 3

Simulácia tekutín

Problematika simulácie tekutín má široké uplatnenie v počítačovej grafike. Jedným z dôvodov môže byť to, že tekutiny sú takmer všade, či už sa jedná o typického predstavitela tekutín - kvapaliny alebo rôzne plyny. Kvalitný simulátor tekutín (angl. *fluid solver*) nájde svoje uplatnenie najmä v oblasti špeciálnych efektov, či už vo filmových scénach, počítačových hrách, či rôznych grafických demách. Rovnako nájde svoje uplatnenie aj v aplikáciách pre malovanie, kde umožňuje vytváranie dojmu akvarelov a olejových malieb.

3.1 Navier-Stokesove rovnice

Jedným z štandardných matematických modelov pre popis toku tekutín sú známe *Navier-Stokesove* rovnice [18]. Vo fyzike rovnako ako v počítačovej grafike často využívame zjednodušené modely, ktoré daný fenomén iba aproximujú, zanedbávajúc nepodstatné detaily. Simulácia tekutín nie je žiadnou výnimkou, konkrétne v prípade Navier-Stokesových rovníc uvažujeme iba tekutiny, ktoré sú nestlačiteľné a homogénne. **Nestlačiteľné** tekutiny sú také, ktorých objem je konštantný v čase pre ľubovoľný subregión tekutiny. **Homogénne** tekutiny sú také tekutiny ktorých hustota ρ je konštantná v priestore. Kombináciou týchto vlastností získame model tekutiny, ktorý je dostatočne všeobecné a presné, aby sme boli schopný nimi popísať reálne tekutiny (voda, vzduch, oheň, dym, atď). Okrem toho matematický model nestlačiteľnej homogénnej tekutiny popisuje tekutinu s konštantnou hustotou v čase aj priestore. Navier-Stokesove rovnice boli odvodené z druhého Newtonovho pohybového zákona - zákona sily

$$\vec{F} = m \cdot \vec{a} \tag{3.1}$$

kde

- $\vec{F}$ je sila
- m je hmotnosť
- $\vec{a}$ je zrýchlenie

V kompaktnej forme, ktorá je často uvádzaná v literatúre predstavujúcou Navier-Stokesové rovnice sú tieto uvedené nasledovne:

$$\nabla \cdot \mathbf{u} = 0 \tag{3.2}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla \mathbf{p} + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{f} \tag{3.3}$$

kde

- ν je kinematická viskozita tekutiny
- ρ je jej hustota
- `f` je externá sila
- ∇ je gradient, ∇^2 značí $\nabla \cdot \nabla$

Gradient ∇ (*nabla*), je definovaný ako vektor priestorových parciálnych derivácií

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^T \tag{3.4}$$

pričom pre pole skalárov $s(x)$ definujeme gradient skalárneho pola

$$\nabla s(x) = \left(\frac{\partial s(x)}{\partial x}, \frac{\partial s(x)}{\partial y}, \frac{\partial s(x)}{\partial z}\right)^T, x \in \mathbb{R}^3 \tag{3.5}$$

Je nutné poznamenať, že sa jedná o zjednodušené Navier-Stokesove rovnice. Získame ich tvrdením, že tekutina zachováva objem (rovnica 3.2) a hybnosť (rovnica 3.3).
Na prvý pohľad vypadajú Navier-Stokesove rovnice odstrašujúco, no pozornému oku neunikne, že napr. v rovnici 3.3 nájdeme v podstate 4 vzťahy pre popis zrýchlenia.

3.1.1 Advekcia

Advekcia predstavuje schopnosť tekutiny prenášať (transportovať) objekty, hustoty a iné vlastnosti v toku. V Navier-Stokesových rovniciach pre popis hybnosti sa jedná o vzťah

$$-(\mathbf{u} \cdot \nabla)\mathbf{u} = -\begin{pmatrix} u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} + u_z \frac{\partial u_x}{\partial z} \\ u_x \frac{\partial u_y}{\partial x} + u_y \frac{\partial u_y}{\partial y} + u_z \frac{\partial u_y}{\partial z} \\ u_x \frac{\partial u_z}{\partial x} + u_y \frac{\partial u_z}{\partial y} + u_z \frac{\partial u_z}{\partial z} \end{pmatrix} \tag{3.6}$$

kde ∇ je gradient a $\mathtt{u} = (u_x, u_y, u_z)^T$ je rýchlosť. Zjednodušene povedané, jedná sa o popis samoadvekcie, teda schopnosti tekutiny, prenášať v toku svoju rýchlosť (prenášanie seba sama). Efektivitu metódy advekcie je do istej miery možné posúdiť *rádom presnosti*. Ten kvantifikuje to ako daná numerická aproximácia diferenciálnej rovnice konverguje k presnému výsledku. Tento vzťah môžme vyjadriť rovnicou:

$$E(h) = Ch^n \tag{3.7}$$

a teda hovoríme, že numerické riešenie danej diferenciálnej rovnice je *v n-tom ráde presnosti* pokiaľ chyba E je úmerná veľkosti kroku h^n. Simulátory založené na predchádzajúcom výskume Josa Stama [18] využívajú pre advekciu tzv. *semi-Lagrangeovskú* metódu. Výhodou tejto metódy je najmä to, že je nepodmienečne stabilná a jej použitím nemôže dojsť k tomu,

že pri väčších časových krokoch dojde k prenosu rýchlostí mimo simulačnú doménu, čo vedie k tomu, že simulácia zlyhá. Jej nevýhodou je to, že produkuje značné numerické odchýlky a simulovaný jav stráca detail. Jej rád presnosti je 1.

Jednou z možných alternatív je advekcia metódou *Mac Cormack*. Táto metóda má vyšší rád presnosti - 2. Jadro Mac Cormackovej advekcie pozostáva z dvoch krokov výpočtu semi-Lagrangeovou metódou a následným výhodnotením výpočetnej chyby. Definovaná je nasledovne:

$$\hat{\phi}^{n+1} = A(\phi^n) \tag{3.8}$$

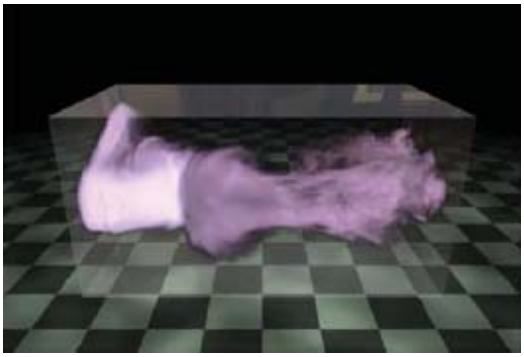
$$\hat{\phi}^{n} = A^R(\phi^{n+1}) \tag{3.9}$$

$$\phi^{n+1} = \hat{\phi}^{n+1} + \frac{1}{2}\left(\phi^n - \hat{\phi}^{n+1}\right) \tag{3.10}$$

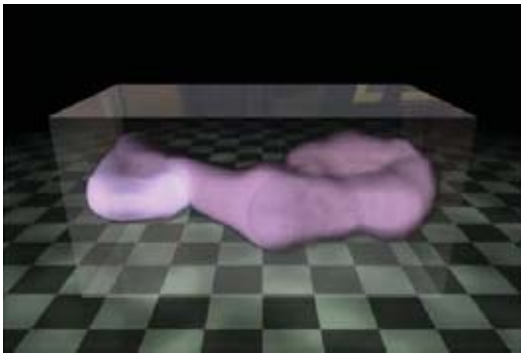
kde:

- ϕ^n je fyzikálna veličina, ktorá je advekovaná
- $\hat{\phi}^{n+1}$ a $\hat{\phi}^n$ sú dočasné veličiny
- ϕ^{n+1} je výsledná advekovaná veličina

Pozn. A^R značí reverznú advekciu. Na obrázku 3.1 je demonštrácia efektivity MacCormackovej metódy advekcie, ktorá v tomto prípade na menšej mriežke simulačnej domény podáva lepšie vizuálne výsledky.



(a) Mac Cormack, simulačná doména o rozmeroch $128 \times 64 \times 64$



(b) semi-Lagrange, simulačná doména o rozmeroch $256 \times 128 \times 128$

Obrázok 3.1: Porovnanie metód advekcie. Prevzaté z [7]

3.1.2 Tlak

Pohybujúce sa molekuly tekutiny majú tendenciu tlačiť na okolité molekuly. Pri pôsobení sily F v tekutine sa táto sila nepropaguje okamžite v celom jej objeme, ale namiesto toho molekuly blízko miesta pôsobenia F tlačia na molekuly vzdialenejšie, čím, vzniká v tekutine tlak. Na základe stredoškolských znalostí fyziky vieme, že platí:

$$P = \frac{F}{S} \tag{3.11}$$

$$\tag{3.12}$$

a teda tlak je sila pôsobiaca na plochu. V Navier-Stokesových rovniciach je tlak popísaný vzťahom

$$-\frac{1}{\rho}\nabla = -\frac{1}{\rho}\left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}\right)^T \tag{3.13}$$

kde p je tlak.

3.1.3 Viskozita (difúzia)

Z bežného života vieme, že nie každá tekutina má rovnakú hustotu. Príkladom môže byť porovnanie toku vody a sirupu. Sirup "tečie" pomalšie, a teda má väčší odpor voči toku. Hovoríme, že hustejšia tekutina má väčšiu viskozitu a odpor, ktorý vzniká pri toku spôsobuje rozptyl hybnosti. V rovnici 3.3 sa jedná o vzťah

$$\nu\nabla^2\mathbf{u} = -\begin{pmatrix} \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + u_z\frac{\partial^2 u_x}{\partial z^2} \\ \frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} + u_z\frac{\partial^2 u_y}{\partial z^2} \\ \frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2} + u_z\frac{\partial^2 u_z}{\partial z^2} \end{pmatrix} \tag{3.14}$$

3.1.4 Externé sily

Tekutina môže byť ovplyvnená externými silami. Jedná sa o sily ktoré nevznikajú pohybom tekutiny ale ovplyvňujú ju na základe externých podmienok. Pôsobenie externých síl vyjadruje vzťah

$$\frac{1}{\rho}\mathbf{f} = \frac{1}{\rho}\left(f_\mathtt{x}, f_\mathtt{y}, f_\mathtt{z}\right)^T \tag{3.15}$$

kde $\mathtt{f} = \left(f_x, f_y, f_z\right)^T$ je súčet všetkých externých síl pôsobiacich na kvapalinu a ρ je jej hustota. V simulácii tekutín rozlišujeme dva typy externých síl `f`:

- **lokálne** pôsobia iba v určitom podregióne v tekutine, napríklad sila, ktorá spôsobuje fúkanie vzduchu
- **telesové/globálne** pôsobia globálne v celej tekutine, napríklad gravitácia

3.1.5 Nestlačiteľnosť

Pre zachovanie konštantného objemu tekutiny je potrebné aby bol čistý tok v simulačnej jednotke rovný 0, čo odpovedá skutočnosti, že množstvo tekutiny vstupujúcej a vystupujúcej do simulačnej jednotky je rovnaké. Preto je potrebné stanoviť podmienku nestlačiteľnosti, ktorá je definovaná uvedenou rovnicou 3.2, pričom platí

$$\nabla\cdot\mathbf{u} = 0 \iff \left(\frac{\partial u_x}{\partial x}, \frac{\partial u_y}{\partial y}, \frac{\partial u_z}{\partial z}\right) = 0 \tag{3.16}$$

3.2 Externé sily pri simulácii dymu a ohňa

V sekcii 3.1.4 je uvedený všeobecný matematický aparát pre popis externých síl pri simulácii tekutín, ktorý je potrebné pri simulácii efektov dymu a ohňa rozšíriť.

3.2.1 Vírivosť

Simuláciou tekutiny na relatívne malej mriežke, ktoré sa v real-time aplikáciách v počítačovej grafike zvyknú využívať chýbajú detaily pohybu tekutiny, keďže sú tlmené odchýlkami použitých numerických metód. Preto sa v simulácii dymu a ohňa zvykne aplikovať externá sila vírivosti (*vorticity confinement*)[20], ktorá pridáva stratené detaily. Výpočet vírivosti popisuje rovnica

$$\omega = \nabla \times \mathbf{u} \tag{3.17}$$

Každý vektor vírivosti predstavuje lokálnú točivosť v toku tekutiny, pričom smer vektoru určuje os otáčania a magnitúda určuje silu vírenia. Následne sa vektory vírivosti použijú pre výpočet pozičných vektorov

$$\mathtt{N} = \frac{\eta}{|\eta|}, \eta = \nabla|\omega| \tag{3.18}$$

Pozičné vektory vírivosti sú vektory ktoré smerujú z oblastí nízkej vírivosti do oblastí vysokej vírivosti. Nakoniec je potrebné zaviesť spočítané hodnoty späť do vektorového poľa rýchlosti, a teda pridať vplyv vírivostnej sily:

$$\mathbf{f}_{conf} = \epsilon h(\mathtt{N} \times \omega) \tag{3.19}$$

kde

- $\epsilon > 0$ je parameter, ktorý určuje silu vírivosti
- h je priestorová diskretizácia

Diskretizáciou, použitím 1. rádu rozdielu stredov na usporiadanej *collocated* mriežke, získame z rovníc vzťahy

$$\omega_{i,j,k} = \frac{1}{2h}\begin{pmatrix} u^z_{i,j+1,k} - u^z_{i,j-1,k} - u^y_{i,j,k+1} - u^y_{i,j,k-1} \\ u^x_{i,j,k+1} - u^x_{i,j,k-1} - u^z_{i+1,j,k} - u^z_{i-1,j,k} \\ u^y_{i+1,j,k} - u^y_{i-1,j,k} - u^x_{i,j+1,k} - u^x_{i,j-1,k} \end{pmatrix} \tag{3.20}$$

a

$$\eta_{i,j,k} = \frac{1}{2h}\begin{pmatrix} |\omega_{i+1,j,k}| - |\omega_{i-1,j,k}| \\ |\omega_{i,j+1,k}| - |\omega_{i,j-1,k}| \\ |\omega_{i,j,k+1}| - |\omega_{i,j,k-1}| \end{pmatrix} \tag{3.21}$$

3.2.2 Vztlaková sila

Ďalším príkladom externej sily pôsobiacej v tekutine, ktorá sa často využíva v simuláciách plynov, je vztlak alebo tepelná vztlaková sila. Táto sila spôsobuje to, že horúci plyn stúpa a chladný klesá. V texte [8] popisujú vztlakovú silu nasledovne:

$$\mathbf{f}T = \beta\mathbf{y}(T - T_0) \tag{3.22}$$

kde:

- β je kladný koefficient, ktorý kontroluje veľkosť vztlakovej sily, odvodený od koeficientu teplotnej expanzie

- $\mathbf{y}$ je vektor určujúci vertikálny smer
- T_0 je priemerná teplota v simulačnej doméne
- T je aktuálne advekovaná teplota v toku tekutiny

Pre pridanie vztlakovej sily je nutná skalárna reprezentácia teploty T, ktorá musí byť v čase aktualizovaná aby jej vývoj prebiehal korektne. V [9] popisujú vývoj teploty rovnicou

$$\frac{\partial T}{\partial t} = \lambda \nabla^2 T - (\mathbf{u} \cdot \nabla) T \tag{3.23}$$

kde:

- λ je koeficient tepelnej distribúcie

Nakoľko je táto rovnica podobná difúznym a advektívnym členom Navier-Stokesových rovníc môžeme k jej riešeniu využiť rovnaké metódy advekcie.

3.3 Riešenie Navier-Stokesových rovníc

Rovnice môžeme riešit analyticky len pre niekoľko fyzických konfigurácií. Okrem toho ich však vieme riešiť inkrementálne numerickými metódami pre riešenie sústavy rovníc [4]. Pri simulovaní toku tekutín nás zaujíma práve vývoj toku veličín prenášaných tekutinou v čase, čo znamená jednoduché použitie týchto metód.

Pre riešenie je potrebné rozložiť rovnicu na sekvenciu jednoduchých krokov, ktoré budeme riešiť postupne. Rovnice je potrebné rozložiť na formu vhodnú pre numerické riešenie. To znamená vhodne diskretizovať doménu simulácie, teda priestor v ktorom simulácia toku prebieha.

3.3.1 Lagrangeova metóda

Flexibilnú metódu využívajúcu časticové systémy bez použitia mriežky predstavuje technika SPH (*Smoothed particle hydrodynamics)*[21], ktorá sa spočiatku využívala pre simuláciu astrofyzikálnych problémov ako sú napríklad kolízie v galaxiách a dynamika formovania hviezd. V SPH je tekutina modelovaná ako kolekcia častíc, ktoré sa pohybujú pod vplyvom hydrodynamickej a gravitačnej sily. Aj keď je SPH dostatočne flexibilná, je použiteľná iba pre riešenie stlačiteľných tokov. Táto metóda však nie je efektívna pre modelovanie vírivých tokov a často sa preto využíva pre simulácie kvapalín a pre simulácie tekutín sa zvyknú využívať Eulerovské metódy.

3.3.2 Eulerova metóda

V prípade Eulerovskej dikretizácie tento priestor rozdelíme do pravidelných výpočetných elementov, ktoré sú v čase nemenné, menia sa len hodnoty veličín, ktoré sú v nich uchované. Prakticky sa to znamená, že sa simulačná doména rozdelí do pravidelných kubických buniek, kde každá z týchto buniek môže uchovávať nielen skalárne ale vektorové hodnoty.

3.3.3 Hybridné metódy

Pre rozsiahle scény, ktoré obsahujú regióny, kde sa vyžadujú vzniky vírivých tokov je vhodné využiť hybridný prístup Lagrangeovských a Eulerovských metód tak, že sa využije Lagrange v globálnom merítku a v oblastiach záujmu sa použije Euler a častice, ktoré vstúpia do daných regiónov budu ovplyvnené vektormi rýchlostí z mriežky. Zaujímavé hybridné metódy poskytujú napríklad [6] a [15].

Kapitola 4

Diskretizácia simulačnej domény

Pre riešenie Navier-Stokesových rovníc je veľmi výhodné použiť numerické metódy. Pre ich vhodnú a efektívnu aplikáciu je výhodné ak simulované veličiny (tlak, teplota, čas, rýchlosť a iné) budú ľahko dostupné pre body, ktoré sú riešené numerickými metódami. Táto skutočnosť poukazuje na to, že diskretizácia rovníc a diskretizácia vektorových polí sú značne prepojené a bolo by vhodné uvažovať o ich spoločnom použití. V tejto kapitole si popíšeme niektoré z metód diskretizácie, ktoré vychádzajú z literatúry o dynamike tekutín. Existuje množstvo metód no iba niektoré z nich sú vhodné pre použitie v rámci tejto práce.

4.1 Rovnomerné kartézske mriežky

Simulátor tekutín je možné postaviť na dvoch základných modeloch reprezentácie domény:

- usporiadané kartézke mriežky *(angl. collocated grids)*
- nestále kartézke mriežky *(angl. staggered grids)*

Oba tieto typy reprezentujú rovnomernú kartézsku mriežku (mriežka, ktorej bunky sú osovo zarovnané), ktorej ľavý spodný roh je umiestnený v počiatku koordinačného systému. Bunky rovnomernej mriežky spĺňajú tú vlastnosť, že ich objem je rovnaký naprieč celým telesom:

$$\Delta x_{b_1} = \Delta y_{b_1} = \Delta z_{b_1} = x, \forall x \in G \tag{4.1}$$

kde x označuje priestorovo diskretizačný faktor. Pri simulácii kvapalín je však výhodné zvoliť práve tento prístup, teda bunky reprezentujúce kockami, nakoľko nepravidelné bunky môžu viesť k neželaním artefaktom, ktoré budú adresované v nasledujúcich kapitolách.
Keďže sa jedná o rovnomerné mriežky, spočítať koordináty prostrednej bunky `C` môžeme na základe vzťahu

$$\mathtt{C}(\mathtt{b}_{\mathtt{x,y,z}}) = \begin{pmatrix} s/2 \\ s/2 \\ s/2 \end{pmatrix} + \begin{pmatrix} x \cdot s \\ y \cdot s \\ z \cdot s \end{pmatrix}, \forall b_{x,y,z} \in G, \tag{4.2}$$

kde
$\mathtt{b} = b_{x,y,z},\ x, y, z \in \mathbb{Z}$

$$0 \leq x < w, \tag{4.3}$$
$$0 \leq y < h, \tag{4.4}$$
$$0 \leq z < d, \tag{4.5}$$

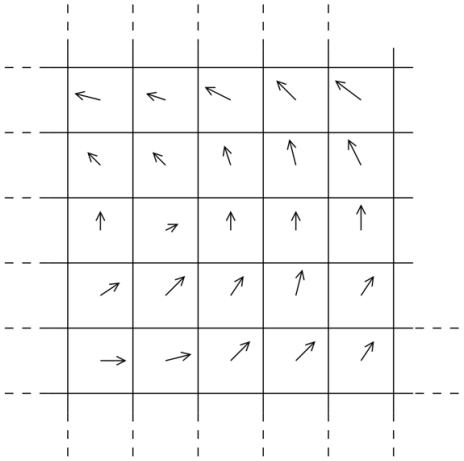
kde w, h, d sú šírka, výška a hĺbka mriežky v tomto poradí.

4.2 Usporiadané mriežky

Stam vo svojich prácach [19][18] využíva špeciálny typ kartézskych mriežok, ktoré reprezentujú veličiny v stredoch buniek simulačnej domény, viď obrázok 4.1. Hodnoty, ktoré sa spočítajú mimo stredov buniek musia byť interpolované z okolitých hodnôt. Táto reprezentácia sa v simuláciách tekutín využíva často najmä vďaka svojej jednoduchosti, no prináša aj nevýhody, ktorými sú numerické odchýlky. Napríklad pri výpočte divergencie vektorového poľa rýchlostí v tomto type mriežky, danej vzorcom $\nabla \cdot \mathbf{u}$ dostávame vzťah:

$$\nabla \cdot \mathbf{u}_{i,j,k} = \frac{\partial u^x_{i,j,k}}{\partial x} + \frac{\partial u^y_{i,j,k}}{\partial y} + \frac{\partial u^z_{i,j,k}}{\partial z} \tag{4.6}$$
$$= \frac{u^x_{i+1,j,k} - u^x_{i-1,j,k} + u^y_{i,j+1,k} - u^y_{i,j-1,k} + u^z_{i,j,k+1} - u^z_{i,j,k-1}}{2h} \tag{4.7}$$

čo znamená, že bod divergencie $\mathbf{p} = \mathcal{C}(c_{i,j,k})$ sa spočíta bez ohľadu na hodnotu rýchlosti $\mathbf{u(p)}$ v danej bunke. V podstate sa jedná o výpočet priemernej odchýlky v stredoch okolitých buniek, čo môže spôsobovať to, že sa tekutina javí hustejšia ako by mala byť.



Obrázok 4.1: Reprezentácia dát v usporiadanej mriežke.

4.3 Nestále mriežky

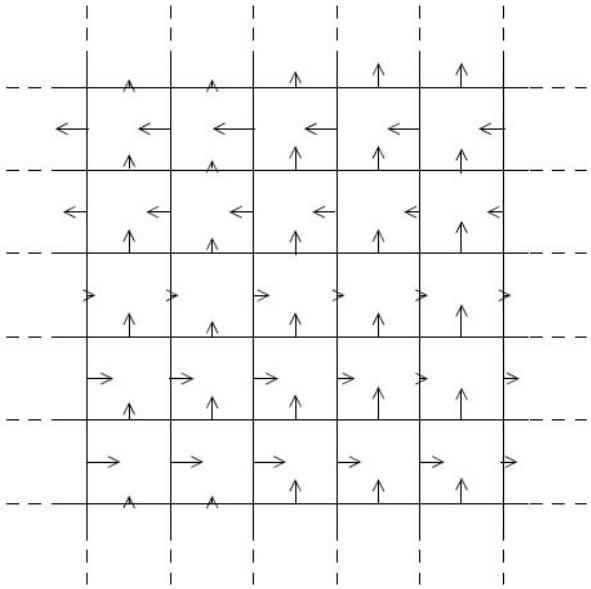
Odlišným prístupom sú tzv. nestále mriežky, uvedené v [9][8][10] ktoré modifikujú usporiadané mriežky tak, že sa rýchlosti reprezentujú na stenách/hranách buniek, čo indikuje tok

medzi nimi. Rýchlosť je jediná veličina, ktorá je v tomto type mriežky reprezentovaná na hranách, pričom tlak, hustota či teplota sú reprezentované podobne ako v predošlom prípade - v stredoch buniek, obrázok 4.2. Reprezentáciou rýchlostí týmto spôsobom minimalizuje numerické odchýlky, nakoľko pri rozdieloch v stredoch buniek platí:

$$\nabla \cdot \mathbf{u}_{i,j,k} = \frac{\partial u^x_{i,j,k}}{\partial x} + \frac{\partial u^y_{i,j,k}}{\partial y} + \frac{\partial u^z_{i,j,k}}{\partial z} \tag{4.8}$$

$$= \frac{u^x_{i+\frac{1}{2},j,k} - u^x_{i-\frac{1}{2},j,k} + u^y_{i,j+\frac{1}{2},k} - u^y_{i,j-\frac{1}{2},k} + u^z_{i,j,k+\frac{1}{2}} - u^z_{i,j,k-\frac{1}{2}}}{h} \tag{4.9}$$

Keďže zložky vektorov rýchlostí nie sú reprezentované na rovnakých pozíciách, je nutná interpolácia, čo vedie k drobnému rozptylu. Taktiež prepočet vektoru rýchlosti môže vyžadovať 3 oddelené výpočty, nakoľko sú vektory rýchlostí rozdielne na pozíciách jednotlivých komponent. Keďže sú ostatné veličiny v tomto type mriežky takisto reprezentované v stredoch buniek, je pri ich aplikácii s vektorovým poľom rýchlostí opäť nutná interpolácia.



Obrázok 4.2: Reprezentácia dát v nestálej mriežke.

Kapitola 5

Vizualizačné techniky

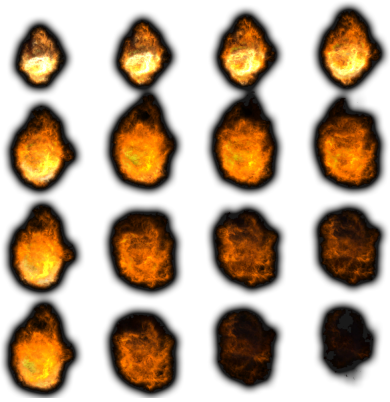
Samotná simulácia tekutín je jadrom pre túto prácu. Jej cieľom však nie je len tekutiny simulovať, ale aj ich vierohodne vykreslovať s cieľom dosiahnutia čo najrealistickejších výsledkov pri vykresľovaní v reálnom čase. V tejto kapitole budú predstavené vhodné vykresľovacie techniky pre vizualizáciu volumetrických efektov. Niektoré z uvedených metód sú pomerne komplexné a vedú k výborným vizuálnym výsledkom, za cenu toho, že vykresľovanie simulácie nemusí byť v reálnom čase pri rozumnom počte vykreslených snímkov za sekundu. Tieto techniky však predstavujú budúcnosť zobrazovania nielen volumetrických efektov, nakoľko výkonnosť grafického hardvéru sa zdokonaľuje každým rokom a otvára tak nové možnosti.

5.1 Časticové systémy

Časticovými systémami počítačovej grafike označujeme techniku zobrazenia veľkého množstva malých objektov (bod, úsečka, obdĺžnik či iné jednoduché 2D/3D geometrické primitíva) s cieľom simulácie určitých typov "fuzzy"fenoménov, ktoré sú typicky veľmi ťažko reprodukovateľné štandardnými renderovacími technikami. Typicky sa jedná o chemické reakcie, prírodné úkazy či iné náhodné systémy. mnohých prípadoch využívajú práve Billboarding. Vďaka svojej jednoduchosti a škálovateľnosti sú časticové systémy aktuálne využívané vo veľkej miere najmä v počítačových hrách pre simuláciu plynov. Populárne herné enginy [1] obsahujú optimalizované moduly pre prácu s časticovými systémami. Častica sa po vytvorení pohybuje na základe svojho počiatočného stavu, ktorý typicky zahrňuje veľkosť, rýchlosť, rotáciu a polohu v priestore. Rýchlosť častice môže byť počas jej existencie ovplyvnená definovanými silami (vietor, odpor vzduchu atď) a jej poloha sa môže meniť napríklad na základe procedurálne vygenerovanej dráhe pohybu. Okrem toho samozrejme častica môže v čase meniť aj svoju veľkosť a rotáciu s cieľom dosiahnuť prirodzený výsledný dojem. Typickou ukážkou sú napríklad častice dymu.

Ďalšou dôležitou vlastnosťou častice je taktiež jej životnosť, nakoľko v časticových systémoch takmer vždy očakávame, že častica sa po určitom čase vymaže, keď splní svoj účel. Pre dosiahnutie realistických výsledkov sa preto často pred samotným odmazaním častice znižuje jej viditeľnosť.

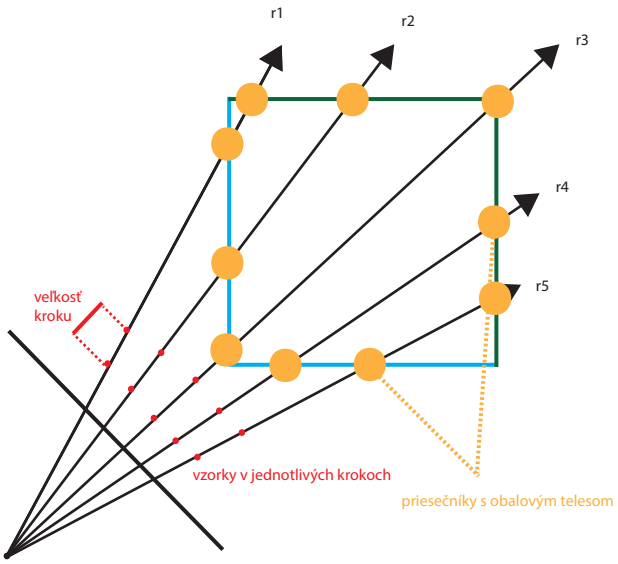
[1] https://unity3d.com/ https://www.unrealengine.com/ https://www.cryengine.com/



Obrázok 5.1: Ukážka atlasu textúr častice pre animáciu ohňa.

5.2 Ray-marching

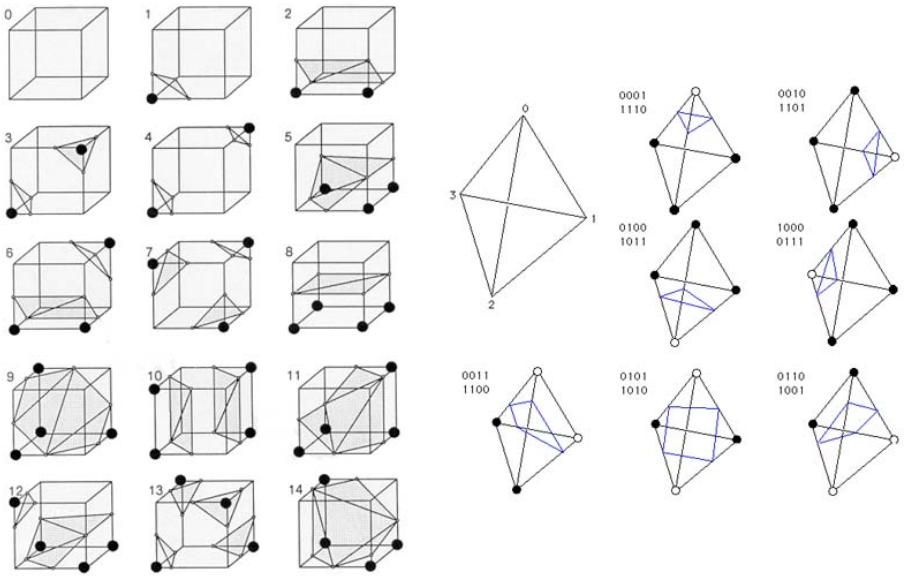
Ray-casting[13] alebo vrhanie lúčov, je metóda vykresľovania pre priame zobrazenie 3D skalárneho poľa bez použitia pomocnej reprezentácie dát ako sú napríklad trojuholníky. Výhodou raycastingu je to, že umožňuje vynechávať prázdne miesta v 3D priestore, je nezávislé na type projekcie, je jednopriechodové a je pomerne jednoduché na implementáciu. Pri volumetrickom ray-castingu[12] sa lúče vrhajú cez objemovú štruktúru a pri jej priechode sa zbierajú vzorky, ktoré nás zaujímajú. Špeciálnou metódou ray-castingu je tzv. *ray-marching*, ktorý funguje takmer rovnako, až na to, že pri prieniku cez objemovú štruktúru sa dáta vzorkujú po predom zvolenom kroku. Ako lúč postupuje skalárne hodnoty sa mapujú na optické vlastnosti použitím prevodových funkcií, ktorých výsledkom je typicky farba v RGBA farebnom modele, ktorá zahŕňa odpovedajúce vyžarovacie a absorbčné koeficienty pre danú vzorku. Následne sa daná farba vyskladá použitím alpha blendingu (spredu dozadu alebo zozadu dopredu).



Obrázok 5.2: Ray-marching.

5.3 Extrakcia iso povrchov

Ďalším zo zaužívaných algoritmov pre vykresľovanie iso povrchov vo volumetrických dátach je tzv. *marching cubes*[14]. Podstatou tejto techniky je to, že môžeme každý voxel definovať hodnotami texelov na jeho 8 rohoch. V prípade, že jeden alebo viaceré z týchto hodnôt sú menšie ako definovaná iso hodnota, a jedna alebo viac hodnôt je väčších, daný voxel tvorí iso povrch. Zistením, ktoré hrany voxelu pretínajú iso povrch sa vytvoria trojuholníkové plochy, ktoré delia voxelovú kocku na segmenty obsiahnuté v a mimo iso povrchu. Spojením všetkých týchto plôch získame reprezentáciu povrchu. Pri triedení rohov voxelu na základe toho, či je hodnota ktorú uchováva menšia alebo väčšia ako zvolená iso hodnota, existuje 256 (2^8) možností ako daný roh klasifikovať. Týchto 256 možností je však možné na základe určitých symetrií delenia kocky zredukovať na 14, viď obrázok 5.3a.



(a) marching cubes (b) marching tetrahedron

Obrázok 5.3: Ukážky variácií prienikov pri algoritmoch marching cubes a marching tetrahedron

Za zmienku stojí aj algoritmus, ktorý vychádza z marching cubes - *marching tetrahedron* (tetrahedron - štvorsten)[2]. Rozdiel spočíva v tom, že sa voxel rozdelí na 6 nepravidelných štvorstenov. Vznikne tak väčší počet hrán (19) voči marching cubes, kde vznikne rozdelením iba 12 hrán. Každý štvorsten má 8 možných konfigurácií (7 z nich je na obrázku 5.3b, ktoré spadajú do troch tried: bez prieniku, prienik iba s jedným trojuholníkom, prienik s dvomi susednými trojuholníkmi. Výhodou[5] tohto algoritmu voči algoritmu marching cubes je to, že nie je tak výrazne ovplyvnený vznikajúcimi nejednozačnými plochami.

[2] http://paulbourke.net/geometry/polygonise/

Kapitola 6

Návrh aplikácie

Návrh softwarovej časti tejto práce je rozdelený do troch celkov. Jadrom práce je knižnica, ktorá slúži na simuláciu a vykreslenie volumetrických efektov ohňa a dymu. Knižnica sa opiera o jednoduchý engine vytvorený takisto v rámci tejto práce, pričom ten zabezpečuje nízkoúrovňové operácie ako manažment aplikačného okna, vstupov z myši a klávesnice, zavádza jednoduchý udalostný systém a zabezpečuje základné grafické nástroje ako načítavanie a správu shaderov, či rendering jednoduchých geometrických primitív. Poslednou súčasťou tejto práce je samotná demonštračná aplikácia, ktorá prepája uvedený engine, knižnicu volumetrických efektov a jednoduché grafické užívateľské rozhranie pre parametrizáciu simulovaného efektu. Táto kapitola ako aj kapitola implementácie sa zameriava na simulačnú a vykresľovaciu knižnicu.

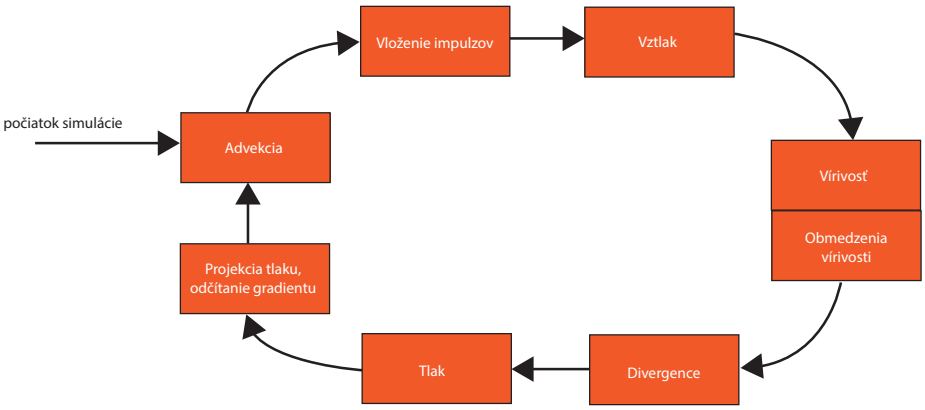
6.1 Využitie GPU

Cieľom pri návrhu implementačnej časti práce bolo využiť akceleráciu grafických kariet, ktorú umožňuje prenos sekvenčných riešení algoritmov na paralelné, ktoré sú riešené SIMD jednotkami. Proces simulácie, ktorý vo svojej podstate prakticky nevyžaduje využitie grafického pipelinu som sa preto rozhodol riešiť s využitím GPGPU výpočtov, realizovaných compute shadermi, pričom tie boli v neskoršej fáze návrhu z časti zavedené aj do fázy vizuálizácie. Analogicky k CPU riešeniu pre reprezentáciu dát poliami, či podobnými statickými štruktúrami, je možné využiť textúry, ktorých implementácia na GPU implicitne umožňujú efektívne vyčítanie z pamäti, či interpoláciu hodnôt. Okrem toho textúry implicitne umožňujú uchovávať nielen skalárne, ale aj vektorové hodnoty, čo je práve pri implementácii simulácie tekutín veľmi výhodné. Akcelerácia prostredníctvom GPU má však aj svoje nevýhody, kde tou najväčšou je nemožnosť čítať a zapisovať do tej istej textúry, ktorá je použitá pre čítanie dát. Preto je nutné zapisovať dáta priamo do framebufferu. Riešením tohto problému je jednak kopírovanie dát z framebufferu spätne do textúry, prípadne použitie ďalšej textúry ako framebufferu, viď *pingponging* v sekcii 6.3.

6.2 Simulačný cyklus

Na základe informácií uvedených v kapitole 3, bolo potrebné rozhodnúť, ktoré členy Navier-Stokesových rovníc sú relevantné pre simulované efekty. Nakoľko cieľom bolo vytvoriť simulátor efektov dymu a ohňa, bolo do simulačného cyklu bezpodmienečne nutné zaradiť operáciu advekcie a riešenia tlaku, pričom bolo možné vypustiť výpočet viskozity. Ďalej

bolo potrebné rozhodnúť akým spôsobom budú veličiny vkladané do simulačnej domény. Rozhodol som sa tento problém riešiť tak, že sa po advekcii vygeneruje silový impulz, ktorý bude simulovaný gausiánom. Ten je potrebné generovať pre každú z vkladaných veličín, nakoľko každá z nich musí byť reprezentovaná samostatným volumetrickým úložiskom. Pre menované efekty bolo tak isto nutné zohľadniť špecifické externé sily, ktoré v týchto fyzikálnych javoch pôsobia, konkrétne vztlakovú silu a vírivosť. Zvážením týchto skutočností bol navrhnutý simulačný cyklus (obrázok 6.1).



Obrázok 6.1: Vizualizácia simulačného cyklu

Pre simuláciu uvedených efektov je nutné koncept rozšíriť o možnosť advekcie ďalších nezávislých veličín. Na základe získaných znalostí o advekčných metódach a rozporuplných názorov na ich efektivitu, je zavedená podpora viacerých z nich, pričom užívateľovi pripadá možnosť voľby. Jednotlivé fázy boli navrhnuté tak, aby boli prístupné prostredníctvom rozhrania triedy simulátoru a bolo možné ich prípadne vypustiť v prípade rozšírenia knižnice o efekt, ktorý ich nevyžaduje.

Veľmi dôležitou voľbou je aj spôsob riešenia rovnice prenosu tlaku. Jedná sa o vhodného kandidáta pre riešenie numerickými metódami. Efektívnou metódou pre riešenie je napríklad metóda Runge-Kutta (RK4), ktorá je však pomerne komplikovaná pre paralelné použitie. Preto som sa rozhodol siahnuť po metóde vhodnejšej pre paralelizáciu - Jacobiho metódu.

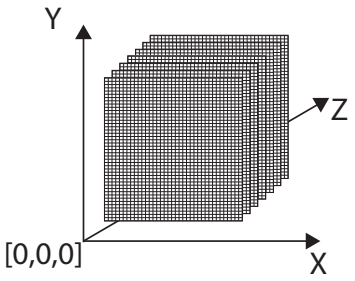
6.3 Reprezentácia simulačnej domény

Pre riešenie Navier-Stokesových rovníc bola zvolená Eulerovská metóda. Simulačná doména preto môže byť reprezentovaná niekoľkými možnosťami:

- 3D textúrami
- riedkymi 3D textúrami
- riedky voxelový strom (*sparse voxel octree*
- poľom 2D textúr

Nakoľko je simulačná doména osovo zarovnaná a pravidelná, rozhodol som sa využiť 3D textúry, obrázok 6.2, ktoré sú na využitie v tejto problematike praktické a práca s nimi je

jednoduchá. 3D textúry taktiež umožňujú jednoduché výpočty nad simulačnou doménou reprezentované usporiadanou kartézskou mriežkou (s hodnotami v strede buniek).



Obrázok 6.2: Reprezentácia mriežky 3D textúrou

Simulácia efektov dymu a ohňa môže advekovať v simulačnej doméne niekoľko typov veličín. Preto bolo nutné pridať podporu pre rôzne komplexné textúry, na základe počtu skalárnych, prípadne vektorových informácií ktoré veličina reprezentuje. Keďže viaceré z výpočetných fáz simulácie pracujú s rozdielnymi typmi dát, je potrebné zohľadniť návrh shaderov tak, aby umožňovali výpočet nad vstupnými dátami nezávisle na počte ich komponent. V prvej iterácii návrhu sa do knižnice zaviedlo niekoľko compute shaderov v závislosti na vstupných dátach. Tento koncept bol neskôr upravený, aby bol generickejší a viedol k tomu, že prakticky je v rámci knižnice zavedený jediný compute shadery pre každú z výpočetných fáz.

Keďže výpočty sú akcelerované na GPU a prebiehajú vysoko paralelne, je nutné zaviesť mechanizmus prevencie prepisovania dát. Problém by bolo možné riešiť napríklad komplikovanou synchronizáciou kernelov, no ako jednoduchšie riešenie bolo zvolené prepínanie textúr. Za týmto účelom je v aplikácii každá z veličín reprezentovaná dvojicou textúr, ktoré sa vždy po zápise vymenia. Tento mechanizmus sa zvykne označovať ako *ping-ponging*.

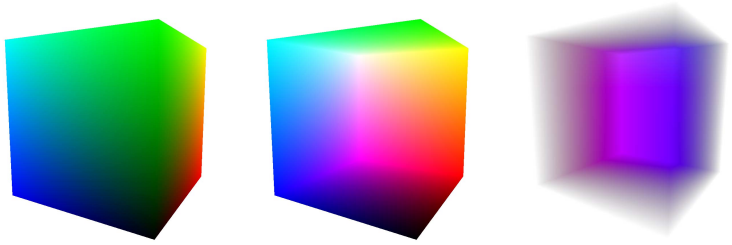
6.4 Post-processing simulačných dát

Pre dosiahnutie výsledkov v reálnom čase ako aj z hľadiska pamäťovej náročnosti je nutné uvažovať nad rozumnou veľkosťou simulačnej mriežky. Keďže toto je závislé na type grafickej karty, je potrebné užívateľovi umožniť zvoliť veľkosť mriežky na základe jeho možností. V súčasnosti však aj výkonné grafické karty nie sú schopné realtime simulácie na mriežke väčšej ako $512 \times 512 \times 512$. Preto je nutné zvoliť približne 2-4$\times$ menšiu mriežku v každej dimenzii, čo reálne predstavuje veľmi malé množstvo informácie. Preto je súčasťou návrhu aplikácie post-processing simulovaných dát ktorého cieľom je vyhľadiť vykresľované dáta pri vykresľovaní vo väčšom rozlíšení ako je simulačná doména. Každú z reprezentovaných veličín preto aplikácia umožňuje filtrovať, pričom je nutné aby bola filtrácia nezávislá na type vstupných dát.

6.5 Zobrazenie volumetrických dát

Pre zobrazovanie volumetrických dát som zvolil techniku ray-marching, nakoľko táto metóda podáva veľmi dobré vizuálne výsledky za cenu vyššej výpočetnej náročnosti. Použiteľnou by boli aj odnože marching cubes metódy, ale tie sú výhodnejšie pre zobrazovanie pevných povrchov, čo dym a oheň nie sú. Štandardným spôsobom návrhu ray-marchingu

nad volumetrickými dátami je dvojpriechodový prístup, kedy sa simulačná doména reprezentuje geometrickým objektom (typicky kváder), ktorý sa vykreslí pomocou rasterizácie, pričom sa v prvom kroku sa vykreslia predné (obrázok 6.3a) a zadné (obrázok 6.3b) steny obalového telesa (`axis aligned bounding box`), ktoré reprezentuje simulačnú doménu.



(a) predné steny simulačnej domény (b) zadné steny simulačnej domény (c) textúrovacie koordináty

Obrázok 6.3: Obaľové teleso reprezentujúce simulačnú doménu

Tie sa uložia do osobitných 2D textúr a následne sa v druhom kroku na základe nich spočíta smerový vektor a vykoná sa samotný ray-marching. Tento prístup som sa rozhodol v implementácii optimalizovať, nakoľko nie je potrebné, aby bolo vykresľovanie dvojpriechodové. Optimalizácia spočíva v tom, že sa oba z uvedených krokov spoja do jedného priechodu tak, že sa obalové teleso reprezentuje parametricky a vykreslí sa až vo fáze ray-castingu nad obdĺžnikom , ktorý typicky pokrýva celú obrazovku (*full screen quad*). Pred samotným ray-marchingom sa spočíta prienik lúča s obalovým telesom priamo v shaderi realizujúcom ray-marching. Výhodou tohto prístupu je to, že je možné okamžite ukončiť výpočet pre lúče, ktoré by nevzorkovali dáta z volumetrických úložisk, a teda tie ktoré nemajú spoločný prienik s obalovým telesom. Rovnako sa tak vyrieši problém, ktorý prináša dvojpriechodový prístup, a síce problém orezania keď je kamera umiestnená vo vnútri simulačnej domény. Výhodou je aj to, že sa tento prístup jednoducho zakomponuje do existujúceho pipelinu, ktorý využíva rasterizáciu.

6.6 Parametrizácia

Účelom navrhnutej simulačnej a vykresľovacej knižnice je poskytnutie hrubej funkcionality, ktorá zabezpečuje výpočet matematicko/fyzikálneho aparátu predstaveného v teoretickej časti, pričom ho sprístupňuje pomocou jednoduchého rozhrania, pre externé použitie. Každá z výpočetných fáz, či už pri simulácii alebo renderovaní je navrhnutá tak, aby bola parametrizovateľná. Nespornou výhodou tohto návrhu je to, že aplikácia nie je viazaná na konkrétne vytvorený efekt, prípadne podmnožinu efektov. Užívateľ tak získava plnú kontrolu výpočtom, čo mu umožňuje vytváranie potenciálne veľkého množstva efektov. Okrem toho návrh umožňuje prostredníctvom parametrizácie rôzne kombinovať použité techniky, čo vedie nielen k jednoduchému použitiu ale aj k jednoduchej možnosti rozšírenia.

Kapitola 7

Implementácia a dosiahnuté výsledky

Aplikácia je implementovaná v jazyku C++, pričom jej jadro spočíva v GPGPU výpočtoch, realizovaných prostredníctvom OpenGL API compute shaderov, pričom pre vykresľovanie je rovnako použitá knižnica OpenGL. Pre načítavanie rozšírení sa využíva grafická knižnica `glew`, matematické operácie a typy zahŕňa knižnica `glm`, pre správu grafického contextu, okna a vstupov z klávesnice a myši je použité `glfw` a načítanie konfiguračných súborov zabezpečuje `rapidjson`. Tak isto je v časti demoštračnej aplikácie aplikovaný kód vykresľovania komplexnejšej scény, využívajúci knižnicu `assimp`. V neposlednom rade pre zobrazenie GUI, umožňujúceho parametrizáciu simulácie a vykresľovania aplikácia využíva `imgui`. Aplikácia bola vyvíjaná najmä pod OS Windows a z časti MacOS, pričom cieľová platforma je Windows. Zdrojové súbory obsahujú predpripravené konfiguračné súbory pre build systém `CMake`.

7.1 Prenos veličín v toku

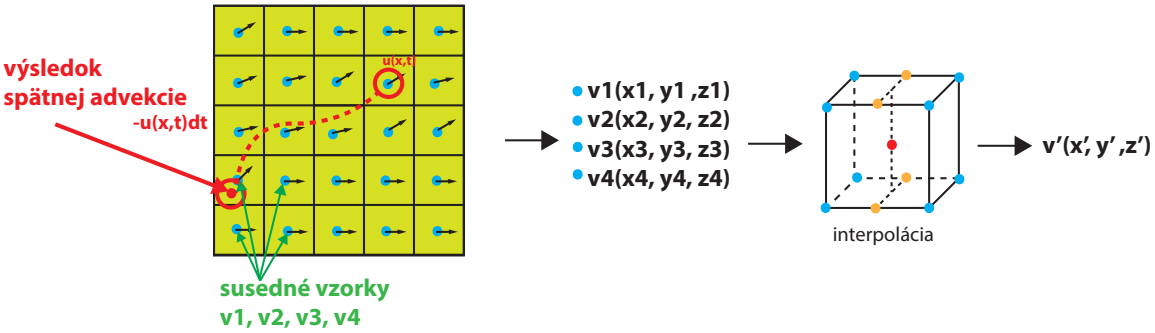
Základným stavebným kameňom pri implementácii simulátora tekutín je integrácia operácie advekcie. Tá je v aplikácii reprezentovaná rozhraním `IAdvection`, ktoré implementujú triedy `SemiLagrangian` a `MacCormack`. Simulácia je preddefinovaná tak, že sa rýchlosť advekuje s použitím semi-Lagrangeovej metódy a ostatné veličiny sú advekované metódou MacCormack, čo podla [16] vedie k najoptimálnejším výsledkom.

7.1.1 Semi-Lagrangeova metóda

Obrázok 7.1 presne popisuje algoritmus semi-Lagrangeovej metódy. Shader ktorý zabezpečuje advekciu touto metódou funguje nasledovne:

1. získa sa hodnota rýchlosti v strede bunky aktuálnej bunky
2. znormalizuje sa aktuálna pozícia na základe `id` invokácie kernelu
3. na základe rozdielu času medzi dvojicou snímkou (ďalej len `deltaTime`), sa spočíta koordinát 'častice' obsiahnutej v tejto bunke pred časom `deltaTime`
4. nakoľko nová pozícia nemusí spadnúť do stredu bunky v simulačnej doméne, je potrebné získať hodnoty okolitých vzorkov a trilineárnou interpoláciou získať výslednú hodnotu

Nevýhodou je, že metóda prináša pomerne značnú numerickú odchýlku a vedie k tomu, že oheň alebo dym sa "vyhladia". Rovnako sa pri tejto metóde môžu strácať detaily napríklad vo výrivosti tekutiny (viď obrázok 8.3 vľavo).



Obrázok 7.1: Sledovanie prenášanej veličiny spätne v čase v 2D.

Advekcia veličín touto metódou je implementovaná v compute shaderoch `advect_1d.comp` a `advect_4d.comp` pričom implementácia si zvolí správny druh shaderu na základe komplexnosti danej veličiny a je parametrizovaná hodnotou rozptylu(*dissipation*) v intervale [0,1]. Tento parameter určuje, ako výrazne dochádza v čase k rozptylu advekovanej veličiny.

7.1.2 Mac Cormackova metóda

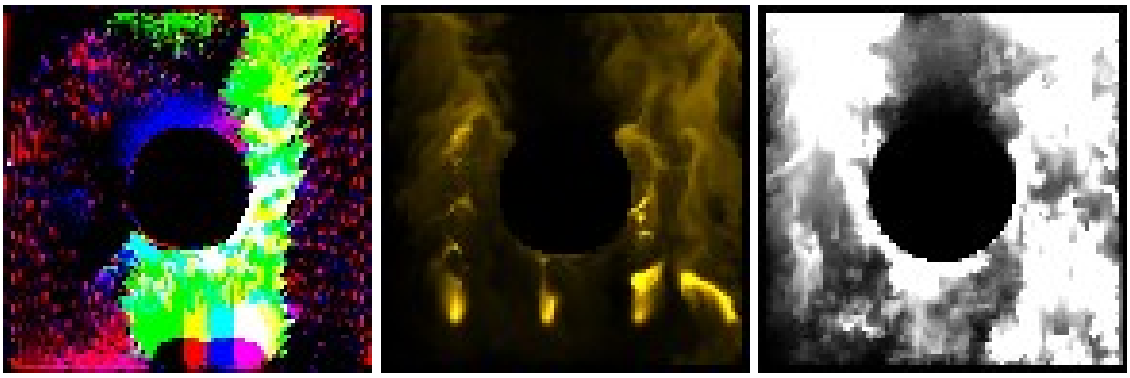
Implementácia metódy Mac Cormack v prvom kroku používa rovnaké shadery ako predošlá metóda, na základe ktorých sa spočíta dopredná a spätná advekcia. Dočasné výsledky sa uložia do objektov triedy `vfx::Image3D`, nakoľko sa nejedná o veličiny, pri ktorých by sme potrebovali vymieňať textúry. Výpočet prebieha nasledovne:

1. advekcia rýchlosti s `+deltaTime` - $\hat{\phi^{n+1}}$
2. advekcia rýchlosti s `-deltaTime` - $\hat{\phi^{n}}$
3. spočíta sa textúrovací koordinát pred časom `deltaTime` podobne ako v krokoch 1. a 2.
4. spočíta sa najbližšia vzdialenosť k hranici simulačnej domény, pokiaľ je vzdialenosť menšia ako 3, získame vzorku z textúry advekovanej veličiny na koordináte z kroku 3., v opačnom prípade spočítame minimum a maximum po zložkách z okolitých bodov vzorku na backtrackovacom koordináte, a pôvodnú hodnotu rýchlosti upravíme o spočítanú chybu z rozdielu dočasných textúr z krokov 1. a 2.
5. získanú vzorku orežeme minimom a maximom spočítaných v predošlom kroku

Pri použití tejto metódy sa v doprednej aj spätnej advekcii uvažuje s **nulovou hodnotou rozptylu**. Kroky 1. a 2. sú implementované v jednom shaderi, pričom výpočetná úloha sa dispatchne pre každý krok osobitne. Kroky 3-5 sú implementované v ďalšom compute shaderi, pričom s jeho použitím sa v jednom dispatchi spočíta odchýlka. Tento krok je nutný kvôli tomu, že táto metóda nie je bezpodmienečne stabilná. V tejto fáze je už znova aplikovaná hodnota rozptylu, no na rozdiel od semi-Lagrangea je táto metóda navyše parametrizovaná hodnotou úpadku (*decay*).

7.2 Veličiny a vkladanie impulzov

V aplikácii je 3D textúra reprezentovaná triedou, ktorá predstavuje wrapper nad GPU objektom. Trieda zaobaľuje celkovo 2 volumetrické dátové úložiská, z ktorých prvé uchováva samotné dáta, a druhé slúži ako potenciálne cieľové uložisko pri post processingu (viac informácií v kapitole 7.6.2. Ako vyplýva z návrhu, každú zo simulovaných veličín je potrebné reprezentovať 2 textúrami, s cieľom umožniť ich prepínanie. Veličiny sú preto reprezentované triedou `vfx::Quantity`, kde každá si prostredníctvom inteligentných ukazateľov uchováva referencie na dvojice volumetrických dát. Prepínanie medzi textúrami je teda implicitné na úrovni ukazateľov. Advekcia rýchlostí je však sama o sebe nedostatočná pre vytvorenie vizuálneho vnemu, nakoľko vizualizácia rýchlostí, obrázok 7.2, nepredstavuje požadovaný efekt.



Obrázok 7.2: Vizualizácia vektorového poľa rýchlosti, hustoty a teploty z jednej úrovne 3D textúry v rámci jedného dispatchu. Rozlíšenie textúr $96 \times 96 \times 96$

Simulácia je preto rozšírená o 2 ďalšie veličiny, ktoré sa šíria tekutinou spolu s rýchlosťou: hustotu a teplotu. Každá z týchto veličín je reprezentovaná triedou `vfx::Quantity`, čím je zaručená možnosť ich advekcie rovnakým compute shadermi ako v prípade rýchlosti. Obe z týchto veličín je potrebné dostať do simulačnej domény. V tomto prípade sa veličiny líšia svojimi vlastnosťami a ich potrebnou reprezentáciou.

7.2.1 Hustota

Hustota je reprezentovaná 4 zložkovým vektorom, ktorá však nie je z fyzikálneho hľadiska spätá s dymom či ohňom. Vkladanie hustoty zabezpečuje compute shader, ktorý generuje tzv. *gaussovský splat*, viď ukážka kódu:

```
gaussian = exp( - distance / 2 * sigma * sigma) / sigma;
quantity = sample(quantityTexture, invocation_id);
injection = quantity * gaussian * deltaTime;
output_value = applyInjectionTypeFactors(injection)
```

Kód 7.1: Generácia gausiánu ako zdrojového silového impulzu

Každá z veličín má však svoje parametre, a preto s cieľom prípadnej rozšíriteľnosti musí byť každý algoritmus definovaný vlastnou triedou. V knižnici je preto trieda `DensityInjection`, ktorá implementuje rozhranie `IInjection`, spoločné pre všetky veličiny.
Keďže advekcia rýchlosti a vloženie silového impulzu do vektorového poľa rýchlosti predstavujú hlavné úskalie simulačného algoritmu, je možné nadvzorkovať textúru hustoty 2× až

4×, čím je možné dosiahnuť zaujímavejšie vizuálne výsledky. V takom prípade je ale nutné interpolovať vektory rýchlostí, čo z numerického hľadiska nie je správne, nakoľko interpolované rýchlosti nie sú bez divergencie. V aplikácii ma však táto textúra z dôvodu zachovania konzistencie dát rovnaké rozlíšenie ako textúra rýchlosti.

7.2.2 Teplota

Pridanie hustoty ako ďalšej veličiny umožňuje jednoduchú vizualizáciu simulovanej tekutiny. Tá je však pomerne neatraktívna, preto je simulácia rozšírená o veličinu teploty. Narozdiel od hustoty má teplota priamy vplyv na správanie tekutiny, nakoľko ovplyvňuje to ako sa tekutina klesá či stúpa. V knižnici je tiež reprezentovaná triedou `vfx::Quantity`, nakoľko je táto veličina taktiež advekovaná. Teplota ako taká je skalárna veličina a preto je dostatočné reprezentovať ju 3d textúrou vo 1 kanálovom formáte s polovičnou presnosťou (`GL_R16F`). Podobne ako v prípade hustoty, je nutné definovať algoritmus vloženia do domény, ktorý reprezentuje trieda `TempInjection`.

7.3 Dym a oheň

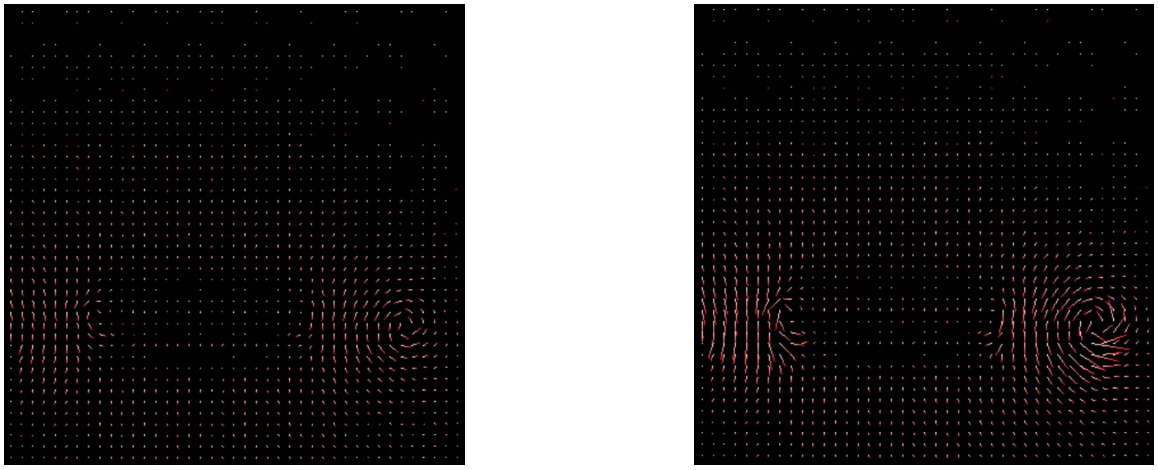
Pre vierohodnú simuláciu efektov dymu a ohňa je potrebné okrem spomínaných veličín zaintegrovať do simulátoru aj externé sily (sekcia 3.2), ktoré sú pre tieto fenomény špecifické. Prvou[1] z implementovaných externých síl je *vztlaková sila*. Navzorkovaním textúr hustoty a teploty sa tieto hodnoty odčítajú pričom je výsledok nutné opraviť vektorom vertikálneho smeru. Takto sa spočíta nový offset smer toku tekutiny, ktorý sa následne aplikuje na aktuálnu vzorku rýchlosti. Výpočet teda neprodukuje žiadne medzivýsledky. Okrem toho je tento výpočet parametrizovaný koeficientom magnitúdy tejto sily. Do simulácie vztlakovej sily sú v aplikácii zavedené okrem koeficientu vztlaku aj ďalšie parametre: **okolitá teplota** a **váha hustoty**. Tieto parametre priamo ovplyvňujú chovanie toku pri stúpaní, najmä z fyzikálneho hľadiska (stúpanie a klesanie na základe teploty). Aj napriek tomu, že sa jedná o typickú vlastnosť plynových efektov, aplikácia umožňuje túto časť simulácie vypustiť, čo otvára možnosť rozšírenia pre iné efekty.

Ďalším implementovaním efektom je *vírivosť tekutiny*. Implementácia efektu pozostáva z dvojice compute shaderov, pričom prvý realizuje výpočet samotnej vírivosti tak, že každá invokácia získa z 6-okolia aktuálnej bunky vzorky rýchlostí a aplikuje operátor popísaný rovnicou 3.17.

Výsledok reprezentovaný 3D smerovým vektorom sa uchová vo farebných kanáloch 4 zložkovej 3D textúry, pričom do alpha kanálu sa prevypočíta dĺžka tohto vektoru ω, ktorá je parametrom v nasledujúceh fáze výpočtu (*confinement*). V tej sa z 6-okolia aktuálneho vzorku z textúry vírivosti spočítanej v predošlom kroku získajú vzorky ω a ich rozdielom v každej dimenzii získame nový smerový vektor, ktorým sa upraví pôvodný smerový vektor rýchlosti. Výpočet je parametrizovaný **magnitúdou** tohto vektoru.

Na obrázku 7.3 je zobrazené vektorové pole rýchlostí bez a s aplikovanou vírivosťou (obrázok s vírivosťou je zveličený pre demonštračne účely).

[1] Je dôležité, aby sa vztlaková sila aplikovala po advekcii, resp. po aplikovaní silových impluzov



(a) bez vírivosti (b) s vírivosťou

Obrázok 7.3: Vizualizácia vektorov rýchlosti a vírivosti, prevzaté z [**?**]

7.4 Projekcia tlaku

Najdôležitejšou časťou simulácie je výpočet projekcie tlaku. Výpočet je implementačne rozdelený do troch častí:

- výpočet divergencie rýchlostí
- riešenie numerických rovníc projekcie tlaku
- aplikácia gradientu

Výpočet *divergencie* rýchlosti, ktorá je potrebná v ďalších štádiách simulácie realizuje compute shader ktorého vstupom je textúra rýchlostí, z ktorej sa spočíta rozdiel v rýchlostiach v okolitých bunkách . Výsledkom je textúra divergencie rýchlostí ktorá uchováva skalárnu informáciu odchýlky. Tá sa následne použije vo výpočte tlaku. Výpočet tlaku je implementovaný *Jacobiho* metódou, ktorá je veľmi jednoduchá na implementáciu na GPU. Každý kernel vyčíta vzorky z textúry tlaku z predošlej iterácie výpočtu z okolitých buniek, pričom sa spočíta ich priemer upravený o divergenciu rýchlostí spočítaný v predošlom kroku. Výsledkom je taktiež textúra skalárnych hodnôt. Compute shader ktorý rieši výpočet produkuje výsledky v rámci jednej iterácie, preto je výpočet cyklicky spúšťaný z CPU, pričom dochádza k prepínaniu vstupných a výstupných textúr. Počet iterácií je parametrizovateľný, pričom štandardne je predvolených 20 iterácií. V každom novom snímku **musi byť vynulovaná textúra tlaku**. Posledným krokom projekcie tlaku je odčítanie gradientu od aktuálneho vektoru rýchlosti, ktorý sa spočíta projekčným compute shaderom vytvorením masky rýchlosti v danej bunke, navzorkovaním tlaku z okolitých rýchlostí a ich odčítaním. Takto sa implementačne vynúti splnenie podmienky nestlačiteľnosti pretože sa vynuluje divergencia rýchlosti. Výpočet je parametrizovaný škálovateľnou **veľkosťou** gradientu tlaku.

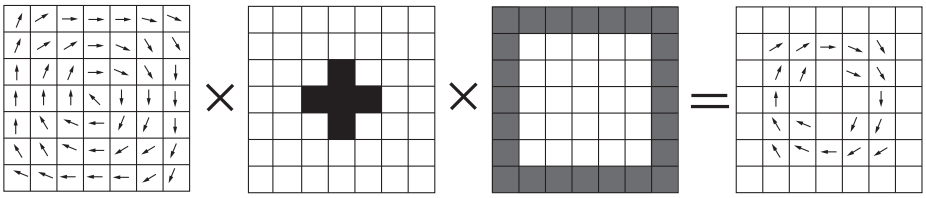
7.5 Prekážky

Pre zvýšenie flexibility, korektnosti a použiteľnosti simulácie tekutín, je do aplikácie pridaná podporu pre použitie prekážok (angl. *obstacles*). Simulácia dokáže uvažovať pevné

prekážky, ktoré daná tekutina musí obtekať. Prekážky sú na strane CPU reprezentované triedami odvodenými od abstraktnej triedy `vfx::Obstacle`. Táto trieda definuje rozhranie pre spoločné operácie vykonávané nad prekážkami ako aj definuje spoločné vlastnosti ako je pozícia v simulačnej doméne. Trieda umožňuje vyplnenie objektu do 3D textúry metódou `fill()` ako aj gaussovskú filtráciu prekážky v diskrétnom 3D priestoru metódou `blur()`. Konkrétne typy prekážok sú definované odvodenými triedami `vfx::Sphere` a `vfx::Cube`, pričom každá z nich definuje potrebné parametre pre svoj typ: *rádius*, prípadne *rozmery*, *compute shader* atď. Objekt je na GPU reprezentovaný jednou 3D textúrou s jedným farebným kanálom s nízkou presnosťou, teda formátom `GL_R8`. Textúra v podstate predstavuje masku simulačnej domény, nakoľko je potrebné uchovávať iba informáciu typu `True`/`False` a teda či daná bunka obsahuje prekážku alebo nie. Následne sa táto textúra zavedie do vybraných fází simulácie, kde sa z nej navzrokujú dáta a podľa danej fázy sa modifikuje výpočet veličiny, ktorú aktuálna fáza spracováva, viď ukážku zavedenie prekážky, ako aj hraníc do výpočtu advekcie na obr. 7.4.

Integrácia prekážok do aplikácie pozostáva z týchto krokov:

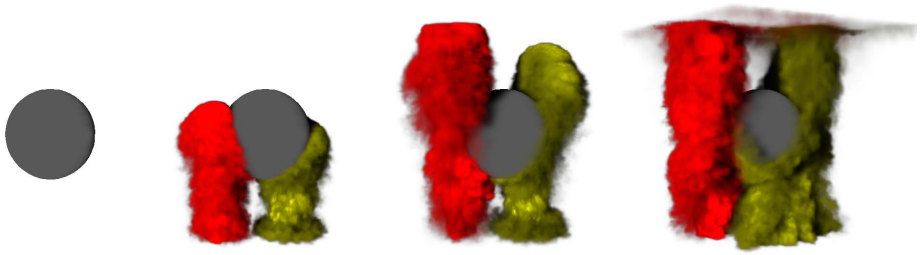
1. vytvorenie shaderu vyplňujúci objekt do diskrétnej domény
2. integrácia do výpočtu tlaku
3. integrácia do výpočtu divergencie rýchlosti
4. integrácia do výpočtu projekcie a odčítaniu gradientu
5. zahrnutie objektu do vykresľovania
6. sprístupnenie parametrizácie prostredníctvom rozhrania knižnice



Obrázok 7.4: Hraničné podmienky a prekážky, advekcia rýchlosti.

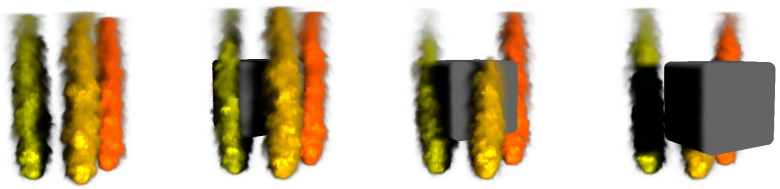
Vykreslenie telesa do 3D textúry je jednoduché: najprv sa textúra vynuluje, aby sa správne reprezentovala maska simulačnej domény a následne sa na základe parametrického popisu geometrického telesa a aktuálneho koordinátu spusteného vlákna a požadovaného polomeru spočíta, či daný voxel spadá do jeho objemu. V implementácii je do tohto kroku taktiež pridané uchovávanie informácie o hraniciach simulačnej domény, pričom je nutné hodnoty hranice a samotného telesa dostatočne diverzifikovať (v tomto prípade boli zvolené hodnoty: $hranica = 0.01$, $teleso = 1.0$).

Jedným z nedostatkov advekčných metód je schopnosť prenikania prenášaných veličín cez hranice objektov ako aj simulačnej domény. Dochádza k tomu pri veľkých časových krokoch počas simulácie. Pridaním prekážky do scény je potrebné pridať dodatočnú kontrolu, ktorou sa tomuto javu vyhneme. Pri vzorkovaní dát z objemovej textúry tlaku šíriaceho sa v doméne



Obrázok 7.5: Dym s prekážkou v tvare gule

v okolí aktuálne spracovávanej bunky, je potrebné zabezpečiť aby v prípade, že tieto bunky sú bunkami pevných telies došlo k vynulovaniu vplyvu tlaku.
Prekážky použité v aplikácii sú štandardne statické, čo znamená, že za behu aplikácie ich prakticky stačí do 3D textúry uložiť iba raz. Užívateľ však môže prostredníctvom rozhrania meniť rôzne parametre týkajúce sa danej prekážky, kde jednou z nich je práve pozícia v rámci simulačnej domény. Táto operácia vynucuje invokáciu compute shaderu vyplňujúceho útvar do textúry jeden krát za jeden snímok, pri kontinuálnom pohybe. Nakoľko simulácia prebieha



Obrázok 7.6: Dym s pohyblivou prekážkou v tvare kocky

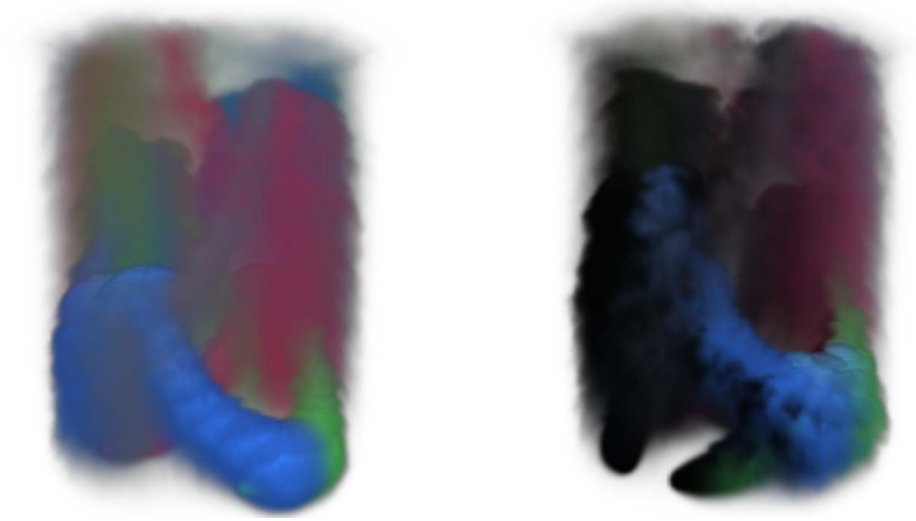
7.6 Rendering simulovaných dát

Nemenej dôležitou súčasťou implementácie volumetrických efektov založených na simulácii tekutín je vizualizácia simulovaného toku veličín v simulačnej doméne. Pre vizualizáciu bola zvolená technika ray-marching, pričom sa pred samotným vykresľovaním vykoná post-processing (gaussova filtrácia) nad výstupom dát zo simulácie. Tie predstavuje skupina 3D textúr, ktoré uchovávajú diskretizované dáta (pamäťovej náročnosti sa venuje kapitola 8, konkrétne tabuľka 8.5). Keďže dáta sú reprezentované 3D textúrami a vzniká potreba filtrácie dát je táto technika veľmi výhodná nakoľko OpenGL poskytuje vstavanú možnosť trilineárnej interpolácie.

7.6.1 Výpočet osvetlenia a tieňov

S cieľom dosiahnutia väčšej hĺbky je do výpočtu osvetlenia zaintegrovaný výpočet tieňov, ktoré vrhá dym, prípadne teleso umiestnené v simulačnej doméne. Aplikácia realizovuje vý-

počet osvetlenia a tieňov prostredníctvom samostatného compute shaderu, pričom výpočet prebieha ako pred-fáza raymarchingu realizujúceho celkový vizuálny výstup. Ten taktiež využíva ray-marching, pričom sa vzorkuje textúra hustoty a prekážok, ktoré sú však na rozdiel od výsledného vizualizačného výpočtu vzorkované s podstatne menším krokom - štandardne je v implementácii preddefinovaný krok 16 vzorkov. V prípade prieniku lúča s prekážkou, je zrejmé, že je tento texel zatienený, v opačnom prípade dochádza k akumulácii osvetlenia. Porovnanie výstupu s výpočtom tieňov (16 vzorkov) a bez neho demonštruje obrázok 7.7.



(a) bez volumetrických tieňov (b) s volumetrickými tieňmi

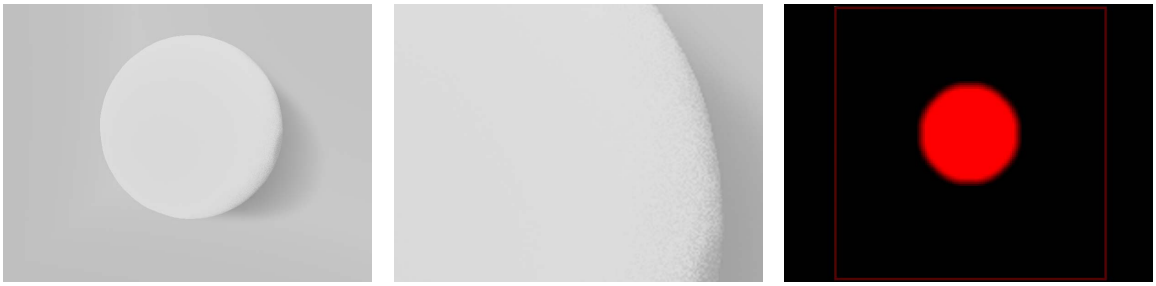
Obrázok 7.7: Integrácia výpočtu tieňov

7.6.2 Filtrácia objemových dát

Nakoľko je simulácia tekutín výpočetne náročná, pre dosiahnutie výsledkov v reálnom čase je na bežnom GPU nutné zvoliť pomerne malé rozmery simulačnej domény. Typicky sa využívajú veľkosti v rozmedzí $100 - 200$ voxelov v jednej dimenzii. V 3D priestore sa však už aj pri tejto veľkosti pohybujeme v rozmedzí niekoľkých miliónov voxelov, pričom presnosť simulácie je aj napriek tomu pomerne nízka a produkuje značne veľký aliasing dát. Práve preto je v aplikácii integrovaná filtrácia rozličných objemových dát, ktorá zjemňuje aliasing a zväčšuje prívetivosť vizuálneho výstupu. Filtráciu realizuje trojdimenzionálny *Gaussov filter*[11], pričom implementácia využíva fakt, že tento filter je separabilný a teda je implemetovaný generickým compute shaderom pre všetky dimenzie. Pred aplikáciou filtru sa najskôr na CPU spočítajú váhy, na základe zvolenej veľkosti filtru, ktoré sú spoločné pre všetky dimenzie. Následne sa pre každú dimenziu spočítajú vektorové offsety, z ktorých sa bude vzorkovať. Oba tieto vektory sú distribuované do shaderu prostredníctvom `Shader Storage Buffer Objectov` (každý osobitne nakoľko OpenGL neumožňuje posielať do shaderu viac polí s neobmedzenou veľkosťou). Nakoniec s použitím týchto hodnôt shader aplikuje filter v danej dimenzii. Je zrejmé, že aj v tomto prípade je nutné využiť ping-pongovanie textúr. Na obrázku 7.8 je ukážka filtrácia prekážky a tieňov v demonštračnej aplikácii s jadrom 5×5, priemerom gule/strana kocky $r = 52$, parametrom rozmazania $\sigma = 3$, raymarching: 64 vzorkov, 16 vzorkov pre výpočet tieňov, jittering($p = 1.0$). Simu-

lačná doména $128 \times 128 \times 128$. Obrázky vľavo zobrazujú teleso po postprocessingu zobrazené v rozlíšení $1920 \times 1080px$. Obrázky v strede zobrazujú detail pred/po aplikácii filtru na teleso a tiene, obrázky vpravo zobrazujú jednu vrstvu zdrojových volumetrických dát.

(a) guľa bez použitia filtru (b) detail bez použitia filtru (c) vrstva 3D textúry bez filtru



(d) guľa s použitím filtru (e) detail s použitím filtru (f) vrstva 3D s použitím filtru

Obrázok 7.8: Filtrácia prekážok a tieňov

Aplikácia celkovo umožňuje filtrovať niekoľko druhov dát:

- veličiny prenášané v toku tekutiny
- prekážky umiestnené v simulačnej doméne
- tiene

Pri vytváraní efektu dymu je veľmi výhodné aplikovať filtráciu na textúru hustoty (obrázok 7.9), ktorá nesie hlavnú informáciu o objem tekutiny. Môžeme sa tak rovnako vyhnúť artefaktom ktoré môžu vznikať.

7.6.3 Jittering

Algoritmus raymarchingu, obohatený o predfiltrované dáta, ktorý bol do tohto bodu implementovaný stále produkuje viditeľné artefakty. Práve pre zlepšenie vizuálneho výstupu je do aplikácie v rámci compute shaderu realizujúceho raymarching zavedený jittering a to pri výpočte osvetlenia počas priechodu lúčov simulačnou doménou ako aj počas výpočtu tieňov. Podobne ako v prípade filtrácie je možné parametrizovať aj jittering. Dôvodom implementácie jitteringu je fakt, že počas marchingu môžu vznikať artefakty pripomínajúce hladké segmenty, ktoré vznikajú na hraniciach dát s veľkým rozdielom v aktuálnej hĺbke pretože sa dáta z textúr vzorkujú v pravidelných krokoch. Tento artefakt je zobrazený na obrázkoch 7.9a, 7.8a, 7.8b. Pri vzorkovaní sa aplikuje jitter na výpočet nového textúrovacieho koordinátu nasledovne:

(a) $\sigma = 0$ (b) $\sigma = 0.5$ (c) $\sigma = 3$

Obrázok 7.9: Filtrácia hustoty hustoty dymu

```
tracingCoord = rayStart + mix(-jitter/2, jitter/2, rand()) * step * rayDir;
```

Kód 7.2: Jittering

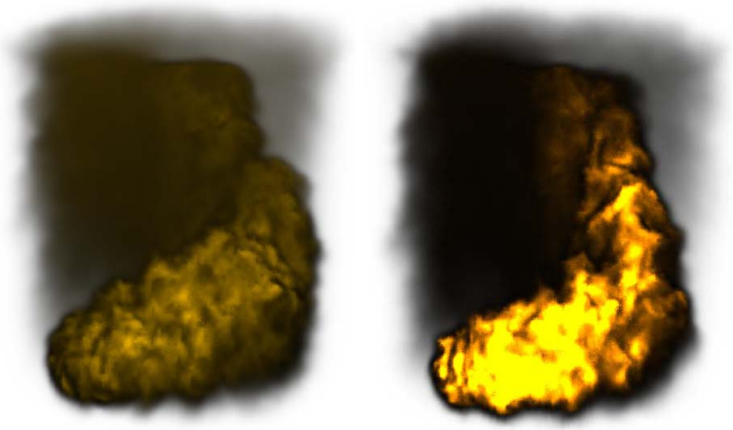
Vďaka podpore rozmazávacieho filtru je možné tieto techniky skombinovať (obr. 7.10) a dosiahnuť tak zaujímavejší výstup.

(a) rozmazanie (b) jitter (c) jittering + rozmazanie

Obrázok 7.10: Aplikácia jitteru

7.6.4 Scattering a vyžarovanie

Ďalším implementovaným efektom je volumetrický scattering, ktorý je pri efektoch dymu a ohňa veľmi dôležitý. Aplikácia tento efekt aproximuje tak, že sa vypočíta osvetlenie a uloží sa do 3D textúry, a pri ray-marchingu sa rozmaže textúra hustoty a skombinuje sa s pôvodnou nerozmazanou textúrou. Aplikácie tohto efektu je možné pomerne presvedčivo aproximovať efekt ohňa (obrázok 7.11b).
Okrem toho je možné aplikovať efekt vyžarovania na základe teploty, ktorý je taktiež implementovaný v rámci ray-marchingu. Jedná sa o aproximáciu, ktorá vhodným skombinovaním farby akumulovanej v priebehu priechodu lúča telesom aplikuje v každom kroku navzorkovanú teplotu (ktorú je taktiež možné prefiltrovať), a na základe parametrizácie atenuácie sa nasledovne spočíta nová farba aktuálneho vzorku:



(a) Bez scatteringu (b) so scatteringom

Obrázok 7.11: Scattering

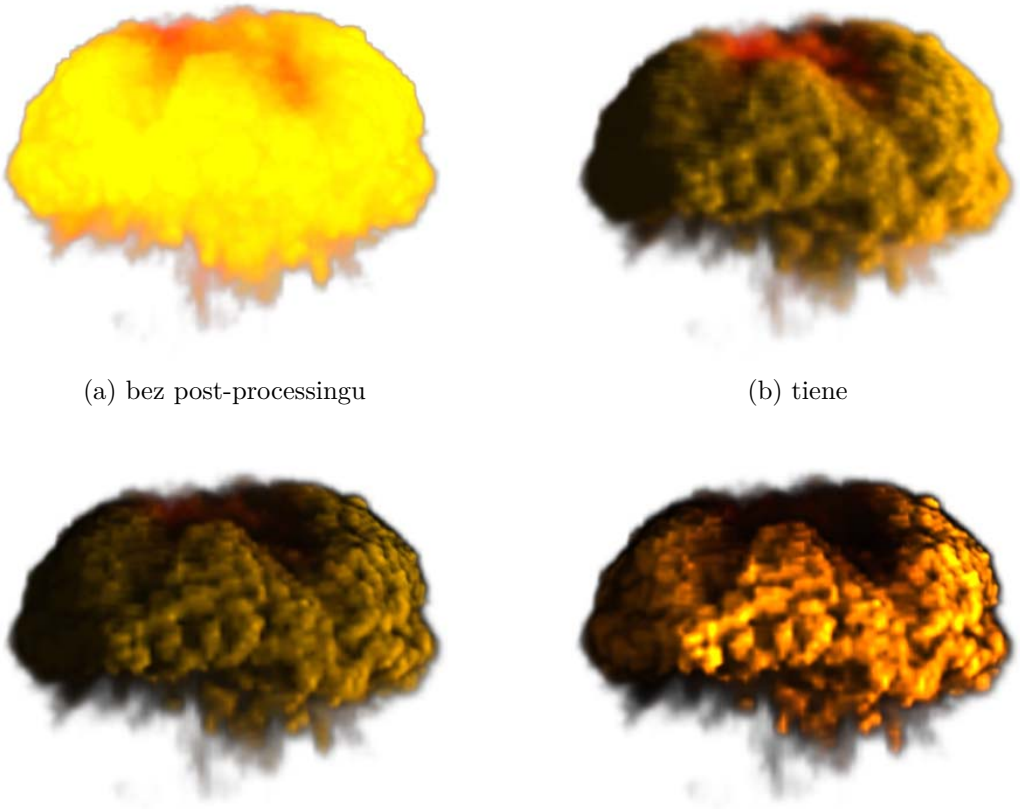
```
radiance_color = lerp(accumulated_color, reference_color, exp(-temperature*↩
    temperature/attenuation));
```

Kód 7.3: Jittering

Tieto efekty je možné rôzne kombinovať a parametrizáciou samotných efektov ako aj osvetlenia, je možné dosiahnuť rozličných vizuálnych výstupov (obrázok 7.12).

7.7 Debugging

Jadro knižnice, ktorá zabezpečuje simuláciu a vykresľovanie volumetrických efektov, tvorí najmä množstvo GPGPU výpočtov využívajúcich compute shadery, čo zvyšuje obtiažnosť implementácie a ladenia programu. Okrem toho knižnica pracuje s 3D textúrami, ktoré rovnako neumožňujú pohodlné skúmanie prípadných chýb pri vývoji. Samotný debugging pri implementácii simulátoru volumetrických efektov zaberá množstvo času, a preto som pri vývoji využil aplikáciu `RenderDoc`, viď obrázok **??**, teda grafický debugger, ktorý poskytuje množstvo užitočných nástrojov ako vizualizácia obsahu volumetrických dát, obsah Shader Storage Buffer Objectov či Uniform Buffer Objectov, uniformných premenných ale aj prehľad jednotlivých dispatchov compute shaderov, descriptor setov, draw calls atď.
RenderDoc je veľmi jednoduchý na používanie, pretože ma intuitívne grafické rozhranie. Stále sa však jedná o externý nástroj. Pri vývoji v Microsoft Visual Studio je vhodné siahnuť po nástroji `Nsight`, ktorý je však obmedzený iba na grafické čipy od spoločnosti NVIDIA, pričom má slabšiu podporu v novších verziách Visual Studia a horšiu podporu starších generácií grafických čipov. Práve preto som zvolil nástroj RenderDoc, ktorý v tejto oblasti nezaostáva.



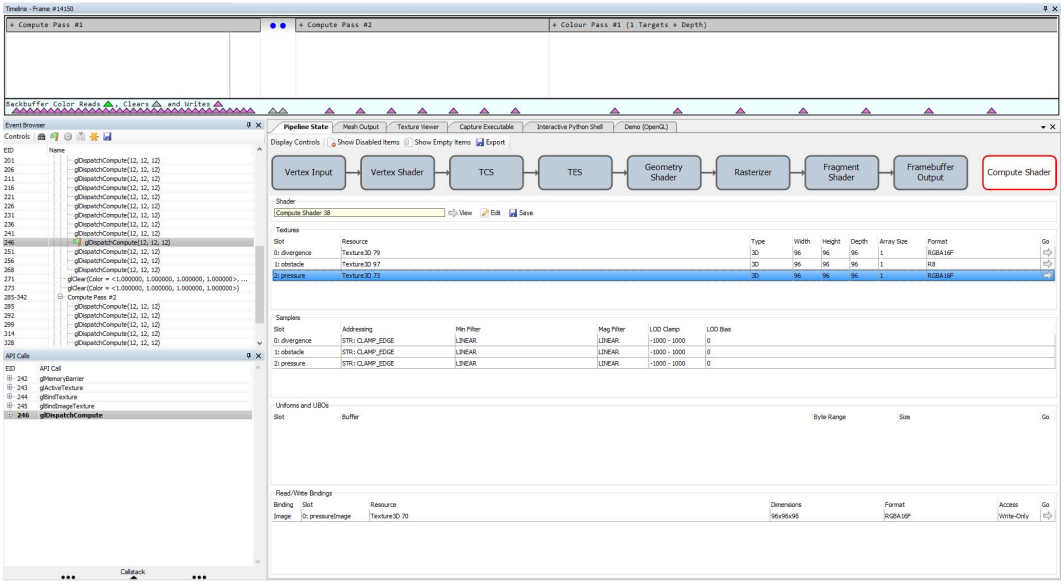
(a) bez post-processingu

(b) tiene

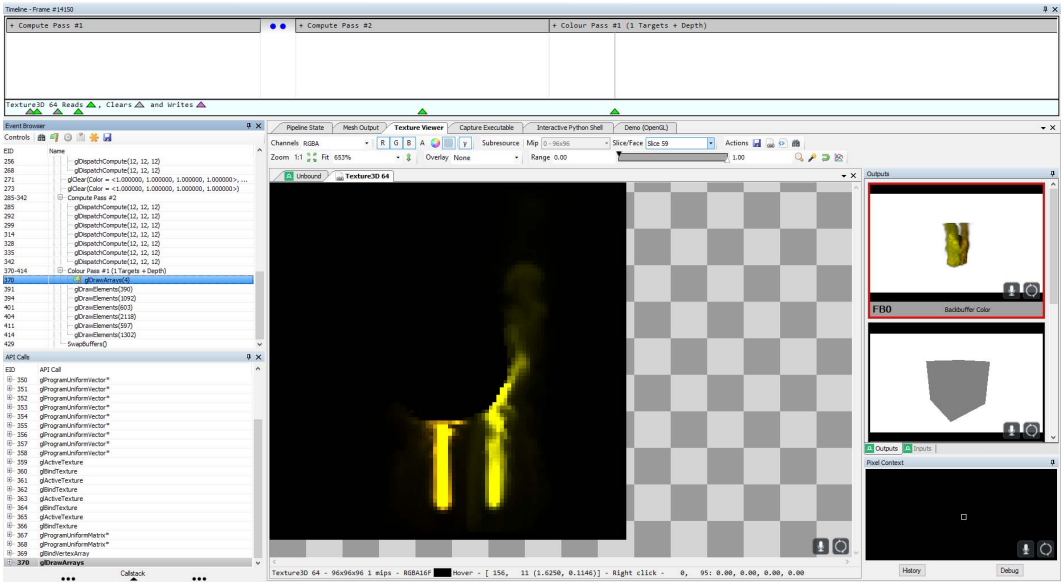
(c) tiene + radiácia

(d) tiene + radiácia + scattering

Obrázok 7.12: Efekty aplikované počas ray-marchingu pri explózii.



(a) Debugging dispatchu úlohu pre výpočet prenosu tlaku Jacobiho metódou



(b) Vizualizácia obsahu framebufferu a vrstvy 3D textúry

Obrázok 7.13: Grafický debugger RenderDoc

Kapitola 8

Analýza a zhodnotenie výsledkov

V tejto kapitole sú uvedené výsledky testov, ktorým bol implementovaný simulátor tekutín podrobený. Rovnako sú uvedené výsledky z testov implementácie vykresľovania. Testovanie prebiehalo pod operačným systémom Windows 10 na troch testovacích zostavách, uvedených v tabuľke 8.1. Pri prvej z uvedených zostáv sa jedná o mobilné verzie CPU aj GPU, v ostatných prípadoch sa jedná o desktopové verzie.

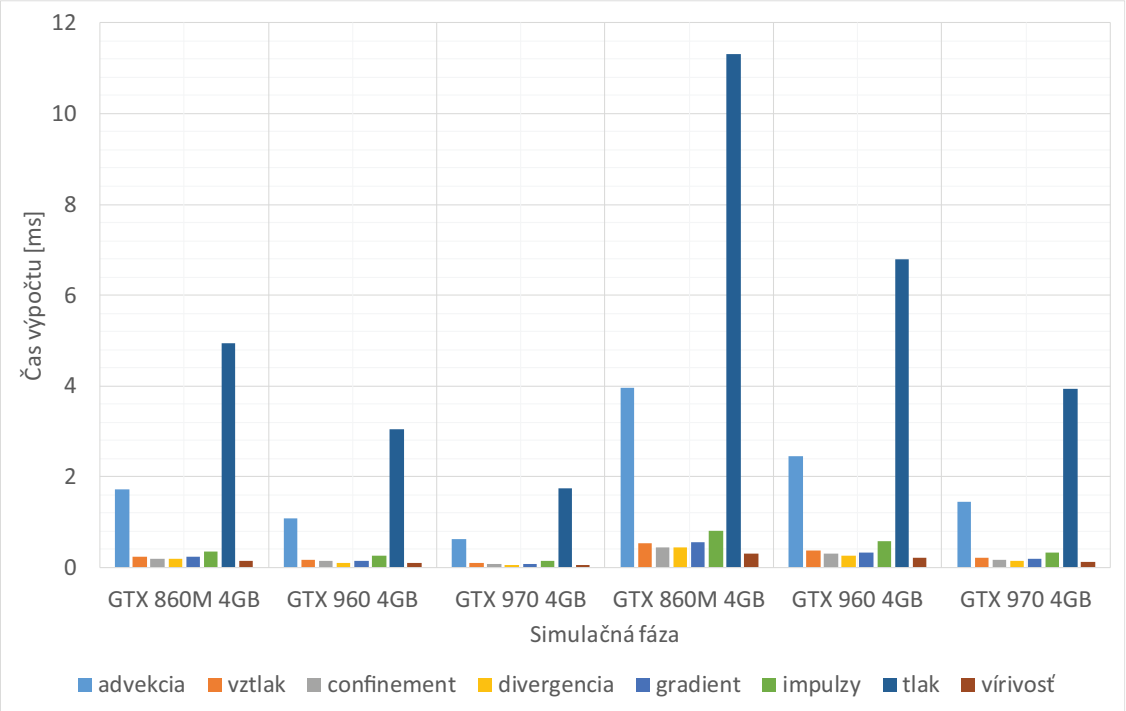
CPU	GPU
Intel(R) Core(TM) i7-4710HQ @ 2.50 GHz	NVIDIA GeForce GTX 860M 4GB
Intel(R) Core(TM) i3-4170 @ 3.7 GHz	GIGABYTE GTX 960 WINDFORCE 2X OC Gaming 4GB
Intel(R) Core(TM) i5-4670K @ 3.40 GHz	GIGABYTE GTX 970 WINDFORCE 3X OC Gaming 4GB

Tabuľka 8.1: Testovacie zostavy

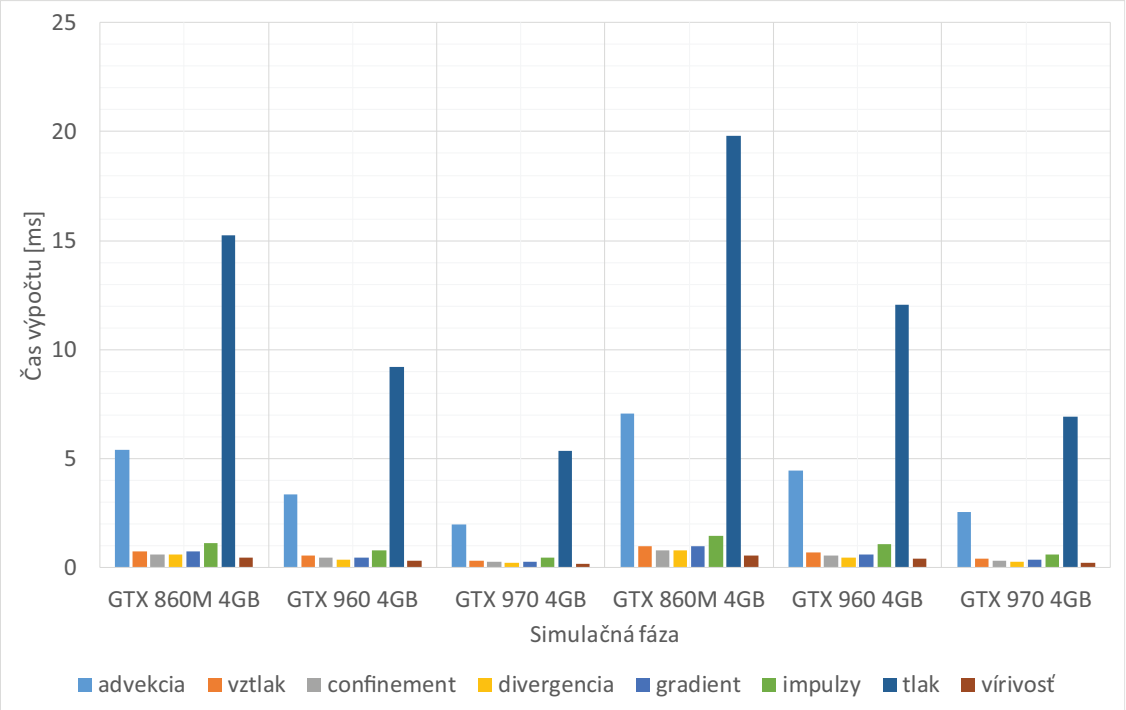
Testovanie sa zameriava najmä na analýzu časov potrebných na výpočet jednotlivých fáz simulácie a vykresľovania, a vplyv parametrov na tieto fázy. Samotné meranie prebiehalo s využitím dedikovaného mechanizmu na profilovanie - `OpenGL queries`. Pre účely analýzy časovej náročnosti jednotlivých fází bol implementovaný jednoduchý profiler, ktorý automatizovane meral a logoval namerané časy do formátu csv.

8.1 Veľkosť simulačnej domény

Jeden zo základných testov implementácie je vplyv veľkosti simulačnej domény na výpočet jednotlivých simulačných fáz. Namerané hodnoty sú uvedené v grafe a tabuľke na obrázku 8.1. Pre test boli zvolené 4 veľkosti mriežky, pričom je dôležité aby tieto rozmery boli násobkom 2. Testované boli domény s rovnakými rozmermi vo všetkých dimenziách ale aj tie nepravidelné. V rámci testu boli zmerané časy výpočtov v jednotlivých fázach 100 krát, pričom meranie sa v každom prípade spúštalo od 150. snímku od počiatku simulácie. Výsledky boli spriemerované. Z nameraných výsledkov je zrejmé, že drobné zmeny vo veľkosti simulačnej mriežky majú obrovský dopad na rýchlosť simulácie - pri zmene veľkosti mriežky v dvoch dimenziách o $\frac{1}{3}$ sa výpočet predĺžil viac než 2×.



(a) Mriežky o rozmeroch $64 \times 128 \times 64$ (vľavo) a $96 \times 128 \times 96$ (vpravo)



(b) Mriežky o rozmeroch $112 \times 128 \times 112$ (vľavo) a $128 \times 128 \times 128$ (vpravo)

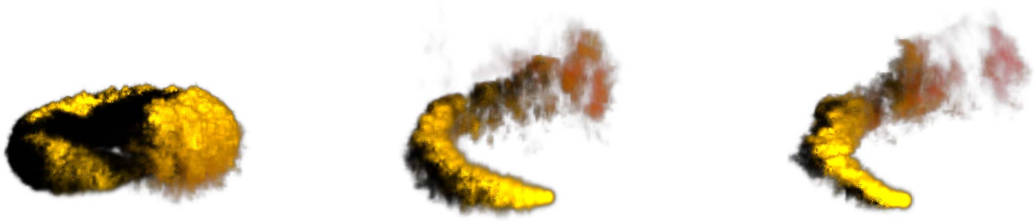
Graf 8.1: Porovnanie času výpočtu jednotlivých fází simulácie na rôznych grafických kartách pri rôznych veľkostiach simulačnej domény.

8.2 Riešenie prenosu tlaku

Okrem toho test ukazuje, že najkritickejšou fázou celej simulácie je výpočet tlaku, na ktorom je simulácia postavená. Test zachytáva výpočet tlaku v 20 iteráciách, ktoré predstavujú rozumnú voľbu s ohľadom na pomer rýchlosť/vizuálny výstup (obrázok 8.2, zobrazuje porovnanie výstupov pri rôznom počte iterácií Jacobiho metódy). Nakoľko implementácia využíva pravidelnú reprezentáciu simulačnej domény, očakávaná časová zložitosť výpočtu tohto člena rovnice Navier-Stokesových rovníc je lineárna. Túto skutočnosť potvrdzuje meranie času výpočtu riešenia rovnice tlaku pre niekoľko zvolených počtov iterácií, ktorých výsledky sú v tabuľke 8.2.

	čas výpočtu [ms]		
počet iterácií	GTX 860M 4GB	GTX 960 4GB	GTX 970 4GB
10	16.45648755	12.66481487	7.35648774
20	32.16729796	23.54145782	14.9191254
30	47.12165463	37.25486496	22.2485421
40	65.89846455	47.52684578	29.6197264
50	82.77437391	60.32685124	37.0245105

Tabuľka 8.2: Meranie času výpočtu riešenia tlaku ako člena Navier-Stokesových rovníc, veľkosť simulačnej domény: $156 \times 156 \times 156$



(a) 1 iterácia (b) 20 iterácií (c) 50 iterácií

Obrázok 8.2: Vizuálny výstup pri rôznom počte iterácií riešenia rovnice tlaku. Na základe experimentácie sa javí ako rozumná voľba počtu iterácií v rozmedzí hodnôt [10,20] nakoľko väčší počet iterácií ma minimálny pozitívny dopad na simuláciu.

8.3 Advekčné metódy

Merania časov výpočtu jednotlivých fází ďalej ukazuje, že advekcia je ďalšou z výpočetne náročných fází simulácie. V rámci testu bola použitá Mac Cormackova metóda advekcie, ktorá produkuje kvalitnejšie výsledky ako metóda semi-Lagrange (vizuálne porovnanie metód advekcie zobrazuje obrázok 8.3), aj keď je výpočetne náročnejšia - viď meranie zachytené v tabuľke 8.3. Pokiaľ však chceme dosiahnuť väčší stupeň detailu, ktorý sa najjednoduchšie dosiahne zväčšením hustoty simulačnej mriežky, je vhodné zmeniť metódu advekcie z

Obrázok 8.3: Porovnanie metód advekcie: Semi-Lagrangeova metóda (vľavo), Mack Cormack (vpravo).

vyšším rádom presnosti, čo bude mať menší negatívny dopad na výkonnosť. Rovnako je potrebné vziať do úvahy počet advekovaných veličín, nakoľko čas výpočtu advekcie jednej veličiny metódou Mac Cormack je takmer 5× väčší ako pri metóde semi-Lagrange, advekcia touto metódou v simuláciách s veľkým počtom veličín prenášaných v toku tekutiny môže byť značne výpočetne náročná.

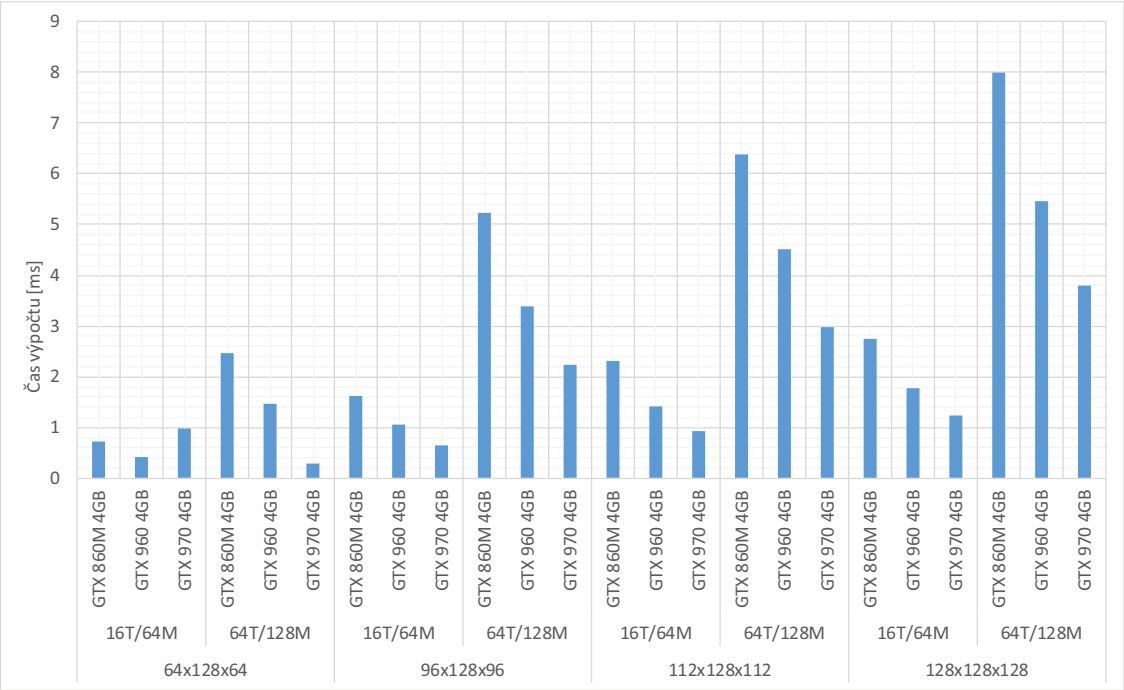
	semi-Lagrange [ms]			Mac Cormack [ms]		
	GTX 860M	GTX 960	GTX 970	GTX 860M	GTX 960	GTX 970
hustota	0.7705	0.5403	0.2381	0.5826	0.5412	0.2299
teplota	0.7661	0.5283	0.2544	3.1404	1.7203	1.5247
rýchlosť	0.5778	0.372	0.2646	3.1212	1.7166	1.5161
spolu	2.1476	1.4406	0.7571	6.9126	3.9781	3.2707

Tabuľka 8.3: Porovnanie výpočetnej náročnosti operácie advekcie. Simulačná mriežka o rozmeroch 128 × 128 × 128. Pozn.: Advekcia rýchlosti pri voľbe metódy Mac Cormack, prebieha metódou semi-Lagrangea, advekcia prenášaných veličín prebieha metódou Mac Cormack.

8.4 Vizualizácia

Predchádzajúce sekcie ukázali, že veľkosť simulačnej mriežky je hlavným faktorom ovplyvňujúcim simuláciu z hľadiska výpočtovej náročnosti. Okrem toho je však nutné zobrať do úvahy to, aký vplyv má zvolenie menšej prípadne väčšej mriežky na rýchlosť výpočtu osvetlenia, tieňov, filtrácie a na celkový vizuálny dojem.

Veľmi dôležitým parametrom s ohľadom na časovú zložitosť výpočtu pri vykresľovaní je veľkosť kroku ray-marchingu, obrázok 8.4, či už pri výpočte osvetlenia, tieňov alebo vzorkovaní hustoty. Rozdielne veľkosti pri každom z týchto priechodov majú však aj rôzny dopad na vizuálny výstup. Pri výpočte tieňov, obrázok 8.5, totiž postačuje veľmi veľký krok pričom pri výslednom marchingu akumulovaním hustoty a osvetlenia dosiahneme kvalitnejšie výsledky s menšími krokmi (obrázok 8.6).



Graf 8.4: Porovnanie časovej náročnosti výpočtu osvetlenia a tieňov pri rôznych veľkostiach simulačnej mriežky, s rôznym počtom vzorkov ako veľkosť kroku: 16T/64M značí, že pri výpočte bol použitý krok $\frac{1}{16}$ vzorkov pri výpočte tieňov a krok $\frac{1}{64}$ vzorkov pre ray-marching.

(a) krok $\frac{1}{16}$ (b) krok $\frac{1}{64}$ (c) krok $\frac{1}{128}$

Obrázok 8.5: Vykreslenie tieňov pri rôznych veľkostiach krokov ray-marchingu. Simulačná doména: $128 \times 128 \times 128$. Použitie príliš malého kroku je pri výpočte tieňov zbytočné, nakoľko má značný dopad na čas výpočtu, a minimálny prínos z vizuálneho hľadiska, viď obrázok v strede a vpravo.

Z hľadiska výpočetnej náročnosti pri vykresľovaní je potrebné zobrať do úvahy aj filtráciu dát. Tabuľka 8.4 zobrazuje časy filtrácie **jednej** 3D textúry v jednom snímku. Keďže dáta sú uložené v pravidelnej mriežke, pri filtrácii dát nedochádza k optimalizácii na základe informácie, či sa v danej oblasti nachádza zvolená veličina alebo nie. Dôsledkom toho je, že je filtrácia rovnako výpočetne náročná pre všetky typy dát uložených v textúrach. V

(a) krok $\frac{1}{64}$ (b) krok $\frac{1}{128}$

Obrázok 8.6: Výstupy pri rôznych veľkostiach krokov pri finálnom ray-marchingu. Väčšie kroky (vľavo) produkujú nepresnejšie výsledky, pretože sa získava menší počet vzorkov. Výsledkom sú jasnejšie osvetlené "častice"za cenu väčšieho šumu.

závilosti na užívateľsky zvolených parametroch (filtrácia textúr hustoty, teploty, osvetlenia, tieňov či prekážok) tak lineárne narastá komplexita výpočtu.

	GTX 860M	GTX 960	GTX 970
64x128x64	1.14618954	0.57674464	0.58614242
96x128x96	2.2494118	1.1080918	0.77358086
112x128x112	2.9738451	1.4332568	0.9943121
128x128x128	3.7857251	1.8009239	1.24077926

Tabuľka 8.4: Doba výpočtu filtrácie 3D textúry rôznych veľkostí.

8.5 Pamäťové nároky

Simulácia používa pomerne veľké množstvo 3D textúr (tabuľka 8.5), z ktorých niektoré je nutné uchovávať 2x kvôli možnosti prepínania, prípadne 3x kvôli filtrácii pred vykresľovaním. Celková pamäťová náročnosť jednej simulačnej domény sa blížia k 100 MB, ktorú je možné vykresliť s viacerými zdrojmi, no pokiaľ sa zvýši potreba simulovať rozličné efekty, je potrebné vytvoriť novú instanciu, čo vedie k nutnosti vytvoriť novú instanciu simulátora.

	Počet	Typ	Rozmery	Pamäť
Rýchlosť	2	GL_RGBA16F	128x128x128	32MB
Hustota	2	GL_RGBA16F	128x128x128	32MB
Teplota	2	GL_R16F	128x128x128	8MB
Divergencia	1	GL_R16F	128x128x128	4MB
Tlak	2	GL_R16F	128x128x128	8MB
Vírivosť	1	GL_RGBA16F	128x128x128	16MB
Prekážky (guľa, kocka, hranice)	1	GL_R8	128x128x128	2MB
Rozmazanie	4	1x GL_RGBA16F, 1x GL_R16F 1x GL_R8	128x128x128 128x128x128 128x128x128	22MB
Osvetlenie a tiene	1	GL_RGBA16F	128x128x128	16MB

Tabuľka 8.5: Pamäťová náročnosť simulácie

Kapitola 9

Možnosti ďalšieho vývoja

Táto práca ponúka stabilný základ pre rendering volumetrických efektov vychádzajúcich z fyzikálnych princípov tekutín. Keďže sa zameriava najmä na efekt dymu a ohňa, otvára sa možnosť rozšíriť funkcionalitu o ďalšie efekty, ako napríklad simulácia vody, či simulácia mračien. Zaujímavým rozšírením je aj podpora dynamických prekážok, ktorá by umožňovala dynamické správanie simulačnej domény. Prekážka by sa v tomto prípade musela v každom snímku zvoxelizovať do 3D textúry (pokiaľ by sa jednalo o komplexnejší model) a zároveň by sa trieda `vfx::Obstacle` musela rozšíriť o textúru, ktorá by uchovávala informáciu o rýchlosti pohybu prekážky v danej bunke simulačnej domény. Z hľadiska výkonnosti a optimalizácií by bolo výzvou využiť na reprezentáciu simulačnej domény pokročilejšie dátové štruktúry deliace priestor do stromových štruktúr ako napríklad *Sparse Octree*, prípadne využiť sparse textúry. Na základe výsledkov analýzy z predošlej kapitoly sa otvára otázka optimalizácie najkritickejšej fázy simulácie a to riešenia tlaku. Možným riešením sú už spomínané stromové štruktúry, nakoľko tie by viedli k relevantným výpočtom iba nad bunkami, ktoré reálne obsahujú dáta. Ďalším z riešení by bola integrácia numerickej metódy, ktorá rýchlejšie konverguje. Jednou z nich je napríklad metóda konjugovaného gradientu[17], ktorá je vhodná pre riešenie riedkych symetrických systémov lineárnych rovníc, prípadne jej kombinácia s multimriežkovými technikami [2]. Z hľadiska zlepšenia vizuálneho výstupu je možné rozšíriť stávajúci model, ktorý uvažuje jednotné rozlíšenia uložených volumetrických tak, že by sa nadvzorkovala napríklad textúra uchovávajúca hustotu. Zaujímavou výzvou môže byť aj odlišný prístup k simulácii [3], predstavený pomerne nedávno, využívajúci Schrödingerovu rovnicu a taktiež Eulerovskú reprezentáciu simulačnej domény.

Kapitola 10

Záver

Cieľom tejto diplomovej práce je predstaviť problematiku volumetrických efektov v počítačovej grafike so zameraním na simuláciu tekutín a vyvoriť demonštračnú aplikáciu, ktorá umožňuje vykresľovanie týchto efektov. V teoretickej časti práce bolo predstavené základné matematické a fyzikálne pozadie správania toku tekutín, ktoré bolo nezbytné naštudovať pre pochopenie tejto problematiky. V úvodných kapitolách sa čitateľ dozvie ľahký úvod do simulácie tekutín a jej spojitosť volumetrickými efektami. Ďalej boli predstavené Navier-Stokesove rovnice, ktoré sú základným stavebným kameňom mnohých minulých aj súčasných simulátorov tekutín. Taktiež je možné sa dozvedieť úvod do problematiky riešenia týchto rovníc a problémov, ktoré počas implementácie takéhoto systému môže čitateľ riešiť, ako je napríklad diskretizácia priestoru, či reprezentácia volumetrických dát. Predstavené boli aj vykresľovacie metódy, ktoré sú vhodné pre puťitie v tejto oblasti, pričom v implementácia sa zameriava na jedinú z nich, ktorá podáva najlepšie výsledky - technika ray-marching. Vytvorená aplikácia umožňuje simuláciu a vykresľovanie volumetrických efektov dymu a ohňa, pričom je možné obe z týchto fáz parametrizovať s cieľom dosiahnutia rozličných efektov. V rámci práce sa taktiež vykonala analýza jednotlivých fáz výpočtu pri simulovaní aj renderovaní, s cieľom identifikácie kritických častí riešenia tejto problematiky.

Literatúra

[1] Biswas, G.: *Introduction to Fluid Mechanics and Fluid Machines,2e*. Tata McGraw-Hill, 2003, ISBN 9780070494978.
URL https://books.google.sk/books?id=Fh18yn0iNOsC

[2] Bolz, J.; Farmer, I.; Grinspun, E.; aj.: Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. *ACM Trans. Graph.*, ročník 22, č. 3, Červenec 2003: s. 917–924, ISSN 0730-0301, doi:10.1145/882262.882364.
URL http://doi.acm.org/10.1145/882262.882364

[3] Chern, A.; Knöppel, F.; Pinkall, U.; aj.: SchrÖDinger's Smoke. *ACM Trans. Graph.*, ročník 35, č. 4, Červenec 2016: s. 77:1–77:13, ISSN 0730-0301, doi:10.1145/2897824.2925868.
URL http://doi.acm.org/10.1145/2897824.2925868

[4] Chorin, A.; Marsden, J.: *A Mathematical Introduction to Fluid Mechanics*. Texts in Applied Mathematics, Springer New York, 2000, ISBN 9780387979182.
URL https://books.google.sk/books?id=0Iglq1WA5PQC

[5] Ciznicki, M.; Kierzynka, M.; Kurowski, K.; aj.: Efficient Isosurface Extraction Using Marching Tetrahedra and Histogram Pyramids on Multiple GPUs. In *PPAM (2)*, editace R. Wyrzykowski; J. Dongarra; K. Karczewski; J. Wasniewski, Lecture Notes in Computer Science, Springer, 2011, ISBN 978-3-642-31499-5, s. 343–352.

[6] Cohen, J. M.; Tariq, S.; Green, S.: Interactive Fluid-particle Simulation Using Translating Eulerian Grids. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, New York, NY, USA: ACM, 2010, ISBN 978-1-60558-939-8, s. 15–22, doi:10.1145/1730804.1730807.
URL http://doi.acm.org/10.1145/1730804.1730807

[7] Crane, K.; Llamas, I.; Tariq, S.: *Real Time Simulation and Rendering of 3D Fluids*, kapitola 30. Addison-Wesley, 2007.

[8] Fedkiw, R.; Stam, J.; Jensen, H. W.: Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, New York, NY, USA: ACM, 2001, ISBN 1-58113-374-X, s. 15–22, doi:10.1145/383259.383260.
URL http://doi.acm.org/10.1145/383259.383260

[9] Foster, N.; Metaxas, D.: Modeling the Motion of a Hot, Turbulent Gas. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, New York, NY, USA: ACM Press/Addison-Wesley

Publishing Co., 1997, ISBN 0-89791-896-7, s. 181–188, doi:10.1145/258734.258838.
URL http://dx.doi.org/10.1145/258734.258838

[10] Griebel, M.; Dornseifer, T.; Neunhoeffer, T.: *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998, ISBN 0-89871-398-6.

[11] Hadwiger, M.: *High-Quality Visualization and Filtering of Textures and Segmented Volume Data on Consumer Graphics Hardware*. Dizertační práce, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2004.
URL
https://www.cg.tuwien.ac.at/research/publications/2004/Hadwiger-thesis/

[12] Hadwiger, M.; Ljung, P.; Salama, C. R.; aj.: Advanced Illumination Techniques for GPU Volume Raycasting. In *ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08, New York, NY, USA: ACM, 2008, s. 1:1–1:166, doi:10.1145/1508044.1508045.
URL http://doi.acm.org/10.1145/1508044.1508045

[13] Hadwiger, M.; Sigg, C.; Scharsach, H.; aj.: Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. *Computer Graphics Forum*, ročník 24, č. 3, 2005: s. 303–312, ISSN 1467-8659, doi:10.1111/j.1467-8659.2005.00855.x.
URL http://dx.doi.org/10.1111/j.1467-8659.2005.00855.x

[14] Lorensen, W. E.; Cline, H. E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, New York, NY, USA: ACM, 1987, ISBN 0-89791-227-6, s. 163–169, doi:10.1145/37401.37422.
URL http://doi.acm.org/10.1145/37401.37422

[15] Pfaff, T.; Thuerey, N.; Cohen, J.; aj.: Scalable Fluid Simulation Using Anisotropic Turbulence Particles. *ACM Trans. Graph.*, ročník 29, č. 6, Prosinec 2010: s. 174:1–174:8, ISSN 0730-0301, doi:10.1145/1882261.1866196.
URL http://doi.acm.org/10.1145/1882261.1866196

[16] Selle, A.; Fedkiw, R.; Kim, B.; aj.: An Unconditionally Stable MacCormack Method. *J. Sci. Comput.*, ročník 35, č. 2-3, Červen 2008: s. 350–371, ISSN 0885-7474, doi:10.1007/s10915-007-9166-4.
URL http://dx.doi.org/10.1007/s10915-007-9166-4

[17] Shewchuk, J. R.: An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technická zpráva, Pittsburgh, PA, USA, 1994.

[18] Stam, J.: Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, ISBN 0-201-48560-5, s. 121–128, doi:10.1145/311535.311548.
URL http://dx.doi.org/10.1145/311535.311548

[19] Stam, J.: Real-Time Fluid Dynamics for Games. 2003.

[20] Steinhoff, J.; Underhill, D.: Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids*, ročník 6, č. 8, 1994: s. 2738–2744, doi:http://dx.doi.org/10.1063/1.868164.
URL http://dx.doi.org/http://dx.doi.org/10.1063/1.868164

[21] Yan, H.; Wang, Z.; He, J.; aj.: Real-time Fluid Simulation with Adaptive SPH. *Comput. Animat. Virtual Worlds*, ročník 20, č. 2, Červen 2009: s. 417–426, ISSN 1546-4261, doi:10.1002/cav.v20:2/3.
URL http://dx.doi.org/10.1002/cav.v20:2/3

Prílohy

Obsah priloženého pamäťového média

Priložené CD obsahuje:

- **/3rdParty** - zdrojové súbory externých knižníc
- **/data** - model Sponza vo formáte wavefront .obj a príslušné textúry
- **/deployment/Debug** - predkompilovaná debug verzia aplikácie
- **/deployment/Release** - predkompilovaná release verzia aplikácie
- **/report** - text práce
- **/report/src** - zdrojové súbory a obrázky textu práce pre LaTeX
- **/vfxEngine** - zdrojové súbory enginu
- **/vfxFluid** - zdrojové súbory knižnice volumetrických efektov
- **/vfxDemo** - zdrojové súbory demonštračnej aplikácie
- **/video** - video k aplikácii
- **/CMakeLists.txt** - Najvyššia úroveň stromu pre build systém CMake
- **/README.txt** - Informácie k spusteniu a ovládaniu aplikácie