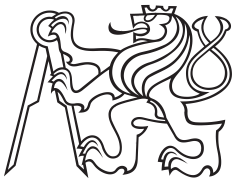
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Simulace vodopádů

Zdeněk Kolář

Vedoucí: Ing. Jaroslav Sloup
Obor: Otevřená informatika
Studijní program: Počítačová grafika
Květen 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kolář** Jméno: **Zdeněk** Osobní číslo: **474379**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačová grafika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Simulace vodopádů

Název diplomové práce anglicky:

Waterfall simulation

Pokyny pro vypracování:

Seznamte se detailně s metodou SPH (Smoothed Particle Hydrodynamics) používanou pro simulaci kapalin [5] a prostudujte literaturu [1-4] využívající tuto metodu pro simulaci vodopádů. Proveďte rešerši existujících metod simulace vodopádů v počítačové grafice. Identifikujte nejdůležitější fyzikální procesy nutné pro dosažení realistické simulace vodopádů a popište způsoby jejich implementace.
Na základě prostudované literatury navrhněte model simulace vodopádů pomocí metody SPH v prostředí tvořeném trojúhelníkovou sítí. Zaměřte se na simulační část, správnou interakci s terénem, vznik vodní tříště a její pohyb. Prostudujte stávající možnosti herního enginu Unity a navrhněte způsob, jakým simulační model do prostředí Unity integrovat. Následně v Unity vytvořte aplikaci implementující navržený model.
Pro ověření funkčnosti aplikace vytvořte alespoň tři testovací scény s různými typy vodopádů (např. kaskádovité, převislé či padající). Porovnáním výsledků simulace s chováním skutečných vodopádů vyhodnoťte, které fyzikální procesy implementovaná simulace reprodukuje dobře a se kterými má problémy (např. pohyb vody, interakce s terénem, vznik vodní tříště, atd.). Po dohodě s vedoucím práce zvolte vhodnou metodu prezentace výsledků.

Seznam doporučené literatury:

[1] Nobuhiko Mukai, Yuto Hizono, Youngha Chang: Waterfall Simulation with Spray Cloud in Different Environment. The Journal of the Society for Art and Science, vol.15, no.3, p.111-119, 2016.
[2] Nobuhiko Mukai, Yuto Hizono, Youngha Chang: Particle Based Waterfall Simulation with Spray Cloud Emerging from Basin. SIMULTECH, pp.55-61, 2018.
[3] Nobuhiko Mukai, Yuka Sunaoshi, Youngha Chang: Study on Spray Cloud Behavior Depending on Waterfall Height. Nicograph International 2019, pp.106-109, 2019.
[4] Y. Guan, W. Chen, L. Zou, L. Zhang, Q. Peng: Modeling and Rendering of Realistic Waterfall Scenes with Dynamic Texture Sprites. Computer Animation and Virtual Worlds, vol.17, no.5, p.573-583, 2006.
[5] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, M. Teschner: SPH Fluids in Computer Graphics. Eurographics 2014 State of the Art Reports.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jaroslav Sloup Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.02.2024** Termín odevzdání diplomové práce: **24.05.2024**

Platnost zadání diplomové práce: **21.09.2025**

Ing. Jaroslav Sloup podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych poděkovat panu Ing. Jaroslavu Sloupovi za užitečné rady, a hlavně za jeho trpělivost při vedení této práce. Dále bych chtěl poděkovat mým nejbližším, kteří mě při psaní této práce vždy podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu a zdroje.

V Praze, 24. května 2024

Abstrakt

Tématem této diplomové práce je simulace velkých vodopádů. Pro tyto vodopády je specifický vysoký průtok a tvorba mlhy v jejich bázi. Primárním cílem je navrhnout model simulace takových vodopádů, který využívá metod Smoothed Particle Hydrodynamics(SPH) pro simulaci dynamiky vodní masy a eulerovské simulace dynamiky tekutin Stable Fluids pro simulování mlhy, a otestovat možnosti tohoto modelu na třech scénách. Model je implementován v herním enginu Unity s využitím masivně paralelního výpočetního modelu grafických procesorů (GPU).

Klíčová slova: vodopád, simulace, dynamika tekutin, GPU, výkon, Unity

Vedoucí: Ing. Jaroslav Sloup

Abstract

The topic of this thesis is the simulation of large waterfalls. These waterfalls are characterized by a high flow and the formation of a mist at their base. The primary goal is to design a simulation model of such waterfalls that uses Smoothed Particle Hydrodynamics (SPH) and Eulerian fluid dynamics simulation Stable Fluids to simulate water mass and mist respectively, and to test the capabilities of this model on three scenes. The model is implemented in the Unity game engine utilizing the massively parallel computing model of graphics processors (GPUs).

Keywords: waterfall, simulation, fluid dynamics, GPU, performance, Unity

Title translation: Waterfall simulation

Obsah

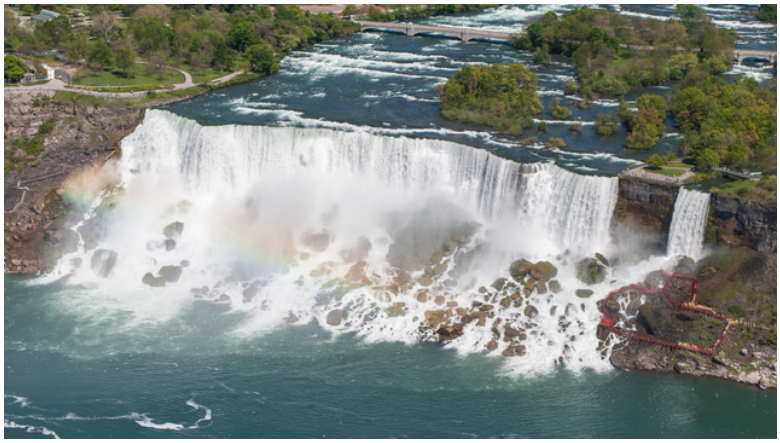
Obrázky

Tabulky

Kapitola 1

Úvod

Vodopády jsou úchvatným přírodním fenoménem, kde vodní masa nepřetržitě padá volným pádem k zemi. Při použití v počítačové grafice dokáže dodat do scény potřebnou dynamiku pro umocnění její realističnosti. Vytváření realistických a poutavých vodopádů ovšem vyžaduje hluboké pochopení fyzikálních procesů spojených s chováním vody, jako je dynamika tekutin, gravitace a interakce s terénem a prostředím, ve kterém se vodopád nachází.



Obrázek 1.1: Niagárské vodopády. Zdroj: [26]

1.1 Cíle práce

Cílem této práce je zmapovat existující modely simulace vodopádů využívající metody simulace dynamiky tekutin, konkrétně Smoothed Particle Hydrodynamics (SPH) a eulerovké simulace dynamiky tekutin ve volumetrické mřížce. Na základě této analýzy dále navrhnout vlastní model pro simulaci vodopádů. Na obrázku 1.1 je typický vodopád, který se modelem simulace snažím napodobit. Navržený model implementovat v herním enginu Unity a vyhodnotit jeho schopnosti a závislost výkonu na jednotlivých parametrech simulace nejméně na třech scénách s vodopádem.

1.2 Struktura práce

Následující text práce je členěn do 4 kapitol. Kapitola 2 se soustředí na analýzu metody SPH, simulaci dynamiky tekutin ve volumetrické mřížce a existujícími modely simulace vodopádů. V této kapitole je také navržen vlastní model simulace vodopádů. Kapitola 3 a 4 je věnována implementaci modelu simulace v herním enginu Unity a testování výsledného produktu na třech scénách. Důraz je kladen především na simulační část a její implementaci na GPU.

Kapitola 2

Analýza a Návrh

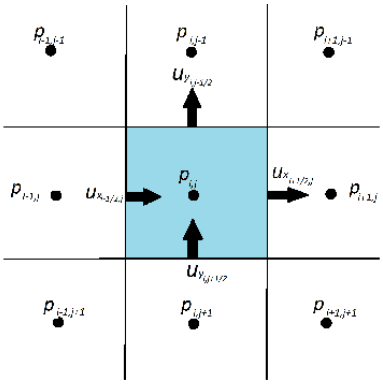
2.1 Simulace dynamiky tekutin

Pro samotné simulování dynamiky tekutin je zapotřebí matematický model, který by popisoval pohyb a fyzikální jevy simulovaného média. V počítačové grafice záleží především na vzhledu při působení deformačních sil. V matematice se tato disciplína, která se zabývá makroskopickým chováním kapalin a tuhých těles, nazývá mechanika kontinua[19]. Konkrétně, mechanika kontinua je disciplína, která se zabývá deformací a přenosem sil v materiálech, které jsou modelovány jako kontinuum, tedy jako souvisle distribuovaná hmota plně vyplňující prostor, namísto diskrétních částic, jako jsou atomy a molekuly.

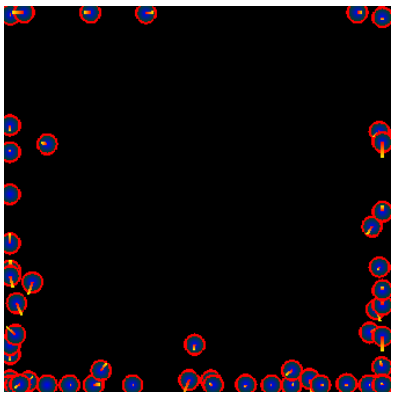
Díky tomu, že je materiál modelován jako kontinuum, je možné zavést pojem materiálové derivace $\frac{D}{Dt}$. Materiálová derivace popisuje velikost změny vlastnosti materiálu v bodě v závislosti na čase t. Matematická podoba materiálové derivace je závislá na tom, jak je kontinuum popsáno.

První způsob je sledovat pohyb a změnu vlastností materiálu z pohledu fixních bodů v prostoru. Tomuto popsání se říká popsání v eulerovských souřadnicích. Podoba rovnice materiálová derivace v těchto fixních bodech je popsána v rovnici 2.1. První výraz popisuje velikost změny vlastnosti A v čase. Jelikož je vlastnost pozorována ve fixních bodech a materiál se pohybuje, je potřeba brát v potaz i pohyb materiálu v čase. Velikost změny vlastnosti A díky pohybu je popsána druhým výrazem rovnice 2.1 $\mathbf{v} \cdot \nabla A^E$.

Druhý způsob, je sledovat vlastnosti materiálu z pohledu bodů, které předsta-



(a) : MAC buňka simulace v mřížce



(b) : Částicová simulace

Obrázek 2.1: Principy simulačních metod (mřížka a částice)[5]

vují nekonečně malý zlomek objemu materiálu. Tyto body v podstatě tvoří modelovaný materiál. Jedná se o popsání kontinua v lagrangeovských souřadnicích. Jelikož se body hýbou s materiálem, je z materiálové derivace vynechán druhý výraz zohledňující pohyb materiálu vůči fixnímu bodu (rovnice 2.2).

$$\frac{DA^E}{Dt} = \frac{\delta A^E}{\delta t} + \mathbf{v} \cdot \nabla A^E \tag{2.1}$$

$$\frac{DA^L}{Dt} = \frac{\delta A^L}{\delta t} \tag{2.2}$$

Popisu v eulerovských souřadnicích využívají metody pro simulaci dynamiky tekutin založené na simulaci v mřížce. Nad doménou simulace je vytvořena mřížka. Jednotlivé buňky tak představují část domény zafixovanou v prostoru. Vlastnosti simulovaných materiálů se pak většinou měří ve středu jednotlivých buněk, nebo jejich stěn. Na obrázku 2.1b je buňka mřížky, kde ve středu buňky je sledován tlak a na stěnách rychlosti v dané souřadnicové ose. Pro tyto metody je typická velká paměťová náročnost, jelikož je potřeba pokrýt i prostor domény, kde se materiál v daném čase nenachází. Jejich výhodou oproti metodám využívající popis v lagrangeových souřadnicích je však větší stabilita a tedy možnost simulovat po delších časových krocích. V této práci je níže zmíněná metoda s názvem Stable Fluids [30][29], která je použita pro simulaci mlhy.

Metody simulace dynamiky tekutin využívající popisu kontinua pomocí lagrangeových souřadnic jsou založené na simulaci jednotlivých částic. Materiál je modelován množinou částic, které jsou v jednotlivých krocích simulace zpracovávány (obrázek 2.1a). Částice samotné nesou hodnoty zkoumaných vlastností simulovaného materiálu. Výhodou oproti metodám využívající popis pomocí eulerovských souřadnic je lineární paměťová závislost na počtu simulovaných částic. Jelikož na sebe částice vzájemně působí, je nutné pro každou částici sledovat, jaké okolní částice ji ovlivňují. V naivním případě

jde o výpočet s kvadratickou časovou složitostí k počtu částic. Pro urychlení je potřeba udržovat akcelerační strukturu pro vyhledání nejbližších sousedů jednotlivých částic. Jednotlivým technikám se věnuje sekce 2.1.2.

Chování nestlačitelných tekutin jako je voda je popsáno Navier-stokesevými rovnicemi 2.3 a 2.4. $\mathbf{v}$ značí rychlost pohybu, ρ hustotu, p tlak, μ kinetickou viskozitu a $\mathbf{f}_{ext}$ externí sílu vstupující do systému. První rovnice 2.3 je získána dosazením $\frac{D\rho}{Dt} = 0$, což je podmínka nestlačitelnosti, do rovnice kontinuity [8]. Druhá rovnice 2.4 je odvozena z rovnice pro zachování hybnosti v kontinuu [7]. Konkrétní odvození lze nalézt například v [10].

$$\nabla \cdot \mathbf{v} = 0 \tag{2.3}$$

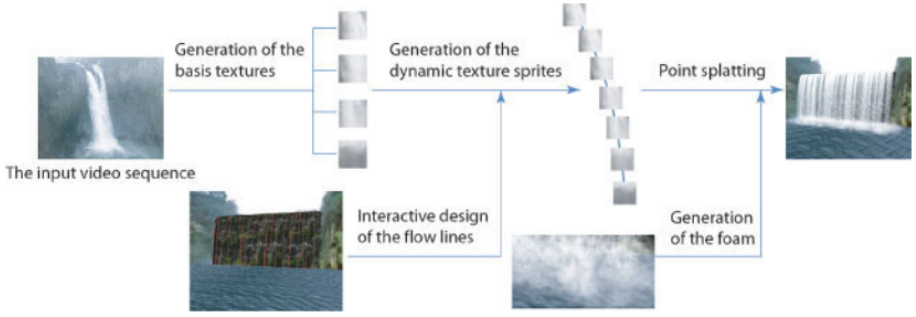
$$\rho \frac{D\mathbf{v}}{Dt} = -\Delta p + \mu \nabla^2 \mathbf{v} + \mathbf{f}_{ext} \tag{2.4}$$

Toto byl stručný úvod do teorie za simulací dynamiky tekutin, jak je popsán v [19]. Následující sekce se věnuje simulaci vodopádů.

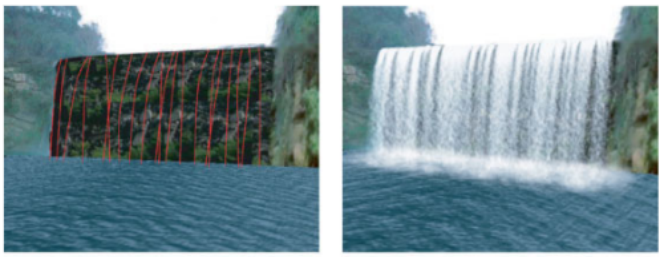
2.1.1 Simulace vodopádů

Vodopády, jak už název naznačuje, jsou ve své podstatě pouze padající voda. S dostatečnými výpočetními zdroji by pro simulaci vodopádů stačilo simulovat jednotlivé mikroskopické vody, které tvoří proud vody tekoucí přes přepad a padající volným pádem do báze vodopádu, kde se roztříští a část z nich utvoří stoupající mlhu. Bohužel výpočetní zdroje nejsou neomezené a proto je potřeba udělat kompromis mezi přesností a realističností simulace a časem potřebným k výpočtu.

Pro dosažení realističnosti pouze po vizuální stránce je možné simulovat vodopád pomocí dynamických textur. Guan Yu a spol. [12] dosáhli vysokého stupně realističnosti pro libovolně komplexní scény vodopádů. Jejich přístup spočíval ve vytvoření databáze textur padající vody vodopádu ze vstupního videa. Následně ve scéně definovat jednotlivé proudy padající vody vodopádu, které určovali jeho tvar. Podél proudů pak byla následné tažena geometrie ve tvaru čtverce, na kterou se vykreslovaly textury padající vody z databáze. Na obrázku 2.2 je zobrazen diagram popisující koncept tvorby simulace a jednotlivé proudy vytvořeného vodopádu. Tímto způsobem je možné vytvořit velmi komplexní scény vodopádů. Nevýhodou tohoto přístupu je obtížná implementace interakce s vnějšími silami. Simulace také není založená na zmíněných navier-stokesových rovnicích pro simulaci dynamiky tekutin.



(a) : Koncept simulace.



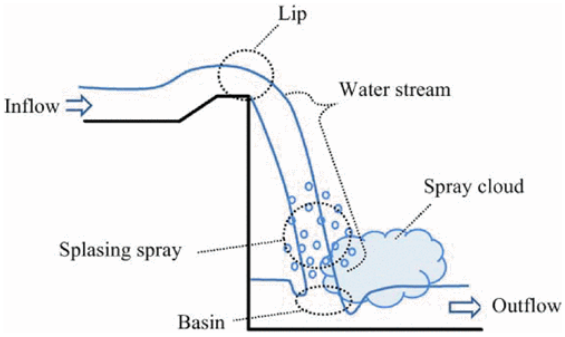
(b) : Proudy vody.

Obrázek 2.2: Simulace vodopádů pomocí dynamických textur.[12]

Modelem simulace vodopádu založeným na navier-stokesových rovnicích se zabýval Mukai a spol. [23]. Jejich model vodopádu je zobrazen na obrázku 2.3. Jejich simulace je založená na propojení částicové simulace se simulací v mřížce. Simulace má tři části:

1. Proud vody
2. Částice reprezentující rozstříknutou vodu
3. Mlha vznikající v bázi vodopádu.

Proud vody je simulován na základě navier-stokesovy rovnice 2.4 metodou Smoothed Particle Hydrodynamics(SPH) což je metoda částicové simulace blíže popsaná v sekci 2.1.2. Částice simulace jsou na základě rovnic 2.5 kategorizovány podle hustoty na částice hlavního proudu, částice povrchu a izolované částice. Izolované částice jsou odstraňovány ze simulace a nahrazovány částicemi reprezentující rozstříknutou vodu. Rychlosti jednotlivých částic jsou dále promítány do simulační domény mlhy, která je simulována metodou Stable fluids, popsanou v sekci 2.1.3, a tím pádem ovlivňují její pohyb.



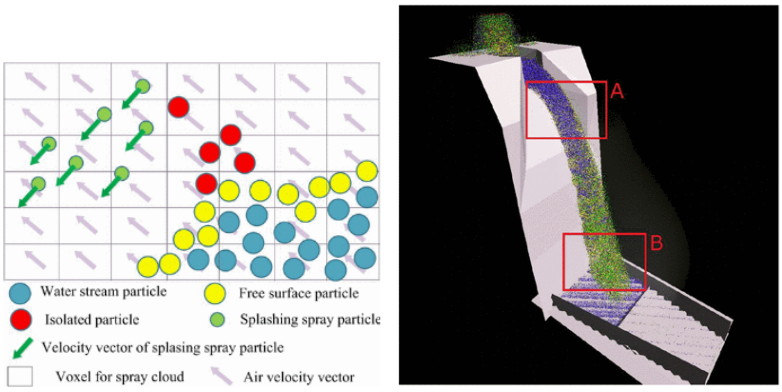
Obrázek 2.3: Model vodopádu použitý v simulaci založené na propojení částicové simulace a simulace v mřížce.[23]

$$\rho_{sur} = \alpha_{sur}\rho_{main} \tag{2.5}$$
$$\rho_{iso} = \alpha_{iso}\rho_{main} \tag{2.6}$$
$$0 < \alpha_{iso} < \alpha_{sur} < 1 \tag{2.7}$$

Částice reprezentující rozstříknutou vodu už nejsou simulovány na základě navier-stokesovy rovnice, jelikož představují jednotlivé kapky, které spolu v podstatě neinteragují. Namísto toho je jejich pohyb ovlivňován vnějšími vlivy jako je rychlost a hustota okolního vzduchu. Síly působící na částice jsou popsány v rovnicích 2.8, kde ρ, u a D značí hustotu, rychlost a průměr a dolní indexy p a a značí, že se jedná o vlastnost částice nebo okolního vzduchu. Vlastnosti okolního vzduchu jsou vzorkovány ze simulační domény poslední fáze simulace (mlhy). Částice rozstříknuté vody v čase uvolňují hustotu do simulace mlhy a po konstantním čase jsou smazány.

$$F_r + F_b = \rho_p \frac{\pi}{6} D_p^3 \frac{du_p}{dt} \tag{2.8}$$
$$F_r = \frac{1}{2}\pi(\frac{D_p}{2})^2 \rho_a C_r |u_a - u_p|(u_a - u_p) \tag{2.9}$$
$$F_b = (\rho_p - \rho_a)\frac{\pi D_p^3}{6} g \tag{2.10}$$

Mlha se skládá z velmi malých částic vody, které jsou unášeny okolním vzduchem. Jelikož by nedávalo smysl simulovat částice jednotlivě, je simulace mlhy realizována simulací v mřížce. Konkrétně metodou Stable fluids popsanou v sekci 2.1.3. Rychlost v simulační mřížce ovlivňuje pohyb částic rozstříknuté vody. Rychlost je dále ovlivňována částicemi ze simulace SPH a



(a) : Model simulace. **(b) :** Příklad simulace.

Obrázek 2.4: Model simulace a ukázka. [23]

také tzv. hustotou páry, která je pro každou buňku mřížky vypočítána podle rovnice 2.11, kde $1.293[kg/m3]$ je hustota suchého vzduchu, $273.15[degree]$ je koeficient změny teploty z Celsiovy stupnice do absolutní teploty, $T[degree]$ je teplota v $°C$, $p[hPa]$ je tlak, $1013.2[mb]$ je tlak vzduchu, 0.378 je koeficient pro změnu z gravitace do gravitace vlhkého vzduchu, a $e[hPa]$ je tlak páry. Následné zrychlení je vypočítáno podle rozdílu hustoty mezi buňkami, ležícími nad sebou podle rovnice 2.12, kde ρ_l je hustota dolní buňky, ρ_u je hustota horní buňky a g je gravitační zrychlení.

$$\rho_v = 1.293\frac{273.15}{273.15+T}\frac{p}{1013.25}(1-0.378\frac{e}{p}) \tag{2.11}$$

$$(\frac{\rho_l-\rho_u}{\rho_l})g \tag{2.12}$$

Na obrázku 2.4a je zobrazen popsaný model simulace a na obrázku 2.4b je ukázka jednoho simulovaného vodopádu. V průběhu let se Mukai dále věnoval různým parametrům jak samotné simulace, tak prostředí [24][21][22].

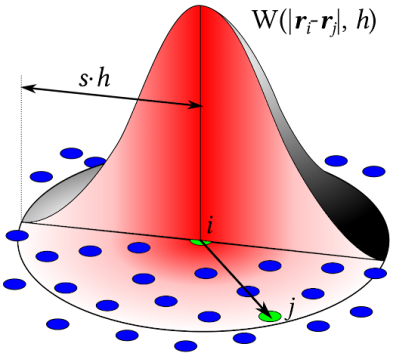
2.1.2 Smoothed Particle Hydrodynamics

SPH je metoda, která byla původně vyvinuta pro simulaci dynamiky toku plynů v astrofyzice. Jedná se o metodu diskretizace, kde každá částice reprezentuje jeden vzorek hodnoty funkce $A(\mathbf{x}_i)$. Hodnota funkce v daném bodě je následně aproximována z váženého součtu hodnot okolních vzorků 2.13 [19]. Obrázek 2.5 zobrazuje princip aproximace s použitím vyhlazovacího kernelu W, jež je popsán níže.

$$A(\mathbf{x}) \approx \int A(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)\,dV \tag{2.13}$$

$$\int A(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)\,dV = \int A(\mathbf{x}')\frac{\rho(\mathbf{x}')}{\rho(\mathbf{x}')}W(\mathbf{x}-\mathbf{x}',h)\,dV \tag{2.14}$$

$$\int \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')}W(\mathbf{x}-\mathbf{x}',h)\rho(\mathbf{x}')\,dV \approx \sum_{j\in\mathcal{S}} A(\mathbf{x_j})\frac{m_j}{\rho_j}W(\mathbf{x}-\mathbf{x_j},h) \tag{2.15}$$



Obrázek 2.5: Princip SPH

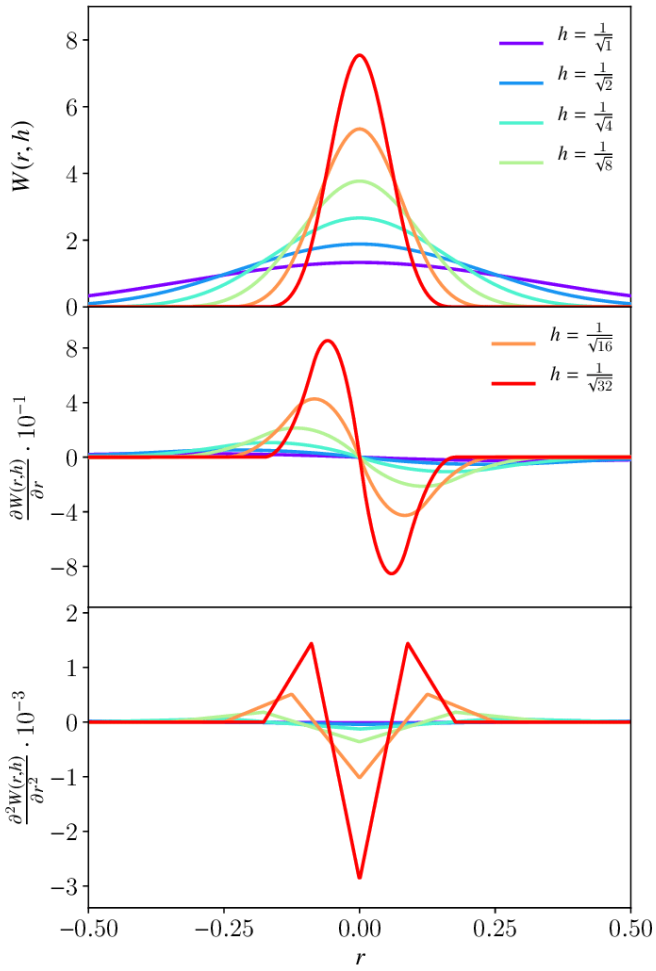
Váha každého vzorku závisí na jeho objemu m_j/ρ_j a funkci vzdálenosti od bodu $W(\mathbf{r},h)$, které se říká kernel. Kernel, nebo také vyhlazovací kernel, je speciální funkce $W : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$, která určuje, jak moc je konečná hodnota ovlivněna vzorky na základě jejich vzdálenosti. Parametr $\mathbf{x}$ je vektor vzdálenosti a h je vyhlazovací poloměr. Aby byla simulace stabilní, měl by být kernel alespoň normalizován 2.16 a symetrický 2.17.

$$\int_{R^d} W(\mathbf{r},h)\,dV = 1 \tag{2.16}$$

$$W(\mathbf{r},h) = W(-\mathbf{r},h) \tag{2.17}$$

Typický příklad vyhlazovacího kernelu je například kubický spline kernel 2.18 [19]. Na obrázku 2.6 je grafické znázornění hodnot kernelu a jeho derivací v závislosti na velikosti vektoru vzdálenosti r a vyhlazovacím poloměru h.

$$W(\mathbf{x},h) = \frac{8}{\pi h^3}\begin{cases} 6(\frac{\|\mathbf{x}\|^3}{h^3} - \frac{\|\mathbf{x}\|^2}{h^2}) + 1 & \text{if } 0 \le \frac{\|\mathbf{x}\|}{h} \le \frac{1}{2} \\ 2(1-\frac{\|\mathbf{x}\|}{h})^3 & \text{if } \frac{1}{2} < \frac{\|\mathbf{x}\|}{h} \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$



Obrázek 2.6: Kubická spline kernel a jeho první a druhá derivace.[19]

Aproximace vlastností kontinua metodou SPH probíhá ve dvou fázích. Prvně je nutné vypočítat objem každé částice (vzorku). Hustota je odvozena podle rovnice 2.19. Ve většině simulací je hmotnost pro všechny částice stejná. Výpočet přibližné hustoty se pak redukuje pouze na sumu hodnot kernelu pro dané vektory vzdálenosti. Po vypočítání hustoty už je možné vypočítat i přibližné hodnoty ostatních vlastností.

$$\rho(\mathbf{x}) \approx \sum_{j \in \mathcal{S}} \rho_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x_j}, h) = \sum_{j \in \mathcal{S}} m_j W(\mathbf{x} - \mathbf{x_j}, h) \tag{2.19}$$

Pro nestlačitelné kapaliny je nutné, aby jejich objem zůstával stejný. Po-

kud se zmenší objem, tak se zvětší tlak v kapalině a naopak. Konstantní objem tedy může být udržován výpočtem tlakového zrychlení $\mathbf{a}^p = -\frac{1}{\rho}\nabla p$. Částice jsou pak přemisťovány z míst s velkým tlakem do míst s nízkým, dokud není dosaženo rovnováhy. Jelikož je velikost tlaku úměrná odchylce v objemu, která je úměrná odchylce hustoty, je možné tlak vypočítat jako násobek odchylky aktuální hustoty od hustoty v klidovém stavu ρ_0 2.20. Konstanta k se označuje jako konstanta tuhosti. Pro nízké hodnoty konstanty k bude objem více oscilovat, než se ustálí. Vysoké hodnoty sice snižují oscilaci objemu, ale také vyžadují nižší časový krok simulace, jelikož s odchylkou hustoty roste tlakové zrychlení strměji.

$$p(\mathbf{x}) = k(\rho - \rho_0) \tag{2.20}$$

Rovnici 2.20 se říká rovnice stavu (State equation) a SPH simulátory, které ji požívají mají zkratku SESPH. Jelikož u těchto metod dochází k mírnému stlačení kapaliny, spadají také do skupiny simulátorů mírně stlačitelných kapalin. Pro dosažení kvalitnější simulace nestlačitelných kapalin, je potřeba vypočítat tlak globálně vyřešením soustavy, který se přezdívá Pressure Poisson Equation (PPE)[28][16][3].
Gradient tlaku pro výpočet konečné síly je realizován symetrickou formulí pro aproximaci gradientu, která vede ke stabilnější simulaci než prostá suma gradientem kernelu 2.13 [19].

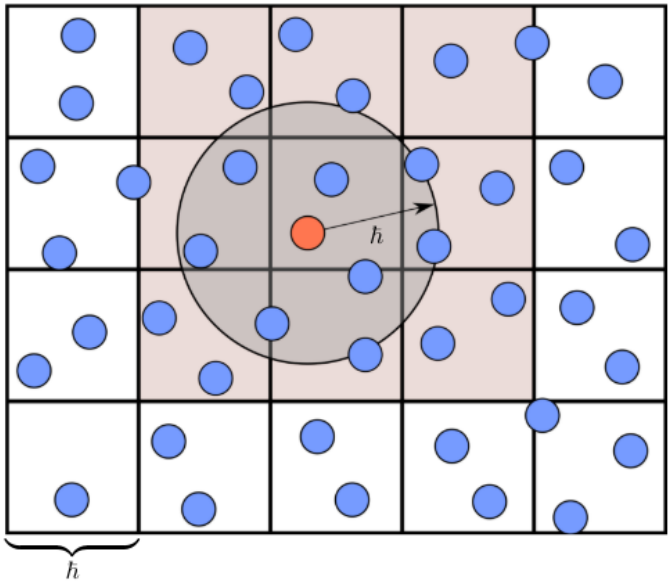
$$\nabla p_i \approx \rho_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{2.21}$$

Poslední operátor, který je potřeba diskretizovat, je Laplaceův operátor ∇^2 pro výpočet síly vlivem viskózního tření. Nejjednodušší způsob diskretizace Laplaceova operátoru je pomocí druhé derivace kernelu 2.22. Nicméně existují sofistikovanější metody, jak tento operátor diskretizovat, produkují přesnější výsledky. [19]

$$\nabla^2 A(\mathbf{x_i}) \approx \sum_j m_j \frac{A(\mathbf{x_j})}{\rho_j} \nabla^2 W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{2.22}$$

Vyhledávání sousedních částic

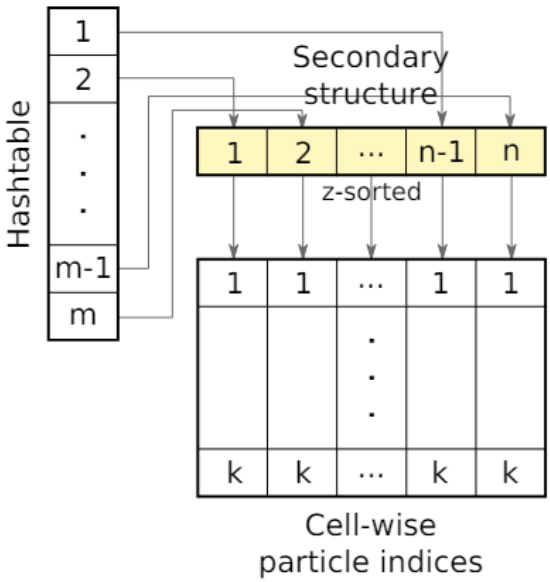
Jak již bylo zmíněno při simulaci materiálu částicemi, je nutné počítat síly vzniklé interakcí s okolními částicemi. V SPH je pro výpočet jakékoli vlastnosti materiálu v daném bodě nutné najít částice, pro které je hodnota kernelu W nenulová. To znamená, že jsou ve vzdálenosti menší než je vyhlazovací poloměr. Naivním přístupem by bylo pro každou částici procházet všechny ostatní a do výsledku započítat pouze ty ve vzdálenosti menší než je vyhlazovací vzdálenost h. To ovšem znamená kvadratickou časovou složitost. Pro urychlení výpočtu je potřeba neprocházet všechny částice, ale pouze ty, pro které je hodnota kernelu nenulová. Toho je možné dosáhnout vytvořením mřížky nad doménou simulace a pro konkrétní částici procházet pouze částice z okolních buněk. Aby tento přístup fungoval, musí být velikost buňky větší než parametr h kernelu (obrázek 2.7).



Obrázek 2.7: Mřížka nad doménou simulace.[19]

Vytvoření mřížky nad doménou ovšem vyžaduje velké množství paměti, které roste exponenciálně s počtem dimenzí. Jelikož okupovaných buněk je nanejvýš tolik jako částic v simulaci, je možné vytvořit hashovací tabulku a ukládat do ní pouze ty buňky (na základě hashovací funkce $hash()$, kde $\mathbf{x}$ je pozice buňky), ve kterých se vyskytují částice. Nevýhodou tohoto přístupu je, že s menší hashovací tabulkou roste pravděpodobnost kolizí jednotlivých buněk, což způsobí pomalejší výpočet. Také je možné v hashovací tabulce ukládat pouze indexy do sekundární struktury, která se dále odkazuje na list konkrétních částic v buňce, jak navrhuje Ihmsen a spol.[15]. Tato struktura má tu výhodu, že separuje hashovací tabulku od úložiště částic v jednotlivých buňkách. Je tak možné buňky skrze sekundární strukturu seřadit, například podle mortonova kódu, a optimalizovat tak přístupy do paměti při následném

procházení sousedních částic. Na obrázku 2.8 je zobrazen diagram celkové struktury.



Obrázek 2.8: hashovací tabulka a sekundární struktura.[19]

Nezávislost výpočtů hodnot vlastností pro jednotlivé částice vybízí k implementaci na masivně paralelních systémech jako je GPU. Vyhledávání sousedů na grafických procesorech využívá stejných principů popsaných výše. Nicméně je nutné brát větší důraz na paměťovou lokalitu vzhledem k architektuře GPU. Na GPU lze zpracování sousedů uskutečnit seřazením částic podle indexu buněk (hash, mortonův kód) paralelním radix sortem [1] a uložením počátků každé okupované buňky pro pozdější procházení sousedů. [17][14][11]

Toto byl velmi stručný úvod do širokého téma SPH metody a částicové simulace tekutin. Pro hlubší pochopení tohoto tématu je možné použít [19] nebo [17] jako úvod do jednotlivých aspektů metody a jako zdroj další literatury.

2.1.3 Stable Fluids

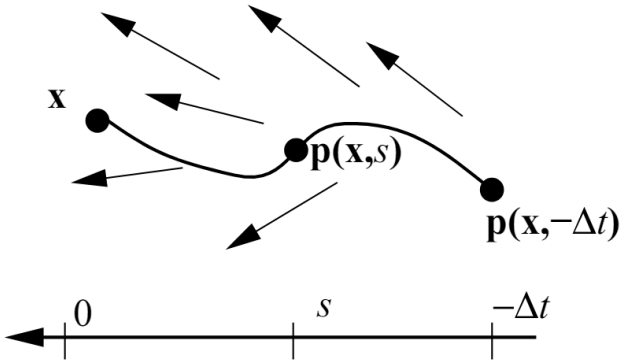
Stable Fluids [30] je metoda pro simulování dynamiky tekutin v mřížce. Hodnoty jsou ukládány ve středu buněk. Celá simulace je vykonána ve čtyřech krocích (obrázek 2.9):

$$\mathbf{w}_0(\mathbf{x}) \xrightarrow{\text{add force}} \mathbf{w}_1(\mathbf{x}) \xrightarrow{\text{advect}} \mathbf{w}_2(\mathbf{x}) \xrightarrow{\text{diffuse}} \mathbf{w}_3(\mathbf{x}) \xrightarrow{\text{project}} \mathbf{w}_4(\mathbf{x}).$$

Obrázek 2.9: Kroky Stable Fluids metody. [29]

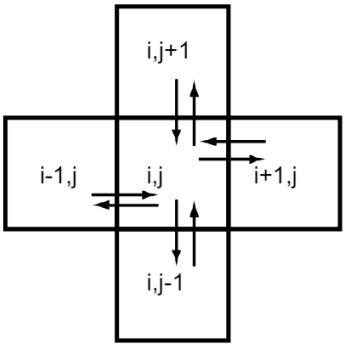
1. Přidání sil
2. Posun podél vektorového pole rychlostí
3. Difuze
4. Projekce

Předpokládá se, že divergence vstupního vektorového pole rychlosti $\mathbf{w}_0$ je ve všech bodech nulová. Posun podél vektorového pole rychlosti znamená, že pro každý střed buňky je sledována zpětná trajektorie podél vektorového pole rychlosti $\mathbf{w}_1$ po dobu časového kroku simulace Δt. Buňce je následně přiřazena hodnota na konci trajektorie. Ta je získána lineární interpolací hodnot z okolních buněk na základě vzdálenosti. Obrázek 2.10 vizualizuje tento proces. Třetím krok simuluje difuzi hodnot do okolních buněk (obrázek 2.11).

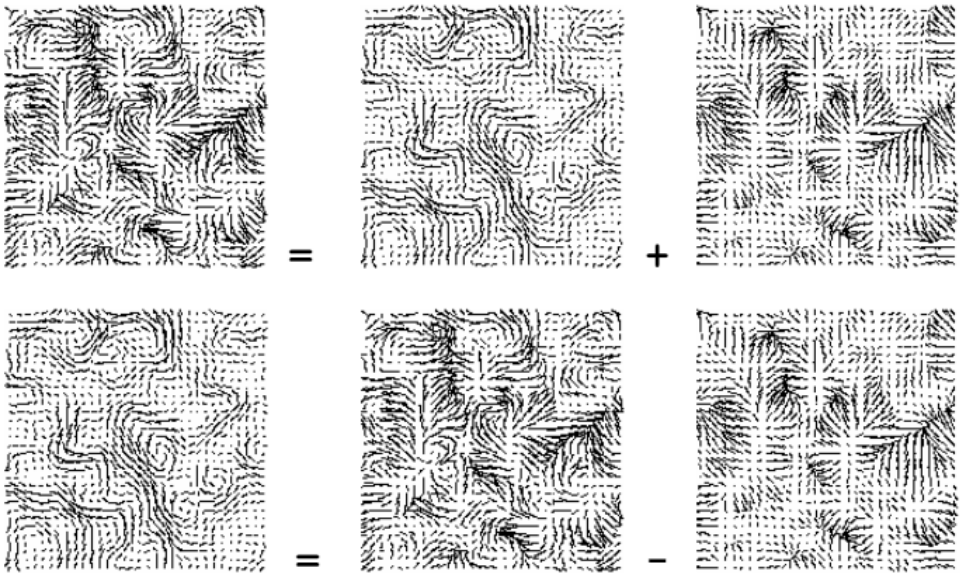


Obrázek 2.10: Posun podél vektorového pole ve Stable Fluids simulaci.[30]

Rovnice 2.23 vyjadřuje vztah hodnoty vlastnosti A v buňce dvourozměrné mřížky na souřadnicích $[x, y]$, kde parametr d vyjadřuje míru difuze. Tento výpočet však vede k nestabilitě při vysokých hodnotách parametru d. Proto je pro výpočet použit vztah 2.24, který vede na řídkou soustavu lineárních rovnic s naznámými $A(-,-)_{t+1}$, kterou lze vyřešit například Gauss-Seidelovou metodou. [29].



Obrázek 2.11: Difuze hodnot do okolních buněk.[29]



Obrázek 2.12: Vizualizace Helmholtzovi dekompozice skalárního pole.[29]

$$A(x,y)_n = A(x,y)_t + d * [A(x \pm 1, y)_t + A(x, y \pm 1)_t - 4 * A(x,y)_t] \quad (2.23)$$

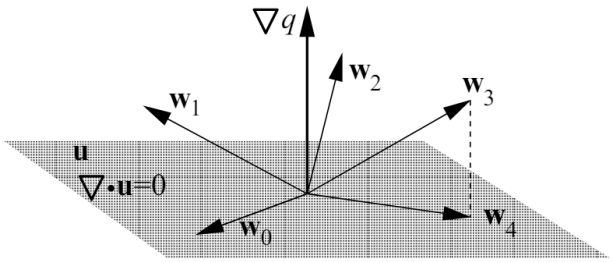
$$A(x,y)_t = A(x,y)_n - d * [A(x \pm 1, y)_n + A(x, y \pm 1)_n - 4 * A(x,y)_n] \quad (2.24)$$

$$n = t + 1 \quad (2.25)$$

Čtvrtým krokem se metoda snaží docílit nulové divergence vektorového pole po difuzním kroku. Myšlenka projekce je založena na Helmholtzově dekompozici, která tvrdí, že jakékoliv vektorové pole je možné rozdělit na vektorové pole s nulovou divergencí a gradient skalárního pole.

$$\mathbf{w}_3 = \mathbf{w}_4 + \nabla p \quad (2.26)$$

Celý vývoj vektorového pole rychlosti od $\mathbf{w}_0$ až po $\mathbf{w}_4$ je zobrazen na



Obrázek 2.13: Vývoj vektorového pole rychlostí. [30]

obrázku 2.13. Simulace může obsahovat libovolné množství vlastností, například hustotu. Tyto vlastnosti jsou aktualizovány stejně jako vektorové pole rychlosti **w** s tou výjimkou, že není vykonáván krok projekce. Pro konkrétní teoretické detaily metody je možné originální článek od Stama [30] a dále [29] pro ukázku jednoduché implementace.

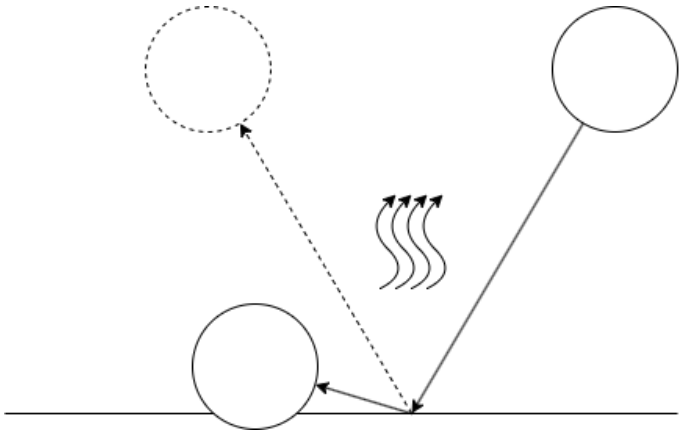
2.2 Návrh vlastního modelu simulace

V této sekci je popsán model vodopádu a návrh zpracování na grafických procesorech.

2.2.1 Model vodopádu

Model vodopádu je založen na modelu Mukai a spol. [24] popsaným v sekci 2.1.1. Vodní proud je simulován metodou SPH a mlha metodou Stable Fluids. Od původního modelu se liší tím, že neobsahuje simulaci malých částic reprezentujících tříšť. Tento krok jsem se rozhodl vynechat z toho důvodu, že v původní práci sloužil především k injekci hustoty do simulace mlhy a částice jsou tak malé, že je velmi obtížné je ve výsledném renderu spatřit, jak zmiňují v [23]. Model simulace je tedy realizován pouze SPH simulací pro vodní proud a Stable Fluid simulací pro simulaci mlhy.

Jelikož je tedy simulace malých částic tříště vynechána, je nutné určit podmínky pro injekci hustoty do simulace mlhy. Zvolil jsem vstřikovat hustotu na základě velikosti akcelerace částic SPH simulace a dále velikosti hustoty.



Obrázek 2.14: Princip vstřikování hustoty do Stable Fluid simulace a zpomalení částice.

Myšlenka je taková, že tříšť a mlha vzniká primárně v bázi vodopádu, kdy naráží na tvrdý podklad. Jelikož se SPH částice nedají dělit na menší, které by se po nárazu rozletěly do okolí, a místo toho se odrazí od povrchu, je tento efekt emulován snížením rychlosti a vstříknutím hustoty do simulace mlhy. Obrázek 2.14 zobrazuje tento proces. Další situací, kdy je hustota vstřikována do simulace mlhy je, pokud má částice menší hustotu než ρ_{iso} z rovnice 2.5.

Dalším rozdílem oproti původnímu modelu je způsob ovlivňování rychlosti ve Stable Fluids simulaci. Původní model ovlivňuje rychlosti každou částicí SPH. Já jsem zvolil ovlivňovat rychlost pouze částicemi s hustotou $\rho < \rho_{sur}$ 2.5. Myšlenka za tímto rozhodnutím je ta, že částice, které mají hustotu větší, než částice na povrchu jsou částice vodního proudu, kde není žádný vzduch. Pokud tedy vodopád padá nepřetržitě, je veškerý vzduch z prostoru vodního proudy vytlačen a okolo něj je vzduch v rovnováze.

Posledním rozdílem v simulaci je výpočet hustoty páry a zrychlení, které indukuje. V původní práci [24] je výpočet hustoty páry 2.11 založen na tlaku, vypočteném při kroku projekce ve Stable Fluids simulaci. Nebyl jsem schopen najít původní výzkum, který by tento vztah popisoval do detailu a proto jsem založil výpočet hustoty na rovnici pro hustotu ideálního plynu 2.27 a vlhkého vzduchu 2.29[9][18]. R je molární plynová konstanta specifická pro daný plyn. $R_A = 286.9$ je molární plynová konstanta suchého vzduchu a R_{AV} mixu suchého vzduchu s párou. ϕ je relativní vlhkost, která by měla být v rozsahu $0 \leq \phi \leq 1$, nicméně v modelu je tato podmínka zanedbána. Jako relativní vlhkost je použita hustota ze Stable fluid simulace. $p_V^{'}$ a p je tlak saturované páry a tlak suchého vzduchu. V simulaci je však použita pouze tlak suchého vzduchu a zlomek v rovnici 2.28 je tedy roven jedné. T je teplota ve stupních Kelvina.

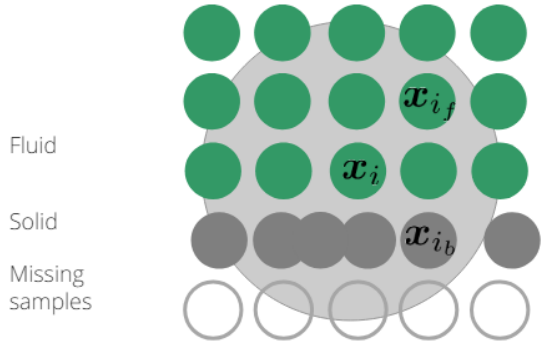
$$\rho = \frac{p}{RT} \tag{2.27}$$

$$R_{AV} = R_A(1 + 0.378\phi\frac{p_V^{'}}{p}) \tag{2.28}$$

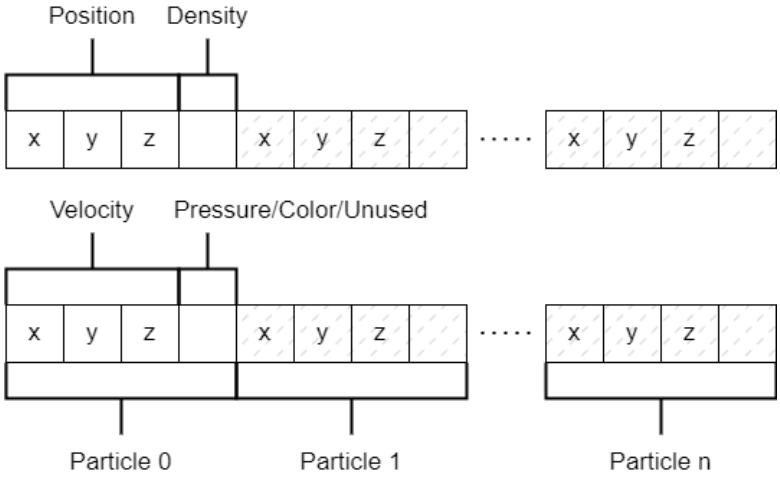
$$\rho_v = \frac{p}{R_{AV}T} \tag{2.29}$$

Kolize v SPH

Aby model vstřikování hustoty na základě velikosti zrychlení částic a samotná simulace působily realisticky, je nutné vyřešit kolize s částicemi prostředím. Kolize jsou založeny na modelu hraničních částic. [2]. Na obrázku 2.15 je zobrazená typická situace, ke které dochází při interakci s částicemi. Ideální hraniční reprezentace pomocí částic je vyplnění celého objemu kolizních předmětů. To ovšem implikuje velkou paměťovou náročnost, a proto jsou částice vzorkovány pouze na povrchu. Chybějící částice, tím pádem nepřispívají do výpočtu tlaku a tento deficit je nutné kompenzovat, jinak budou částice pod tlakem ze strany tekutiny propadávat skrze hraniční částice. V této práci je kompenzace dosaženo zvýšením frekvence vzorkování hraničních částic a také jejich hustoty při výpočtu. Tento způsob kompenzace umocňuje odrážení částic od povrchu. Nicméně existují sofistikovanější metody pro kompenzaci chybějících částic, které produkují přesnější simulaci[19][2].



Obrázek 2.15: Hraniční částice (šedě) a částice tekutiny (zeleně).[19]



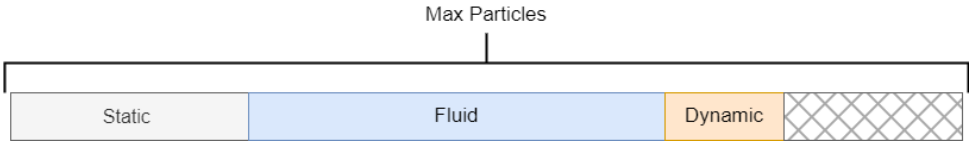
Obrázek 2.16: Data částic a jejich rozložení do dvou polí.

2.2.2 Simulace SPH na GPU

Vedlejším cílem práce bylo navrhnout simulaci na grafickém procesoru. Vzhledem k tomu, že se každá částice aktualizuje nezávisle na ostatních se paralelizace přímo nabízí. Při návrhu implementace na GPU jsem vycházel z práce Billoty a spol. [4]. Konkrétně jsem z jejich práce převzal myšlenku zpracovávat částice v buňkách akcelerační mřížky po pracovních skupinách vláken GPU (u NVIDIA karet warpy po 32 vláknech, u AMD workgroups po 64 vláknech), kterým pro zjednodušení budu nadále říkat *skupiny vláken* anglicky *waves*. Tento způsob zpracování zrychlí konstrukci seznamu sousedů až o 50%. Dále ukládání dat částic v separátních listech podle toho v jakých fázích simulace jsou využívána. Například během tvorby listu sousedů je potřeba pouze informace o pozici částic a nic víc. Na obrázku 2.16 je znázorněno jaká data částice obsahují a jak jsou rozděleny do dvou polí podle použití. Každý čtvereček je 32-bitový float. Pro hustotu nedává smysl vytvářet separátní pole, jelikož kdykoli se v SPH simulaci přistupuje k pozici, přistupuje se i k hustotě.

Simulaci SPH je možné rozdělit do dvou fází a to zpracování částic pro rychlé nalezení sousedů a samotný krok simulace s výpočtem hustoty, sil a integrace pozice. Pro nalezení sousedů jsem zvolil vytvořit list sousedů. ten si jde představit jako tabulku, kde sloupce jsou jednotlivé částice simulace a v řádcích jsou odkazy na sousední částice. Podle Koshiera a spol.[19] stačí pro dosažení dostatečné aproximace pro přesnou simulaci stačí přibližně 40 sousedních částic.

Rozložení částic ve vstupním poli je zobrazeno na obrázku 2.17. Statické částice jsou na začátku pole, dále jsou v poli uložené částice tekutiny a na konci jsou dynamické částice. Statické částice jsou na začátek pole vloženy při



Obrázek 2.17: Rozložení částic podle typu ve vstupním poli

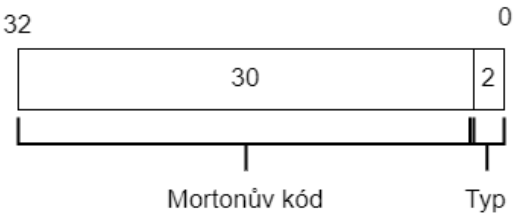
spuštění aplikace a dále už nejsou aktualizovány. Dynamické hraniční částice je nutné na začátku každého kroku aktualizovat po tom, co jsou do simulace vloženy nové částice tekutiny. Na začátku je nutné určit maximální celkový počet částic v simulaci, jinak může dojít k vynechání dynamických částic nebo dokonce i částic tekutiny ze simulace.

Vytvoření seznamu sousedů

Během vytváření seznamu sousedů uvažuji, že je nad doménou simulace postavena uniformní mřížka s buňkami minimálně o velikosti vyhlazovacího poloměru. Vytvoření seznamu sousedů probíhá v pěti krocích:

1. Přiřazení kódu částicím (mortonův kód buňky + typ částice)
2. Seřadit indexy částic podle kódu
3. Zkopírovat částice do sekundárního pole podle seřazených indexů
4. Identifikovat segmenty částic podle buňky mřížky a zaznamenat potřebná data pro jejich zpracování
5. Zpracování částic každé buňky jednou skupinou vláken.

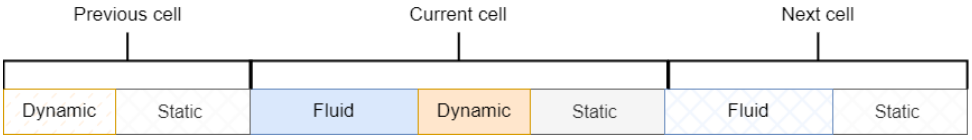
Přiřazení kódu částicím spočívá ve spočítání mortonova kódu buňky, ve kterém se částice nachází a následně ho zkombinovat s kódem pro typ částice. Typ částic je rozdělen na statické (kód `0x2`) a dynamické (kód `0x1`) hraniční částice a částice tekutiny (kód `0x0`). Stačí tedy dva bity. Zbylých 30 bitů je použito pro mortonův kód (obrázek 2.18. Se 30 bity na mortonův kód je možné simulovat v doméně o velikosti 2^{30} buněk nebo také $1024 \times 1024 \times 1024$. Poslední možná hodnota je `0xffffffff`, kterou v simulaci používám na vyřazení neplatných částic odstraněných v předchozím kroku simulace (sekce 2.2.2). Tím že je mortonův kód uložen na těch nejvíce významných bitech a typ částice na nejméně významných způsobí, že po seřazení je rozložení v paměti stejné jako na obrázku 2.19.



Obrázek 2.18: Kód částice.

Seřazení indexů částic podle kódu může být realizováno libovolným řadícím algoritmem. Konkrétní algoritmus použitý v implementaci je zmíněn v kapitole 3.

Zkopírováním do sekundárního pole jsou data částic zkopírována do sekundárního pole podle seřazených indexů. V tomto poli jsou částice rozloženy podle buněk a typů. Buňky jsou seřazeny podle tzv. z-orderu, což by mělo zaručit lepší paměťovou lokalitu při konstrukci listu sousedů a následném výpočtu sil. Obrázek 2.19 vizualizuje rozložení částic v sekundárním poli. Na obrázku 2.20 je dále vizualizován proces od přiřazení kódu až po zkopírování do sekundárního pole. Pro snadnější pochopení jsou v obrázcích uvažovány pouze částice tekutiny bez neplatných částic.

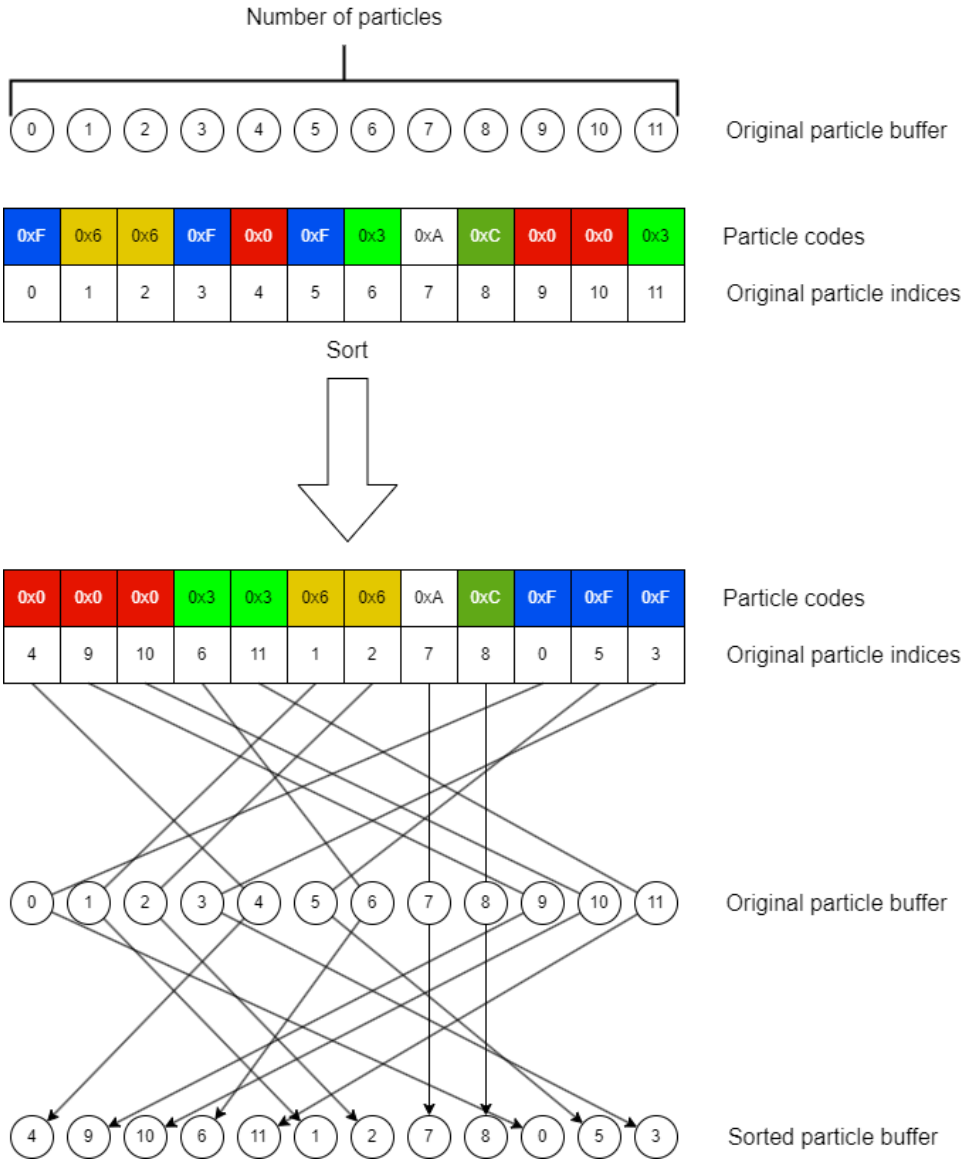


Obrázek 2.19: Rozložení částic v paměti po seřazení.

Zaznamenání počátků buněk. Po zkopírování seřazených dat, jsou zaznamenány potřebná data pro konečnou konstrukci seznamu sousedů. Pro každou buňku jsou zaznamenány následující údaje:

1. Index počáteční částice v buňce
2. Mortonův kód buňky
3. Exkluzivní prefixová suma částic tekutiny napříč celým sekundárním polem
4. Exkluzivní prefixová suma dynamických hraničních částic napříč celým sekundárním polem

Exkluzivní prefixové sumy je nutné zaznamenat kvůli zpětnému zkopírování dat do originálního pole částic a indexaci při výpočtu sil. Pro statické částice

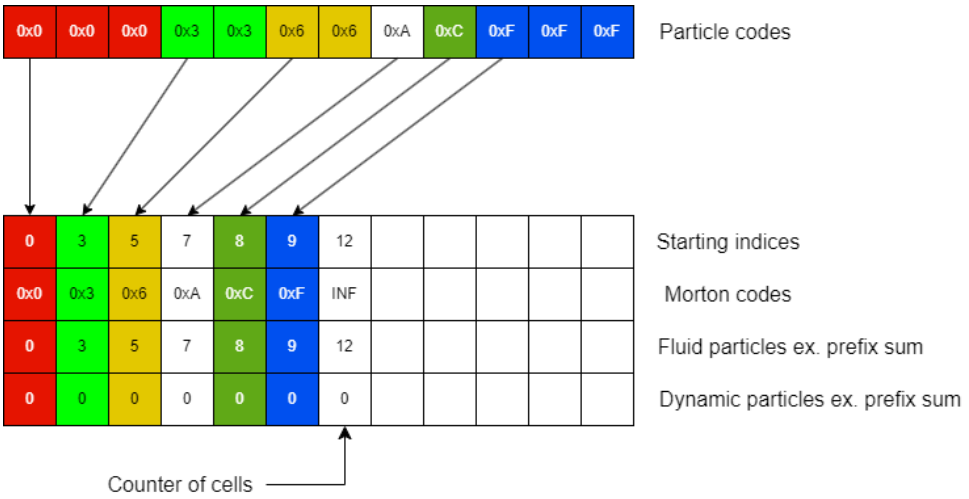


Obrázek 2.20: Přiřazení kódu, seřazení a následné zkopírování dat částic do sekundárního pole

totiž není nutné zaznamenávat sousedy, jelikož se jejich vlastnosti v simulaci nemění. Kvůli prokládanému rozložení částic v buňce ale není možné indexovat sloupce seznamu sousedů podle indexu částice. Dynamické hraniční částice dále prochází sousední částice pro určení síly vyvolané okolní kapalinou. Pro ně je ovšem nutné procházet pouze částice kapaliny za předpokladu, že kolize mezi jednotlivými objekty, které tvoří dynamické částice, jsou zpracovány v separátním enginu pro zpracování kolizí, jako je tomu například v Unity.

Přestože to pro funkčnost není nutné, jsou informace o buňkách seřazeny podle indexu počáteční částice pro zlepšení přístupů do paměti v následujícím

kroku. Stačí spočítat exkluzivní prefixový součet počátků buněk napříč seřazeným polem částic. Tato operace je v podstatě zadarmo, protože je možné prefixový součet počítat společně s počtem částic tekutin a dynamických hraničních částic. Obrázek 2.21 znázorňuje na příkladu proces uložení informací o buňkách.

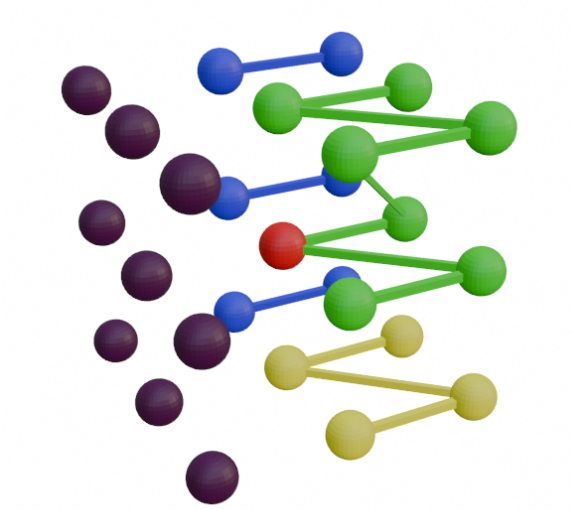


Obrázek 2.21: Ukládání dat buněk.

Hledání sousedů probíhá ve dvou fázích. Zpracovávaná buňka je určena indexem skupiny vláken $i_{wave} = \lfloor \frac{t}{c} \rfloor$, kde i_{wave} je index skupiny vláken, t je index vlákna a c je počet vláken ve skupině na konkrétní architektuře. Pro buňku je nejdříve nutné najít její sousedy. Ty jsou vyhledány binárním vyhledáváním v poli se záznamy mortonových kódů buněk. Každé vlákno ve skupině vyhledává jednoho souseda. To znamená, že počet vláken ve skupině musí být ve 3D alespoň 27 (buňka sousedí i sama se sebou). Po nalezení sousedních buněk jsou postupně porovnávány vzdálenosti jejich částic s částicemi zpracovávané buňky. Protože jsou buňky seřazeny podle mortonova kódů, je možné je zpracovávat v takovém pořadí, aby bylo zpracováno více buněk najednou. Na obrázku 2.22 jsou znázorněni sousedi zpracovávané buňky (červená) a jejich návaznost v paměti (napojené buňky) díky seřazení.

Po nalezení sousedů jsou po jednom zpracováni. Nejdříve jsou z buňky zpracovány částice tekutiny v jednom průchodu sousedů a následně dynamické částice v druhém průchodu. Důvod je ten, že výpočty sil působících na tyto částice jsou odlišné. Pro tekutinu se započítávají všechny typy částic, ale pro dynamické částice pouze částice tekutiny. Ve sloupcích seznamu sousedů jsou nejprve uloženy sousedi částic tekutin a až potom sousedi dynamických hraničních částic. Konkrétní index sloupce je určen z exkluzivní prefixové sumy pro daný typ částic, indexu vlákna ve skupině a pro dynamické hraniční částice dále celkovým počtem částic tekutiny v simulaci.

Samotné částice jsou zpracovány následovně. Nejprve jsou pro zpracovávané

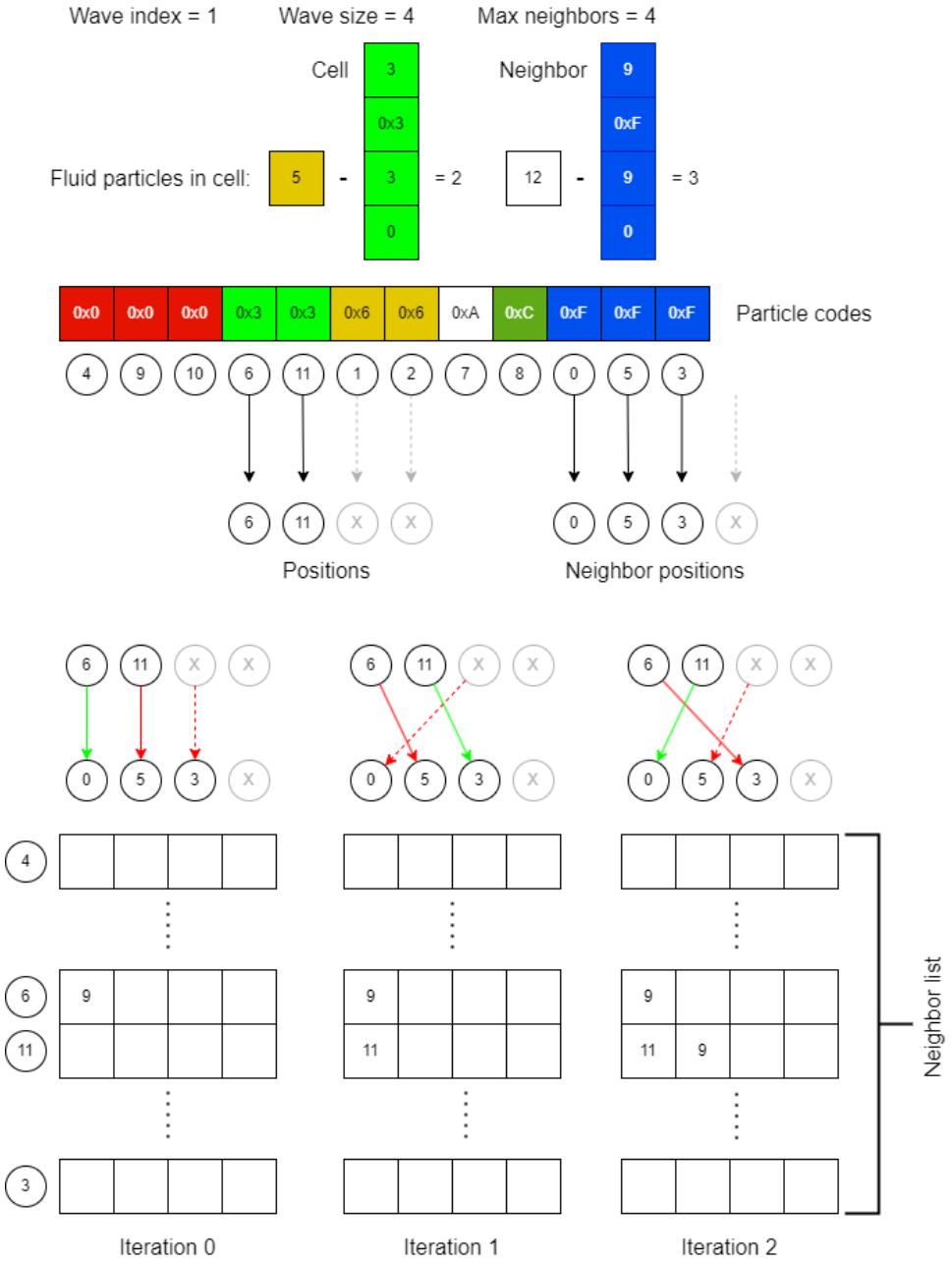


Obrázek 2.22: Sousedi červené buňky a jejich návaznost v paměti.

buňky načteny každým vláknem skupiny pozice částic do registru. Pokud je počet vláken větší než počet částic v buňce, pozice je neplatná, ale vlákno zůstává aktivní. Pokud je pozice platná, je index částice v sekundárním poli nahrán do prvního řádku seznamu sousedů pro zpětné nalezení částice. Následně je určena velikost souvislého bloku sousedních částic v paměti. Potom jsou všemi vlákny načteny pozice sousedních částic do registrů. Následuje iterace, kdy každé vlákno načítá hodnotu registru s pozicí sousední částice a porovnává ji s pozicí aktuální částice zpracovávané buňky. Tento proces je znázorněn na obrázku 2.23. Pro jednoduchost byl vynechán krok odvození souvislého bloku paměti a je zpracována pouze jedna buňka. Sousedních částic je celkem 3, takže je potřeba iterovat třikrát. Seznam sousedů je také transponovaný, takže sousední částice jsou uloženy v řádcích. Konkretní rozložení seznamu sousedů v paměti je zobrazeno na obrázku 2.24 a je popsáno v sekci 2.2.2. Jak již bylo naznačeno, pro částice tekutin je iterováno přes všechny sousední částice a pro dynamické hraniční pouze přes sousední částice tekutiny. Po zaznamenání všech sousedů je na poslední pozici indexu následujícího souseda uložen příznak konce sousedů, pokud není list plně zaplněn.

Výpočet hustoty, sil a integrace

Jakmile je vytvořen seznam sousedů je možné vypočítat síly působící na jednotlivé částice v simulaci. Jak již bylo zmíněno v sekci 2.1.2, nejdříve je nutné vypočítat hustotu. Ta je vypočítána rovnicí 2.19. V simulaci jsou použity stejné kernely jako v referenčním modelu [23][21][24], které jsou převzaty z práce Mullera a spol. [25]. Konkrétně se jedná o kernely W_{poly} 2.30



Obrázek 2.23: Konstrukce seznamu sousedů.

pro výpočet hustoty, W_{spiky} 2.31použitý při výpočtu tlaku a $W_{viscosity}$ 2.32 použitým pro výpočet síly způsobené viskozitou. Tlak je vypočítán symetrickou rovnicí pro aproximaci gradientu tlaku 2.21, kde tlak p je vypočítán stavovou rovnicí 2.20. Výpočet síly způsobené viskozitou 2.33 je opět převzat z práce Mullera a spol.[25] Jedná se o variaci na výpočet tzv. umělé viskozity. SPH simulace, které se tento výpočet používají pro určení síly, se nazývají XSPH [19]. Přičítáním váženého rozdílu rychlostí okolních částic dochází k

"vyhlazování"pole rychlostí.

$$W_{poly6}(\mathbf{r},h) = \frac{315}{64\pi h^9}\begin{cases}(h^2-\|\mathbf{r}\|^2)^3 & \text{if } 0 \le \|r\| \le h \\ 0 & otherwise\end{cases} \tag{2.30}$$

$$W_{spiky}(\mathbf{r},h) = \frac{15}{\pi h^6}\begin{cases}(h-\|\mathbf{r}\|)^3 & \text{if } 0 \le \|r\| \le h \\ 0 & otherwise\end{cases} \tag{2.31}$$

$$W_{viscosity}(\mathbf{r},h) = \frac{15}{2\pi h^3}\begin{cases}-\frac{\|r\|^3}{2h^3}+\frac{\|r\|^2}{h^2}+\frac{h}{2\|r\|}-1 & \text{if } 0 \le \|r\| \le h \\ 0 & otherwise\end{cases} \tag{2.32}$$

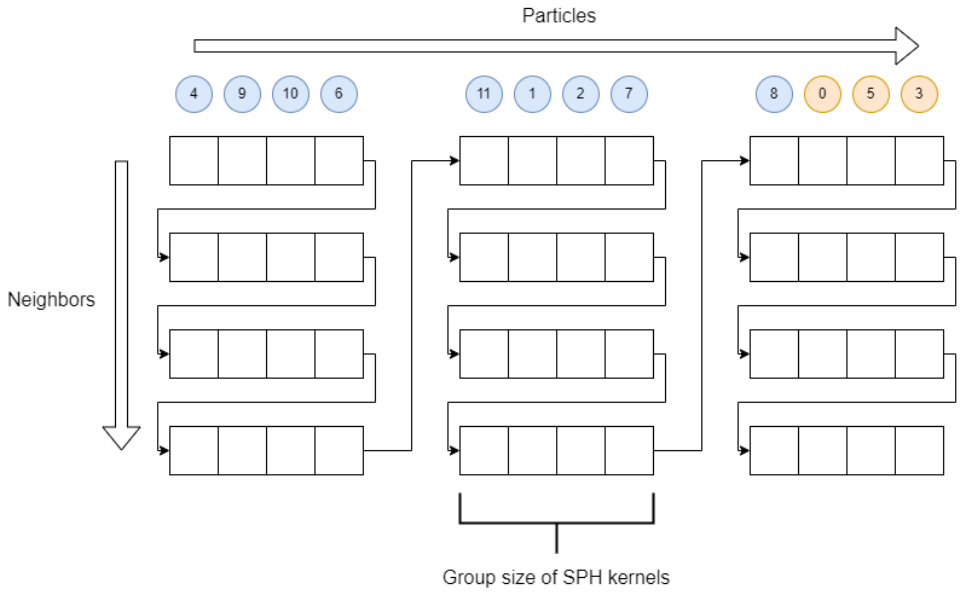
$$\mathbf{f}_i^{viscosity} = \mu\sum_j m_j \frac{\mathbf{v}_j-\mathbf{v}_i}{\rho_j}\nabla^2 W_{viscosity}(\mathbf{x}_i-\mathbf{x}_j,h) \tag{2.33}$$

Samotné zpracování probíhá následovně:

1. Načíst index zpracovávané částice z prvního řádku seznamu sousedů
2. Iterovat přes řádky, dokud není dosažen příznak konce seznamu, nebo poslední řádek. v každém kroku:
   a. Načíst index sousední částice
   b. Načíst částici ze sekundárního pole
   c. Přičíst kontribuci částice do finální hodnoty

Aby byly indexy sousedních částic co nejblíže v paměti, je seznam sousedů rozdělen na bloky podle maximálního počtu sousedů a velikostí bloků(Groups) vláken vyvolávaných na GPU. Seznam sousedů pro jeden blok tvoří souvislý úsek paměti seřazený po řádcích, tyto úseky na sebe vzájemně navazují. Na obrázku 2.24 je vizualizováno popsané rozložení. Dále je naznačeno, jak jsou částice tekutin před částicemi dynamických hranic (přestože jsou po celou dobu v obrázcích uvažovány pouze částice tekutiny)

Síla působící na dynamické hraniční částice je vypočítána z tlaku v částic tekutiny (tzv. pressure mirroring [19]). Po vypočtení síly je nutné dále vypočíst sumu sil vzhledem k hmotnému bodu těla, kterému částice náleží a také kroutivý moment způsobený tím, že síly nepůsobí přímo směrem k hmotnému bodu. Po sečtení je nutné síly aplikovat v simulaci pro fyziku tuhých těles, kde je ošetřována integrace pozice a rychlosti a kolize mezi těmito tělesy.



Obrázek 2.24: Rozložení seznamu sousedů v paměti.

Integrace rychlosti a pozice je řešena tzv. semi-implicitní eulerovskou metodou, která je jednou z nejvíce používaných metod numerické integrace v SPH simulacích [19]. Její matematická podoba je znázorněna v rovnici 2.34.

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \mathbf{a}_n \tag{2.34}$$

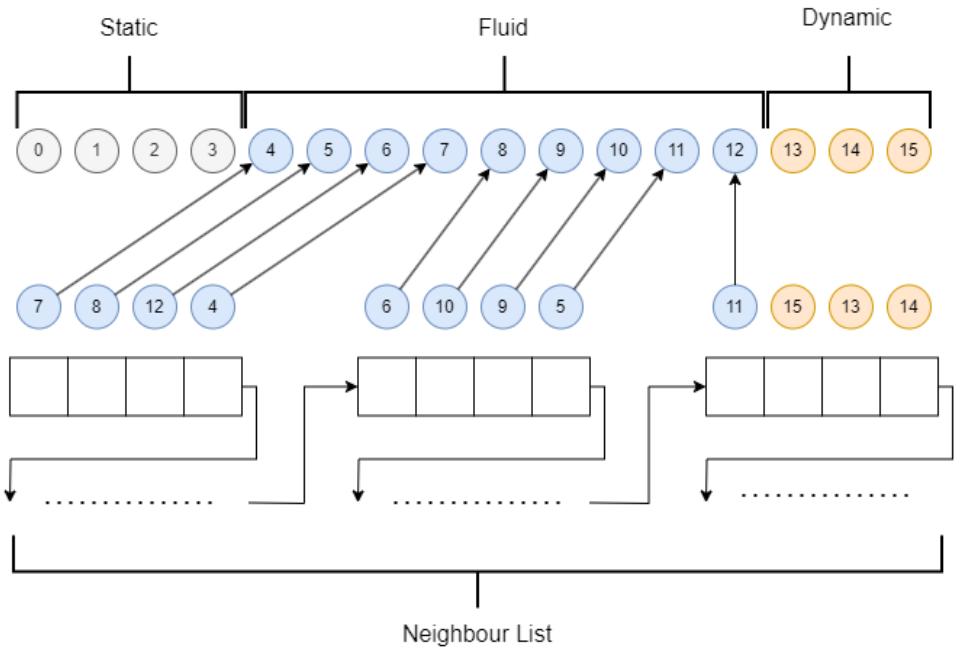
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \mathbf{v}_{n+1} \tag{2.35}$$

Invalidní částice. Jelikož se simuluje tekoucí voda, je nutné částice odebírat ze simulace, aby bylo možné vkládat nové. Částice jsou odebírány na základě výškové souřadnice. Pokud jejich poloha klesne pod danou výšku, jsou označeny jako neplatné a v následujícím kroku jsou vyfiltrovány seřazením mimo pole aktivních částic, jak je popsáno v sekci 2.2.2 v odstavci o přiřazování kódu částicím. Invalidní částice je označena nastavením její hustoty na zápornou hodnotu.

Post processing

Po SPH kroku je nutné částice zkopírovat zpět částice tekutin do originálního pole. Statické částice není nutné kopírovat, protože nejsou nijak aktualizovány a dynamické nejsou aktualizovány v SPH simulaci, ale jsou aktualizovány na základě posunu tuhého tělesa, kterému náleží. Zkopírování je uskutečněno extrakcí indexu částice z prvního řádku seznamu sousedů a následným zkopírováním dat částice ze sekundárního pole do původního pole na pozici, která

se rovná sloupci v seznamu sousedů. Obrázek 2.25 zobrazuje tento proces pro snadnější pochopení.



Obrázek 2.25: Zkopírování částic tekutin do původního pole

2.2.3 Stable Fluids na GPU

Implementaci Stable Fluid simulace na grafických procesorech je v podstatě stejná jako na procesoru[29]. Jediné co je potřeba udělat, je správně namapovat struktury použité v implementaci na procesoru na struktury používané v programování na GPU.

- Pole ⇒ Textury
- Vnější `for` cykly ⇒ zavolání shaderu.

Samotný výpočet je potom stejný jako při implementaci na procesoru. Pro simulaci mlhy vodopádu je jsou nutné pouze čtyři údaje, rychlost v jednotlivých osách prostory (x, y, z) a hustotu, které jsou uloženy v jedné 3D textuře s texely obsahujícími čtyři desetinná čísla. Celkem jsou alokovány tři textury. Dvě pro výpočet kroku simulace, jelikož není možné přepisovat údaje v jedné, protože během jednotlivých kroků simulace je přistupováno k hodnotám

sousedních buněk. Třetí textura slouží k přenosu přidané hustoty a rychlosti z SPH simulace do Stable Fluids simulace. Pro krok projekce jsou dále alokované tři další textury. Jedna pro uložení divergence v první části výpočtu projekce a další dvě pro ukládání vypočtených hodnot tlaku[29]. Všechny tři textury nesou v texelech informaci o jednom desetinném čísle a mají stejné rozlišení jako hlavní textury s daty o rychlosti a hustotě. Synchronizace je implicitně zaručena voláním shaderu, nad stejnými zdroji. Po volání shaderu je dále nutné prohodit odkazy vstupní a výstupní textury, jelikož výstupní textura z jednoho shaderu je vstupní následujícího.

Kapitola 3

Implementace

V této kapitole je popsána implementace navrženého modelu v herním enginu Unity ve verzi 2022.1.16f1.

3.1 Herní engine Unity

Herní engine Unity je jeden z nejpopulárnějších herních enginů. Svým uživatelům nabízí komplexní vývojové prostředí umožňující vývoj her většiny existujících žánrů. Vývojové prostředí unity se skládá z mnoha komponent. Pro účely této práce jsou však využity pouze tři. Konkrétně:

- Skriptovací API
- Programování ComputeShaderů
- Vykreslovací engine

Skryptovací API

Pomocí skriptovacího API nabízeným Unity může uživatel vytvářet skripty v jazyce C#. Skripty lze komunikovat s interními systémy Unity a vytvářet tak

doplňující funkcionality editoru nebo přímo herní mechaniky. V této práci jsou skripty použity k synchronizaci volání jednotlivých shaderů a předávání dat mezi simulacemi SPH a Stable fluid. Tyto skript dědí od třídy `MonoBehavior`, která umožní Unity editoru skript identifikovat. Další důležitou třídou je třída `GameObject`. Instance třídy `GameObject` je možné umisťovat do scény, kde se hra má odehrávat. K těmto objektům je možné připojovat komponenty ovlivňující jeho chování. Těmito komponentami jsou právě třídy dědící od třídy `MonoBehavior`. Pokud je objektu komponenta přiřazena již v editoru, je pro ni vygenerováno uživatelské rozhraní a vývojář je tak schopen editovat proměnné komponenty přímo v editoru. Instance třídy `MonoBehavior` jsou dále součástí herní smyčky Unity enginu, takže je možné jejich stav pravidelně aktualizovat, například ve funkci `Update()`. Díky třídě `MonoBehavior` je také možné skrze volání funkce `OnGizmosDraw()` vykreslovat pomocné objekty do prostoru scény, pro snadnější manipulaci. Toho je v práci využito pro vizualizaci objemu simulací. Více Unity lze nalézt v dokumentaci [32].

Programování ComputeShaderů

Součástí Unity je i backend pro multiplatformní programování shader programů. Shader programy jsou v Unity programovány s jazyce HLSL. Unity je následně při sestavení aplikace namapuje na jazyk cílové platformy například GLSL, Vulkan, Metal apod. Unity dále poskytuje široké spektrum vestavěné funkcionality v podobě `.hlsl` souborů pro snadnější ošetření rozdílných paradigmat mezi platformami. V této práci využívám především rozhraní pro programování ComputeShader. Ty jsou V podstatě identické s ComputeShadery platformy DirectX.

Vykreslovací engine

Unity nabízí svým uživatelům širokou škálu funkcionalit v oblasti vykreslování. Pro účely této práce je však využita pouze ta nejzákladnější. Pro vykreslování byl použit balíček Universal Render Pipeline (URP) [33] a jeho komponenta ShaderGraph pro vykreslování SPH částic.

Použité třídy

v této sekci jsou popsány vestavěné třídy Unity enginu, které jsem při implementaci použil. Třída `MonoBehavior` byla již popsána výše.

RenderTexture. Třída `RenderTexture` představuje textury, do níž je možné zapisovat data na GPU. Obsahuje dva buffery a to `ColorBuffer` a `DepthBuffer`. Podle formátu textury je určen buffer, z něhož je čteno. Zapisováno může být do obou bufferů. V práci jsou tyto třídy použity ve Stable Fluid simulaci pro ukládání dat jednotlivých buněk mřížky. Více informací k třídě `RenderTexture` lze nalézt v dokumentaci [31, stránka "RenderTexture"].

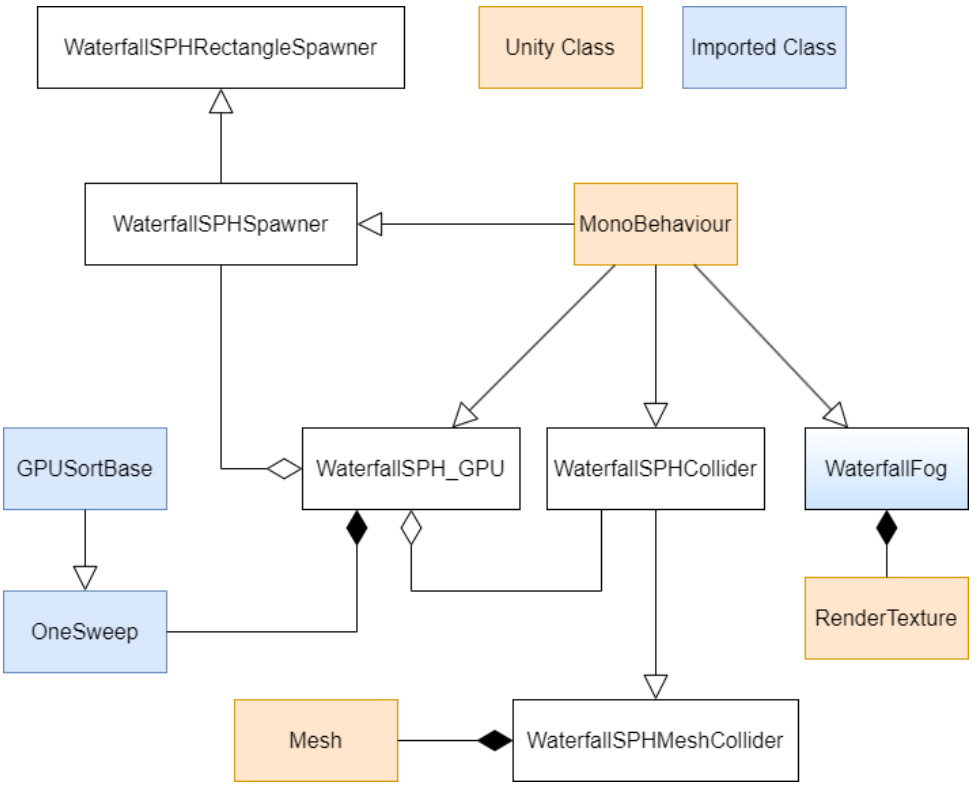
Mesh. Třída `Mesh` je používána k reprezentaci polygonových mřížek. Kromě tohoto použití jsou v práci pouty také jako nosiče informace o pozicích hraničních částic v poli `vertices`. Kompletní rozhraní třídy `Mesh` je k nahlédnutí v dokumentaci [31, stránka "Mesh"].

ComputeBuffer. Třída `ComputeBuffer` slouží jako wraper pro buffery alokované na v paměti grafické karty. Skrze tuto třídu je možné posílat a zároveň vracet data z procesoru na GPU a obráceně. tyto buffery mohou být několika typů. v práci jsou konkrétně použity buffery pro ukládání parametrů nepřímého vyvolání shader programu a dále pro ukládání strukturovaných dat, což je základní typ. Kompletní rozhraní třídy `ComputeBuffer` je k nahlédnutí v dokumentaci [31, stránka "ComputeBuffer"].

3.2 Implementace

finální implementace se skládá celkem z osmi tříd. Na obrázku 3.1 je objektový návrh implementace. modré obdelní4ky značí importovaný kód. Konkretně jde o soubory ve složce `Assets/Sctipts/Onesweep` a částečně o kód třídy `WatefallFog`. Následuje popis jednotlivých tříd.

WaterfallSPH_GPU. Ve třídě `WaterfallSPH_GPU` se odehrává veškerá logika a synchronizace SPH simulace. Implementace je rozdělena do dvou částí.



Obrázek 3.1: Objektový návrh implementace.

První, v souboru `WaterfallSPH_GPU.cs`, obsahuje definice parametrů a veškerou funcionalitu simulace. Druhá, v souboru `WaterfallSPH_GPU_ComputeShaders.cs`, obsahuje boilerplate kód pro získání jednotlivých odkazů uniformních proměnných na grafické kartě. Dále nastavování těchto proměnných pro jednotlivé kernely a také kód pro samotné vyvolávání kernelů a alokaci dat potřebných k simulaci.

WaterfallFog. Ve třídě `WaterfallFog` se odehrává naopak odehrává simulace Stable Fluids. V Objektovém návrhu má modrý gradient z toho důvodu, protože je založena na implementaci GPUStableFluids, kterou je možné nalézt na online [6]. Ve kód je upraven tak, aby bylo k simulaci potřeba pouze dvou 3D `RenderTexture` objektů a dále aby bylo možné vstřikovat hustotu do simulace skrze SPH. Nakonec jsou oproti původní implementaci optimalizovány jednotlivé fáze výpočtu. Konkrétně tak, aby se na některých místech v kódu nemusely kopírovat textury s daty o rychlosti, jak je tomu v originální implementaci.

OneSweep. Třída `OneSweep` je implementace state-of-the-art radix sortu OneSweep[1] na GPU. Původní implementaci pro unity lze opět nalézt online

jako stáhnutelný balíček[27]. Algoritmus je naprogramovaný správně, nicméně jednotlivé ComputeShadery nelze po stažení zkompilovat. V práci se tedy nachází pouze extrahovaný kód z balíčku upravený o potřebné prvky, aby již kompilace byla možná.

Ostatní. Zbylé čtyři třídy slouží jako kontainery pro data potřebná k simulaci SPH. `WaterfallSPHCollider` poskytuje rozhraní pro získání dat o pozicích částic v objektových souřadnicích. `WaterfallSPHSpawner` poskytuje zase rozhraní pro získání částic vkládaných do simulace.

3.2.1 ComputeShadery

Jelikož simulace běží na GPU, obsahuje implementace několik ComputeShaderů s definovanými kernely pro výpočet simulace. Konkrétně jde o tyto ComputeShadery a kernely:

1. SPH Shadery
   a. **ParticlePrepNeibFind**, kde jsou částice označeny kódem.
   b. **ReorderSortedParticle**, kde jsou částice zkopírovány do sekundárního bufferu.
   c. **RecordCellstarts**, kde jsou zaznamenány data jednotlivých buněk. Pro výpočet prefixového součtu je použit stejný princip jako ve radix sortu OneSweep[1].
   d. **FindNeighbors**, kde je vytvořen seznam sousedů. Tento shader společně s **RecordCellStarts** používá instrukce pro skupiny vláken, což umožňuje zpracování buněk po skupinách vláken.
   e. **ComputeDensity**, kde je vypočítána hustota částic.
   f. **ComputeForces**, kde je vypočítána síla působící na částice, integrována rychlost a zároveň vstřikována hustota do Stable fluids simulace.
   g. **SPHColorTest**, což je shader pro obarvení částic pro debugování simulace.
   h. **CopyParticlesBack**, kde jsou částice zkopírovány zpět do originálního pole.
   i. **Advect**, kde je integrována pozice částic a zároveň jsou částice označeny jako neplatné pokus jsou pod hraniční výškou
   j. **VolumeCollision**, což je poslední shader volaný v kroku simulace, který zajistí, že částice zůstanou v objemu simulace.

2. StableFluids shadery
    a. **AddValueFromTexture**, kde je vstřikována hustota a rychlost z SPH simulace do Stablefluids simulace.
    b. **Advection**, kde jsou vlastnosti přenášeny podél pole rychlosti.
    c. **Diffusion**, kde je implentován jedna iterace Gauss-Seidelovy metody pro vyřešení lineárního systému rovnic simulující difuzi.
    d. **ProjectionPt1,2 a 3**, kde je realizován krok projekce pole rychlostí a výpočet tlaku, jehož gradient je ve třetí části odečten od pole rychlosti v simulaci.
    e. **Shadery pro nastavení hraničních buněk a inicializaci dat.**

    Tyto shadery jsou volány v pořadí, v jakém jsou zde vypsány.

Některé shadery využívají funkce dostupné pouze na zařízeních podporujících platformu Directx12. Je tedy možné, že na starších zařízeních nebude implementace zcela funkční

Kapitola 4

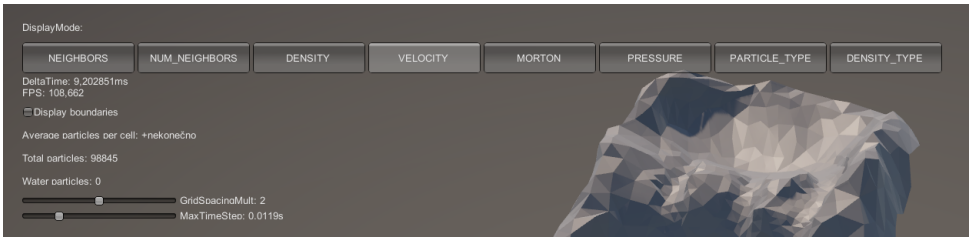
Výsledky

V této kapitole je představena demonstrační aplikace využívající implementovaný model pro simulaci vodopádu. Jsou zde představeny jednotlivé scény a jejich parametry. Pro každou scénu jsou dále změřeny časy vykonávání jednotlivých kernelů a na závěr jsou výsledky shrnuty a zmíněny slabá místa modelu. Jelikož grafické API Unity neumožňuje přístup k měřícím nástrojům GPU, byli časy jednotlivých kernelů změřeny v programu PIX verze 2312.08 funkcí GPU Capture. Pro každou scénu bylo pořízeno celkem 10 snímků časů kernelů jednoho snímku aplikace a byl vybrán ten, co se nejvíce přibližoval průměrné hodnotě času jednoho snímku. Měření jsem provedl na stolním počítači s procesorem AMD Ryzen 5 1600 Six-Core Processor 3.20GHz, grafickou kartou NVIDIA GeForce GTX 1070 a 32GB RAM paměti. Měření proběhlo ve výchozí pozici kamery po zapnutí aplikace a simulace(P), jakmile se počet částic tekutiny stabilizoval.

4.1 Demonstrační aplikace

Demonstrační aplikace má následující ovládání:

- Pohyb v prostoru ⇒ W/A/S/D/space/Y
- Otáčení kamery ⇒ držení prostředního tlačítka myši a táhnout
- Zastavení celé SPH simulace ⇒ P



Obrázek 4.1: Uživatelské rozhraní aplikace.

- Zastavení pouze kroku SPH ⇒ O
- Krok SPH v módu pozastavení pouze SPH kroku ⇒ I
- Resetování celé simulace ⇒ R
- Vypnutí aplikace ⇒ Alt+F4
- Přepínání mezi mody zobrazení ⇒ alpha1 až alpha8

V uživatelském rozhraní (obrázek 4.1) je možné nastavovat následující parametry:

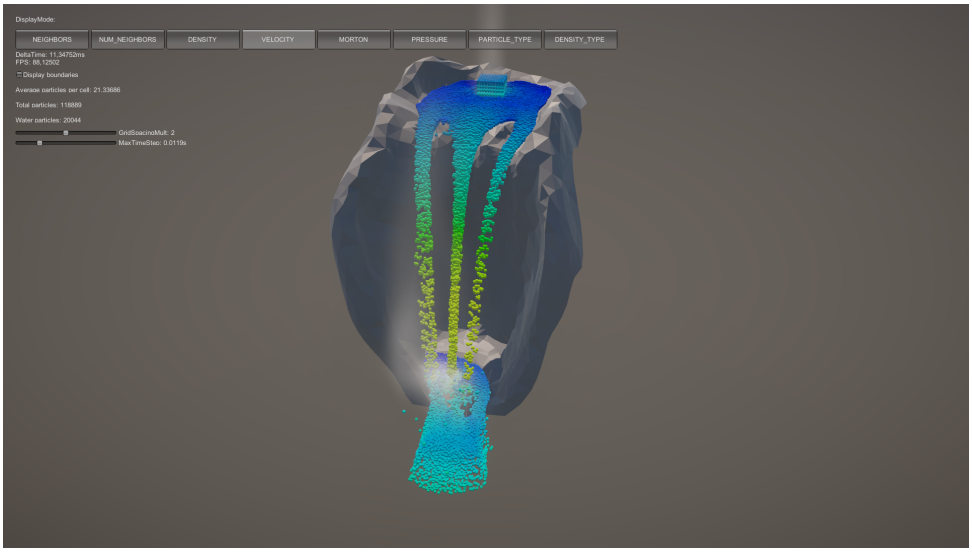
- Mód zobrazení
- Vykreslení hraničních částic
- Velikost mřížky, jako násobek vyhlazovacího poloměru
- Maximální časový krok simulace

Celkem jsem sestavil tři aplikace. Pro každou scénu jednou. Nicméně jednotlivé aplikace se kromě modelů vodopádu liší pouze v nastavení rychlosti vkládání nových částic do simulace SPH a hustoty do Stable Fluid simulace. Co se týče parametrů samotných simulací, jsou pro všechny tři až na nepatrné výjimky stejné.

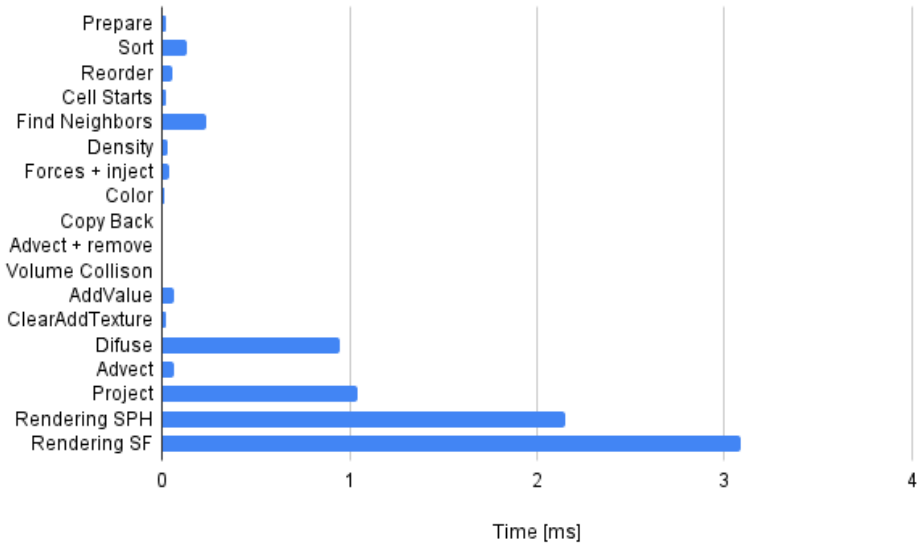
4.1.1 Horsetail vodopád

Vodopád typu horsetail je specifický úzkým ústím a vysokým spádem, na obrázku 4.2 je vyobrazen model vodopádu v aplikaci. Vodopád má spád zhruba 40 metrů. Konkretní parametry simulace SPH jsou zobrazeny na

obrázku 4.4. Dále je na obrázku 4.3 vykreslen graf časů jednotlivých kernelů simulace. Konkretní časy v milisekundách a počty částic jsou vypsány v tabulce 4.1. SPH simulace probíhá od kernelu *Prepare* až po kernel *Volume Collision*. Kernely *AddValue*, *ClearAddTexture*, *Diffuse*, *Advect* a *Project* pokrývají StableFluid simulaci s tím, že časy difuzního a projekčního kroku jsou sečteny dohromady.



Obrázek 4.2: Horsetail vodopád

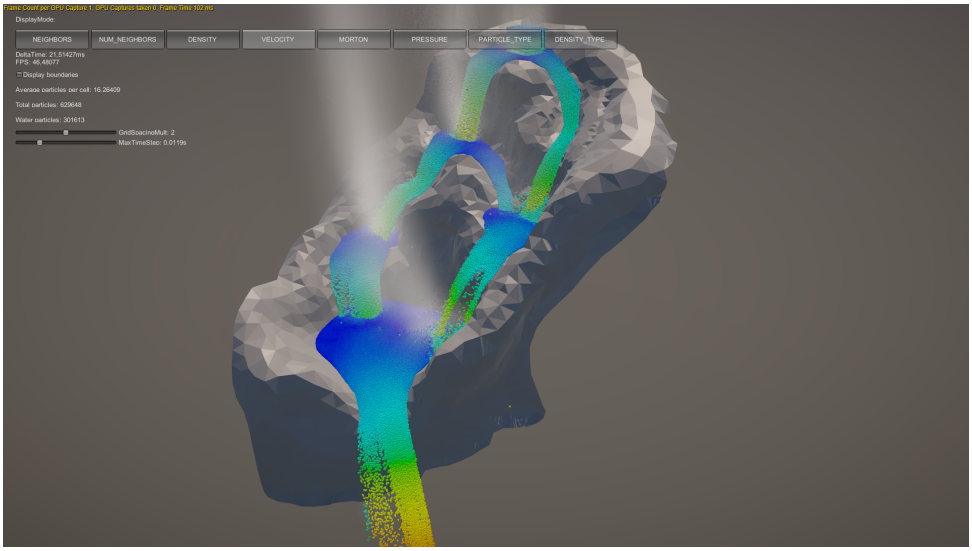


Obrázek 4.3: Časy jednotlivých kernelů pro vodopád typu horsetail.

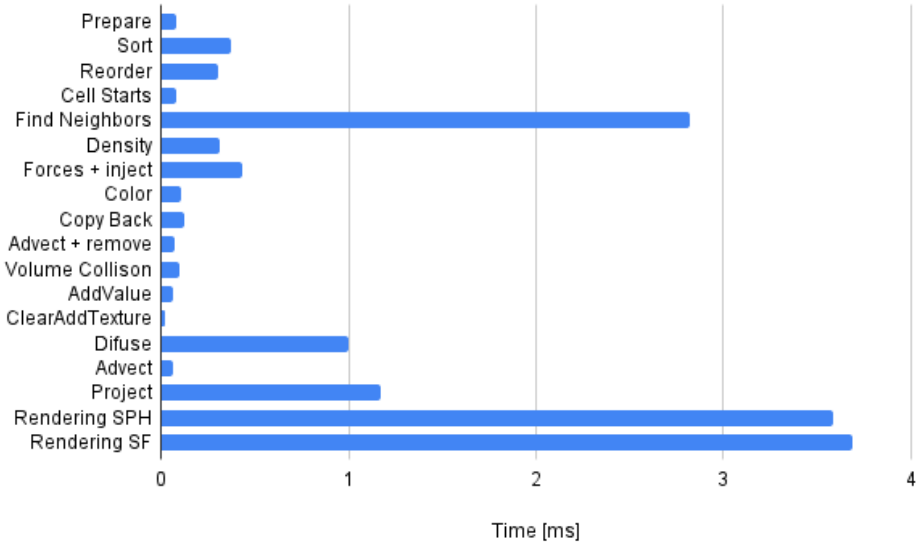
Obrázek 4.4: Parametry SPH simulace pro vodopád horsetail.

4.1.2 Multistep vodopád

Pro multistep nebo také kaskádový vodopád je specifický vysoký celkový spád rozdělený do několika kroků. Voda napříč spádem vytváří malá jezírka a vyplňuje propadliny v terénu, na obrázku 4.5 je vyobrazen model vodopádu v aplikaci. Vodopád má spád zhruba 80 metrů. Konkretní parametry simulace SPH jsou zobrazeny na obrázku 4.7. Dále je na obrázku 4.6 vykreslen graf časů jednotlivých kernelů simulace. Konkretní časy v milisekundách a počty částic jsou vypsány v tabulce 4.1. Seřazení kernelů v grafu je stejné jako u horsetail vodopádu.



Obrázek 4.5: Kaskádový vodopád

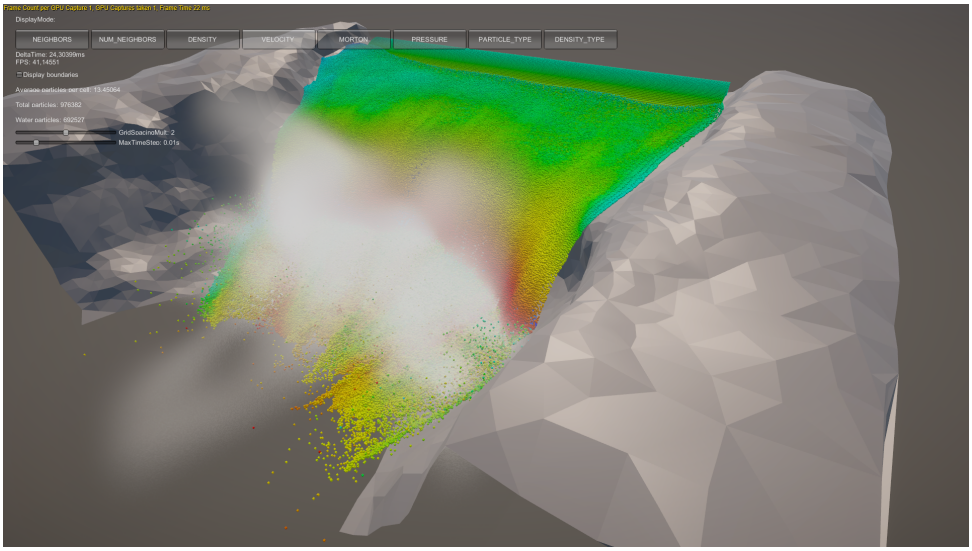


Obrázek 4.6: Časy jednotlivých kernelů pro kaskádový vodopád.

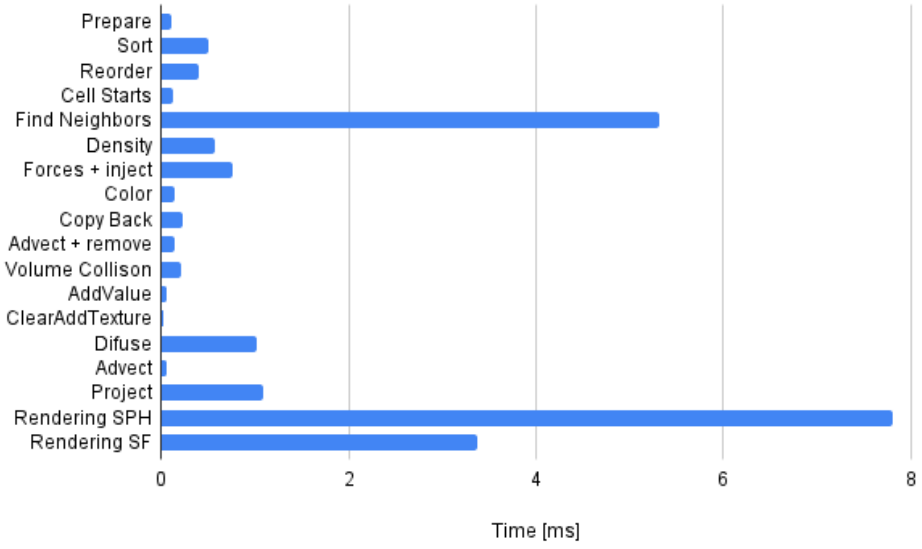
Obrázek 4.7: Parametry SPH simulace pro kaskádový vodopád.

4.1.3 Blokový vodopád

Inspirací pro tento vodopád byli Niagarské vodopády. u těchto vodopádů je specifický vysoký průtok a také rovná základna, na kterou voda dopadá a široký spád, na obrázku 4.8 je vyobrazen model vodopádu v aplikaci. Tento vodopád má spád podobný Niagarkým vodopádům a to přibližně 55 metrů. Konkretní parametry simulace SPH jsou opět zobrazeny na obrázku 4.10. Dále je na obrázku 4.9 vykreslen graf časů jednotlivých kernelů simulace stejně jako u předchozích dvou scén. Konkretní časy v milisekundách a počty částic jsou vypsány v tabulce 4.1. Seřazení kernelů v grafu je stejné jako u horsetail vodopádu.



Obrázek 4.8: Blokový vodopád.



Obrázek 4.9: Časy jednotlivých kernelů pro blokový vodopád.

4.2 Zhodnocení výsledků

4.2.1 Výkon

V tabulce 4.1 jsou vyneseny počty simulovaných částic a časy kernelů pro jednotlivé scény. Pro větší přehlednost jsou také vyneseny v grafu na obrázku

Obrázek 4.10: Parametry SPH simulace pro blokový vodopád.

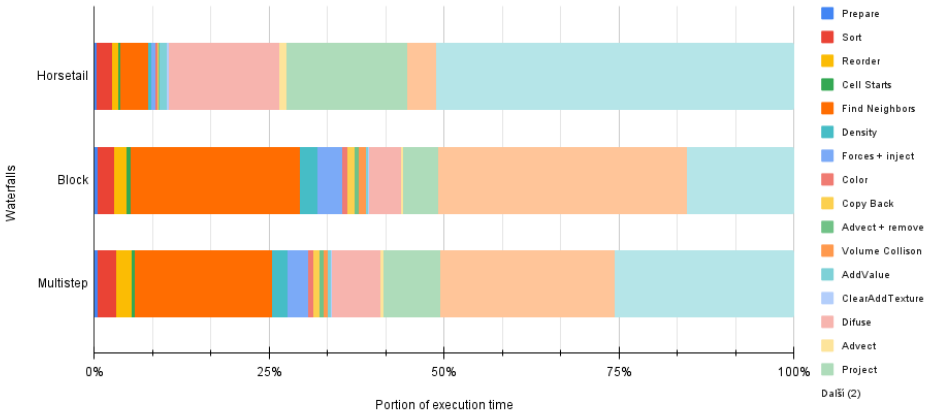
4.13. Včetně časů vykreslování *Rendering SPH* pro čas vykreslování částic SPH simulace a *Rendering SF* pro čas vykreslování objemu simulace Stable Fluids. Na obrázku 4.11 jsou dále relativní časy i s vykreslováním a na obrázku 4.12 relativní časy bez započítání vykreslování. Z grafů je vidět, jak s narůstajícím počtem částic tekutiny převládá čas potřebný pro simulaci SPH nad StableFluid simulací, která má pro všechny tři scény stejné rozlišení. Dále je možné identifikovat slabé části simulace. Nejvíce, času je stráveno konstrukcí seznamu sousedů. Jelikož je nutné také částice vykreslovat, představuje vykreslování částic a objemu simulace Stable Fluids největší slabinu současné implementace. To je především tím, že vykreslování nebyla v této práci věnována žádná pozornost a je nutné v budoucnu navrhnout efektivnější a realističtější způsoby vykreslování částic SPH a objemu simulace Stable Fluids.

Co se týče paměťové náročnosti, ta je pro všechny tři scény stejná. Pro

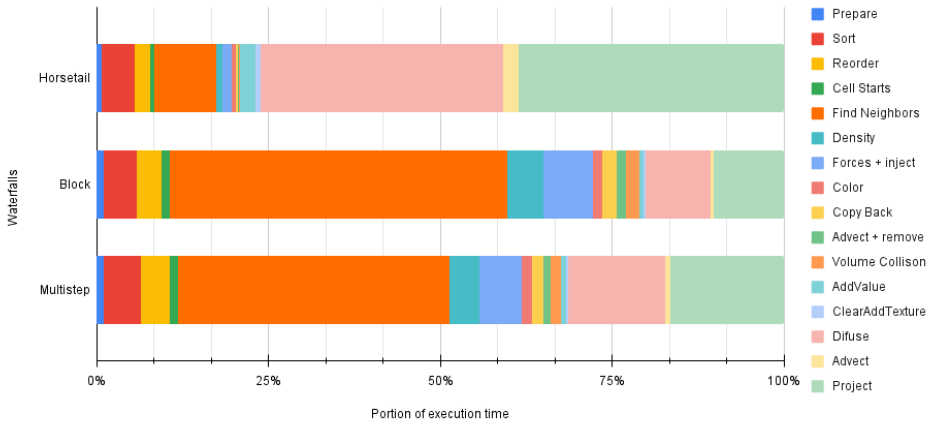
SPH simulaci je alokována paměť pro 1000000 částic a tedy celková velikost přibližně 286.1MB. Pro Stable fluid simulaci s rozlišením $64 \times 64 \times 64$ to je 15MB, což na schopnosti dnešních grafických karet není mnoho.

Waterfalls	Horsetail	Block	Multistep
Static particles [#]	98845	283855	328035
WaterParticles [#]	20000	700000	330000
Kernels		Times[ms]	
Prepare	0.0190	0.1149	0.0774
Sort	0.1303	0.5054	0.3749
Reorder	0.0588	0.3997	0.3058
Cell Starts	0.0199	0.1198	0.0819
Find Neighbors	0.2396	5.3062	2.8190
Density	0.0266	0.5705	0.3115
Forces + inject	0.0364	0.7664	0.4305
Color	0.0156	0.1470	0.1105
Copy Back	0.0070	0.2278	0.1220
Advect + remove	0.0043	0.1531	0.0740
Volume Collison	0.0050	0.2091	0.0997
AddValue	0.0614	0.0637	0.0608
ClearAddTexture	0.0210	0.0211	0.0221
Difuse	0.9501	1.0222	0.9976
Advect	0.0636	0.0626	0.0599
Project	1.0409	1.0942	1.1751
Rendering SPH	0.2493	7.8058	3.5866
Rendering SF	3.0844	3.3736	3.6885
Total	6.0333	21.9631	14.3979
Total (no rendering)	2.6996	10.7837	7.1228

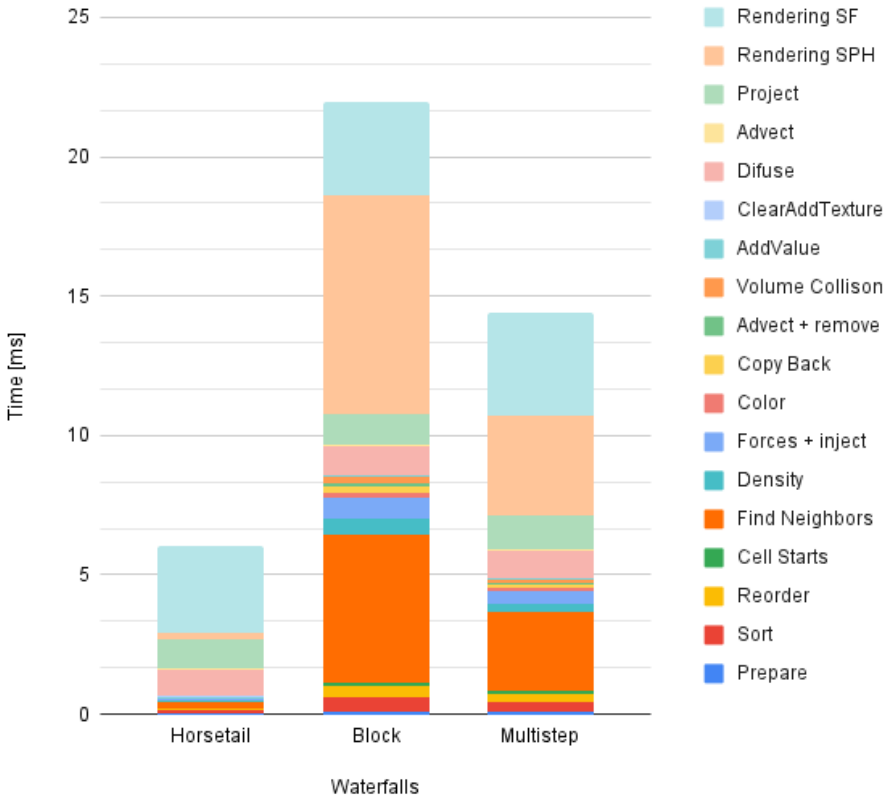
Tabulka 4.1: Naměřená data pro jednotlivé vodopády



Obrázek 4.11: Relativní časy kernelů i s vykreslováním.



Obrázek 4.12: Relativní časy kernelů bez vykreslování.



Obrázek 4.13: Absolutní časy v součtu i s vykreslováním.

Kapitola 5

Závěr

V této práci jsem se zabýval problematikou simulace velkých vodopádů. Hlavním cílem bylo navrhnout model umožňující takových vodopádů. Dále v herním enginu Unity vytvořit demonstrační aplikaci implementující tento model na GPU k simulaci alespoň tří scén vodopádů. Na těchto scénách následně změřit, jak se projevuje počet částic a parametry simulace na výkonu modelu.

V kapitole 2, kde proběhla analýza metod pro simulaci dynamiky tekutin a vodopádů, byl navržen model simulace vodopádu pomocí metody SPH a StableFluids na GPU.

Navržený model byl v kapitole 3 implementován v herním enginu Unity verze 2022.1.16f1. Bohužel jsem z časových důvodů nestihl implementovat funkcionalitu pro dynamické hraniční částice. Nicméně současná implementace je navržena tak, aby její doplnění působilo co nejmenší potíže.

S využitím herního enginu Unity byly dále vytvořeny tři demonstrační aplikace představené v kapitole 4, které testují možnosti modelu na třech scénách vodopádu. Na těchto scénách bylo změřeno, jak se projevuje nastavení parametrů scény a simulace na výkonu vizuálním vzhledu vodopádů.

Po uskutečněných měřeních lze vyvodit, že největší slabinou současné implementace je vykreslování částic a mlhy vodopádu. Co se týče samotné simulace, představuje proces konstrukce seznamu sousedů největší výpočetní zátěž a je nutné prozkoumat další možnosti, jak daný výpočet urychlit. Nabízí se hierarchické zpracování, kdy jsou malé buňky sloučeny do větších, čímž by se zlepšilo využití vláken skupin, které při malém počtu částic v buňkách pracují "naprázdno". Po vizuální stránce je model nejvíce vhodný pro vodopády typu

block, kde je velkým průtokem způsobena vysoká tvorba mlhy a vodopád tak nepůsobí, jako kdyby se z něj voda vypařovala, jak je možné pozorovat u dalších dvou vodopádů.

Reference

[1] Andy Adinets a Duane Merrill. "Onesweep: A Faster Least Significant Digit Radix Sort for GPUs". In: (čvn. 2022). DOI: 10.48550/arXiv.2206.01784.

[2] Nadir Akinci et al. "Versatile Rigid-Fluid Coupling for Incompressible SPH". In: *ACM Trans. Graph.* 31 (čvc. 2012), 62:1–62:8. DOI: 10.1145/2185520.2185558.

[3] Jan Bender a Dan Koschier. "Divergence-Free SPH for Incompressible and Viscous Fluids". In: *IEEE Transactions on Visualization and Computer Graphics* 23 (2017), s. 1193–1206. URL: https://api.semanticscholar.org/CorpusID:14623432.

[4] Giuseppe Bilotta et al. "Fast, feature-rich weakly-compressible SPH on GPU: coding strategies and compiler choices". In: *ArXiv* abs/2207.11328 (2022). URL: https://api.semanticscholar.org/CorpusID:251040105.

[5] Colin Braley a Adrian Sandu. "Fluid Simulation For Computer Graphics: A Tutorial in Grid Based and Particle Based Methods". In: 2009. URL: https://api.semanticscholar.org/CorpusID:18598591.

[6] Matthias Broske. *GPUStableFluids*. en. 24. květ. 2024. URL: https://github.com/matthiasbroske/GPUStableFluids/tree/main (cit. 24. 05. 2024).

[7] *Cauchy momentum equation*. en. 2024. URL: https://en.wikipedia.org/wiki/Cauchy_momentum_equation (cit. 20. 05. 2024).

[8] *Cauchy momentum equation*. en. 2024. URL: https://en.wikipedia.org/wiki/Continuity_equation (cit. 20. 05. 2024).

[9] *Density of the mix of dry air and water vapor - moist humid air*. en. 2024. URL: https://www.engineeringtoolbox.com/density-air-d_680.html (cit. 16. 05. 2024).

[10] *Derivation of the Navier–Stokes equations*. en. 2024. URL: https://en.wikipedia.org/wiki/Derivation_of_the_Navier%E2%80%93Stokes_equations (cit. 20. 05. 2024).

[11] Simon Green. *Particle-based Fluid Simulation based Fluid Simulation*. en. 2008. URL: https://developer.download.nvidia.com/presentations/2008/GDC/GDC08_ParticleFluids.pdf (cit. 20. 05. 2024).

[12] Yu Guan et al. "Modeling and rendering of realistic waterfall scenes with dynamic texture sprites". In: *Journal of Visualization and Computer Animation* 17 (pros. 2006), s. 573–583. DOI: 10.1002/cav.156.

[13] Mark Harris. "Fast Fluid Dynamics Simulation on the GPU". In: *Fluid Dynamics* 1 (led. 2005), s. 637–666. DOI: 10.1145/1198555.1198790.

[14] Rama C. Hoetzlein. *FAST FIXED-RADIUS NEAREST NEIGHBORS: INTERACTIVE MILLION-PARTICLE FLUIDS*. en. 2013. URL: https://on-demand.gputechconf.com/gtc/2014/presentations/S4117-fast-fixed-radius-nearest-neighbor-gpu.pdf (cit. 20. 05. 2024).

[15] Markus Ihmsen et al. "A parallel SPH implementation on multi-core CPUs". In: *Comput. Graph. Forum* 30 (břez. 2011), s. 99–112. DOI: 10.1111/j.1467-8659.2010.01832.x.

[16] Markus Ihmsen et al. "Implicit incompressible SPH". In: *IEEE transactions on visualization and computer graphics* 20 (čvc. 2013). DOI: 10.1109/TVCG.2013.105.

[17] Markus Ihmsen et al. "SPH Fluids in Computer Graphics". In: *Eurographics*. 2014. URL: https://api.semanticscholar.org/CorpusID:13302656.

[18] Radomír Kalinay. "CFD Modelling of Horizontal Water Film Evaporation". Dipl. pr. Czech Technical University in Prague, 2011. URL: http://hdl.handle.net/10467/73241.

[19] Dan Koschier et al. "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids". In: květ. 2019. DOI: 10.2312/egt.20191035.

[20] *Material derivative*. en. 2024. URL: https://en.wikipedia.org/wiki/Material_derivative (cit. 20. 05. 2024).

[21] Nobuhiko Mukai, Yuto Hizono a Young Soo Chang. "Waterfall Simulation with Spray Cloud in different Environments". In: *The Journal of the Society for Art and Science* (2016). URL: https://api.semanticscholar.org/CorpusID:212529424.

[22] Nobuhiko Mukai, Yuto Hizono a Youngha Chang. "Particle based Waterfall Simulation with Spray Cloud Emerging from Basin". In: *Proceedings of 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SIMULTECH 2018. Porto, Portugal: SCITEPRESS - Science a Technology Publications, Lda, 2018, s. 55–61. ISBN: 9789897583230. DOI: 10.5220/0006896500550061. URL: https://doi.org/10.5220/0006896500550061.

[23] Nobuhiko Mukai, Yasuomi Sakai a Youngha Chang. "Waterfall Simulation by Using a Particle and Grid-Based Hybrid Approach". In: *2014 International Conference on Cyberworlds*. 2014, s. 23–30. DOI: 10.1109/CW.2014.12.

[24] Nobuhiko Mukai, Yuka Sunaoshi a Youngha Chang. "Study on Spray Cloud Behavior Depending on Waterfall Height". In: *2019 Nicograph International (NicoInt)*. 2019, s. 106–109. DOI: 10.1109/NICOInt.2019.00028.

[25] Matthias Müller, David Charypar a Markus Gross. "Particle-Based Fluid Simulation for Interactive Applications". In: sv. 2003. Čvc. 2003, s. 154–159. ISBN: 1581136595.

[26] *Niagara Falls*. en. 2024. URL: https://en.wikipedia.org/wiki/Niagara_Falls (cit. 16. 05. 2024).

[27] Thomas Smith. *OneSweepUnity*. en. 24. květ. 2024. URL: https://github.com/b0nes164/GPUSorting/tree/main (cit. 24. 05. 2024).

[28] Barbara Solenthaler. "Predictive-corrective incompressible SPH". In: *ACM Trans. Graph. Article* 28 (zář. 2009). DOI: 10.1145/1576246.1531346.

[29] Jos Stam. "Real-Time Fluid Dynamics for Games". In: (květ. 2003). URL: http://graphics.cs.cmu.edu/nsp/course/15-464/Fall09/papers/StamFluidforGames.pdf.

[30] Jos Stam. "Stable Fluids". In: *ACM SIGGRAPH 99* 1999 (lis. 2001). DOI: 10.1145/311535.311548.

[31] Unity Technologies. *Unity Scripting Reference 2020.3 (LTS)*. en. 24. květ. 2024. URL: https://docs.unity3d.com/ScriptReference/index.html (cit. 24. 05. 2024).

[32] Unity Technologies. *Unity User Manual 2020.3 (LTS)*. en. URL: https://docs.unity3d.com/Manual/index.html (cit. 24. 05. 2024).

[33] Unity Technologies. *Universal Render Pipeline overview*. en. 24. květ. 2024. URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@17.0/manual/index.html (cit. 24. 05. 2024).

[34] Matthias Teschner et al. "Optimized Spatial Hashing for Collision Detection of Deformable Objects". In: *VMV'03: Proceedings of the Vision, Modeling, Visualization* 3 (pros. 2003).

Příloha A

Seznam příloh

- Tato práce jako pdf soubor
- Sestavené scény pro Windows x86_64 s grafickým rozhraním DirectX12
  - Je nutné sloučit `_Data` složku do složky se spustitelným souborem
- Unity projekt
  - veškerý kód se nachází ve složce `Assets`
  - projekt je nastavený tak, aby mohl být okamžitě sestaven stejně jako byla sestavena aplikace, na které probíhalo měření
- Obrázky pořízené v aplikacích