Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Voxelový terén a částicové simulace vody

2022

Bc. Daniel Bazala

Vedoucí práce:
RNDr. Eduard Bartl, Ph.D.

Studijní program: Aplikovaná informatika,
Specializace: Vývoj software

Bibliografické údaje

Autor:	Bc. Daniel Bazala
Název práce:	Voxelový terén a částicové simulace vody
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2022
Studijní program:	Aplikovaná informatika, Specializace: Vývoj software
Vedoucí práce:	RNDr. Eduard Bartl, Ph.D.
Počet stran:	44
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Bc. Daniel Bazala
Title:	Voxel terrain and water particle simulations
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2022
Study program:	Applied Computer Science, Specialization: Software Development
Supervisor:	RNDr. Eduard Bartl, Ph.D.
Page count:	44
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Diplomová práce navazuje na výsledky bakalářské práce, která se zabývala implementací realistického terénu pomocí voxelové reprezentace. V diplomové práci studuji částicové systémy a možnosti jejich simulací. Zvláště se zaměřuji na částicovou simulaci kapalin a rekonstrukci povrchu. Důraz je kladen na korektní chování kapaliny dle platných fyzikálních zákonů a následné zobrazení kapaliny pomocí standardních vykreslovacích technik pro polygonovou grafiku.

Synopsis

Master's thesis follows up my bachelor's thesis results which dealt with realistic terrain implementation using voxel representation. In my master's thesis I study particle systems and possibilities of their simulations. I especially focus on the fluid particle simulation and surface reconstruction. The emphasis is laid on the correct fluid behaviour according to valid physical laws and subsequent fluid visualization using standard rendering techniques for polygonal graphics.

Klíčová slova: terén; simulace; voda; sph; částicové simulace; 3D; voxel; marching cubes; unreal engine

Keywords: terrain; simulation; water; sph; particle simulation; 3D; voxel; marching cubes; unreal engine

Děkuji vedoucímu této práce RNDr. Eduardu Bartlovi, Ph. D. za podněty k tématu a pomoc se zpracováním problematiky. Dále bych chtěl také poděkovat celému pedagogickému sboru KI PřF UPOL za kvalitní a individuální přístup při vzdělávání.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Ve své bakalářské práci jsem se zabýval voxelovým terénem a nástroji jeho editování. Diplomová práce využívá poznatků a implementace této bakalářské práce, dále ji však rozšiřuje o novou kvalitu, a to částicové simulace vody. Jak voxelový terén, tak částicové simulace vody do velké míry využívají podobné techniky a algoritmy, proto se jeví velmi vhodné tyto dvě technologie spojit do jedné aplikace, jež bude využívat přínosu obou technologií v jejich vzájemné interakci.

Diplomová práce nejprve stručně seznamuje s výsledky a závěry předchozí bakalářské práce. Prochází problematiku voxelové reprezentace (jež je důležitá i pro částicové simulace) a problematiku polygonové reprezentace (jež je zásadní pro renderování). Dále také studuji vzájemné vztahy a souvislosti mezi různými reprezentacemi. Takto navržený dynamický terén založený na voxelové reprezentaci nabízí vhodný základ pro další rozšíření, např. simulace kapalin, jejíž implementaci v interakci s terénem se věnuji v poslední části této práce.

Samotná simulace vody, jež vychází z obecné teorie částicových systémů, může být realizována rozličnými způsoby. Hlavním faktorem určujícím způsob realizace simulace je poměr výkonu vůči preciznosti dle fyzikálních zákonů. Pokud bychom chtěli dosáhnout ideální, detailní simulace, která se bude co možná nejvíce blížit reálnému chování vody, pak bychom zvolili simulaci na molekulární úrovni. Taková simulace by však pro větší objemy vody byla výpočetně velmi náročná a pro aplikace v reálném čase nepoužitelná. Algoritmy implementace tedy vždy využívají určitou míru aproximace pro získání vyššího výkonu aplikace za cenu ztráty preciznosti a detailů simulovaného systému.

Práce se dále zabývá metodami, jež řeší problematiku simulace kapalin, zvláště pak metodou Smoothed Particle Hydrodynamics (SPH) navrženou pro simulaci astronomických a hydrodynamických jevů. Nejprve jsou nastíněny teoretické základy vycházející z fyzikálního modelu popsaného pomocí Navier-Stokesovy rovnice, dále metody přístupu dle Eulera a dle Lagrangeho. Obsáhleji pak práce rozebírá metodu SPH, jejíž variantu jsem použil v praktické aplikační části svojí práce. Analyzuji zde teorii metody SPH, její fáze, implementaci a nakonec i optimalizace, jež jsou z hlediska výkonu aplikace velmi důležité. Práce také podrobněji studuje metody rekonstrukce povrchu, které jsou důležité pro správnou vizualizaci simulace.

2 Voxelový terén

Mnoho současných her a aplikací simulujících reálný svět používá trojrozměrný terén. Dle požadavků a využití takových aplikací můžeme sledovat různé přístupy k problému od jednoduchých statických ploch budících zdání skutečného povrchu, přes složitější procedurálně generované terény, až po pokročilé terény založené na principu voxelové reprezentace.

Nejjednodušším přístupem je reprezentovat terén jako statickou síťovinu o $n \times n$ bodech, kde jsou sousedící body navzájem propojené hranami. Jednotlivé body pak mají různou výškovou souřadnici, čímž je docíleno zdání zvrásnění terénu. Pro nastavení výškových souřadnic může být použita textura s pozvolným gradientem, tzv. výšková mapa. Iluzi povrchu skutečného terénu navodíme pomocí vhodných textur - difuzní, normálové, odrazové či bump textury. [2]

O něco složitějším přístupem je procedurálně generovaný terén. Takový terén využívá algoritmy popisující jeho strukturu a vzhled. Algoritmus může terén buďto předgenerovat, anebo běžet přímo v reálném čase v rámci aplikace a generovat terén v místech, kde to aplikace zrovna vyžaduje. Tímto způsobem můžeme dosáhnout potenciálně nekonečného terénu, rozděleného na menší segmenty, které se přidávají k již existující ploše terénu, např. dle pohybu hráče. Navíc však musíme řešit problémy s přechodem segmentů tak, aby na sebe segmenty dokonale navazovaly, a to včetně textur a normálových vektorů, jež jsou důležité pro správné vykreslování. [2]

Více pokročilým přístupem je reprezentovat terén pomocí voxelové reprezentace. Voxelová reprezentace na rozdíl od obou předchozích přístupů zachycuje nejen povrch terénu, ale i jeho objem. Voxel je trojrozměrnou alternativou ke grafickému pojmu pixel. Voxelový terén se tedy skládá z voxelů o určitě velikosti, tuto konstantu budu dále nazývat velikost voxelu. [2]

Voxelová reprezentace daleko lépe vystihuje terén tak, jak existuje ve skutečnosti. Skutečný terén má určitý objem, skládá se z malých částic, na něž působí fyzikální zákony, z částic, které můžeme přidávat či odebírat. Voxelový terén je aproximací tohoto skutečného terénu, velikost voxelu pak určuje do jaké míry se chceme přiblížit detailům skutečného terénu a do jaké míry simulace zůstává odhadovanou aproximací. Jelikož algoritmy řešící voxelový terén obvykle prochází voxely ve třech souřadnicových osách, bývá časová složitost takových algoritmů v $O(n^3)$, kde n je počet voxelů v jedné souřadnicové ose. Zvětšením velikosti voxelu pak algoritmus vyřeší větší výsek trojrozměrného prostoru při stejném výpočetním výkonu, nicméně za cenu hrubší aproximace. Klíčovým faktorem při volbě velikosti voxelu pak zůstává poměr výkonu a detailnosti terénu.

2.1 Voxelová reprezentace

Voxelová reprezentace je důležitá jak pro editaci a vykreslování voxelového terénu, tak pro zobrazení částicové simulace kapaliny, kterému se věnuji v dalších kapitolách. V principu se jedná o způsob, jak zachytit objem objektů pomocí

3D skalárního pole – voxelové mřížky. Ta se skládá z diskrétních bodů v 3D prostoru, kterým říkáme voxely. Voxely jsou obvykle v mřížce pravidelně rozmístěny ve stejných vzdálenostech a každý voxel nese určitou hodnotu, nejčastěji skalár nebo vektor. [2]

Voxel si neuchovává informaci o své poloze, jeho poloha bývá dána souřadnicemi v datové struktuře (např. trojrozměrném poli). Můžeme si jej představit také jako krychli, jejíž střed obsahuje přesnou hodnotu v daném místě 3D prostoru. Naopak čím více se přiblížíme okrajům této krychle, získáme nepřesnější hodnotu, jelikož se vzdalujeme od místa, kde byla vypočtena či uložena přesná hodnota. Abychom snížili nepřesnost odhadu hodnoty v libovolném místě voxelové mřížky, můžeme použít polynomiální interpolaci s okolními voxely. [2]

Voxelovou mřížku definují specifické parametry jako její rozlišení, velikost voxelu a pozice v prostoru. Rozlišení uvádíme v počtu voxelů ve třech souřadnicových osách, např. rozlišení 64x64x128 voxelů. Rozměry mřížky pak můžeme vypočítat jako její rozlišení krát velikost voxelu. Na konstantě velikosti voxelu zároveň závisí, jak detailní aproximaci objemu provádíme. Čím větší stanovíme velikost voxelu, tím nepřesnější aproximace bude. [2]

2.2 Polygonová reprezentace

Většina současných herních engine a grafických programů pracuje především s polygonovou grafikou. Polygonová grafika nezachycuje objem objektů, ale pouze aproximuje jejich povrch pomocí mnohoúhelníků (polygonů). Skupině vzájemně propojených mnohoúhelníků, které dohromady reprezentují model, říkáme polygonová síť. Polygonová reprezentace je výhodná, protože nad ní snadno definujeme efektivní algoritmy pro její transformace a vykreslování. [2]

Pro snadné a efektivní vykreslování terénu a stejně tak simulace vody proto na základě voxelové reprezentace generuji ještě reprezentaci polygonovou. Tomuto procesu se říká polygonizace. Jejím cílem je najít povrch objektu a aproximovat jej za pomocí polygonů. Voxelový terén tak má dvě reprezentace, jednu vnitřní (skrytou) voxelovou a druhou vnější (viditelnou) polygonovou reprezentaci. Simulace vody má dokonce reprezentace tři. První je částicová reprezentace sloužící k simulaci algoritmu SPH (vysvětleno později). Druhá je reprezentace voxelová, která slouží pro vygenerování polygonů, a je jakýmsi přechodným mezistupněm. A třetí je samotná polygonová reprezentace, díky které může být voda snadno vykreslována v libovolném moderním engine pracujícím s polygony.

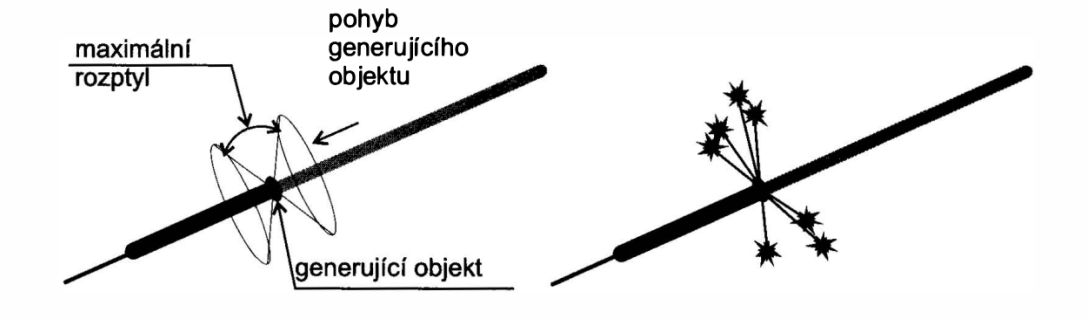
K polygonové reprezentaci se váží ještě další techniky zajišťující realistické vykreslování modelů. Mezi nimi jsou především důležité normálové vektory vrcholů, UV mapa a materiál. Díky normálovým vektorům vrcholů může být správně spočítáno osvětlení modelu a jeho stínování. Technika UV mapování určuje, jakým způsobem má být 2D textura namapována na povrch 3D modelu. Vše se pak kombinuje v materiálu, který určuje výsledný vzhled povrchu modelu a ovlivňuje jeho vlastnosti jako barevnost, hrubost, odlesky nebo průhlednost.

3 Částicové systémy

„Silnou modelovací a zobrazovací technikou jsou systémy částic (particle systems), které se používají zejména k modelování objektů, jejichž tvar je natolik členitý, nebo se mění takovým způsobem, že ho není možno reprezentovat jako povrch. Takovými objekty jsou hejna ptáků či ryb, padající sníh, déšť, oheň, mlha, dým, tráva, les, atp." [19]

Kromě polygonové a voxelové reprezentace můžeme použít ještě reprezentaci pomocí částic. Každá částice se chová jako samostatný objekt a má specifické vlastnosti jako polohu, rychlost, zrychlení, velikost nebo tvar. Pro zvláštní účely můžeme definovat také dodatečné vlastnosti jako např. hmotnost, hustotu, tlak, síly, jak tomu dělám při simulaci kapalin. Částice mohou interagovat mezi sebou navzájem nebo s cizími objekty. Soubor částic stejného typu nazýváme částicový systém.

Velká část systémů pracuje s nahodilostí při simulaci jevů jako oheň, kouř nebo mlha. Není tomu však pravidlem. V případě, že nám jde pouze o vizuální efekt, pak náhodné transformace částic napomáhají k jejich přirozenému chování, které v reálném světě vypadá zdánlivě chaoticky. Částicové systémy mohou být nicméně použity i při simulaci exaktních fyzikálních systémů, kde prvek nahodilosti nahradíme přesnými fyzikálními rovnicemi. Tento způsob používá i moje aplikace při částicových simulacích kapalin.



Obrázek 1: Prskavka jako systém částic [19].

4 Popis chování kapalin

4.1 Navier–Stokesova rovnice

Numerické řešení chování viskózní kapaliny, jež pro simulaci tekoucí vody dále používám, vychází z matematicko-fyzikálního modelu. Základy pro tento model položil již Sir Isaac Newton roku 1687, když ve své publikaci "Principia" poprvé popsal dynamické chování viskózní kapaliny při konstantní vizkozitě. Tyto kapaliny dnes na jeho počest nazýváme newtonovskými kapalinami. Později Daniel

Bernoulli a Leonhard Euler odvodili rovnice popisující chování neviskózních kapalin. Až roku 1845 Sir George Stokes odvodil rovnici popisující chování viskózní kapaliny, když do Eulerových rovnic doplnil podmínky pro viskózní proudění. Vznikla tak výsledná rovnice, dnes známá jako Navier–Stokesova rovnice. [16]

Navier–Stokesova rovnice popisuje chování nestlačitelné newtonovské kapaliny. Jedná se o soustavu parciálních diferenciálních rovnic, které matematicky vyjadřují zákon zachování hybnosti a zákon zachování hmotnosti pro newtonovské kapaliny [16]. Matematický zápis vypadá následovně:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \vec{u} + \vec{f}. \tag{1}$$

Vyřešením Navier-Stokesovy rovnice získáme pole vektorů $\nabla \vec{u}$ reprezentující rychlost toku v kapalině. Vektorové pole nám udává směr a velikost rychlosti toku kapaliny pro každý bod kapaliny v libovolném čase t. Rovnice uvažuje gradient tlaku v kapalině ∇p a hustotu kapaliny ρ při určité konstantní teplotě. Konstanta ν v rovnici označuje viskozitu kapaliny. Do rovnice jsou rovněž započteny další síly $\vec{f}$, jež na kapalinu působí, např. gravitační síla.

Navier-Stokesova rovnice popisuje pouze zákon zachování hybnosti. Aby bylo možné proudění kapaliny popsat úplně, bývá rovnice doplněna ještě o zákon zachování hmotnosti. Ten vyjadřuje tzv. rovnice kontinuity:

$$\nabla \cdot \vec{u} = 0. \tag{2}$$

Operátor nabla ∇ je diferenciální operátor sloužící k výpočtu gradientu – vektorového pole, které značí směr a velikost největší změny skalárního pole. Operátorem nabla ∇ ve skalárním součinu s vektorovým polem získáme divergenci tohoto pole. Divergence gradientu pole pak udává expanzi nebo kompresi tohoto pole, např. zda v daném bodě kapalina odtéká pryč či přitéká. Tato rovnice nám jinými slovy říká, že divergence rychlosti proudění je vždy nulová. Pokud by nulová nebyla, znamenalo by to, že v kapalině hmotnost resp. objem přibývá nebo naopak ubývá. To by porušovalo zákon zachování hmotnosti. Taková situace by mohla nastat pouze v případě nějakého zřídla či výpustě, uvažujeme však pouze uzavřené systémy, nikoliv systémy otevřené.

Obecné řešení Navier-Stokesovy rovnice ve třech dimenzích zatím nebylo nalezeno. Rovnici lze analyticky řešit v učitých specifických případech. V ostatních případech můžeme použít nepřesné numerické řešení. Jednou z implementací tohoto numerického řešení je algoritmus SPH, který jsem ve své aplikaci použil a který v dalších kapitolách podrobně popisuji. Důkaz existence hladkého řešení Navierovy-Stokesovy rovnice ve třech dimenzích bylo Clayovým matematickým institutem zařazeno na seznam sedmi nejdůležitějších matematických problémů, tzv. Problémů tisíciletí.

4.2 Používané přístupy

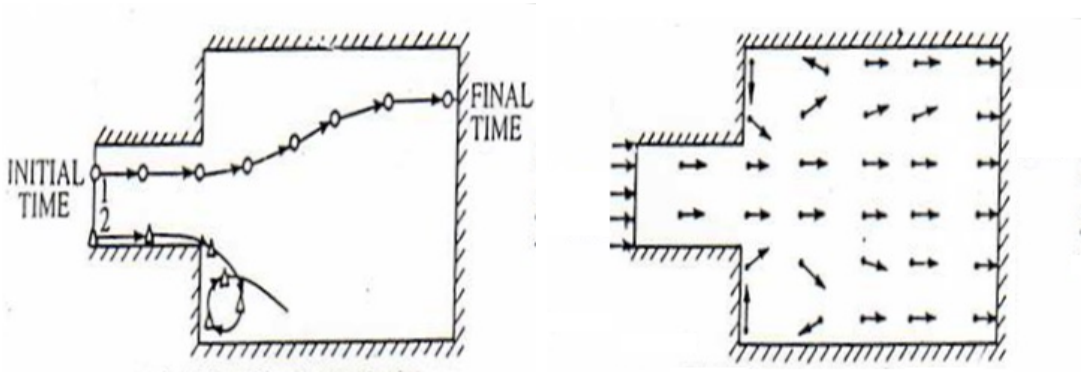
Na problematiku proudění tekutin lze nahlížet ze dvou pohledů známých jako Eulerův a Lagrangeův přístup. Tyto přístupy jsou dobře známé při fyzikálně-numerickém modelování a jejich implementace se používají při simulacích kontinua rozličných látek.

Pro zjednodušení představy pohybu kapaliny uvažujeme pojem spojitého kontinua, ve kterém je hmotnost rovnoměrně rozložena. Vlastnosti kapaliny pak vyšetřujeme pouze v určitých diskrétních geometrických bodech, které aproximují své okolí.

Eulerův a Langrangeův přístup se liší především v místě pozorování. Eulerův přístup používá k popisu proudění kapaliny pevnou diskrétní mřížku v prostoru, skrze kterou kapalina protéká. Kapalinu tedy pozorujeme jako vnější pozorovatelé, sledující vlastnosti kapaliny na konkrétním místě v prostoru v určitém čase.

Lagrangeův přístup naproti tomu sleduje přímo každou částici. Pozorovatel se pohybuje spolu s částicí a v určitém čase pozoruje vlastnosti této částice. Při tomto přístupu pozorovatel již nemá přehled o vlastnostech kapaliny v konkrétním bodě prostoru, ale místo toho sleduje trajektorii částice vůči její počáteční poloze.

Popsané přístupy vykresluje obrázek 2. Oba přístupy mají své výhody a nevýhody stejně jako specifické způsoby využití, které rozvádím dále v následujících podkapitolách.



(a) (b)

Obrázek 2: Lagrangeův přístup (a), Eulerův přístup (b). Zdroj: https://player.slideplayer.com/30/9529653/data/images/img2.jpg

4.2.1 Eulerův přístup

Eulerův přístup zaznamenává, resp. vypočítává hodnoty v pevných bodech prostoru. Tyto body jsou většinou rovnoměrně rozloženy v rámci určité diskrétní mřížky. Eulerův přístup můžeme formalizovat jako matematickou funkci f:

$$\vec{u} = f(x, y, z, t), \tag{3}$$

která pro libovolný bod o souřadnicích $[x, y, z]$ a pro libovolný čas t definuje vektor rychlosti $\vec{u}$ proudění kapaliny v daném bodě. Výpočet rychlostí je pro Eulerův přístup typický, mohli bychom však vypočítávat i jiné vlastnosti.

Při použití Eulerova přístupu vznikají typicky dva problémy, oba jsou však řešitelné. Prvním problémem je omezenost diskrétní mřížky pouze na určitý výsek prostoru. Jelikož se mřížka nepohybuje spolu s kapalinou, může nastat situace, kdy kapalina odtéká mimo sledovaný prostor. Tento problém lze vyřešit poměrně jednoduše pomocí dynamické mřížky, která může expandovat do požadovaného prostoru.

Druhým problémem je hrubost vzorkování. Při použití příliš hrubé mřížky nejsme schopni zachytit jemné nuance v kapalině, jako například detailní zvlnění povrchu kapaliny. Tento problém lze samozřejmě řešit jemnější mřížkou, výrazně však roste paměťová náročnost. Takovou situaci pak můžeme vyřešit pomocí sofistikovanějších struktur, jako např. oktalových stromů nebo tzv. řídkých mřížek, které adaptivně používají jemnější vzorek pouze v místech, kde to simulace vyžaduje.

Mezi výhody tohoto přístupu patří vyšší přesnost simulovaného děje. Při numerickém řešení získáváme v daných bodech prostoru poměrně přesné výsledky, proto se tento přístup častěji používá u fyzikálních simulací, jež vyžadují exaktnější výsledek. Jednou z těchto metod je finite volume method [15]. K tomuto přístupu přirozeně vedou také simulace sestavené z dat fyzikálního měření, jelikož takto pořízená data jsou většinou získána bodovým měřením, např. sondou v určitém místě kapaliny. Mezi nevýhody patří vyšší paměťová a výpočetní náročnost, která výrazně roste se zjemňováním (vyšším rozlišením) diskrétní mřížky.

4.2.2 Lagrangeův přístup

Lagrangeův přístup řadíme mezi částicové přístupy, jelikož kontinuum toku kapaliny aproximujeme pomocí částic, z jejichž pohledu simulaci pozorujeme. Každá částice si s sebou nese své vlastnosti jako např. hmotnost, hustotu, tlak. Pomocí těchto částic numericky řešíme parciální diferenciální rovnice dle Navier-Stokesova modelu.

Lagrangeův přístup můžeme formalizovat jako matematickou funkci g:

$$\vec{u} = g(t), \tag{4}$$

která pro libovolnou částici aproximující kapalinu definuje vektor rychlosti $\vec{u}$, kterou se částice v libovolném čase t pohybuje v rámci kontinua toku kapaliny. Na rozdíl od Eulerova přístupu tedy nemůžeme zjišťovat rychlost proudění na určitých souřadnicích, ale pouze rychlost popř. pozici částice během její trajektorie, kterou urazí během své cesty proudem kapaliny.

Nevýhodou může být nutnost přistupovat k okolním částicím, jelikož se tyto částice v průběhu simulace vzájemně ovlivňují. Pokud bychom chtěli najít okolní částice v blízkosti sledované částice, museli bychom prohledat všechny částice kapaliny a zjistit, které z nich podmínku blízkosti splňují. To by vedlo k časové

složitosti v $O(n^2)$, kde n je počet částic v simulaci. Takový výpočet by byl při vysokém počtu částic, který je v simulaci běžný, výpočetně velmi náročný. Naštěstí lze tento problém řešit prostorovou hashovací mřížkou (trojrozměrná analogie hashovací tabulky), díky které lze najít blízké částice v konstantním čase.

Mezi výhody tohoto přístupu patří poměrně jednoduché paradigma, kdy si každá částice nese své vlastnosti s sebou. Takový přístup lze celkem přirozeně uchopit pomocí objektově orientovaného modelu a zároveň jednoduše paralelizovat. Navíc má tento přístup oproti Eulerovu přístupu nižší paměťovou náročnost, jež je přímo závislá na počtu částic v simulaci.

Zatímco Eulerův přístup se většinou používá pro exaktnější fyzikálně orientované simulace, Lagrangeův přístup využívá velká část herního a filmového průmyslu. Důvodem je snažší a efektivnější implementace Lagrageova přístupu pro vizuální zobrazení kapaliny.

Mezi metody vycházející z Lagrangeova částicového přístupu patří také algoritmus Smoothed particle hydrodynamics (SPH), jež jsem využil jako hlavní algoritmus pro simulace kapalin ve své práci. V následující kapitole podrobně rozeberu teoretický model i implementaci tohoto algoritmu.

5 Metoda Smoothed Particle Hydrodynamics

Smoothed-particle hydrodynamics (SPH) je algoritmická metoda pro částicovou simulaci kontinua pohybujícího se média. Metoda byla původně vyvinuta pro řešení astrofyzikálních problémů, její použití je však výrazně širší. Jedná se o bezsíťovou metodu vycházející z Lagrangeova přístupu, založenou na aplikaci Navier-Stokesovy rovnice zachování hybnosti a hmotnosti. Dnes patří mezi nejpoužívanější algoritmy pro simulaci kapalin [16].

5.1 Obecný popis metody

Základní myšlenkou metody SPH je aproximace kapaliny pomocí částic. Každá částice ovlivňuje okolní částice v rámci určitého poloměru, který nazýváme vyhlazovací poloměr. Abychom mohli simulovat síly, které působí na libovolnou částici, musíme nejprve vypočítat vlastnosti všech částic v simulaci.

Nejprve potřebujeme vypočítat hustotu, dále tlak a poté teprve můžeme vypočítat tlakové a viskózní síly působící na částice. Po sečtení všech sil působících na částice včetně konstantní síly gravitační, můžeme provést integraci rychlosti a aktualizovat pozice částic v čase.

Hustotou se myslí koncentrace hmotností v okolí vypočítávané částice. Není tím tedy myšlena fyzikální hustota, která by v celém objemu jediné kapaliny měla být stejná. Tuto hustotu nazýváme klidovou hustotou. V případě, že hustota částice je vyšší než klidová hustota, na částici budou působit síly, které se ji budou snažit přesunout na místo s nižší hustotou tak, aby nastal ideální stav kapaliny.

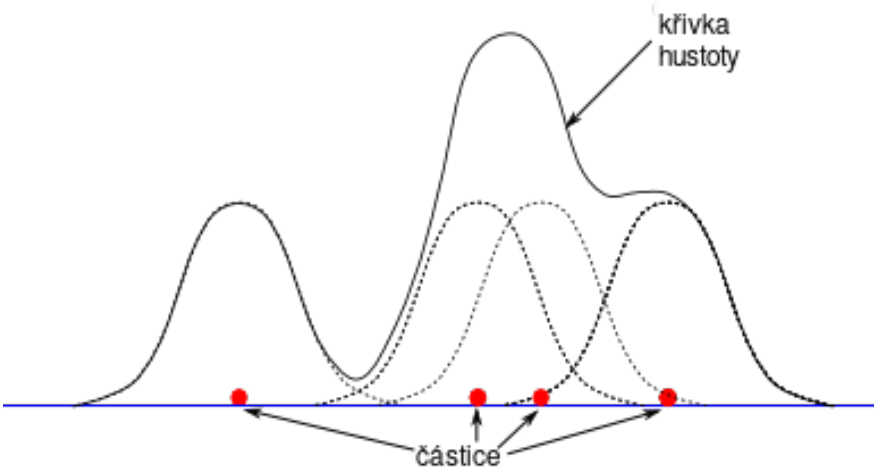
Pro každou částici je vypočtena její hustota v závislosti na hmotnostech okolních částic dle vyhlazovacího jádra. Vyhlazovací jádro definujeme jako matematickou funkci, která nabývá nejvyšší hodnoty pro částice v těsné blízkosti vypočítávané částice, zatímco pro částice u hranice vyhlazovacího poloměru se její hodnota blíží nule. Taková funkce zajišťuje, aby bližší částice měly vyšší váhu, a tudíž více ovlivnily vypočítávanou částici. Mezi nejčastěji používaná vyhlazovací jádra patří Gaussova funkce, funkce B-Spline nebo polynom šestého stupně:

$$W_{Gauss}(r,h) = \begin{cases} \frac{e^{-r^2}}{\pi^{\frac{3}{2}}h^3} & 0 \leq r \leq h \\ 0 & \text{jinak,} \end{cases} \tag{5}$$

$$W_{Poly6}(r,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \leq r \leq h \\ 0 & \text{jinak.} \end{cases} \tag{6}$$

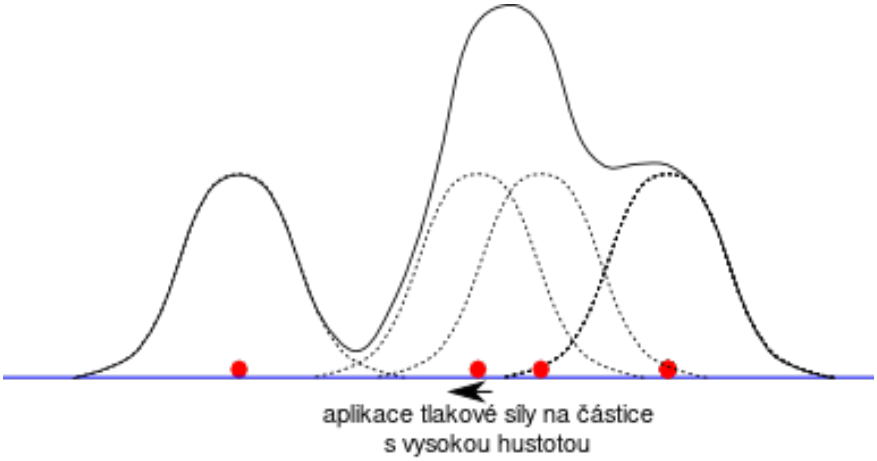
Parametr r je vzdálenost okolní částice od vypočítávané částice a parametr h je vyhlazovací poloměr. Pokud vzdálenost okolní částice přesahuje vyhlazovací poloměr, nemá smysl vyhlazovací jádro vůbec počítat a taková částice nemá na výpočet žádný vliv.

Situaci po vypočtení hustoty nám ilustruje obrázek 3, který je zjednodušením problému do jedné dimenze. Na tomto obrázku vizualizujeme křivku hustoty vzniklou sečtením hustot jednotlivých částic.



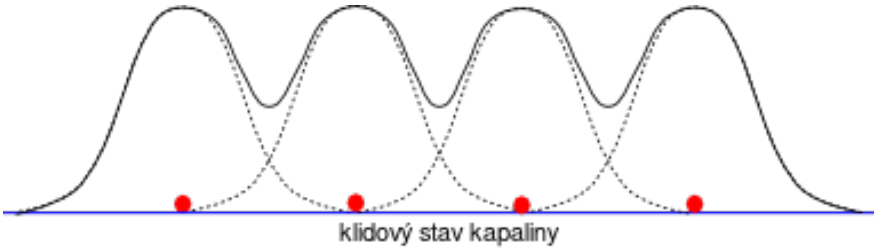
Obrázek 3: Výpočet hustot dle rozložení částic [8].

V další fázi vypočteme tlak pro každou částici. Tlak můžeme vypočítat jednoduše jako rozdíl hustoty vypočtené v předchozím kroku a konstantní klidové hustoty. Na základě rozdílu tlaků mezi místy v kapalině pak vypočteme tlakovou sílu působící na částici. Tato síla bude částici přirozeně tlačit do oblasti s nižším tlakem v závislosti na velikosti rozdílu mezi tlaky v těchto oblastech, viz obrázek 4.



Obrázek 4: Vznik tlakové síly působící na částici [8].

Tento postup se během simulačních kroků stále opakuje, dokud nedojde k ustálení kapaliny a jejímu klidovému stavu, viz obrázek 5. Při dostatečně malých krocích simulace tak můžeme poměrně věrohodně napodobit nestačitelné chování kapaliny.



Obrázek 5: Stav kapaliny po ustálení v klidovém stavu [8].

Zde jsem vysvětlil pouze základní koncept metody. Kapalinu ovlivňuje ještě řada dalších faktorů a sil, které jsou důležité pro její správné a realistické chování. Patří mezi ně viskózní síly, jež vznikají uvnitř kapaliny vlivem vzájemného tření částic a které mají velmi významný podíl na specifickém chování kapalin. Dále také síly povrchového napětí, vznikající v důsledku přitažlivosti molekul a projevující se zvláště na povrchu kapaliny. Zde přitahují povrchové molekuly ke kapalině a způsobují tak minimalizaci povrchu kapaliny.

5.2 Matematický popis

Z hlediska formálního matematického popisu se jedná o numerickou integrační metodu vycházející z aplikace Navier-Stokesovy rovnice.

Dle Lagrangeova částicového přístupu počítáme pro každou částici i její pozici r_i v čase t integrací její rychlosti $\vec{v_i}$:

$$\frac{\mathrm{d}r_i}{\mathrm{d}t} = \vec{v_i}. \tag{7}$$

Rychlost částice je nutno vypočítat na základě sil, které na ni působí. Tyto síly jsou závislé na hustotě a tlaku, popř. dalších veličinách, kterými kapalinu popisujeme. Jelikož používáme Lagrangeův částicový přístup, nemůžeme na rozdíl od Eulerova přístupu zjišťovat tyto veličiny v rámci prostoru a času. Namísto toho provádíme interpolaci veličin z okolních částic pomocí výše zmíněného vyhlazovacího jádra:

$$A_i(r) = \int_\Omega A(r')W(|r - r'|, h)\,\mathrm{d}r'. \tag{8}$$

Integrační metodou počítáme veličinu $A_i(r)$ částice i nad doménou Ω. Vyhlazovací jádro W určuje váhu atributu $A(r')$ okolní částice v závislosti na vzdálenosti částic $|r - r'|$ a vyhlazovacího poloměru h. Numerickým přepisem dostaneme výpočet:

$$A_i = A(r_i) = \sum_j A_j V_j W(|r_i - r_j|, h). \tag{9}$$

Výpočet probíhá sumací přes všechny částice j v simulaci. Tímto způsobem aproximujeme určitý integrál pro numerický výpočet.

Objem částice V vypočítáme podle vztahu hmotnosti m, hustoty ρ a objemu V:

$$V = \frac{m}{\rho}. \tag{10}$$

Po dosazení do rovnice 9 dostáváme výslednou rovnici:

$$A_i = A(r_i) = \sum_j A_j \frac{m_j}{\rho_j} W(|r_i - r_j|, h). \tag{11}$$

Výpočet hustoty získáme z rovnice 11 specifikací hustoty, jako počítané veličiny. Uvažujeme, že hmotnost částic se v průběhu výpočtu nemění, jedná se o předem definovanou konstantu. Na základě hmotností okolních částic m_j pak můžeme vypočítat hustotu ρ částice i:

$$\rho_i = \rho(r_i) = \sum_j \rho_j \frac{m_j}{\rho_j} W(|r_i - r_j|, h) = \sum_j m_j W(|r_i - r_j|, h). \tag{12}$$

Při výpočtu tlaku můžeme vyjít ze stavové rovnice ideálního plynu:

$$\begin{aligned} pV &= mRT, \\ p\frac{m}{\rho} &= mRT, \\ p\rho &= RT, \\ p &= k\rho, \end{aligned} \tag{13}$$

kde p je tlak, ρ hustota, V objem, m hmotnost, R měrná plynová konstanta a T teplota. Po úpravě rovnice zjišťujeme, že tlak závisí na hustotě, měrné plynové

konstantě a teplotě. Protože se v našem případě jedná o izotermickou kapalinu, u níž nepředpokládáme změnu teploty v čase, a protože měrná plynová konstanta naší kapaliny se také nemění, můžeme veličiny R a T nahradit konstantou k.

Abychom systém učinili stabilnějším, do výpočtu tlaku zavedeme ještě klidovou hustotu ρ_0:

$$p_i = k(\rho_i - \rho_0). \tag{14}$$

Klidová hustota ρ_0 je konstanta, která dovoluje mít v kapalině určitou kladnou hustotu bez nutnosti, aby v kapalině vznikal tlak, který by vedl k její nestabilitě. Dle implementace pak nižší hodnoty hustoty, než je klidová hustota, vedou buď nulovému tlaku, anebo k tlaku zápornému, který částice k sobě naopak přitahuje.

Dále počítáme síly působící na částici. Základními silami ovlivňující částice kapaliny jsou síla tlaková a síla viskózní. Tlakovou sílu vypočítáme dle rovnice:

$$\vec{f_i^p} = \vec{f^p}(r_i) = -\nabla p_i = -\sum_{j \neq i} p_j \frac{m_j}{\rho_j} \nabla W(|r_i - r_j|, h). \tag{15}$$

Tlaková síla zodpovídá za nestlačitelnost tekutiny. Kdykoliv se na určitém místě kapaliny nahromadí přílíš velký počet částic, vznikne zde přetlak a tlakové síly částice přesunou do oblasti s nižší hustotou tak, aby se tlakové rozdíly vyrovnaly.

Takto formulovaný výpočet tlakové síly by však porušoval Newtonův třetí pohybový zákon - zákon akce a reakce. Pokud bychom vzali v úvahu dvě částice s rozdílnou hustotou a tlakem, pak by jedna částice působila na druhou větší silou než ona na ni. Dle Newtonova zákona však víme, že tyto síly musí být symetrické - pokud jedna částice působí na druhou částici silou, pak i druhá částice musí působit na první silou stejně velkou, ale opačného směru.

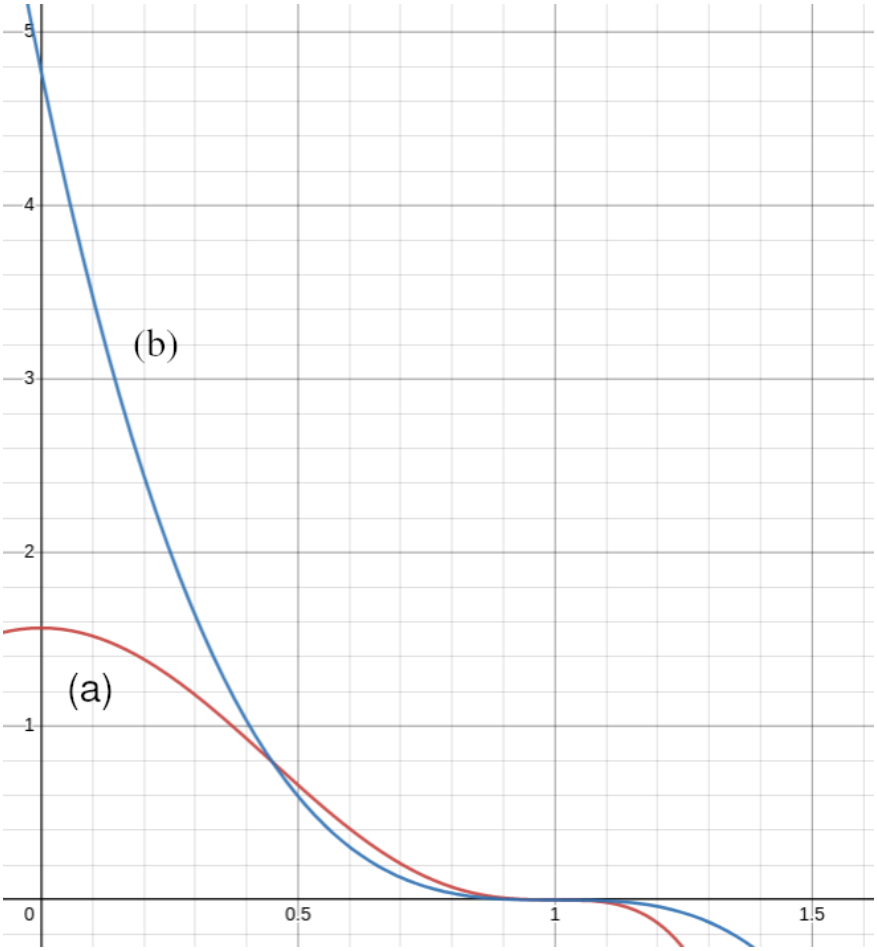
Potřebujeme tedy upravit výpočet tlakové síly tak, abychom získali symetrické síly. To můžeme udělat například následujím způsobem:

$$\vec{f_i^p} = \vec{f^p}(r_i) = -\sum_{j \neq i} \frac{p_i + p_j}{2} \frac{m_j}{\rho_j} \nabla W(|r_i - r_j|, h). \tag{16}$$

Pro potřeby výpočtu tlakových sil potřebujeme najít vhodnější vyhlazovací jádro, než jaké jsme použili pro výpočet předchozích veličin. Tato vyhlazovací jádra byla vhodná pro aproximaci hustoty či tlaku, velikost tlakových sil však musí výrazněji růst se snižující se vzdáleností mezi částicemi. Pro tento účel se lépe hodí tzv. špičaté vyhlazovací jádro:

$$W_{Spiky}(r, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3 & 0 \leq r \leq h \\ 0 & \text{jinak.} \end{cases} \tag{17}$$

Špičaté jádro vhodněji reaguje na blízké vzdálenosti mezi částicemi, jak ukazuje obrázek 6, který jej srovnává s již dříve použitým jádrem polynomu šestého stupně. Obě funkce jsou vykresleny pro vyhlazovací poloměr 1, osa x reprezentuje vzdálenost mezi částicemi.



Obrázek 6: Vyhlazovací jádro Poly6 (a) a jádro Spiky (b).

Dále je potřeba spočítat viskózní sílu, která je zodpovědná za specifické chování kapaliny vlivem vnitřního tření molekul. Tato síla brzdí tekutost kapaliny a brání jí ve změně tvaru. Kinetická energie molekul se vlivem tření mění na teplo, které ale ve výpočtu zanedbáváme. Výpočet viskózní síly odvodíme z viskózního členu Navier-Stokesovy rovnice:

$$\vec{f_i^v} = \vec{f^v}(r_i) = \nu\nabla^2\vec{u}(r_i) = \nu\sum_{j\neq i}\vec{u_j}\frac{m_j}{\rho_j}\nabla^2 W(|r_i - r_j|, h). \tag{18}$$

Viskózní síla závisí především na rychlosti okolních částic $\vec{u_j}$ a konstantě viskozity ν. Čím rychleji se budou okolní částice pohybovat a čím vyšší nastavíme konstantu viskozity, tím větší třecí síly budou na částici působit a budou ji tak zpomalovat v jejím dalším pohybu kapalinou. Rychlost částice jsme sice ještě nepočítali, ve výpočtu však uvažujeme její hodnotu z předchozího kroku simulace.

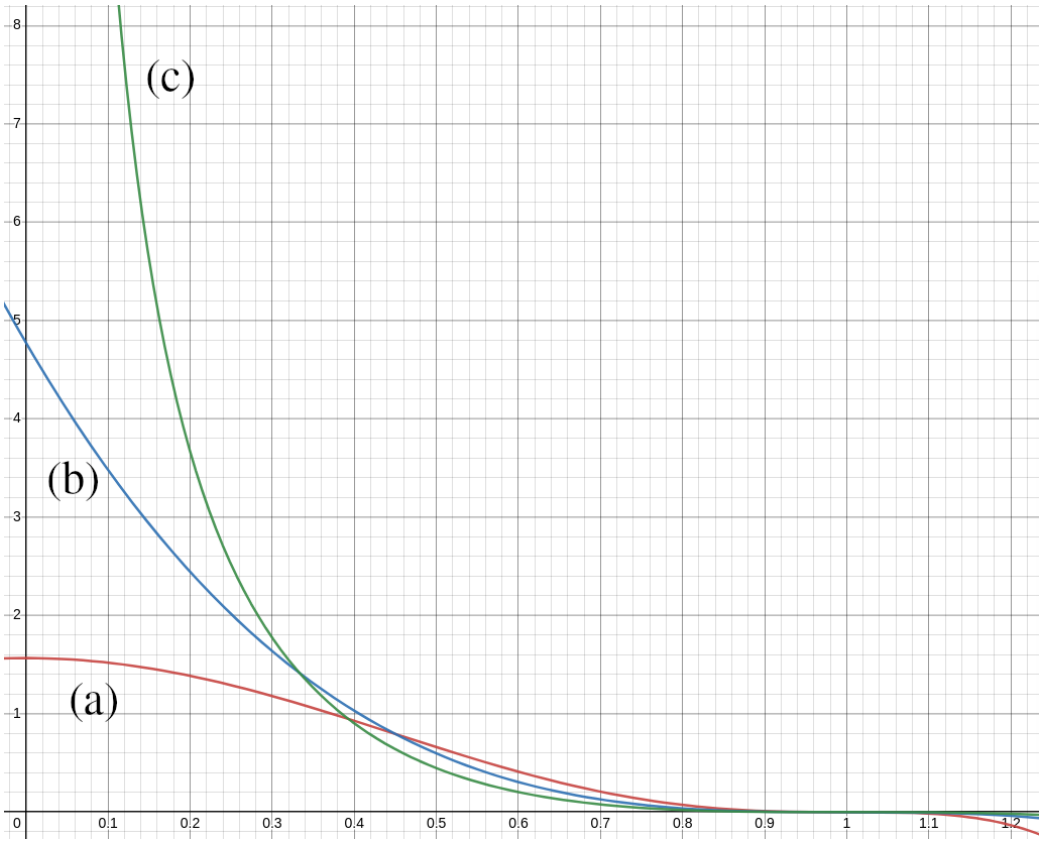
Takto definovaná viskózní síla opět není symetrická, proto výpočet upravíme:

$$\vec{f_i^v} = \vec{f^v}(r_i) = \nu \sum_{j \neq i} m_j \frac{\vec{u_i} - \vec{u_j}}{\rho_j} \nabla^2 W(|r_i - r_j|, h). \tag{19}$$

Zároveň potřebujeme nové vyhlazovací jádro, které bude vhodné pro správnou aproximaci viskózní síly:

$$W_{Viscosity}(r,h) = \frac{15}{2\pi h^3} \begin{cases} (-\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1) & 0 \leq r \leq h \\ 0 & \text{jinak.} \end{cases} \tag{20}$$

Srovnání viskózní jádra s předchozími jádry ukazuje obrázek 7.



Obrázek 7: Vyhlazovací jádro Poly6 (a), jádro Spiky (b) a jádro Viscosity (c).

Nyní máme vypočteny hydrodynamické síly působící na částice. Po započtení konstantní gravitační síly $\vec{f^g}$ tak můžeme vypočítat výsledkou sílu:

$$\vec{f_i} = \vec{f_i^p} + \vec{f_i^v} + \vec{f^g}. \tag{21}$$

A následně vypočítat rychlost částice v čase t integrací výsledné síly $\vec{f_i}$:

$$\frac{\mathrm{d}\vec{v_i}}{\mathrm{d}t} = \vec{f_i}. \tag{22}$$

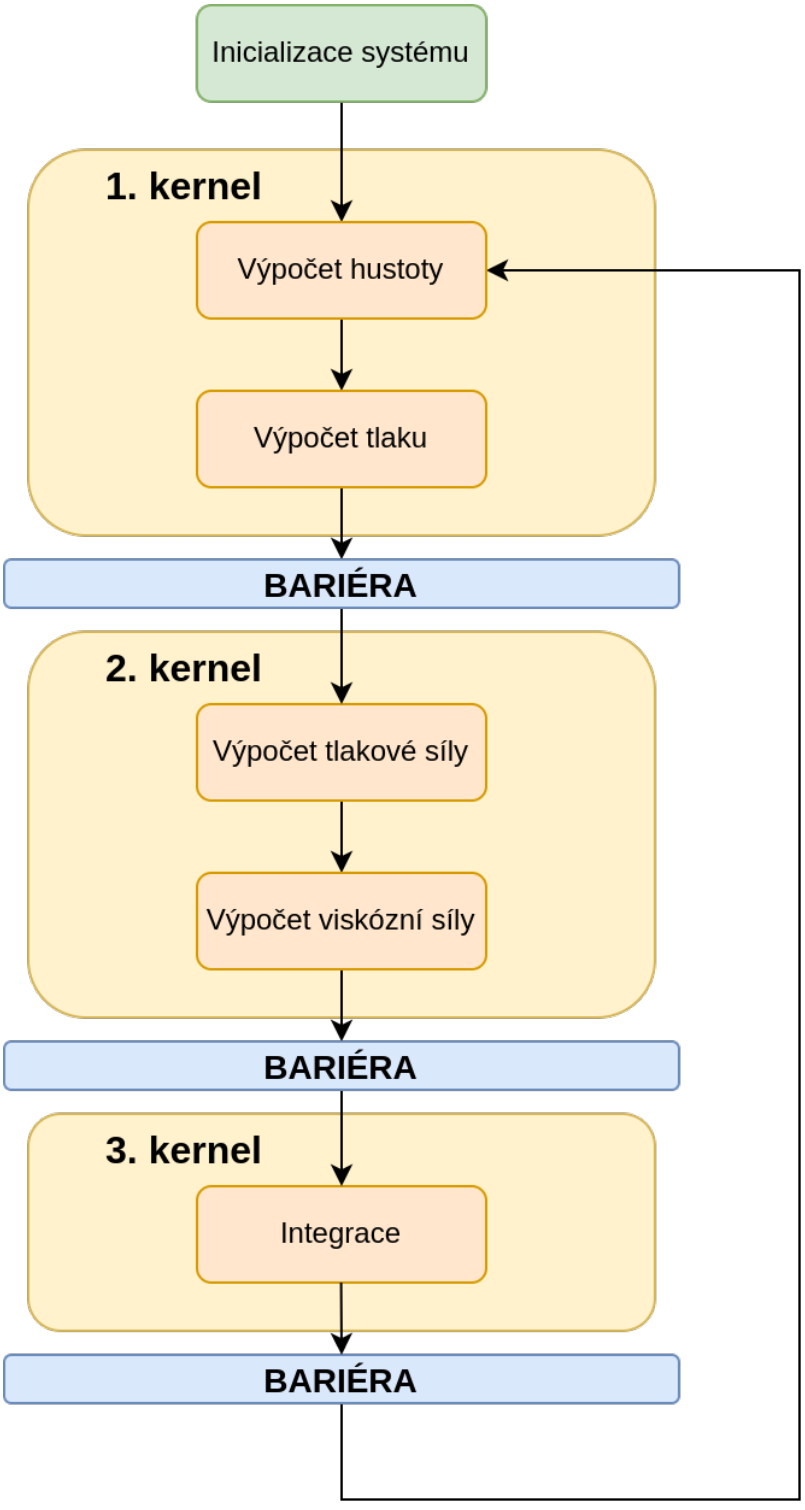
5.3 Popis algoritmu

Cílem algoritmu je vypočítat rychlosti jednotlivých částic, abychom částice mohli interpolovat v čase dle směru a velikosti jejich rychlostí. Tento výpočet a posun se odehrává v každém simulačním kroku, čas mezi kroky simulace by měl být natolik malý, aby simulace byla hladká a nedocházelo k příliš nepřesným přesunům částic.

Algoritmus metody SPH sestává z několika kroků, které jsou na sobě závislé. Následující krok vždy předpokládá dokončení výpočtu předchozího kroku pro všechny částice. V rámci jednoho kroku však výpočet můžeme paralelizovat, jelikož částice vypočítávané vlastnosti akumulují v sobě a okolní částice neovlivňují. V případě paralelizace tedy musí být mezi jednotlivými kroky bariéra. Krok, uvnitř něhož je možná paralelizace, dále nazývám kernel dle terminologie výpočtů na grafických kartách.

Moderní GPU jsou implicitně uzpůsobené pro paralelní výpočty. Skládají se ze stovek či dokonce tisíců výpočetních jader (kernelů), z nichž každé může samostatně vykonávat kód (tzv. shader). Algoritmus SPH proto může být vcelku jednoduše paralelizován tím způsobem, že každá částice vykonává svůj kód na odděleném kernelu. Jakmile svůj výpočet ukončí, kernel se uvolní pro výpočet další částice a již vypočtená částice pak čeká na bariéře, až svůj výpočet dokončí všechny částice. Pro přístup k vlastnostem ostatních částic se využívá sdílená paměť.

Algoritmus SPH může být rozdělen do tří kernelů, jak ukazuje obrázek 8. V prvním kernelu počítáme hustotu a tlak. Výpočet tlaku sice závisí na hustotě, tento výpočet se však děje pouze v rámci jedné částice a nemůže tedy dojít k nechtěnému souběhu. V druhém kernelu, jakmile mají všechny částice vypočtenou hustotu a tlak, můžeme společně vypočítat tlakové a viskózní síly. Na závěr ve třetím kernelu provedeme integraci. Sečteme všechny síly včetně sil externích, vypočteme rychlost a aktualizujeme pozice částic dle jejich rychlosti. Zároveň musíme provést kontrolu kolizí částic vůči prostředí, aby částice neprošly skrze stěny či jiné pevné překážky.



Obrázek 8: Diagram algoritmu SPH.

5.4 Kontrola kolizí

Během fáze integrace aktualizujeme rychlost částic a počítáme jejich nové pozice. Výpočet nových pozic závisí na směru a velikosti jejich rychlosti, která se počítá z výslednice sil působících na částice. Tyto síly ovšem zahrnují pouze hydrodynamické síly, které jsme vypočítali v předchozím kernelu a konstantní sílu gravitační. Neuvažujeme nárazové síly, které vzniknou při kolizi částice s pevným tělesem (např. stěnou nádoby). Tyto síly působí jen po velmi krátkou dobu a mají značnou velikost.

Bez započtení nárazových sil bychom nemohli simulovat chování kapalin uvnitř jiných těles a zkoumat tak specifické chování či vznik vln v kapalině. Jelikož však simulace neprobíhá kontinuálně, ale je rozdělena na časové kvanta do simulačních kroků, bylo by značně komplikované zachytit přesný moment vzniku těchto nárazových sil. Proto se používají jiné přístupy, z nichž základní dva dále rozeberu.

5.4.1 Naivní algoritmus

Naivní algoritmus nejprve vypočítá rychlost částice, aktualizuje její pozici a až poté kontroluje kolize s prostředím. Mohou nastat dvě situace:

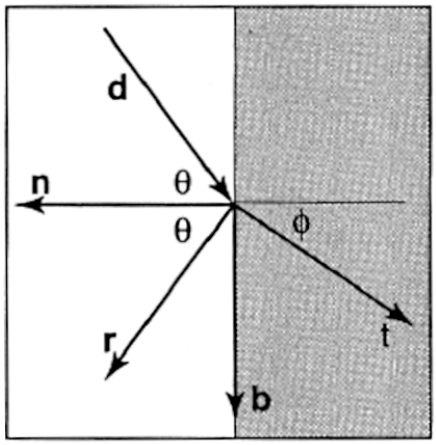
1. Částice nekoliduje s žádným prostředím.
2. Došlo ke kolizi částice a prostředí. Částice buďto částečně proniká cizím pevným tělesem, anebo je uvnitř tělesa celá. Nastává korekce.

Způsoby detekce kolize částice a prostředí se mohou různit dle implementace. Jedním ze způsobů může být analytické řešení pomocí výpočtů nad geometrií. Takové řešení by ovšem při velkém počtu částic, které se obvykle simulace účastní, mohlo vést k velice neefektivnímu algoritmu. Proto bychom měli použít metodu, která nám informaci o kolizi poskytne v konstantním čase. Moje implementace v UE4 například využívá technologie Mesh Distance Fields (vzdálenostní pole)[6], která vytváří volumetrickou texturu, jež v sobě nese informaci o vzdálenosti k nejbližšímu povrchu. Pro body prostoru mimo tělesa obsahuje kladné hodnoty, pro body uvnitř tělesa hodnoty záporné. Jedná se tedy o variantu voxelové mřížky, která se k tomuto účelu dobře hodí.

Korekci při kolizi můžeme provést následujícím způsobem:

1. Zjistíme normálový vektor k povrchu v místě průniku částice s povrchem tělesa.
2. Vypočítáme vektor odrazu z vektoru rychlosti a normálového vektoru.
3. Vektor rychlosti částice nastavíme na vektor odrazu.
4. Částici lineárně interpolujeme mezi její minulou a současnou pozicí tak, aby se nacházela mimo těleso.

Normálový vektor je kolmice k povrchu tělesa v místě průniku. Vektor odrazu můžeme najít z vektoru rychlosti a normálového vektoru. Situaci ve 2D ukazuje obrázek 9:



Obrázek 9: Vektor dopadu $\vec{d}$ a vektor odrazu $\vec{r}$ zrcadlený podle normály $\vec{n}$. Zdroj: https://i.imgur.com/9BHM2.png

Matematicky jej vypočteme vzorcem:

$$\vec{r} = \vec{d} - 2(\vec{d} \cdot \vec{n})\vec{n}, \tag{23}$$

kde $\vec{r}$ je vektor odrazu, $\vec{d}$ je vektor dopadu, $\vec{n}$ je normálový vektor k povrchu a $\vec{d} \cdot \vec{n}$ je skalární součin vektorů. Vektor $\vec{n}$ musí být normalizovaný.

Při reálné kolizi částice s prostředím dojde ke zpomalení částice vlivem nárazu a třecích sil. Proto vektor odrazu můžeme ještě násobit skalárem z intervalu $[0, 1]$, abychom částici po nárazu zpomalili. Částice tak část své kinetické energie předá cizímu tělesu.

Lineární interpolaci částice do oblasti mimo těleso můžeme provést díky vzdálenostnímu poli, resp. voxelové mřížce, která nese informaci o tom, jak hluboko v tělese se částice nachází. Pamatujeme-li si pozici částice v předchozím simulačním kroku, můžeme pozici částice linárně interpolovat mezi těmito dvěma body dle vzdáleností od povrchu tělesa tak, aby hledaná vzdálenost byla nulová.

5.4.2 Ohraničení pomocí částic

Ačkoliv naivní algoritmus plní svůj účel a kolidující částice odráží zpět do volného prostoru, takové řešení problému nevystihuje reálné chování kapaliny dokonale. Pokud bychom potřebovali exaktnější řešení problému, existují k tomuto účelu pokročilejší metody. Jednou z nich je metoda ohraničení pomocí částic (boundary particles). Tato metoda rozlišuje dva typy částic – klasické pohyblivé částice a částice pevné, jejichž účelem je pohyblivé částice pomocí hydrodynamických sil odpuzovat pryč od hranic pevných těles.

Dle implementace metody pak můžeme používat buď tzv. stínové částice (ghost particles), anebo kombinaci repulzivních a dynamických částic. Stínová částice, jak již název napovídá, je zrcadlovým obrazem skutečné částice, která se objeví, pokud se skutečná částice přiblíží moc blízko povrchu tělesa (blíže než je vyhlazovací poloměr). Virtuální stínová částice má stejnou hustotu a tlak jako skutečná částice, ale opačný vektor rychlosti. Díky tomu skutečnou částici odpudí a poté stínová částice zase zmizí. [5] [11]

Repulzivní a dynamické částice oproti tomu svou pozici v čase nemění a zůstávají uvnitř a na povrchu tělesa. Tlak a hustota repulzivních částic se neaktualizuje, tyto částice jsou rozmístěny na povrchu tělesa. Dynamické částice svou hustotu a tlak mění dle výpočtů algoritmu SPH a nachází se pod povrchem tělesa. Kombinací těchto částic lze docílit efektu vhodného odpuzování částic v blízkosti těles. [5] [11]

6 Rekonstrukce povrchu

Abychom mohli zrekonstruovat povrch kapaliny, musíme provést následující postup:

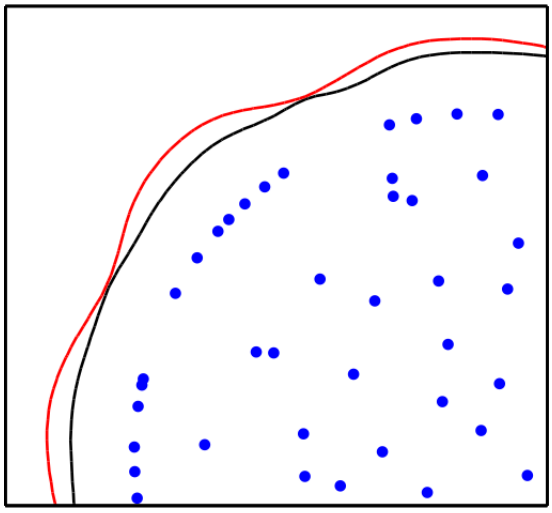
1. Vhodným algoritmem z pozic částic vygenerovat voxelovou mřížku tak, aby hodnoty voxelů udávaly, jak je voxel vzdálený od povrchu kapaliny.

2. Na voxelovou mřížku aplikovat algoritmus pro polygonizaci. V aplikaci jsem použil algoritmus Marching cubes.

Pro první část problému existuje několik řešení, která byla popsána ve vědeckých časopisech. Nejstarší z nich pochází z roku 1982 od Jamese F. Blinna. Dnes ji nazýváme jako tzv. "blobbies"(kuličkovou) rekonstrukci, jelikož na povrchu kapaliny vytváří viditelné kuličkové útvary v místech s vyšší koncentrací částic. Takové chování není nutně vizuálně závadné u nepravidelně tvarovaných objemů, avšak v případě pravidelných tvarů jako rovinná plocha či kužel vytváří na povrchu nechtěné "boule". Z Blinnova řešení vychází většina moderních metod rekonstrukce povrchu kapaliny, jež jsou jeho vylepšením.

Mezi současné populární metody rekonstrukce povrchu kapaliny patří například metoda M. Müllera et al. [13], která oproti Blinnovi přidává určitá vylepšení. Müller et al. navíc vyvažují vliv částic na tvorbu povrchu kapaliny tím, že částice dělí jejich aproximovanou hustotou vypočtenou z algoritmu SPH. Tím částečně redukují vznik kuliček na povrchu kapaliny, nikoliv však úplně.

Výraznější vylepšení metody navrhli Y. Zhu a R. Bridson [18]. Zhu a Bridson se ve svém výzkumu primárně zabývali simulací písku, využili však již existující simulátor vody, který jen mírně upravili. Zvlášť výrazný příspěvek přinesla část jejich práce řešící problém rekonstrukce povrchu z částic, kde srovnávají jejich nově navrženou metodu s již existující blobbies metodou, jak ukazuje obrázek 10:



Obrázek 10: Srovnání metody navržené Zhu a Bridson s blobbies metodou při rekonstrukci povrchu kruhu definovaného částicemi. Blobbies je vnější červená křivka, Zhu a Bridson vnitřní černá křivka [18].

Zhu a Bridson přicházejí s jiným řešením, kdy pro každou částici vypočítávají pole vzdáleností. Pro částici x_0 s poloměrem r_0, definují funkci ϕ takovou že:

$$\phi(x) = |x - x_0| - r_0, \tag{24}$$

kde parametr x je bod v prostoru, jehož vzdálenost od částice x_0 počítáme.

Abychom pole vzdáleností upravili pro rekonstrukci povrchu kapaliny, namísto jediné částice x_0 uvažujeme bod $\bar{x}$, který vypočítáme jako vážený průměr okolních částic, a poloměr $\bar{r}$, který vypočítáme jako vážený průměr jejich poloměrů:

$$\phi(x) = |x - \bar{x}| - \bar{r}, \tag{25}$$

$$\bar{x} = \sum_i w_i x_i, \tag{26}$$

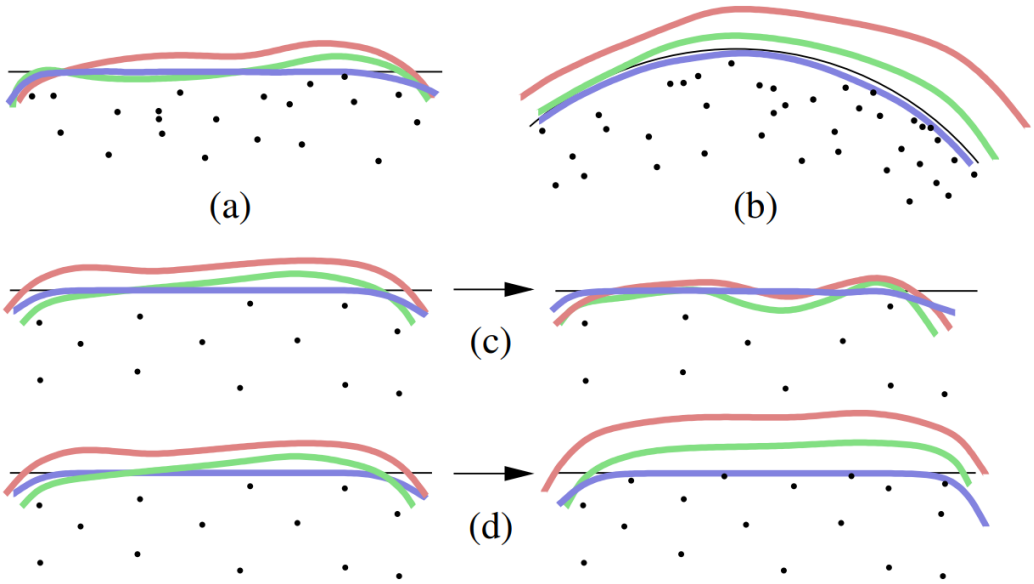
$$\bar{r} = \sum_i w_i r_i, \tag{27}$$

$$w_i = \frac{k(|x - x_i|/R)}{\sum_j k(|x - x_j|/R)}. \tag{28}$$

Funkce k je vhodné vyhlazovací jádro, autoři použili funkci:

$$k(s) = max(0, (1 - s^2)^3). \tag{29}$$

Konstanta R je poloměr pro vyhledání částic v okolí bodu x. Obvykle volíme konstantu R jako dvojnásobek průměrné vzdálenosti mezi částicemi. Správná hodnota této konstanty je poměrně důležitá. Příliš malá hodnota by zahrnovala



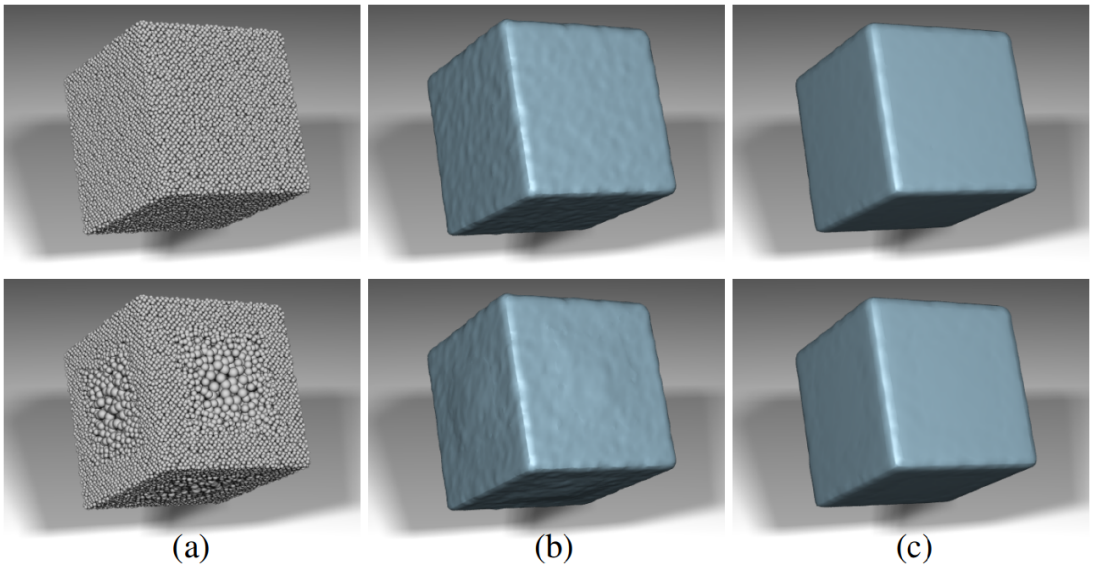
Obrázek 11: Srovnání definice povrchu metodou B. Adams et al (modrá křivka), metodou Zhu a Bridson (zelená křivka) a blobbies metodou (červená křivka). Reprezentovaný povrch je černá křivka. Reprezentujeme rovnou čáru (a) a oblouk kružnice (b). Efekt převzorkování (přidání nebo odebrání částic) ukazují ilustrace (c) a (d) [1].

jen úzký okruh částic, což by vedlo k nepřesným výsledkům rekonstrukce. Příliš velká hodnota by naopak vedla k přesným výsledkům, ale na druhou stranu neúměrně zatížila výpočetní výkon.

Mezi další moderní metody rekonstrukce povrchu patří například metoda B. Adamse et al. [1], anizotropní metoda J. Yu a G. Turk [17], nebo metoda slovenského týmu Onderik et al. [14]. B. Adams et al. srovnávají svou metodu s metodou Zhu a Bridson a blobbies metodou viz obrázek 11 a obrázek 12. J. Yu a G. Turk dále ve své práci vizuálně porovnávají nejznámější metody viz obrázek 13.

Pro potřeby mé aplikace jsem implementoval metodu navrženou Zhu a Bridson. Tato metoda sice ve srovnání s dalšími metodami neposkytuje nejvěrohodnější rekonstrukci povrchu kapaliny, na druhou stranu se ale jedná o metodu poměrně jednoduchou a rychlou. Jelikož jedním z cílů mojí práce bylo umožnit simulaci vody v reálném čase, rychlost algoritmu je tedy klíčová. Vizuální kvalita rekonstrukce touto metodu pro potřeby mé aplikace dostačuje.

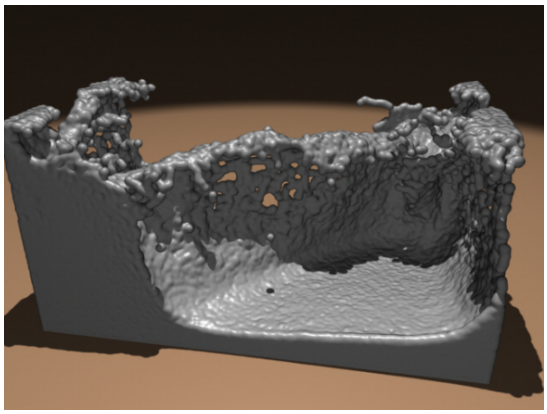
J. Yu a G. Turk ve své práci rovněž analyzují rychlost nejznámějších metod, jak ukazuje tabulka 1.



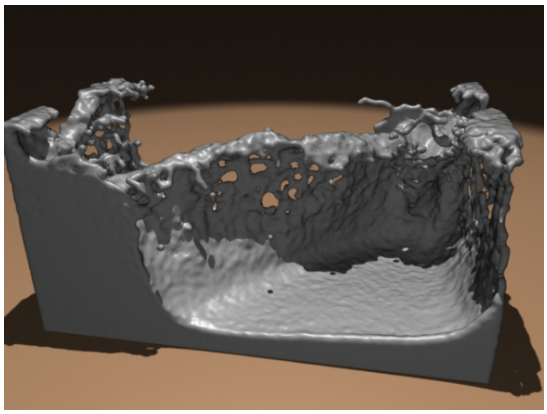
Obrázek 12: Vizuální srovnání rekonstrukce povrchu metodou Zhu a Brison (b) a metodou Adams et. al (c). První řádek ukazuje krychli se 100k částicemi, druhý řádek krychli po adaptivním převzorkování na 20k částic [1].

Metoda rekonstrukce	Průměrný čas rekonstrukce
Blobbies	0.39
Zhu a Bridson	0.50
Adams	1.76
Anisotropic	0.96

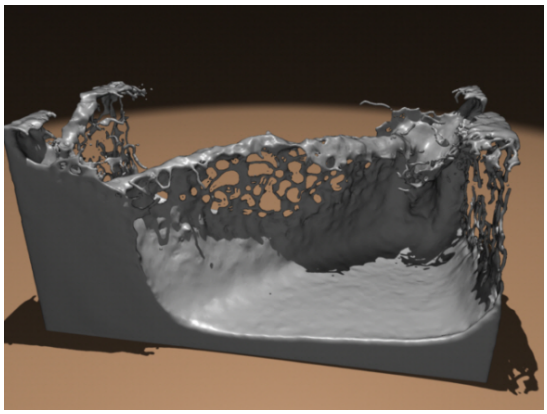
Tabulka 1: Průměrný čas jednoho snímku (v minutách) pro čtyři různé metody rekonstrukce povrchu při simulaci double dam break [17].



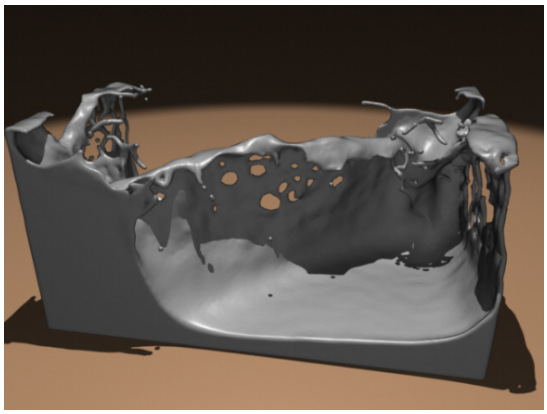
Metoda M. Müller et al. [13]



Metoda Zhu a Bridson [18]



Metoda B. Adams et al. [1]



Metoda J. Yu a G. Turk [17]

Obrázek 13: Vizuální srovnání metod rekonstrukce povrchu [17].

7 Polygonizace

Druhým krokem postupu rekonstrukce povrchu kapaliny je samotná polygonizace. Prvním krokem jsme získali 3D skalární pole (voxelovou mřížku), kde nám hodnoty jednotlivých voxelů udávají, jak je voxel vzdálený od povrchu kapaliny. Z těchto prostorových dat jsme nyní schopni aproximovat povrch kapaliny pomocí polygonů.

Postup polygonizace jsem podrobně vysvětlil ve své bakalářské práci [2], kde se zabývám voxelovým terénem. Problematice je zde věnována celá kapitola pět, ve které rozebírám polygonizaci obecně, algoritmus Marching squares, adaptivní funkci a algoritmus Marching cubes. Stejný algoritmus Marching cubes používám i v případě polygonizace povrchu kapaliny. Proto zde ocituji jen ty nejdůležitější části z této práce a případné zájemce odkazuji na kapitolu pět mojí bakalářské práce.

7.1 Algoritmus Marching cubes

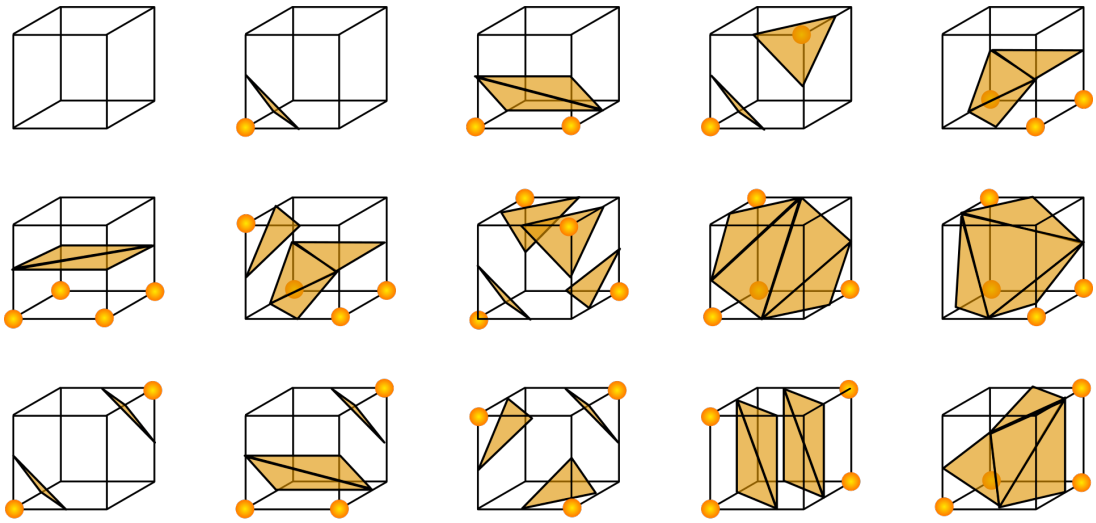
Ve své bakalářské práci nejprve vysvětluji algoritmus Marching squares, který slouží k tvorbě obrysů z 2D skalárního pole. Dále rozebírám algoritmus Marching cubes:

„Algoritmus Marching cubes (česky můžeme volně přeložit jako Pochodující kostky) je 3D varianta algoritmu Marching squares. Slouží k extrakci polygonové síťoviny ze 3D skalárního pole (voxelové mřížky). Tato metoda byla poprvé představena na konferenci SIGGRAPH v roce 1987 [12]. Od té doby prošla několika úpravami a vylepšeními, řešící možné problematické situace, které v algoritmu můžou nastat. Metoda se často používá k vizualizaci medicínských dat získaných pomocí CT nebo MRI." [2]

Využití algoritmu je velmi široké a různorodé. Dnes se často používají také některé jeho vylepšení např. algoritmus Dual Contouring. Pro účely mé práce jsem si nicméně vystačil s původní variantou algoritmu, jak ji popsal William E. Lorensen a Harvey E. Cline [12] a ve vylepšené verzi implementoval Paul Bourke [4]. Marching cubes prochází námi polygonizovaný 3D prostor pomocí krychlí:

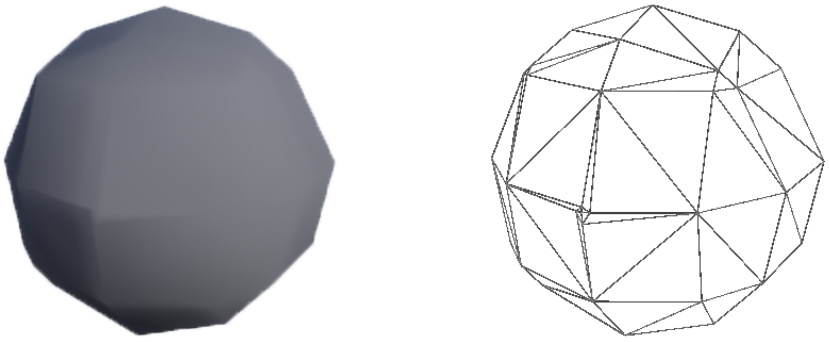
„Zatímco u algoritmu Marching squares jsme procházeli pole po čtvercích stejné velikosti, ve 3D prostoru budeme pole procházet pomocí krychlí. V případě, že všechny vrcholy krychle leží uvnitř objemu, nebo mimo objem voxelů, neprochází touto krychlí žádný polygon povrchu objektu. V opačném případě krychlí prochází jeden či více polygonů, které chceme vygenerovat. V závislosti na vrcholech, které jsou uvnitř nebo vně objektu, můžou nastat různé kombinace průchodů krychle polygony. Ve 2D (čtverec má 4 vrcholy) těchto kombinací bylo $2^4 = 16$*. Krychle má ovšem 8 vrcholů a ve 3D tedy těchto možných kombinací máme celkem* $2^8 = 256$*. Většina kombinací je ovšem pouze rotace nebo zrcadlení (nebo oboje) jiných případů. Pokud odstraníme redundantní případy, získáme minimální tabulku 15 možných případů."* [2]

Tabulka 15 možných případů nám ukazuje základní konfigurace, které mohou při průniku krychle a povrchu nastat:



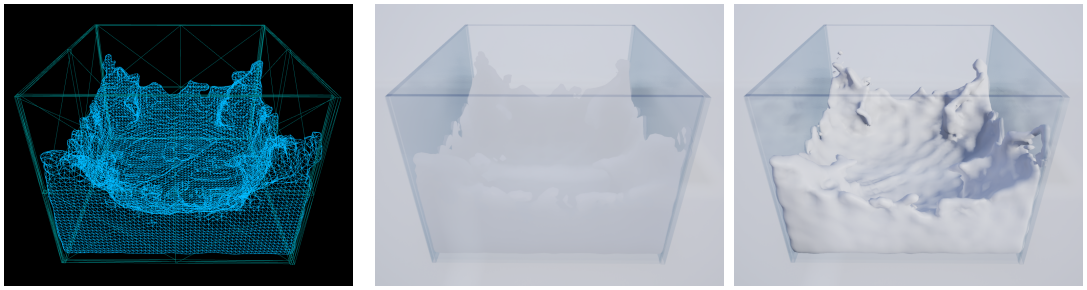
Obrázek 14: Minimální tabulka kombinací algoritmu Marching cubes [2].

Výsledkem algoritmu aplikovaného při rekonstrukci koule je polygonová síť aproximující tuto kouli. Díky adaptační funkci jsou vrcholy polygonů rozmístěny tak, aby co nejlépe aproximovaly rekonstruovaný objekt:



Obrázek 15: Rekonstrukce koule pomocí Marching cubes s adaptační funkcí [2].

Polygonizace povrchu kapaliny v průběhu simulace metody SPH může vypadat následovně:



Polygonová geometrie | Geometrie bez normál | Geometrie s normálami

Obrázek 16: Polygonizovaný povrch kapaliny v průběhu simulace.

Kromě polygonové struktury, skládající se z vrcholů a polygonů, je důležité spočítat ještě normálové vektory daných vrcholů. Díky nim může proběhnout korektní stínování povrchu kapaliny a kapalina tak začne vypadat realisticky. Normálové vektory vrcholů můžeme vypočítat přímo z voxelové mřížky. Pro každý voxel spočítáme jeho normálový vektor z hodnot sousedních voxelů v každé ose:

$$\begin{aligned}\vec{n}(x,y,z) = &(v(x+1,y,z) - v(x-1,y,z),\\ &(v(x,y+1,z) - v(x,y+1,z),\\ &(v(x,y,z+1) - v(x,y,z+1)),\end{aligned} \tag{30}$$

kde $\vec{n}(x,y,z)$ je normálový vektor voxelu na souřadnicích (x,y,z) a $v(a,b,c)$ je hodnota voxelu na souřadnících (a,b,c). Hodnota voxelu je zde skalár udávající nejmenší vzdálenost k povrchu kapaliny.

Normálové vektory jednotlivých vrcholů pak můžeme spočítat během algoritmu Marching cubes interpolací normálových vektorů voxelů podle polohy daného vrcholu. Díky tomuto postupu můžeme velmi snadno a efektivně získat přibližné normály vrcholů a zlepšit tak stínování kapaliny.

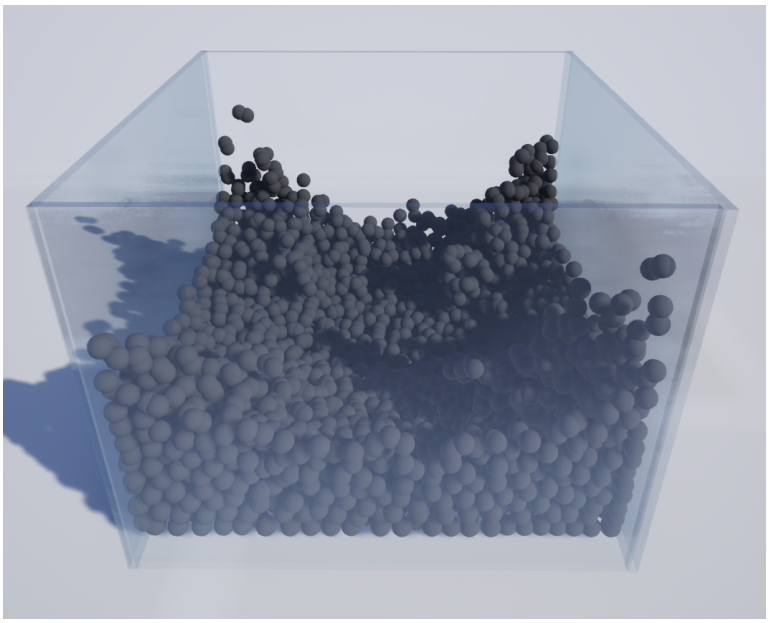
8 Vykreslování kapalin

Důležitým krokem při simulaci kapalin je samotné vykreslování kapaliny. Díky němu můžeme graficky správně interpretovat výsledky výpočetního kroku simulace a zobrazit je uživateli jako projekci na 2D monitor. V zásadě můžeme simulaci buď před-vykreslit, anebo vykreslovat v reálném čase. Moje aplikace používá vykreslování v reálném čase. Základní způsoby vykreslování v reálném čase jsou vykreslení pomocí bodů, vykreslení objemu a vykreslování polygonů.

8.1 Bodové vykreslování

Nejjednodušším způsobem je částice vykreslit jako body. Takové vykreslování je zároveň velmi rychlé a vhodné pro účely ladění. Protože body nejsou prostorovými tělesy, každé částici definujeme určitý poloměr, kolem kterého bude

vykreslena koule. Jak se během simulace mění poloha částice, vizuální reprezentace v podobě koule se posouvá spolu s ní. Díky tomu můžeme snadno sledovat chování jednotlivých částic a aplikaci lépe ladit. Nevýhodou tohoto typu zobrazení je, že výsledek není přílíš realistický viz obrázek 17.



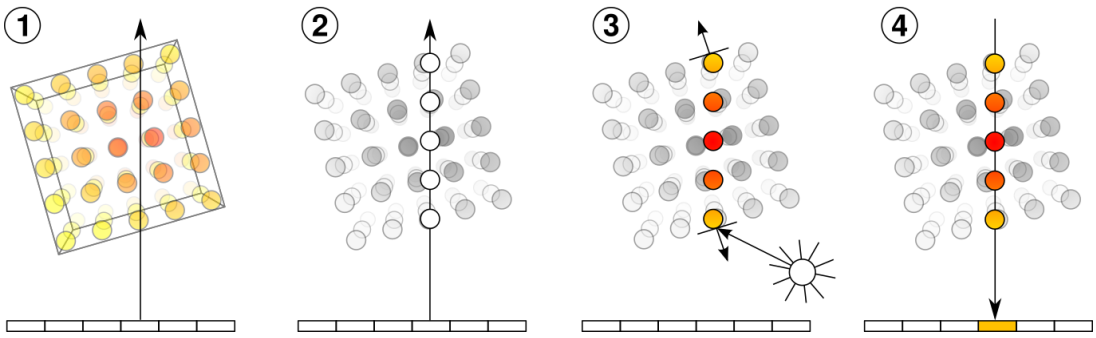
Obrázek 17: Bodové vykreslování simulace.

8.2 Objemové vykreslování

Další možností vykreslování kapaliny je použití objemových technik vykreslování. Tyto techniky jako metoda Volumetric ray casting zpracovávají objemová data (voxelovou mřížku) do 2D snímků, které mohou být zobrazeny na monitoru. V principu se jedná o způsob vyzařování paprsků, které vycházejí z výsledného obrázku směrem k vykreslovanému objemu.

Vykreslování objemu má následující kroky:

1. Pro každý pixel výsledného obrázku vyzařujeme paprsek do sledovaného prostoru skrze určitý objem.
2. Podél paprsku odebíráme vzorky objemu pomocí polynomiální interpolace voxelové mřížky.
3. Pro každý vzorek zjišťujeme barvu, průhlednost a osvětlení objemu v místě vzorku.
4. Zpětně skládáme vzorky od posledního k prvnímu a jejich složením získáme výslednou barvu zjišťovaného pixelu.



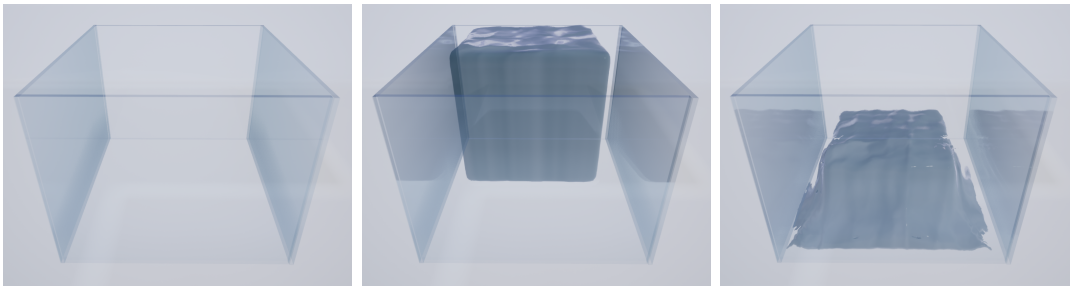
Obrázek 18: Čtyři fáze metody Volumetric ray casting [9].

Ačkoliv vykreslování objemu metodou vyzařování paprsků přináší vizuálně vynikající a přesné výsledky, pro většinu aplikací běžící v reálném čase se příliš nehodí. Stačí si uvědomit, že současné monitory mají přes dva miliony pixelů a pro každý pixel potřebujeme odebrat desítky či stovky vzorků. Existují sice pokročilé adaptivní techniky, které celý tento proces výrazně optimalizují a zrychlují, ale pro většinu běžných aplikací je stále výhodnější vykreslovat polygonové modely. Objemové vykreslování nachází využití ve specifických vědeckých či medicínských aplikacích.

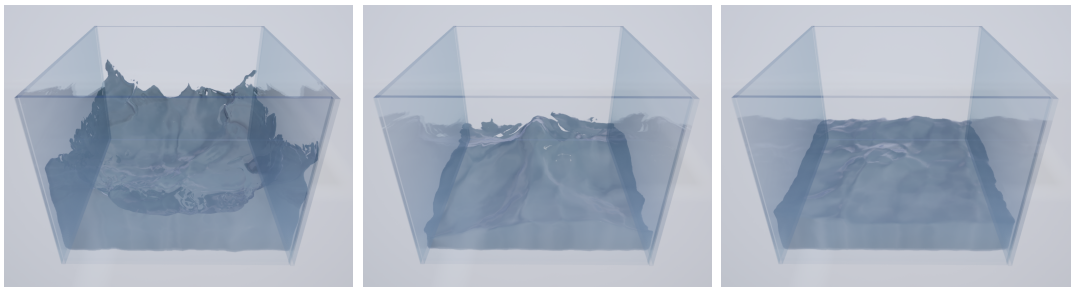
8.3 Polygonové vykreslování

Zatřetí můžeme kapalinu vykreslovat pomocí standardních technik pro polygonové síťoviny. Jak už jsem popsal v předchozích kapitolách, na kapalinu je nejprve nutné aplikovat algoritmy pro rekonstrukci povrchu a polygonizaci. I když tyto algoritmy stojí určitý výpočetní výkon, výslednou vygenerovanou síťovinu už pak můžeme efektivně vykreslovat. Navíc vykreslování polygonové reprezentace považujeme za standardní techniku, kterou implementuje naprostá většina herních engine i grafických programů. Díky polygonové reprezentaci se tak simulaci otevírá širší využití.

Pro výsledný vzhled kapaliny hraje důležitou roli kromě samotné polygonové struktury také její materiál. V případě průsvitných kapalin velmi záleží na průhlednosti a refrakci. Průhlednost v kombinaci s refrakcí dodává kapalině typickou vlastnost, kdy skrze ni vidíme. Zároveň dochází k lomu světla vlivem hustšího prostředí a následnému zkreslení obrazu, který se nachází za kapalinou. Správně nastavený materiál kombinující barevnost, průhlednost i refrakci pak dodá simulaci kapaliny realistický vzhled.



$t = 0s$ $t = 1s$ $t = 2s$



$t = 3s$ $t = 4s$ $t = 5s$

Obrázek 19: Polygonové vykreslování s aplikovaným materiálem v průběhu simulace.

9 Interakce s terénem

Ve svojí bakalářské práci jsem se zabýval voxelovým terénem, který používá podobnou reprezentaci a techniky jako částicová simulace kapaliny. Jak voxelový terén, tak částicová simulace kapaliny pracuje s objemovou reprezentací a využívá voxelovou mřížku. Částicová simulace kapaliny je oproti voxelovému terénu výrazně dynamická, její výpočty musí probíhat v každém simulačním kroku. Voxelový terén oproti tomu stačí aktualizovat pouze v případě, kdy došlo ke změně ve voxelové struktuře. Algoritmy nad voxelovým terénem mohou být asynchronní, a tudíž nezpůsobují zpomalení herní smyčky. Simulační krok kapaliny je synchronní, většinou jej provádíme přímo ve vykreslovací smyčce, aby každý snímek zobrazoval aktualizovaný stav kapaliny.

I přes tyto rozdíly může částicová simulace s voxelovým terénem interagovat a to právě díky společnému voxelovému základu. Každá částice simulace kapaliny má k dispozici přístup do voxelové mřížky terénu pro čtení a může z ní zjišťovat potřebné údaje pro simulaci. Pokud se tedy částice svou pozicí a vektorem rychlosti blíží pod povrch terénu, můžeme adekvátně zakročit detekcí a řešením kolize. Částice se tak od terénu odrazí a výsledná simulace realizuje tok kapaliny po povrchu terénu.

Částicové simulaci kapaliny stačí k voxelové mřížce terénu přístup pro čtení. Není zde důvod, aby kapalina měnila strukturu terénu, snad jen kdybychom uvažovali nějakou formu hydroeroze nebo výbušnou či leptavou kapalinu. Tyto

případy však neberu v úvahu. Jelikož simulace kapaliny probíhá na GPU, přístup k voxelové mřížce terénu, která je uložena v RAM, může být velmi pomalý. Řešením je buď přesunout výpočty voxelového terénu taktéž na GPU, anebo úpravy ve voxelové struktuře terénu přenášet pomocí technologie DMA transfers.

Simulace kapaliny tak podporuje voxelový terén jako svou volitelnou závislost. Oba systémy jsou modulární a mohou fungovat jeden bez druhého. Simulace kapaliny však v případě přítomnosti modulu voxelového terénu detekuje jeho přítomnost a umožní vzájemnou interakci obou systémů.

10 Optimalizace

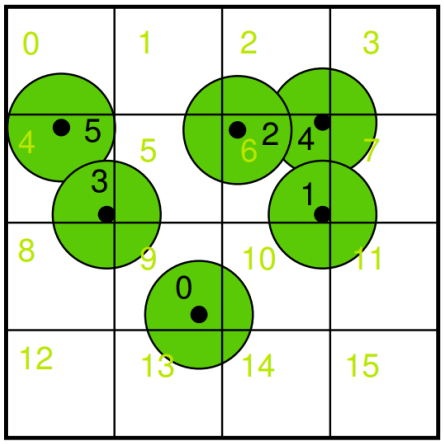
Pro efektivní implementaci simulace v reálném čase jsou důležité optimalizační techniky, které pomáhají k rychlejšímu řešení algoritmů. Možností, jak optimalizovat simulaci kapalin, je celá řada, popíšu zde jen ty nejdůležitější.

10.1 Paralelní výpočty

Díky moderním CPU, která obsahují více jader, můžeme využít paralelní výpočty, jež jdou na simulace kapalin dobře aplikovat. Ještě lepšího přínosu paralelizace můžeme dosáhnout, pokud výpočty přesuneme na GPU, která obsahuje stovky výpočetních jader podporujících paralelní výpočty. Paralelizovat můžeme jak samotnou metodu SPH (za splnění podmínky dodržení bariér mezi kernely), tak rekonstrukci povrchu a polygonizaci pomocí algoritmu Marching cubes.

10.2 Prostorová mřížka

Jednotlivé částice v průběhu simulace interagují s okolními částicemi. Kdybychom neměli k dispozici žádnou pomocnou strukturu pro vyhledávání částic, časová složitost takových algoritmů by byla minimálně v $O(n^2)$, kde n je počet částic v simulaci. Každá částice by totiž musela projít všechny částice v simulaci, aby zjistila, které částice jsou v jejím blízkém okolí a mohla s nimi interagovat. Z datových struktur známe hashovací tabulku, která nám umožňuje vyhledávat podle klíče v konstatním čase. Prostorová mřížka je 3D verze hashovací tabulky, která nám umožňuje roztřídit částice do uniformních bloků prostoru tak, abychom je následně mohli snadno vyhledat.



Obrázek 20: Roztřídění částic v 2D verzi mřížky [7].

Protorová mřížka rozděluje prostor na stejně velké krychle (buňky), do kterých jsou částice roztříděny. Každá částice může v jednom okamžiku patřit jen do jedné buňky. Vyhledávací algoritmus pak prochází nejen buňku aktuální částice, ale i sousední buňky až do vzdálenosti vyhlazovacího poloměru. Částicím je přiřazen hash kód podle jejich prostorových souřadnic, který každou částici identifikuje s konkrétní buňkou. Algoritmus třídění a vyhledávání částic v mřížce vysvětluje Simon Green ve své publikaci [7].

10.3 Efektivní výpočty

Další možností optimalizace je využívat efektivní výpočty. Je známo, že některé matematické operace jsou řádově dražší než jiné. Při prostorových výpočtech často pracujeme se vzdáleností mezi dvěma body. Pokud bychom chtěli vypočítat vzdálenost mezi body $A(x_1, y_1, z_1)$ a $B(x_2, y_2, z_2)$, použili bychom vzorec:

$$|AB| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{31}$$

Operace odmocnění však patří mezi výpočetně poměrně drahé operace. Pokud bychom ji počítali v desítkách či stovkách iterací, na výkonu bychom to téměř nijak nepoznali, ovšem miliony iterací s odmocninou v každém snímku už výkon výrazně sníží. Z tohoto důvodu, pokud to výpočet přímo nevyžaduje, můžeme namísto vzdálenosti dvou bodů pracovat s jejich umocněnou vzdáleností:

$$|AB|^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2. \tag{32}$$

Další možností, jak optimalizovat výpočty, je předpočítání konstantních hodnot či předem známých situací, jež mohou nastat. Např. funkce polynomiálního vyhlazovacího kernelu:

$$W_{Poly6}(r, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \leq r \leq h \\ 0 & \text{jinak} \end{cases}, \tag{33}$$

obsahuje číslo:

$$k = \frac{315}{64\pi h^9}, \tag{34}$$

kde vyhlazovací poloměr h je během celého výpočtu konstantní, a proto i číslo k je konstanta. Stačí jej předpočítat jednou na začátku programu a ušetřit tak procesorový čas v iteracích.

Stejně tak algoritmus Marching cubes může využít předpočítanou tabulku možných konfigurací průniku povrchu, aby tyto situace nemusel počítat vždy znovu. Díky statickému poli pak danou konfiguraci vyhledá v konstantním čase a ušetří tak procesorový čas.

11 Uživatelská příručka

Přiložené DVD obsahuje aplikaci VoxelWaterTerrain pro operační systém MS Windows, která demonstruje výsledky této práce. Samotná aplikace ke svému běhu potřebuje knihovnu DirectX, verzi alespoň 10. Doporučený minimální hardware je procesor AMD Ryzen 5 3600 a grafická karta NVIDIA GTX 1050 Ti. Všechny prerekvizity by aplikace měla nainstalovat při svém prvotním spuštění (vyžaduje úroveň oprávnění administrátora).

Po spuštění aplikace se uživatel ocitá v menu, které obsahuje tři módy, informace o aplikaci a ukončení aplikace. První mód umožňuje procházení po voxelovém terénu z pohledu první osoby, včetně kolizí a štětců pro editování terénu. Druhý mód vizualizuje algoritmus pro simulaci kapalin. Třetí mód demonstruje voxelový terén s interakci s částicovou simulací vody.

Ovládání aplikace je následující:

`W, S, A, D` - pohyb v 3D prostoru
`Levé tlačítko myši` - přidávání voxelů štětcem
`Pravé tlačítko myši` - odebírání voxelů štětcem
`Kolečko myši` - změna prostorového štětce
`Levý shift` + kolečko myši - změna velikosti štětce
`Tlačítka +/- na numerické klávesnici` - změna velikosti štětce
`Esc` - návrat do menu

Závěr

Diplomová práce podrobně prošla problematiku částicových simulací kapalin. V úvodní části práce jsem navázal na výsledky své bakalářské práce, která se týkala voxelového terénu, s nímž diplomová práce souvisí. Zvláštní pozornost jsem zaměřil na popis chování nestlačitelné newtonovské kapaliny, a to jak z fyzikálního, tak i matematického a aplikačního přístupu.

Dále jsem analyzoval metodu Smoothed particle hydrodynamics, která patří mezi nejpoužívanější metody moderního simulování kapalin. Předložil jsem zde matematické vzorce a výpočty nutné ke správnému chování částic kapaliny s ohledem na fyzikální zákony reálného světa. Popsal jsem algoritmus implementující tuto metodu a jeho případné optimalizace.

Významná část práce byla zaměřena na rekonstrukci povrchu kapaliny, která je pro správnou vizualizaci výsledků klíčová. Srovnávám zde několik současných řešení dle různých publikací i s ohledem na kvalitu a výkon předložených řešení. Teoretické poznatky doprovázím obrázky z vlastní aplikace, která implementuje navržená řešení. Praktická implementační část práce zpracovaná v moderní technologii herního engine Unreal Engine 4 je přílohou této práce.

Conclusions

Master's thesis went in detail through the issues concerning fluid particle simulations. In introduction I followed up my bachelor's thesis results which had been dealing with voxel terrain that is related to the master's thesis. I focused more closely on the description of incompressible Newtonian fluid behaviour from physical as well as mathematical and application approach.

Furthermore I analysed the Smoothed particle hydrodynamics method which belongs to the most commonly used modern fluid simulation methods. In my work I presented mathematical formulas and calculations which are necessary for the correct behaviour of fluid particles due to real world physical laws. I described the algorithm which implements this method and its possible optimalizations.

Significant part of my thesis concentrated on fluid surface reconstruction which is crucial for the correct visualization of the results. I have compared several modern solutions based on various publications taking into account their quality and performance. Theoretical knowledges have been accompanied by pictures from my own application which implements the proposed solutions. Practical implementation part of the thesis which is processed in the modern game engine technology Unreal Engine 4 is attached to the thesis.

A Obsah přiloženého CD/DVD

Přiložené CD/DVD obsahuje:

`bin/`
: Aplikaci VoxelWaterTerrain pro operační systém MS Windows, spustitelnou přímo z média. Aplikace vyžaduje knihovnu DirectX, verzi alespoň 10. Doporučený minimální hardware je procesor AMD Ryzen 5 3600 a grafická karta NVIDIA GTX 1050 Ti. Instalace prerekvizit proběhne při prvním spuštění aplikace (vyžaduje úroveň oprávnění administrátora).

`doc/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

`src/`
: Zdrojové kódy a projekt v Unreal Engine 4.22, ze kterého byla aplikace sestavena.

`readme.txt`
: Instrukce pro spuštění aplikace VoxelWaterTerrain, včetně všech požadavků pro její bezproblémový provoz.

Literatura

[1] ADAMS B., PAULY M., KEISER R., GUIBAS L. J. Adaptively sampled particle fluids. ACM Trans. Graph. 26, 3 (2007), 48.

[2] BAZALA, Daniel. Voxelový terén a nástroje jeho editování. Univerzita Palackého v Olomouci (2020).

[3] BLINN J.: A generalization of algebraic surface drawing. ACM Transactions on Graphics (TOG) 1, 3 (1982), pp. 235–256.

[4] BOURKE, Paul. Polygonising a scalar field [online]. (1994) [cit. 2022-05-04]. Dostupné z: http://paulbourke.net/geometry/polygonise/

[5] CRESPO A. J. C., GÓMEZ-GESTEIRA M., DALRYMPLE R. A.: Boundary conditions generated by dynamic particles in sph methods, Computers, Materials & Continua, vol. 5, no.3 (2007), pp. 173–184. [cit. 2022-05-04]. Dostupné z: https://www.techscience.com/cmc/v5n3/23429/pdf

[6] EPIC GAMES, INC, MeshDistanceFields, Unreal Engine 4 Documentation [online] [cit. 2022-05-04] Dostupné z: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LightingAndShadows/MeshDistanceFields/

[7] GREEN Simon: Particle Simulation using CUDA. NVIDIA Corporation (2010). [cit. 2022-05-04] Dostupné z: https://developer.download.nvidia.com/assets/cuda/files/particles.pdf

[8] GREEN Simon: Fluid Simulation in Alice: Madness Returns. (2011) Dostupné z: https://developer.nvidia.com/content/fluid-simulation-alice-madness-returns

[9] HOFMANN Florian: Volume ray casting. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/7/76/Volume_ray_casting.svg

[10] LIND Steven J., ROGERS Benedict D., STANSBY Peter K. Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling Proc. R. Soc. A.47620190801201908O1 (2020) [cit. 2022-04-21] Dostupné z: http://doi.org/10.1098/rspa.2019.0801

[11] LOBOVSKÝ Libor, KREN Jirí: Smoothed particle hydrodynamics modelling of fluids and solid. (2007). University of West Bohemia. Plzeň. [cit. 2022-05-04]. Dostupné z: https://www.kme.zcu.cz/acm/old_acm/full_papers/acm_vol1no2_p062.pdf

[12] LORENSEN, William E., CLINE, Harvey E. Marching cubes: A high resolution 3D surface construction algorithm. ACM SIG-GRAPH Computer Graphics

[online]. (1987) [cit. 2022-05-04]. Dostupné z: https://dblp.org/db/conf/siggraph/siggraph1987

[13] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In SCA'03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[14] ONDERIK Juraj, CHLÁDEK Michal, ĎURIKOVIč Roman. SPH with small scale details and improved surface reconstruction. Proceedings - SCCG 2011: 27th Spring Conference on Computer Graphics (2011). 29-36. 10.1145/2461217.2461224.

[15] RAPP Bastian E.: Microfluidics: Modelling, Mechanics and Mathematics (2017). ISBN 978-1-4557-3141-1

[16] WHITE, Frank. Viscous Fluid Flow. 3rd Edition. McGraw-Hill Mechanical Engineering (1991). ISBN-10: 0072402318.

[17] YU, J., AND TURK, G. Reconstructing surfaces of particlebased fluids using anisotropic kernels. In Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2010), SCA '10, pp. 217–225. 2, 4

[18] ZHU Y., BRIDSON R.: Animating sand as a fluid. In ACM SIGGRAPH 2005 Papers (2005), ACM, p. 972.

[19] ŽÁRA, J.; BENEŠ, B.; SOCHOR, J.; FELKEL, P. Moderní počítačová grafika. Vyd. 2. Brno: Computer Press, a.s. (2010). ISBN 80-251-0454-0.