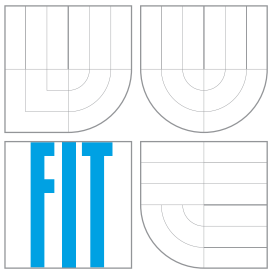
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SIMULACE KAPALIN VE 2D

SIMULATION OF FLUIDS IN 2D

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VLASTIMIL PAZDERA

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Dr. Ing. PAVEL ZEMČÍK,

BRNO 2016

Zadání bakalářské práce/18936/2015/xpazde13

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Pazdera Vlastimil**
Obor: Informační technologie
Téma: **Simulace kapalin ve 2D**
Simulation of Fluids in 2D
Kategorie: Počítačová grafika

Pokyny:

1. Nastudujte dostupnou literaturu na téma simulace kapalin se zaměřením na simulaci ve 2D (v řezu).
2. Vyberte vhodnou metodu pro simulaci kapalin, která by byla implementovatelná i na mobilním zařízení.
3. Navrhněte způsob implementace vybrané metody a diskutujte její vlastnosti.
4. Implementujte metodu a demonstrujte na vhodném příkladě.
5. Diskutujte dosažené vlastnosti a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího práce

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Zemčík Pavel, prof. Dr. Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

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

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá simulací kapalin ve 2D řezu na mobilních zařízeních pomocí techniky Smoothed Particle Hydrodynamics implementovaném v herním enginu Unity. Výsledný program je použitelný na mobilních zařízeních jako stavební prvek her a interaktivních aplikací. Pomocí parametrů lze měnit vlastnosti kapaliny jako např. viskozita. Práce se zaměřuje na co největší využitelnost v mobilních aplikacích a zohledňuje požadavky a limity zařízení k dosažení co nejlepších vizuálních a fyzikálních vlastností kapaliny.

Abstract

This thesis deals with the simulation of fluids in 2D cut for mobile devices using technique Smoothed Particle Hydrodynamics implemented in game engine Unity. The resulting program is usable on mobile devices as a structural element of the games and interactive applications. With the parameters you can alter the properties of liquids for example viscosity. The work focuses on the greatest applicability in mobile applications and takes into account the requirements and limits of the devices to achieve the best visual and physical properties of the simulated liquid.

Klíčová slova

Simulace kapalin v reálném čase, simulace ve 2D řezu, interaktivní simulace, fyzikální simulace, částicový systém, Smoothed Particle Hydrodynamics.

Keywords

Simulation of fluids real-time, interactive simulation, physical simulation, simulation in 2D, particle system, Smoothed Particle Hydrodynamics.

Citace

Vlastimil Pazdera: Simulace kapalin ve 2D, bakalářská práce, Brno, FIT VUT v Brně, 2016

Simulace kapalin ve 2D

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vlastimil Pazdera
18. května 2016

Poděkování

Na tomto místě bych rád poděkoval prof. Ing. Pavlu Zemčíkovi, Dr. za vedení a užitečné rady pro vypracování této práce.

Obsah

Kapitola 1

Úvod

Simulace kapalin ve virtuálním prostředí je složitý úkol, jež vyžaduje mnoho výpočetního výkonu. V současné době má vývoj nových realistických a efektivních technik svoje důležité místo, jelikož se kapaliny a interakce s nimi využívá v počítačových hrách, animovaných filmech nebo v interaktivních aplikacích.

Základní kameny výpočtů dynamiky tekutin položili Claude Navier a George Stokes, když v 19. století nezávisle na sobě zformulovali rovnici popisující dynamiku kapalin. Tato rovnice je též známá jako Navier-Stokesova.

Z této rovnice, která komplexně popisuje chování kapalin, čerpají poznatky i techniky simulace kapalin na počítačích. Od té doby je na jejich simulaci nahlíženo dvěma základními pohledy - Lagrangeovým a Eulerovým. Cílem práce bylo prostudovat oblasti simulace kapalin, jejich možnosti a vybrat takovou, která by byla implementovatelná i na mobilních zařízeních, navrhnout způsob její implementace a dále také vytvořit ukázkovou aplikaci jež bude demonstrovat dosažené výsledky.

V kapitole 2 jsou diskutovány základní techniky, pohledy a možnosti pro simulaci kapalin s různými vlastnostmi. V kapitole 3 je popsán návrh řešení zvolené metody. Kapitola 4 se zabývá implementací zmíněné metody podle předchozího návrhu řešení. V kapitole 5 jsou provedeny testy implementace a shrnuty výsledky práce.

Kapitola 2

Teoretický základ práce

2.1 Co jsou to kapaliny

Jsou to látky, které mohou volně proudit prostorem, ovlivňovány některými silami přítomnými v tomto prostoru. Jedná se především o tekutiny(jako třeba voda), plyn, kouř nebo oheň.

Kapaliny nás obklopují a hrají důležitou roli v každodenním životě. Realističnost zobrazení a interakce s kapalinami může výrazně podpořit realističnost v počítačových hrách, kde je na úkor výkonu zanedbána přesnost simulace, která zde není oproti přesným vědeckým fyzikálním simulací vyžadována. Ačkoli se tyto látky zdají běžné v našem životě, jejich simulace je velmi komplexní a tudíž velmi náročná.

2.2 Vlastnosti kapalin

Všechny kapaliny mají některé společné vlastnosti. Nemají stálý tvar a zaujímají tvar nádoby. Jejich další vlastností je, že se dají přelévat. Jejich molekuly jsou v neuspořádaném pohybu a nejsou pevně vázané [2].

Všechny jevy, ve kterých hrají roli tekutiny, mají společný znak a tím je proudění částic. Se simulací kapalin souvisí i simulování kouře či ohně, kde se částice pohybují daným způsobem ve vzduchu.

2.3 Základní metody reprezentace kapalin

Existují dva základní pohledy na jak lze popsat pohyb kapalin - Eulerův pohled a Lagrangeův pohled. Liší se od sebe jak reprezentací kapalin v prostoru, tak jejich vyhodnocením. Zatímco Eulerův pohled používá k reprezentaci pevnou mřížku nebo pole, Lagrangeův pohled využívá kolekci mezi sebou interagujících částic. Oba pohledy jsou užitečné, často se také kombinují [2].

2.3.1 Eulerův pohled

Eulerova mřížka používá pevné nepohyblivé body ve fixním poli k reprezentaci toku nebo pohybu tekutiny. Přes tyto body se v čase kapalina pohybuje. Pouze v těchto bodech měříme všechny veličiny jako je tlak, hustota, teplota atd.

Často se používá k velmi přesným simulacím toku kapalin, ale je jen málo známých použití v interaktivních aplikacích, kde se pro svoje nevýhody a fixní souřadnice nevyužívá skoro vůbec. Také se zde obtížněji vyhodnocují reakce na kolize s okolními předměty. Metoda má také problémy se zachováním hmoty.

2.3.2 Lagrangeův pohled

Tento pohled využívá k reprezentaci tekutiny pohyblivé body, které se mohou volně pohybovat prostorem a nejsou vázány žádnou mřížkou. Těmto bodům se říká částice. Každá částice reprezentuje částečku kapaliny o určité hmotnosti. Kapalina je tak tvořena množinou částic pohybujících se v prostoru a ovlivňujících se navzájem. Každá částice také nese údaje o své pozici a rychlosti. Jiné vlastnosti těchto částic a celé kapaliny se dají získat z vyhodnocování dráhy jejich pohybu [2].

Největší nevýhoda částicových simulací je obtížná reprezentace hladkého povrchu kapalin. Toho lze však dosáhnout použití většího množství drobnějších částic, což činí simuaci náročnější na výkon. Obtížně se také dosahuje povrchového napětí.

2.4 Smoothed particle hydrodynamics

Smoothed particle hydrodynamics (dále jen SPH) je technika pro simulaci toku kapalin, založená na Lagrangeově pohledu využívající množinu částic k jejich reprezentaci. Vyvinuli ji Gingold, Monaghan and Lucy v roce 1977 původně k řešení a simulaci složitých astrofyzikálních problémů jak je uvedeno v publikaci [3], lze ji však použít i k simulaci kapalin všeho druhu.

Jedná se o druh řešení nestlačitelné Navier-Stokesovy rovnice kapaliny. Metoda pracuje na principu konvolučního filtrování, kdy počítáme s parametry jednotlivých částic a jejich blízkých sousedních částic a pomocí tzv. vyhlazovacích jader (smoothing kernels) interpoluje veličiny, které při sledování drah částic v prostoru těžko určíme. Pomocí těchto interpolací lze vypočítat tlak, hustotu a další atributy v místě částice. Podle vypočtených hodnot se upravuje chování a parametry částic. Jak je uvedeno v práci [1] praktických metod řešení SPH je mnoho.

Kapitola 3

Návrh řešení

Následující kapitola formuluje problémy a nastiňuje návrh jejich řešení. Uvádí také zvolené metody a postupy pro implementaci.

3.1 Formulace problému

Cílem je realisticky vypadající simulace kapalin na hardware mobilních zařízení ideálně alespoň při 60 snímcích za vteřinu. Simulovat se budou pouze viskózní tekutiny jako je voda, sirup atp. Fyzikální výsledky nemusí být zcela přesné, interaktivita a výpočetní jednoduchost aplikace bude upřednostněna nad přesností, která pro tyto účely není vyžadována. Cílem je též dosáhnou uspokojivé realističnosti výsledného zobrazení povrchu kapaliny.

Bude tedy nutné zvolit metodu, která by byla implementovatelná i na mobilních zařízeních a splňovala všechny výše uvedené požadavky. Všechny studované algoritmy jsou navrženy a vyhodnocovány ve 3D prostoru, je však možné je využít bez problémů i pro 2D prostor.

Dále bude nutné zvolit vhodný engine a metodu vykreslení povrchu částic jako kapaliny.

3.2 Volba algoritmu

K řešení problému byl zvolena metoda Smoothed particle hydrodynamics založená na reprezentaci kapaliny pomocí částic. Volba této metody umožňuje jednoduše a tvárně upravovat vlastnosti kapaliny parametry a změnou koeficientů Lze ji také použít na neomezené velikost scény. Poskytuje nejen realistické zobrazení toku ale i hladiny kapaliny.

Na kvalitu simulace bude mít vliv také počet částic, a proto bude třeba dosáhnout co největšího počtu částic ve scéně za rozumnou cenu výkonu. Při vytváření obsahu scény a náplně hry je však také nutné počítat se zatížením scény výpočty ostatních objektů, proto bude počet částic při využití kapaliny ve scéně s mnoha výpočetně složitými objekty ještě menší než při samotné simulaci v prázdné demonstrační scéně.

3.2.1 Popis algoritmu

Jedno z vhodných řešení je založeno na algoritmu [1], kterým lze simulovat tekutiny různé viskozity a dále také simulovat viskoelastické látky jako je třeba pryž nebo želé. Tato práce je však soustředěna na simulaci kapalin, a proto zde viskoelastické chování nebude řešeno.

Metoda přináší stabilní řešení nestlačitelnosti a předchází problému shlukování částic. Klíčová procedura je nazvána *double density relaxation* a je založena na výpočtu dvou

různých hustot v místě částice - jedna při výpočtu zohledňuje počet sousedních částic, a druhá kvantifikuje počet blízkých sousedních částic. Poskytuje též možnost hladké zobrazení hladiny. Hladkost však může být ovlivněna nastavením koeficientů částic. Dále přirozeně dosahuje povrchového napětí, jehož sílu lze také koeficienty ovlivňovat. Stabilita řešení zůstává i při velkých časových intervalech mezi snímky.

Tato metoda využívá schéma tzv. predikce pozice. Částice je v čase posunována tak, že na začátku simulačního kroku je nejprve posunuta do předpokládané pozice, a poté je v průběhu dílčích kroků algoritmu z této pozice postupně posunována do cílové pozice pomocí získaných impulzů. Ty se postupně aplikují na atributy částice. Na konci simulačního kroku je tak částice umístěna na pozici, kterou získala působením sil sousedů a kolizemi s okolní geometrií.

Místo implementace v trojdimenzionálním prostoru bude upraven do dvou dimenzí. To však nijak vyhodnocování algoritmu neovlivní. Konverze z 3D prostoru na 2D by měla mít pozitivní dopad na náročnost simulace co se týče prostředků.

3.3 Vyhledávání sousedních částic

Hledání všech částic v interakčním rádiu h částice je problémem ve všech částicově založených metodách. Kvadratická složitost obyčejného řešení $O(N)^2$, kdy se porovnávají vzdálenosti každé jednotlivé částice se všemi ostatními je z hlediska výkonu nepřípustná.

Protože se impulzy částice počítají pouze pomocí částic nacházejících se v interakčním rádiu, je nutné provádět vyhledávání každý simulační krok. V takto frekventované úloze je nutné použít jinou strukturu s menší složitostí. Protože rádius h je konstantní a stejný pro každou částici, lze jako řešení tohoto problému využít speciální hashovací tabulku *spatial hash table* o velikosti buňky h.

Prostorová tabulka funguje na principu třídění elementů v prostoru do buněk tabulky. Často se využívá také při zjišťování kolizí.

Nejprve se rozdělí prostor na mřížku čtvercových buněk o hraně čtverce h, a poté se pomocí hashovací funkce zjistí příslušnost objektů do dané buňky. Buňky poté obsahují objekty, které jsou v prostoru blízko u sebe. Pro naše využití se bude porovnávat sousední buňka s právě zpracovávanou částici a 8 okolních buněk. Zmenšuje tak počet porovnání. Hashovaci funkce může vypadat po konverzi na 2D souřadnice například podle zdroje [5] jako:

$$hash(x, y) = (\ x\ *\ 73856093\ xor\ y\ *\ 83492791\ )\ mod\ n \tag{3.1}$$

Vzorec 3.1 vykazuje málo kolizí v mapování buněk v tabulce v případě nekonečného prostoru. Předpokládá se, že vyhledávání sousedních částic bude časově nejsložitější část simulačního kroku. Řešení v prostoru 2D místo ve 3D přináší také výhodu zjednodušení výpočtu, jelikož místo prohledávání 27 buněk s možnými sousedy se jich bude prohledávat pouze 9. Na mobilních zařízeních je toto usnadnění velmi vítáno.

Hlavní nevýhodou hashovací tabulky je fakt, že se bude muset během každého simulačního kroku ustavená tabulka vyprázdnit a znovu naplnit roztříděnými částicemi do buněk podle hashovací funkce. Jak se pozice částice v čase mění, částice cestuje z jedné buňky do druhé. Tuto změnu je proto nutné reflektovat každý simulační krok. Částice tedy bude přesunovat z jedné buňky do jiné a to je nutné zaznamenat a částici správně zařadit.

Při vyhledání sousedních částic jsou všechny možné nalezeny v buňce s částicí, pro niž je vyhledáváme, a v 8 okolních buňkách.

3.4 Kolize

Reakce na kolize s předměty bude stejně jako v práci [1] založena na impulzu vypočteném mezi částicí a objektem při kolizi. Impulzy ve formě vektorů jsou založeny na kolizním modelu zvaným zero-restitution s mokrým třením.

Výpočet vyžaduje znalost vektoru rychlosti částice $\vec{v_i}$, rychlost objektu v bodu kontaktu s částicí $v_b\vec{ody}$ a normálového vektoru $\vec{n}$ objektu. Z toho je zjištěna relativní rychlost částice $\overline{v} = \vec{v_i} - v_b\vec{ody}$. Ta je rozdělena na normálovou a tangentovou složku. Impulz je zjištěn podle vztahu jako $I = v^{normal} - u \ * v^{tangent}$, kde u je třecí parametr povolující nebo zakazující klouzání po povrchu objektu.

Pokud je vlivem velké rychlosti částice během počítání kolize již pronikla částice dovnitř těla objektu, je na konci vyhodnocení impulzu po jeho aplikaci na rychlost částice extrahována z těla objektu a posunuta do pozice těsně před objekt blízko původního místa kolize.

Vzhledem k žádoucí interakci kapaliny s okolními předměty se proto nabízí možnost využití *spatial hash table* i pro uložení pozice objektů. Vytvořila by se nová tabulka pro objekty, které mohou kolidovat s částicemi. V každé buňce tabulky budou pomocí hashovací funkce uloženy objekty překrývající buňku. Při vyhodnocování kolizí částice s objekty se tak budou porovnávat pouze objekty, které jsou nejblíže částici, tedy v buňce s částicí. Pro každou částici je podle její pozice vyhledána v tabulce buňka s objekty a pro ty jsou vyhodnoceny kolize a spočteny a aplikovány impulzy.

3.5 Vykreslení povrchu kapaliny

Jako metoda vykreslování byla vybrána technika metaballs z důvod své jednoduchosti.

3.6 Volba enginu

Aby šla simulace využít na co nejvíce zařízeních s různými operačními systémy a mohla tak oslovit co nejvíce potencionálních hráčů nebo zákazníků, je volba enginu zásadní. Výhodu mají samozřejmě multiplatformní herní enginy.

Výhodné je při vývoji na mobilní zařízení zvolit multiplatformní herní engine, jež by poskytoval schopnost uniformního vytváření pro více operačních systémů najednou. V takových enginech je možné napsat program v jednom jazyce a bez jakýchkoli větších modifikací vytvářet aplikace na různé druhy zařízení. Zvolit herní engine ve kterém lze dělat na mobilní zařízení se zvoleným operačním systémem. Poskytují tedy širokou škálu zařízení a možných nasazení simulace. Hlavními kandidáty, mezi kterými se rozhoduje jsou herní enginy Unity a Cocos2d.

3.7 Využití více vláken

Další možností jak zvýšit efektivitu výpočtu a množství částic kapaliny ve scéně je navržení algoritmu simulace s využitím paralelních výpočtů. využití paralelních výpočtů více vláken. Jak je vidět v práci [4] při využiti 4 vláken na 4 jádrovém zařízení se zvýší počet částic ve scéně asi 3x. Nicméně je nutné vybrat anebo navrhnout algoritmus tak, aby se nechoval konkurentně ale paralelně. Při využití více vláken je třeba vyřešit problém jejich synchronizace při přístupu ke sdíleným prostředkům. K tomu však existují nástroje v každém moderním programovacím jazyce.

Avšak je nutné cílit aplikaci na zařízení s určitým počtem jader. I když to zvyšuje výkon a realističnost simulace a počet použitých částic ve scéně při stejné snímkové frekvenci, limituje to nasazení aplikace pouze na zařízení s těmito parametry. Vícevláknový program na 2 nebo 1 jádrovém smartphonu může za určitých okolností být spíše na škodu. Je nutno si to rozhodnout v době vytváření hry nebo aplikace a zvolit cílenou skupinu zařízení.

Kapitola 4

Implementace

Následující kapitola popisuje praktickou část práce - vytvoření programu na mobilní zařízení, jenž bude simulovat chování kapaliny v prostoru 2D. Při řešení se vychází z poznatků z kapitol 1 a 2. Kapitola obsahuje základní popis implementace algoritmu pomocí pseudokódu a rozebírá jej na dílčí kroky.

Vývoj probíhal na platformě Windows v herním enginu Unity v jazyce C#. Řešení využívá aplikačního rozhraní enginu, standardních prostředků jazyka C#. Základem každého objektu je zdědění vlastností a metod z bázové třídy MonoBehaviour, jež poskytuje všechny potřebné třídy a prostředky k implementaci v prostředí Unity. Program je řešen objektově-orientovaným paradigmatem. Respektuje uspořádání a návrh virtuální scény při pozdějším používání v aplikacích.

4.1 Unity

Tento herní engine byl zvolen z důvodu jednoduchosti a předchozích pozitivních zkušeností autora. Je základem mnoha herních společností zabývajících se vývojem her převážně na mobilní zařízení a osobní počítače. Umožňuje vývoj her na více jak 20 platforem mezi které patří: osobní počítače, mobilní zařízení, herní konzole, zařízení s virtuální a rozšířenou realitu a platformy využívající WebGL.

Jeho velkou výhodou je také silná komunita vývojářů sdružující se na oficiálním fóru. Na fóru lze čerpat nespočet rad, informací a zkušeností od úspěšných profesionálních vývojářů.

Scéna v Unity editoru se skládá z herních objektů, do nichž lze vkládat tzv. komponenty, které určují chování a vlastnosti herního objektu. Komponentou se rozumí instance skriptu, instance vestavěné komponenty jako např. *Transform* mající schopnost určovat pozici, rotaci a měřítko ve scéně atp. Chování objektů se programově vytváří instanciací tříd řídících skriptů jako komponent do herních objektů. Skripty mohou být vytvořeny v jazycích C# anebo Javascript. Mezi herními objekty lze také vytvořit vztah rodič-potomek. Lze tak vytvářet hierarchie objektů.

4.2 Architektura scény

Celý projekt se skládá ze 4 hlavních herních objektů. Tím prvním je SPH Controller. Je vytvořen jako prázdný herní objekt. Plní roli kontejneru pro instance třídy *sph*, která implementuje hlavní simulační smyčku algoritmu. Také je v něm instance třídy *TouchLogic*, která obsahuje metody pro zpracování vstupu při dotyku obrazovky.

Druhým herním objektem je samotný objekt částice. Obsahuje následující komponenty: Transform, Sprite renderer a instanci třídy *particle*. Třída slouží pro inicializace a jako datový kontejner pro atributy částice. Mezi atributy patří seznam sousedních částic, dále vektory rychlosti, pozice, pozice v předchozím kroku, skaláry pro hustoty a tlaky v okolí částice, koeficient .

Třetím je soustava 2 kamer, které snímají scénu. Mají na starosti vykreslování kapaliny a geometrie scény.

Čtvrtým je čtverec s povrchovým shaderem, který na svém povrchu vykresluje výstup z kamery zpracovaný renderovací texturou, pomocí níž je vykreslována kapalina. Je umístěna ve scéně naproti kamerám.

Všechny jsou uloženy jako tzv. *prefab* - jakési hotové šablony herních objektů s nastavenými komponentami.

Celá hlavní část algoritmu simulace je implementována třídou *sph*. Třída dědí základní chování, možnosti a vlastnosti enginu ze třídy *MonoBehaviour*.

4.3 Algoritmus

Zvolený algoritmus pro implementaci SPH je založen na řešení [1]. Vyhovuje požadavkům na chování a vlastnosti uvedeným v kapitole 2. Díky své jednoduchosti je k tomuto účelu vhodný. Představuje robustní a stabilní metodu výpočtu pro simulaci. Umí ovládat viskózní chování nastavováním parametrů. Také lze snadno získat viskoelastické chování.

Přesto jsou třeba některé modifikace algoritmu. Bude třeba vypustit viskoelastické chování, protože zatěžuje výpočetně systém a není třeba při simulování tekutin jako je voda.

Algoritmus 1: Simulační krok

```
1 aplikování gravitace();
2 nalezení sousedních částic();
3 aplikace viskozity();
4 uložení předchozí pozice();
5 double density relaxation();
6 vyhodnocení vstupů();
7 vyhodnocení kolizí();
8 aktualizace rychlosti();
```

Algoritmus 1 představuje pseudokód, jež popisuje výpočetní části jednoho simulačního kroku z pohledu nejvyšší úrovně programu. Většina funkcí je konstruována cyklem *foreach* iterující přes seznam všech částic a vykonávající nějaký druh výpočtu.

4.3.1 Aplikování gravitace

V tomto kroku se mění vektor rychlosti částice podle gravitace. Gravitace je získána ze statické třídy *Physics2D*. Rychlost částice se modifikuje na základě výpočtu pomocí hodnoty vektoru. Vektor gravitace je součástí nastavení 2D scény editoru. Hodnotu gravitačního vektoru lze během simulace měnit. Tím bylo docíleno zasazení do globálního nastavení scény. Tak může kapalina přirozeně reagovat na změnu gravitace společně s ostatními objekty ve scéně.

4.3.2 Nalezení sousedních částic

V této části jsou pro každou částici nalezeny sousední částice v okruhu h. Ve speciálním konstruktoru třídy *sph* je vytvořena a inicializována hashovací tabulka. Ta je implementována třídou *SpatialTable*. Prostor scény je tedy rozdělen na buňky.

Poté se shromáždí částice z buňky ve které je právě zpracovávaná částice a 8 okolních buněk, které obklopují tuto buňku. Poté jsou pomocí vektorů pozic vypočteny vzdálenosti mezi částicí a jejími možnými sousedními. Pokud soused leží v interakčním rádiu h, jsou přidány do seznamu obsahující sousední částice.

4.3.3 aplikace viskozity

V tomto bodě se aplikuje působení viskozity na částici. Aplikuje se jako vnitřní radiální impulz mezi páry částic. Pro každou částici je spočten impulz síly I. Impulz, jenž má formu dvourozměrného vektoru je počítán ze sousedů, je závislý na vzdálenosti obou částic, je získán lineárním kernelem $(1 - r_{ij}/h)$. Impulz pak modifikuje rychlost obou částic.

4.3.4 Uložení předchozí pozice

Zde se uloží stará pozice částice do atributu třídy *Particle* každé částice. Poté proběhne posun do předpokládané pozice vypočtené podle deltatime a její rychlosti. Neposouvá se však přímo přiřazením do komponenty *Transform*, nýbrž je pouze modifikována atribut *pos*. Testováním bylo zjištěno, že přímý posun přiřazením do komponenty je výpočetně náročnější než modifikace atributu, a proto byl pro tyto účely vytvořen.

4.3.5 Double density relaxation

Klíčový krok který je srdcem celé simulace. Funkce výpočte pseudotlak a pseudohustotu zohledněním sousedních částic. Nejdříve je pro každou částici vypočtena hustota a blízká hustota. Poté je z obou hustot vypočten tlak a blízký tlak. Nakonec je z tlaků vypočten vektor D, který se použije ke změně pozice sousední částice. Všechny D vektory jsou nakumulovány do pomocného vektoru dx a na závěr je částice posunuta na pozice přičtením vektoru dx k vektoru pozice. Také se ihned provede posun částice přiřazením do komponenty.

4.3.6 Vyhodnocení vstupů od uživatele

v této metodě je implementováno vymazání scény testováním statického instanční proměnné typu *input*. Jelikož vstup je asynchronní událost a tak se musí snímat zde, aby nedošlo k vyvolání výjimek a negativnímu ovlivnění simulace. Natavením atributu se zajišťuje bezpečné odstranění objektů částic ze scény.

4.3.7 Vyhodnocení kolizí

Hranice scény jsou pevně dány intervaly pro x <-10,10> a y <-5,5>. Kolize jsou z pohledu sil vyhodnocovány tak, jak je popsáno v sekci 3.4, ale kolize a interakce s předměty nebyly implementovány.

4.3.8 Aktualizace rychlosti

Na závěr je spočtena nová počáteční rychlost do dalšího simulačního kroku. Spočte se odečtením vektoru původní pozice částice od vektoru současné pozice a podělením časem, který uplynul mezi dvěma snímky.

4.4 Vykreslení povrchu kapaliny

K vykreslování scény jsou použity 2 kamery - Hlavní kamera pro vykreslování všech objektů kromě částic a druhá speciální pouze pro vykreslování částic pomocí techniky metaballs.

Hlavní kamera je v hierarchii objektů jako rodič Metaballs kamery. Tak je závislá na souřadnicovém systému rodiče.

Objekty částic jsou umístěny do vrstvy *metaballs*. Vstup kamery pro vykreslování částic je nastaven na vykreslování pouze této vrstvy. Výstup kamery je posílán do renderovací textury. Obraz vzniklý na textuře je poté promítán na čtverec zpracovávaný shaderem *Metaballs*, jenž na něm vytvoří efekt metaballs. Čtverec je umístěn přesně naproti kamerám. Obraz promítaný na čtverec a zpracovávaný jeho shaderem pak vytváří povrch kapaliny v místech, kde se částice pohybují.

4.5 Využití více vláken

Během pokročilých fází práce bylo zjištěno, že aplikační rozhraní Unity není zabezpečeno pro operace více vláken - je thread non-safe. Nelze tudíž přistupovat k aplikačním voláním mimo hlavní vlákno, jinak za běhu program vyvolá výjimku. Výsledný kód není spolehlivý, jestli volání bude v danou chvíli fungovat je dílem náhody. Proto zjednodušení výpočtů pomocí více vláken nemůže být použito. To se negativně odrazí na výkonu.

Kapitola 5

Výsledky a vyhodnocení

Cílem testů bylo posoudit kvalitu simulace, zjistit náročnost aplikace na hardwarový výkon. Další cíl je zjistit hratelné množství částic na průměrném smartphonu. Posuzováno bylo podle snímkové frekvence aplikace. Hodnocení kvality vykreslení povrchu kapaliny bylo posuzované subjektivně. Testy byly prováděny na 4 zařízeních s operačním systémem Android a to:

- Sony Xperia M s 2 jádrovým procesorem o taktu 1 Ghz, 1 GB RAM, operační systém Android verze 4.3.
- Samsung GALAXY S4 mini se 4 jádrovým procesorem o taktu 1 Ghz, 1 GB RAM, operační systém Android verze 4.4.
- Huawei Ascend G6-L11 se 4 jádrovým procesorem Quad-core o taktem 1,2 Ghz, 1,0 GB paměti RAM, operační systém Android verze 4.3.
- Samsung GALAXY S3 Neo se 4 jádrovým procesorem o taktu 1,4 Ghz, 1,5 GB RAM, operační systém Android verze 4.4.

Všechna zařízení patří do kategorie střední až nižší třídy, pro účely testů to však postačuje. Výkonnější zařízení pochopitelně budou vykazovat lepší výsledky a budou schopny simulovat kapalinu detailněji větším počtem částic. Při vývoji aplikací je cílem oslovit co nejširší množství potencionálních uživatelů.

5.1 Chování simulované kapaliny

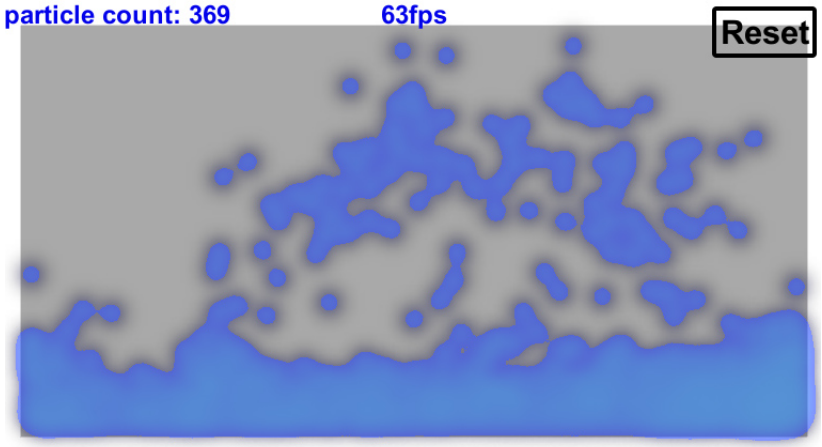
Chování kapaliny lze ovlivnit změnou parametrů simulace.

Při naivním řešení vyhledávání bylo možné simulovat asi 80 částic ve scéně. Po implementaci pomocí hashovací tabulky jež je popsána v kapitole 2, bylo dosaženo řádově lepších výsledků.

5.2 měření množství částic

Jak ukazuje tabulka 5.1, na zařízeních je dosaženo při 60 snímcích za vteřinu jen málo částic. Pokud budeme počítat zařazení do hry nebo aplikace.

Po grafické stránce není simulace moc náročná. V aplikaci je prostor pro rozvoj grafické stránky hry, ale není z hlediska zatížení a využití CPU. Drtivou většinu zdrojů využívá výpočet hlavní smyčky simulace.



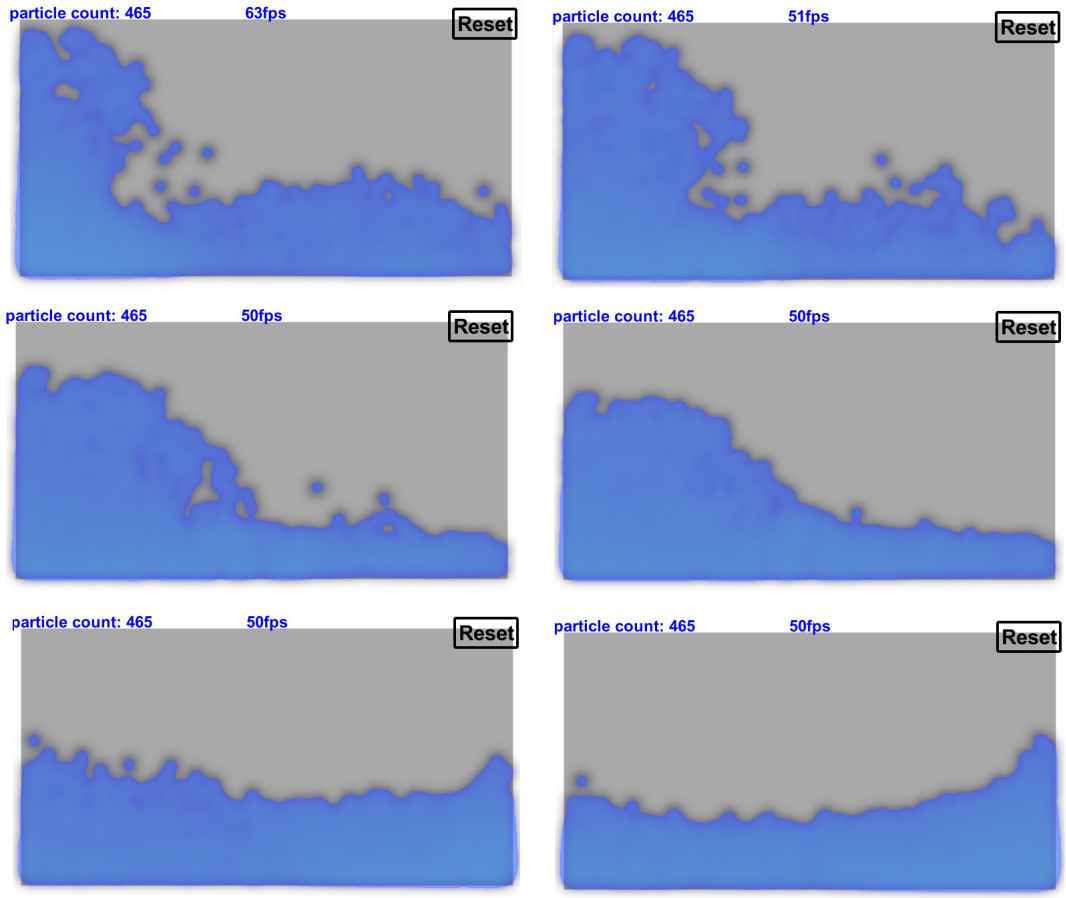
Obrázek 5.1: Výsledek implementace při běhu na PC.

zařízení/fps	60	45	30
Xperia	80	200	300
Samsung S4	100	200	400
Huawei	120	200	450
Samsung S3	50	80	330

Tabulka 5.1: závislost počtu částic na počtu snímků za vteřinu jednotlivých zařízení

Při zvoleném měřítku částic jejich množství vyplní sotva polovinu displeje. Situace se však zlepší na výkonnějších zařízeních. Analýzou pomocí vestavěného nástroje *Profiler* bylo také zjištěno, že 50-60% výpočetního času tráví program v části algoritmu pro vyhledávání sousedů, konkrétně znovu naplňování hashovací tabulky.

Testy dále odhalily, že simulace na smartphonech není nijak graficky náročná, většinu času zabírá výpočet samotného alogritmu.



Obrázek 5.2: Šíření vlny scénou

Kapitola 6

Závěr

Výsledný program byl otestován na mobilních zařízeních s různými hardwarovými parametry. V aplikaci je dosaženo asi 100 částic při 60 fps na průměrném zařízení, což je uspokojivé množství jen pro účely jednoduchých her. Simulace vyžaduje mnohem výkonnější hardware, než bylo původně předpokládáno. Pokud by měla být simulace nasazena ve hrách a interaktivních aplikacích, budou tyto vyvíjeny spíše na high-end zařízení. Při implementaci nebylo využito vláken ani nebylo vytvořeno viskoelastické chování. Testy simulace ukázaly, že z po grafické stránce a z hlediska vykreslování není simulace vůbec náročná. Většinu výpočetního času a výkonu zabere CPU výpočty simulačního algoritmu.

Návaznost by mohla být v podobě implementace viskoelastického chování. Dalším rozšířením práce by se mohla stát implementace vykreslování kapaliny pomocí metody pochodujících čtverců a porovnat výslednou kvalitu renderování povrchu a hladiny tekutiny u obou řešení.

Literatura

[1] Clavet, S.; Beaudoin, P.; Poulin, P.: Particle-based Viscoelastic Fluid Simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM SIGGRAPH, 2005, s. 219–228.

[2] Gourlay, M. J.: Fluid Simulation for Video Games(part 1). [Online]. Poslední modifikace: 30-4-2012. [vid. 2015-12-28].
URL https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1/

[3] Lucy, L. B.: A numerical approach to the testing of the fission hypothesis. In *Astronomical Journal 82*, 1977, s. 1013–1024.

[4] Mansson, D.: *Interactive 2D Particle-based Fluid Simulation for Mobile Devices*. bakalářská práce, KTH Royal Institute of Technology, Stockholm, Švédsko, 2013.

[5] Teschner, M.; Heidelberger, B.; Mueller, M.; aj.: Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *Vision, Modeling, and Visualization*, 2003, s. 47–54, [Online]. Poslední modifikace: 30-4-2012. [vid. 2016-5-10].
URL
http://matthias-mueller-fischer.ch/publications/tetraederCollision.pdf

Přílohy

Seznam příloh

Příloha A

Obsah CD

A.1 Technická zpráva

Technická zpráva ve formátu pdf se nachází v kořenovém adresáři CD. Zdrojové soubory pro LaTeXa soubor Makefile se nacházejí v adresáři `bp`. Po případných změnách lze technickou zprávu ve formátu pdf vygenerovat pomocí příkazu `make`.

A.2 Balíček pro import do enginu Unity

Balíček s názvem `sph2d.unitypackage` pro import do projektu v Unity se nachází v kořenovém adresáři CD.

A.3 Instalační soubor sph2d.apk

Instalační soubor demonstrační aplikace pro operační systém Android se nachází v kořenovém adresáři CD.

A.4 Zdrojové kódy

Všechny zdrojové kódy se nachází v adresáři `src`.