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

Bakalářská práce

Simulování proudění vzduchu v aplikacích VR

Plzeň, 2009 Ivo Zelený

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

V Plzni dne 23.*4.2009, Ivo Zelený,*

Abstract

The bachelor thesis is focused on wind flow modeling for applications of virtual reality (VR). The work is divided into two parts. An analysis of real wind flowing and a survey of methods for flow modeling are situated in the first part. The description of my proposed system and performed tests are described in the second part. The appendices of this thesis contain the user guide and screenshots of my application. The proposed system is based on the calculation of wind vector field from a temperature field mapped over the terain. During the simulation, this temperature field is modified using a particle system. The simulation is random and variable due to the particle system and that is why simulation behaviour looks like almost real wind flowing. Although the system works only for two dimensional temperature field. The test application, which was implemented in C#, was able to achieve more than 25 fps for 250x250 temperature field when running on 1,66GHz duo core, 3GB memory, 256MB graphics card and Windows Vista.

Obsah

1 Úvod

Důvodem vzniku práce je vytvořit nástroj pro modelování větru, který by mohl být použit v aplikacích virtuální reality místo často používaného nereálně chovajícího se větru. Například pohyb vegetace ve větru se většinou modeluje jen jako pohyb sem a tam. Dalším příkladem jsou mraky, jejichž pohyby jsou modelovány po přímých trasách, které jsou odhadnutelné a kazí tak celkový dojem celé scenérie. Na druhou stranu je modelování větru komplexní úlohou, která nemá přímočaré řešení, jelikož na vítr působí mnoho obtížně definovatelných faktorů z prostředí. Samozřejmě již existují způsoby pro popis a modelování proudění a tedy i proudění větru, které je potřeba prozkoumat a zjistit, proč se hojně nevyužívají. Důvodem může být asi nejdůležitější požadavek a tím je běh programu v reálném čase. Právě splnění požadavků na reálný běh a na reálného chování jsou hlavní cíle práce.

Ze zadání a po konzultacích s vedoucím bakalářské práce jsem si vytyčil následující cíle práce:

- Zorientovat se v současných možnostech pro simulování vzduchového proudění. Všechny dostupné způsoby simulace zanalyzovat a pak navrhnout vhodnou metodu pro využití v oboru virtuální reality.
- Naimplementovat větrnou knihovnu, která bude obsahovat vhodné funkce typu načtení pole větru, určení nového stavu, namapování pole na povrch, určení transformace pro volný objekt na pozici terénu. V ideálním případě by knihovna měla obsahovat takový systém pro modelování větru, který by se choval nepředvídatelně, samostatně se vyvíjel a to i po několika hodinách běhu.
- Vytvořit testovací grafickou aplikaci, která bude umožňovat volné procházení po povrchu zvrásněného terénu. V aplikaci bude důležité vhodným způsobem zobrazovat vektorové pole například obarvením vrcholů terénu dle rychlosti nebo směru větru. Případnou další možností jak zobrazovat pole, by mohlo být osázení terénu vegetací či tyčkami, které budou příslušně ohýbány (transformovány) dle vektoru větru v daném místě mapy.
- Posledním úkolem je knihovnu otestovat na zatížení a stabilitu tak, aby byly známy důležité vlastnosti a možnosti použití.

2 Vítr v reálném světě

2.1 Definice větru

Kopáček ve své knize [KB2005] popisuje vznik tlakového pole takto:

„*V důsledku toho, že na různých místech zemského povrchu existují vzájemně odlišné hodnoty atmosférického tlaku redukovaného na mořskou hladinu a podobně hladiny konstantního tlaku v atmosféře zpravidla neleží v konstantní nadmořské výšce, působí v ovzduší horizontální složky síly tlakového gradientu, které jeví tendenci vyrovnávat právě zmíněné tlakové rozdíly a vytvářet tak homogenní tlakové pole.*“

Ono nerovnoměrné tlakové pole působí na částice vzduchu silou a vzniká tak pohyb vzduchových mas, který nazýváme vítr. Z výše uvedené definice je zřejmé, že kdyby v reálné atmosféře probíhalo pouze výše popisované proudění do míst s nižším tlakem bez dalších sil a vlivů, došlo by velmi rychle k vyrovnání tlakových rozdílů a nemohly by se dále vytvářet žádné tlakové útvary s odlišnými hodnotami tlaku vzduchu a tudíž by nevznikalo ani žádné další proudění vzduchu. Faktorů působících na vznik a vývoj větru je mnoho a některé z nich budou uvedeny níže.

2.2 Tlak vzduchu (atmosférický tlak)

V předešlé kapitole se několikrát objevuje pojem atmosférický tlak. Tlak vzduchu má zásadní vliv na veškeré pohyby vzduchu v atmosféře, a proto se jím budu v této kapitole zabývat.

Atmosférický tlak [KB2005] je působení tíhy vzduchového sloupce na plochu. Jednotkou tlaku je pascal (Pa). Tlak 1 pascalu je definován jako síla 1 newtonu působící na plochu o velikosti 1 meter čtvereční. Pascal je dosti malá hodnota, proto se v praxi používají hlavně hektopascaly(hPa). Pro porovnání se zavádí referenční hodnota tlaku vzduchu, označujeme ji jako **normální tlak vzduchu** (**normální atmosférický tlak**) a jeho hodnota je 1013,25 hPa. Tato hodnota odpovídá tlaku vzduchu při mořské hladině na 45° severní šířky při teplotě 15 °C a při tíhovém zrychlení $g_n = 9{,}80665\ ms^{-2}$.

Základní rovnicí pro ideální plyn (plně stlačitelný a nepůsobí v něm tření) je stavová rovnice, která je definována takto:

$$p\,\alpha_d = R_d\,T$$

Vzorec 2.1 - Stavová rovnice

Ve vzorci p označuje tlak, α_d je rovna $1/\rho_d$, což je převrácená hodnota hustoty, neboli měrné hmotnosti (hmotnost vzduchu v jednotkovém objemu). Na pravé straně figurují hodnoty R_d (měrná plynová konstanta vzduchu) a T (teplota v kelvinech). Z uvedeného vzorce plyne, že hustota vzduchu roste s tlakem a klesá s rostoucí teplotou. Hustotu vzduchu také ovlivňuje složení vzduchu, protože například vzduch obsahující vodní páru je lehčí než suchý vzduch a má tedy menší hustotu.

Jako další podstatnou rovnici lze uvést základní hydrostatickou rovnici.

$$-dp = \rho\, g\, dz$$

Vzorec 2.2 - Hydrostatická rovnice

Kde -dp představuje pokles tlaku, ρ je opět hustota, g gravitační zrychlení a dz odpovídá příbytku výšky. Atmosférický tlak s výškou klesá. S výškou ovšem většinou klesá i

teplota. Z těchto předpokladů plyne, že hustota může s výškou klesat (ve většině případů), být konstantní či růst.

Na závěr této kapitoly uvádím tabulky s některými extrémními hodnotami tlaku a teploty, které byly doposud naměřeny.

Naměřené extrémní hodnoty tlaku		
místo	**rok**	**tlak[hPa]**
Praha	1907	1027,7
	1896	950,9
Ve světě	1968	1083,8
	1979	870

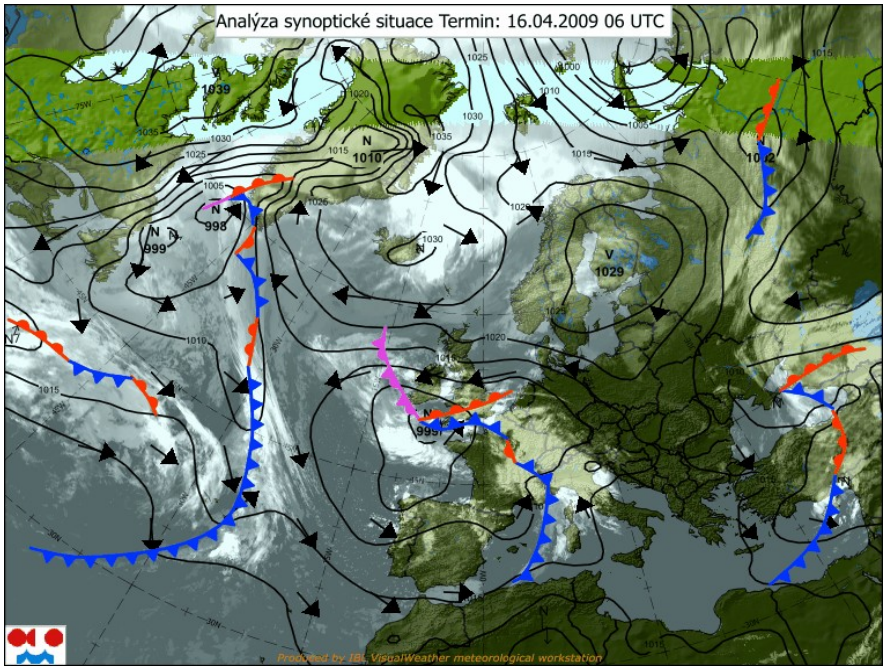
Tabulka 2.1 - Extrémní hodnoty atmosférického tlaku [Kop2005].

Naměřené extrémní hodnoty teploty		
místo	**rok**	**teplota[˚C]**
Tropolis	1922	57,8
Kalifornie	Průměrně v červenci	38,9
Antarktida	1983	-89,2
Sibiř	Průměrně v lednu	-51,2

Tabulka 2.2 - Extrémní hodnoty teploty vzduchu [Kop2005].

2.3 Tlakové útvary v atmosféře

V atmosféře v průběhu času vznikají, zanikají a pohybují se tlakové útvary [Kop2005], které, jak víme, mají na celkové proudění větru vliv, a tak je vhodné další části podrobně rozebrat. Oblasti atmosféry, kde je velikost atmosférického tlaku v daném čase konstantní, jsou označovány jako izobarické hladiny. Takovéto hladiny se zakreslují do takzvaných synoptických map, které slouží k zaznamenání a především zobrazení meteorologických jevů. Hlavní použití naleznou v meteorologii pro předpověď počasí.



Obrázek 2.1 - Příklad synoptické mapy [2].

Na obrázku (Obrázek 2.1) je možné vidět útvary, které významně ovlivňují proudění vzduchu v atmosféře, proto se také často označují jako akční centra atmosféry. Mezi tyto útvary řadíme především tlakové níže, tlakové výše, hřebeny vysokého tlaku a brázdy tlaku nízkého.

Tlaková níže (cyklóna) je oblast, která se vyznačuje nižším tlakem než jaký se vyskytuje v jejím okolí. Zároveň platí, že se směrem ke středu cyklóny tlak snižuje. V synoptických mapách se označují písmenem N (případně L v anglických mapách). Kolem centra tlakové níže navíc musí existovat uzavřené izobary, jinak danou oblast označujeme jako **brázdu nízkého tlaku**. Uprostřed tlakové níže vzduch stoupá vzhůru, přičemž dochází ke kondenzaci vodní páry, což má za následek oblačné a deštivé počasí.

Tlaková výše (anticyklóna) je naopak jev vznikající nad oblastmi, kde se vyskytuje vysoký tlak. Obdobně jako u cyklóny platí, že čím blíže se přibližujeme ke středu, tím více tlak stoupá. V mapách se označují písmenem V (respektive H). V tlakové výši sestupuje dolů těžký studený vzduch, který se následně otepluje a na rozdíl od tlakové níže nedochází ke kondenzaci ale k vysušování, proto jsou anticyklóny většinou spojeny s teplým a jasným počasím. I u anticyklóny platí, že její centrum musí být obklopeno uzavřenými izobarami, jinak se oblast označuje jako **hřeben vysokého tlaku**.

V zemské atmosféře existují oblasti, které trvale vykazují extrémní hodnoty a tvoří tak výše popisované útvary. K těmto oblastem patří především pás okolo rovníkových oblastí s nízkým tlakem. Další významné oblasti jsou například Islandská a Aleutská (poblíž Aljašky) tlaková níže či Azorská, Havajská tlaková výše.

2.4 Faktory ovlivňující vítr

Tato podkapitola se především zabývá otázkou, proč nikdy nedojde k naprosté rovnováze a následně k úplnému bezvětří. Existuje totiž několik důležitých faktorů [Kop2005], které mají na vývoj počasí a tedy i veškerých povětrnostních podmínek zásadní vliv.

2.4.1 Coriolisova síla

Zásadní vliv na vítr má zemská rotace. Sílu, která se projevuje působením na částice vzduchu v důsledku rotace Země, nazýváme **Coriolisova síla.** Tato síla má jak horizontální, tak vertikální složku. Vertikální složka je ale asi 10000 krát menší, proto se kromě speciálních aplikací (vypouštění družic) většinou zanedbává. Horizontální složka je naopak velice podstatná. Kolem rovníku je tato složka nulová. Všude jinde je kolmá k vektoru pohybu částic vzduchu a to vpravo na severní polokouli a vlevo na jižní polokouli ve směru pohybu. Velikost Coriolisovy síly lze spočítat podle vzorce:

$$c = 2\Omega v \sin(\varphi)$$

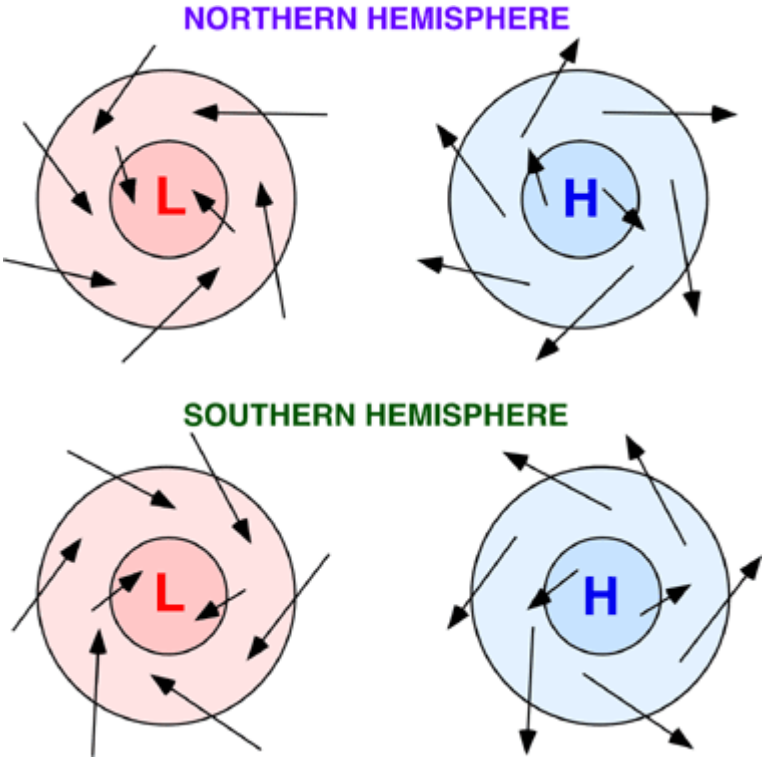
Vzorec 2.3 – Velikost Coriolisovy síly

Kde Ω je rovna úhlové rychlosti zemské rotace s hodnotou 7,29 * 10^{-5} s^{-1}, v je rychlost pohybu částice vůči Zemi a φ je zeměpisná šířka.

Pokud by došlo k rovnováze mezi Coriliosovo silou a silou tlakového gradientu (síla ve směru největšího poklesu tlaku), pak by se částice pohybovaly přímo po izobarách. Tomuto jevu se říká geostrofické proudění. K tomu ovšem v reálné atmosféře téměř nedochází. Rozdíl mezi reálným prouděním a geostrofickým prouděním se nazývá ageostrofické proudění a dosahuje velikosti kolem desetiny geostrofického proudění. Je zřejmé, že ageostrofická složka proudění přispívá k vývoji tlakového pole planety.

V tlakových nížích (výších) dochází při proudění kolem izobar, kde jsou hodnoty

vyššího (nižšího) tlaku vpravo (vlevo) vůči vektor pohybu, ke stáčení proti (po) směru hodinových ručiček. Rozdíl je patrný z níže uvedeného obrázku.



Obrázek 2.2 – Pohyb větrů v cyklóně a anticyklóně na severní a jižní polokouli [PS2008].

2.4.2 Aperiodické změny tlaku vzduchu

Proměnlivost atmosférického tlaku je prostorově i časově ovlivněna termickými a dynamickými příčinami. Mezi tyto příčiny patří:

- nerovnoměrné zahřívání planety Sluncem. U rovníku je planeta ohřívána nejintenzivněji, naopak u pólů nejméně
- cirkulace teplého lehkého vzduchu a studeného těžkého vzduchu
- vliv proudění a jeho charakteristiky

2.4.3 Periodické změny tlaku

Periodické změny rozlišujeme na roční a denní. Roční periodické změny jsou závislé především na konkrétním prostředí a střídání ročních období. Na kontinentech se v nižších oblastech pravidelně objevují maxima tlaku v zimě a minima v létě. Opačně je tomu v horských oblastech, kde jsou maxima v létě a minima v zimě. U oceánů se v průběhu roku vyskytují čtyři extrémy. Maxima tlaku je dosaženo v létě a zimě a minima na jaře a na podzim. Oproti oceánům je tomu v polárních oblastech naopak.

Denní periodické změny jsou způsobeny kolísáním teploty, a tak jsou maxima dosažena v 10 a 22 hodin a naproti tomu minima obvykle nastávají ve 4 a 16 hodin. Rozdíl mezi extrémy klesá se zeměpisnou šířkou.

2.4.4 Ostatní vlivy

Mezi další vlivy na nerovnoměrné ohřívání má také rozložení pevniny a oceánů. Pevnina se zahřívá pomaleji a déle si udržuje teplotu a naopak oceány se zahřívají rychleji, ale

rychleji chladnou. Dosti velký vliv má také různorodost povrchu, neboť rychlost větru například závisí na tom, zda jde o otevřený terén, lesnatý terén či předměstí nebo centrum města. V neposlední řadě uvádím vlhkost, neboť výrazně ovlivňuje hustotu vzduchu. Koncentrace vodní páry se však v daném místě a čase může neustále měnit. Příčinami proměnlivosti jsou například vypařování, kondenzace či horizontální proudění suchého a vlhkého vzduchu.

Výše zmíněné faktory mají vliv především na rychlost a směr větru. Pro představu o rychlostech větru uvedu následující tabulku, která se nazývá **Beaufortova tabulka rychlostí větru**. Tabulka se vztahuje na průměrnou rychlost větru.

Beaufortova tabulka rychlostí větru		
stupeň	**název větru**	**rychlost [m/s]**
0	bezvětří	< 0,5
1	vánek	~ 1,25
2	větřík	~ 3
3	slabý vítr	~ 5
4	mírný vítr	~ 7
5	čerstvý vítr	~ 9,5
6	silný vítr	~ 12
7	mírný vichr	~ 14,5
8	čerstvý vichr	~ 17,5
9	silný vichr	~ 21
10	plný vichr	~ 24,5
11	vichřice	~ 29
12 - 17	orkán	> 30

Tabulka 2.3 – Beufortova stupnice rychlostí větru [Vaš2008].

3 Způsoby modelování větru

3.1 Numerické metody

Numerické metody [KB2005] spočívají v postupném řešení rovnic pro různá měřítka a pro každý časový okamžik. Hlavním popisem proudění tekutin pro většinu numerických metod jsou Navier-Stokesovy pohybové rovnice. Tyto rovnice vycházejí ze vztahu mezi složkami horizontálního a vertikálního zrychlení pohybů proudění v atmosféře a silami, které na částice působí (síla tlakového gradientu, Coriliosova síla, gravitační síla, tření, vztlakové síly související s horizontální nehomogenitou vzduchu). Dále se do výpočtů zahrnuje: první termodynamická věta (zachování energie teplotní výměny), rovnice difúze vodní páry a nečistot, rovnice kontinuity vyjadřující spojitost proudění (zachování hmoty při aerodynamických dějích). Při výpočtech také uvažujeme charakteristiky vzduchu (mezi tlakem, teplotou, hustotou) jako např. Stavová rovnice ideálního plynu či vztahy popisující fázové změny stavů vody (pára, led).

S uvažováním všech výše uvedených vlivů vzniká takzvaný matematický model, který v důsledku obsahuje soustavy s parciálními diferenciálními rovnicemi nelineárního typu. Matematické modely Navier-Stokesových rovnic jsou bohužel tak složité, že je nelze řešit přímo. Ovšem za určitých podmínek (např. vhodná volba simulovaného měřítka) není přesné znění rovnic potřeba pro přiblížení se ke skutečnému proudění. Řešení takto zjednodušených rovnic již lze určit na výkonných vědeckých počítačích, případně pomocí velkého paralelního zpracování.

Pro kontrolu a opravu funkce modelů, popřípadě k zavedení do sítě uzlových bodů, se používají reálné hodnoty naměřené na pozemních meteorologických stanicích například: tlak, teplota, vlhkost, rychlost.

Dále je vhodné v kontextu numerických metod uvést Reynoldsovo číslo [Ur2007], neboť představuje souvislost mezi setrvačnými a vazkými silami. Z hodnoty čísla pak lze určit charakter proudění. Proudění s malými čísly se označují jako laminární a naopak s vysokými čísly jako turbulentní. Velikost Reynoldsova čísla může dosáhnout miliónů až miliard.

Existuje mnoho dostupných numerických modelů od přímočarých řešení po přímé řešení matematických modelů.

3.1.1 Lineární řešení

Lineární modely [Hei2008] se vyznačují snadnou implementací, ale nedosahují takových přesností. Příkladem může být model WASP (Wind Atlas Analysis and Application program), který vychází z konceptu lineárních modelů (navrhnuto Johnsonen a Hunten 1975). Obsahuje jednoduché modely pro turbulence a tvary terénu. Jsou vhodné pouze pro jednoduché konfigurace, u kterých dosahují rychlých a přesných výsledků jako je například stálé proudění přes hornatý terén. Naopak jsou velice nevhodné pro popis cirkulací, či pro simulaci proudění ve složitém prostředí.

3.1.2 DNS (Direct Navier-Stokes)

Počínaje touto metodou spadají všechny další do kategorie komplexních řešení, které jsou naopak v závislosti na metodě více či méně obtížné na implementaci a výpočet, za to však produkují mnohem přesnější řešení.

Metoda DNS [LYD2001] spočívá v řešení všech měřítek pohybu vzduchu od

nejmenších po největší, přičemž poměr měřítek odpovídá (po matematických úpravách) Reynoldsovu číslu. Řešit všechna měřítka například při výpočtu proudění kolem překážky (např. Křídlo), kde požadujeme výpočet pohybů od vizkózních vrstev (vrstvy s vnitřním třením s měřítkem kolem 10^{-6}) až po vrstvy s měřítky řádově jednotek až desítek metrů, představuje tento přístup velké výpočetní nároky. Při použití dnešních supervýkonných počítačů lze simulovat prostorové mřížky, které obsahují okolo 10^7 bodů, což je postačující pouze pro velmi malá Reynoldsova čísla. Se současným tempem vývoje výpočetní techniky může trvat i desítky let, než bude možné pomocí DNS simulovat tak velká Reynoldsova čísla, jaká se vyskytují v přírodě.

3.1.3 RANS (Reynolds Averaged Navier Stokes)

Princip metody RANS [LYD2001] je založen na zavedení Reynoldsovy napěťové rovnice a časové diskretizace Navier-Stokesových rovnic. Další přidané rovnice vedou ke zvýšení počtu neznámých a nejsou tak dále přímo řešitelné. Pro vyřešení je proto nutné volit mnoho nastavitelných parametrů, které mají velký vliv na celkové chování proudění. Je zřejmé, že pro odlišné druhy proudění bude nutné nastavit i odlišné konstanty. Z toho plyne, že RANS nejsou dost univerzální při proměnném působení vnějších sil (rotace, tepelné rozvrstvení) a musí se vyvíjet nejvhodnější nastavení pro danou situaci.

Model je určen zejména pro stabilní proudění nebo pro proudění, jehož vlastnosti se s časem příliš nemění nebo jen velice pomalu. Výhodou naopak je, že díky dlouhodobému vývoji modelu jsou již parametry pro specifické simulace známy a vedou ke stabilním stavům. Další nespornou výhodou jsou menší (stále však vysoké) nároky na výpočetní techniku oproti DNS.

3.1.4 LES (Large Eddie Simulation)

Metoda LES [LYD2001] představuje střed obou předchozích metod, jak ve smyslu nároků na výpočet, tak ve smyslu přesnosti a univerzálnosti. Založeno na principu modelování malých měřítek a deterministickém simulování měřítek velkých, přičemž proudění malých měřítek ovlivňuje proudění v měřítcích velkých. V důsledku se dále musí řešit velkoměřítková nestejnorodost a nestálost. Tyto skutečnosti jsou důležité, neboť víry v malých měřítcích neobsahují takovou energii a nejsou tak podstatné jako proudění velkých měřítek, které hlavně odpovídá za podstatnou část turbulentních přesunů částic a například i výměnu tepla.

Použití tohoto modelu oproti předchozímu je značně univerzálnější a může být aplikováno na různé druhy a konfigurace proudění pod vlivem vnějších sil bez modifikace. Bohužel požaduje především jednodušší terén. Na realistickém povrchu a zejména blízko povrchu dosahuje špatných výsledků. Model je omezený i vůči velmi jednoduchým konfiguracím proudění. Pro vědecké účely pak představuje velký pokrok. Je extrémně užitečný k pochopení a výzkumu souvislých vírů a struktur v turbulenci. I zde je v důsledku technického omezení možné simulovat mřížky o velikostech maximálně 10^7 bodů. Doba simulace je v řádu hodin a vyžaduje tak opět masivní paralelní zpracování. Typickým měřítkem jsou desítky metrů.

3.1.5 Další numerické modely

Samozřejmě je možné vytvářet kombinace z předchozích modelů. Takovým hybridním modelem s praktickým využitím je například model DES (Detached Eddie Simulation model) [Hei2008], který vzniká spojením modelů RANS a LES.

Na výše uvedených principech vznikaly i další modely. Významným modelem je

například WRF(Weather Research and Forecast model) [Hei2008], který vytváří meteorologické modely pro předpověď počasí. Zde je typicky používanou jednotkou délky hodnota z intervalu 1 – 100 km. Je postaven na využití dostupných vstupních dat jako jsou: terén, vlastnosti povrchu, meteorologické podmínky. I tento model však vyžaduje masivní paralelní zpracování.

3.2 Buněčné automaty

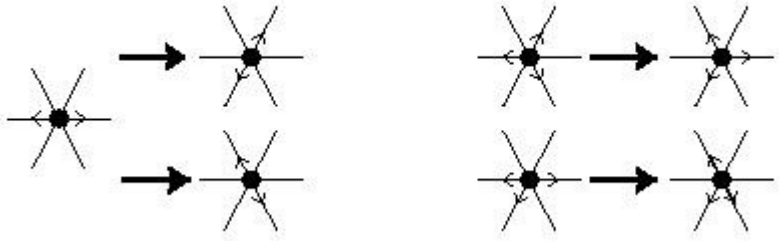
Jiný přístup pro modelování proudění i Navier-Stokesovy rovnice je využití buněčných automatů[WLMK2003], které se v tomto smyslu také nazývají LGA (Lattice Gas Automata). Tento přístup představuje rychlejší a jednodušší způsob simulování pohybů plynů. Jednotlivé metody pak vycházejí z principu simulovat mikroskopické pohyby částic tekutiny na základě pravidel aplikovaných na buňky mřížky. Makroskopické vlastnosti jsou určeny z průměru vlastností částic v daném okolí. Na proudění se tak nahlíží jako na samoorganizovaný proces vývoje od mikroskopických pohybů atomů po makroskopické proudění. Buněčné automaty tedy řeší pohyb atomů a molekul a především jejich kolize, které podstatně ovlivňují celkové chování.

3.2.1 HPP automat

První LGA představili pánové Hardy, Pazzis a Pomeau (odtud zkratka HPP) [HPP1976], jak je uvedeno v [WLMK2003]. Automat je definován na čtvercové mřížce a kvantum vzduchu se mohlo pohybovat po jejích hranách, tudíž z každého bodu je možný pohyb čtyřmi směry. Kolize částic v bodu mřížky se řeší změnou směru jejich pohybu a to tak, aby nová orientace vektoru byla ve směru, odkud ani jedna z částic do bodu nevstoupila. HPP automat zachovává celkové množství i moment částic, ale bohužel jeho chování není izotropní.

3.2.2 FHP automat

Další významný LGA byl představen pány Frisch, Hasslacher a Pomeau [FHP1986], jak je uvedeno v [WLMK2003], a opět nese název podle jejich jmen. Tentokrát je ale založen na hexagonální mřížce. Každá buňka tak má 6 nejbližších sousedů a tedy i 6 možných směrů pohybu. Nový stav je určen na základě 2 pravidel, která se nazývají propagace a kolize. Propagace znamená přesun částic ve směrem jejich pohybu. Velice důležitou součástí jsou pak kolize, neboť mohou úplně změnit směr pohybu. Kolize jsou vyhodnoceny takzvaným kolizním operátorem, který je definován tak, aby uspokojil nároky na zachování kvanta a momentu. Možná řešení kolizního operátoru pro kolizi z 2 a 3 směrů v jednom bodě mřížky jsou na obrázku 3.1.

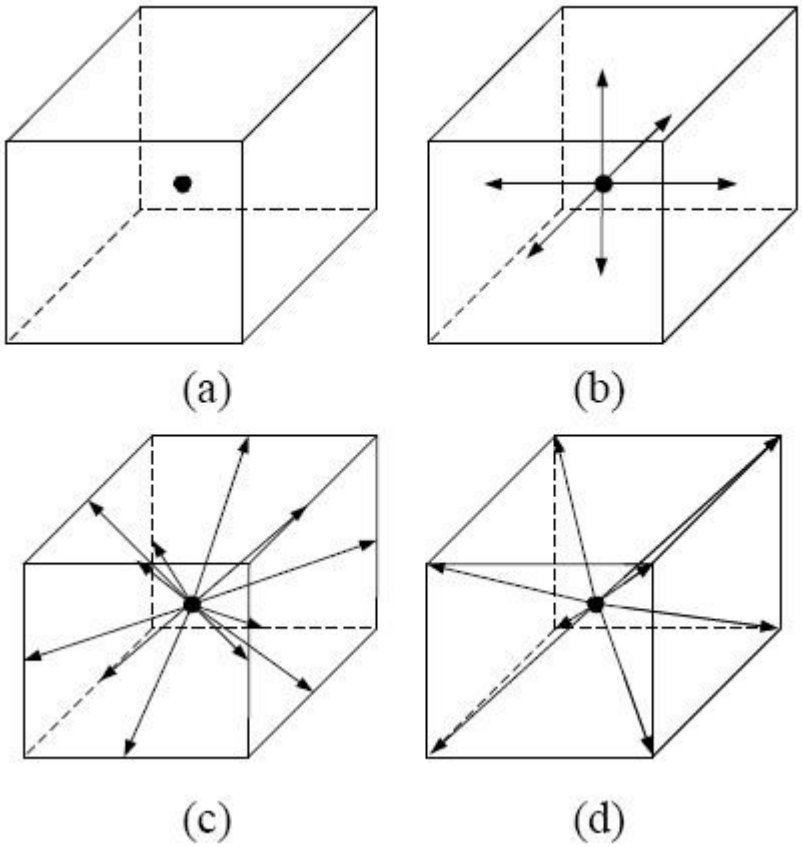


Obrázek 3.1 – Řešení kolizí v FHP [WLMK2003].

3.2.3 LBM (Lattice Boltzman Model)

V současné době je hojně používaným LBM [CD1998], jak je uvedeno v

[WLMK2003]. Existuje v provedení na 2D i 3D diskrétní mřížce, kde je čas i stav každé z buněk diskrétní. Ve většině případů se používá 3D varianta, kde je mřížka složená z krychlí, přičemž je pro každý bod definováno až 26 sousedů s různou vzdáleností od bodu. Pokud budeme uvažovat vzdálenost od středu krychle k její stěně rovnou 1, pak ve vzdálenosti 1 existuje 6 sousedů, ve vzdálenosti √2 existuje 12 sousedů a v √3 je 8 sousedů. Všechny jednotlivé sousednosti jsou uvedeny na níže uvedeném obrázku 3.3.



Obrázek 3.2 - Všechny možné sousednosti pro LBM [WLMK2003].

Ze zkoumání Center for Visual Computing v New Yorku (uvedeno v [WLMK2003]) plyne, že použití všech sousedností je moc výpočetně náročné a vynechání například sousednosti ve vzdálenosti √2 vede k numerické nestabilitě. Jako optimální pak vyhodnotili vynechat sousedy ve vzdálenosti √3, což je optimální vůči stabilitě i výpočetním nárokům. Obdobně jako v předchozím modelu i zde jsou použita 2 základní pravidla propagace a kolize (opět se vyhodnocují vhodným kolizním operátorem).

LBM simulace použitá institutem Center for Visual Computing v New Yorku pro modelování kouře generuje pro malé mřížky (bez vylepšení asi 16^3 bodů) v reálném čase správné rychlostní pole, do něhož je možné začlenit i pole teplot, pokud chceme uvažovat i vznosné síly teplých plynů. Po výzkumu a vylepšeních od uvedeného institutu je možné bez větších problémů v reálné čase simulovat mřížky obsahující 32^3 bodů. Při dalším nárůstu mřížky dochází při každém zdvojnásobení počtu bodů ve všech osách (64^3, 128^3, 256^3) přibližně k desetinásobnému nárůstu výpočetního času (0.05s, 0.5s, 5s na snímek při použití vhodného hardwaru).

3.3 Částicové systémy

Další technikou pro modelování různých přírodních fenoménů jsou částicové systémy [Rev1983]. Použití naleznou především v modelování mraků, kouře, vody, ohně,

tedy v případech, kde selhávají klasické techniky počítačové grafiky jako například modelování pomocí ploch. Tyto takzvané fuzzy objekty totiž nejsou hladké a snadno definovatelné jako většina zobrazovaných objektů, ale komplexní nepravidelné plochy s velmi složitým popisem. Dalším problémem je jejich dynamika, protože se nejedná o tuhá tělesa s jednoduchými transformacemi. U fuzzy objektů tak je obtížné modelovat především jejich vznik, zánik či změnu tvaru a proudění. Budeme-li vítr považovat za přesouvající se a měnící se objem vzduchu, jenž může vznikat a zanikat, lze si představit i použití částicových systémů pro simulaci větru (ne však plnohodnotný model proudění).

Pro reprezentaci pomocí částicových systémů platí:

- Objekt není reprezentován jako množina primitivních ploch (polygony či jiné geometrické objekty s vymezenými okraji), ale jako jakýsi mrak částic definující jeho objem.
- Objekt není statickou entitou, ale téměř neustále mění svůj tvar i pozici v průběhu času.
- Objekt není deterministický, neboť jeho tvar ani formu nelze celkově specifikovat. Naopak částice je triviální objekt se snadným zobrazováním a lze jich tak zobrazit velké množství.
- Částicové systémy mají definovaný postup aktualizace systému (uveden níže) a jsou řízeny náhodnými čísly, což jsou faktory usnadňující návrh systému (není potřeba vytvářet složité matematické popisy ploch).
- Model je v určitém smyslu živý, neboť i když je řízen stochastickými procesy, tak se náhodně mění a vyvíjí v čase.

Ve většině aplikací mají částice tyto atributy: pozice, rychlost, směr, velikost, barva, průhlednost, tvar, doba života. Počet částic se většinou řídí náhodně kolem hodnoty velikosti obrazovky případně jiné pevně stanovené hodnoty.

V každé aktualizaci systému (většinou v každém snímku) se opakuje následující sekvence kroků:

- generování nových částic
- každé nové částici jsou určeny její vlastní atributy
- každá částice, která existuje v systému i po její předepsané době života, je ze systému odstraněna
- zbývající částice jsou přesunuty, transformovány a případně změněny (například barva či průhlednost) dle jejich dynamických atributů
- render (vykreslení) výsledku

4 Analýza metod

Všechny výše uvedené postupy nejsou jen prostými návrhy, ale mají své reálné využití. Úkolem této práce je však vítr simulovat v reálném čase, aby mohl být použit v aplikacích virtuální reality, přičemž vítr má být především simulován a ne nutně přesně modelován.

Výše uvedená podmínka výpočtů v reálném čase je velice podstatná. Numerické metody při použití současných počítačů nelze kromě přímých metod použít, neboť z rozborů numerických metod vyplývá, že ani jediná z nich není schopna podmínku rychlého výpočtu splnit. Přímé numerické metody však řeší jen jeden druh proudění, ale my od modelu požadujeme zajištění proměnlivosti větru. Ostatní numerické metody sice simulují přesně proudění tekutin v nejrůznějších prostředích a například modely pro předpověď počasí s uvažováním všech faktorů by byly pro simulaci ve vnějším prostředí ideální, ale tyto nástroje jsou však příliš složité a pro použití k našim účelům simulace větru až moc detailní a přesné. Pro virtuální realitu není tak podstatné přesně modelovat proudění okolo libovolných předmětů či dodržovat Navier-Stokesovy rovnice a dodávat jim potřebné meteorologické údaje k dosažení správných výsledků. Také není nutné v modelu uvažovat například vlhkost, nečistoty či další faktory, které neúměrně zlepšují přesnost modelu na úkor jeho výpočetních nároků a celkové složitosti.

Na rozdíl od numerických metod je použití buněčných automatů v reálném čase možné. Dokonce i Boltzmannův model lze při malých rozměrech mřížky použít. Naleznou však použití především pro simulování kouře, čili takových pohybů vzduchu, které vznikají z nějakého zdroje jako cigareta, konvice a podobně. Samozřejmě je lze použít i pro simulace pohybů vzduchu v atmosféře či místnosti. V tomto smyslu využití ale požadují přísun okrajových podmínek do modelu, například směr a velikost větru. Lze je tak využít při výzkumu proudění okolo nejrůznějších předmětů ke zjištění vlivů předmětů na proudění a naopak. Tyto okrajové podmínky ovšem chceme ve svém důsledku vytvářet a ne dodávat. Využití buněčných automatů by tak přicházelo v úvahu například při zjišťování lokálního působení větru. Nicméně použití této techniky modelování s oproštěním se od okrajových podmínek by mohlo vést k cíli. Nabízí se například možnost použít vektorové pole, jehož buňky by představovaly buňky automatu a okraje vektorového pole by splývaly. Tím bychom mohli dosáhnout vypuštění okrajových podmínek, neboť by byly dodávány z protějšího okraje vektorového pole.

Poslední z prozkoumávaných technik je modelování větrného proudění pomocí částicových systémů. Tato metoda plně umožňuje požadovanou simulaci v reálném čase. Mezi typické použití však modelování větru nepatří, avšak po určitých úvahách a zjednodušeních si lze použití v tomto směru představit. Částice by mohly představovat kvantum přesouvaného vzduchu a na jejich pohyb by působily některé podstatné síly uvedené v kapitole o přírodním větru. Částice by kromě chtěné vizualizace proudění nebyly zobrazovány, což je asi hlavní rozdíl oproti typickému použití, kde se snažíme modelovat a zobrazovat netriviální objekty a jejich pohyby či změny tvaru. Částice navíc samy o sobě nemohou tvořit vítr, ten by se musel vytvářet na základě aktuálního pohybu částic v daném okolí. Otázkou také je, zda by bylo možné využít tohoto modelu při simulování turbulencí v prostoru. Zajisté se zde nabízí možnost vytvářet částice ve více než jedné vrstvě, avšak tato metoda povede k velkému zvyšování celkového počtu částic. To nás přivádí na další otázky, například: Jaký může být maximální celkový počet částic, se kterými bude probíhat výpočet, přičemž má být dodržen požadavek na simulaci v reálném čase? Jaký počet částic je optimální vzhledem k reálnosti simulace a zároveň k nárokům na paměť a výpočetní dobu? Také co bude zdrojem částic, je další typická otázka, kterou nelze bez analýzy a testů odpovědět, i když jako vhodný předpoklad se jeví použít tlakovou výši jako zdroj.

5 Návrh a implementace

5.1 Volba programovacího jazyka a prostředků

Jako programovací jazyk jsem zvolil jazyk C# a prostředek pro zobrazení výsledků DirectX (dle požadavků zadání). Tuto volbu jsem učinil zejména z důvodů grafického zaměření práce, neboť spojení C# a DirectX je pro grafické aplikace doporučované a snadno dostupné, zároveň mám s touto variantou, zejména díky jiným předmětům (např. KIV/ZPG nebo KIV/PRJ5), zkušenosti a mohu tak navázat na mé předchozí práce.

5.2 Implementace větru pomocí buněčných automatů

Z analýzy vyplývají dvě možné varianty implementace větrného proudění. Jako první implementaci jsem zvolil postup podle buněčných automatů. Tento přístup vychází z principu vítr reprezentovat pomocí vektorového pole, jehož buňky se s buňkami ve svém okolí navzájem ovlivňují, čímž generují nový stav vektorového pole.

5.2.1 Mapování vektorového pole na terén

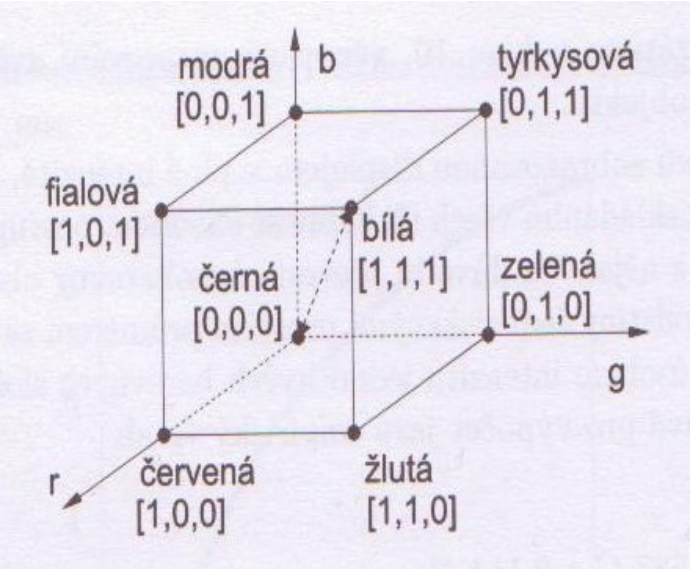
Při vytváření instance větrné knihovny jsou do konstruktoru mimo jiného předány i souřadnice levého dolního rohu a pravého horního rohu. Díky těmto souřadnicím je možné určit šířku a výšku terénu. Protože nyní známe výšku, šířku a alespoň jeden z rohů mapy, na kterou mapujeme vektorové pole, je možné určit pro každý bod mapy jeho relativní pozici vůči známému rohu mapy. Tím dojde k převodu souřadnic na interval <0,1>, který je možné univerzálně použít pro jakkoli veliké vektorové pole, například přenásobením počtu řádků a sloupců s následným převodem na datový typ integer lze jednoduše získat pozici řádku a sloupce odpovídající buňky pole.

5.2.2 Načtení výchozího stavu vektorového pole

Aby bylo možné určovat nový stav na základě předchozího, je nutné nejprve načíst výchozí stav vektorového pole například ze souboru či pomocí grafického rozhraní. V mé aplikaci jsem využil možnost první a pro uložení pole jsem zvolil bitmapový soubor, kde každý pixel představuje vektor vektorového pole. Bitmapový obrázek jsem zvolil zejména kvůli snadné přípravě vektorového pole pomocí libovolného grafického programu pro úpravu obrázků.

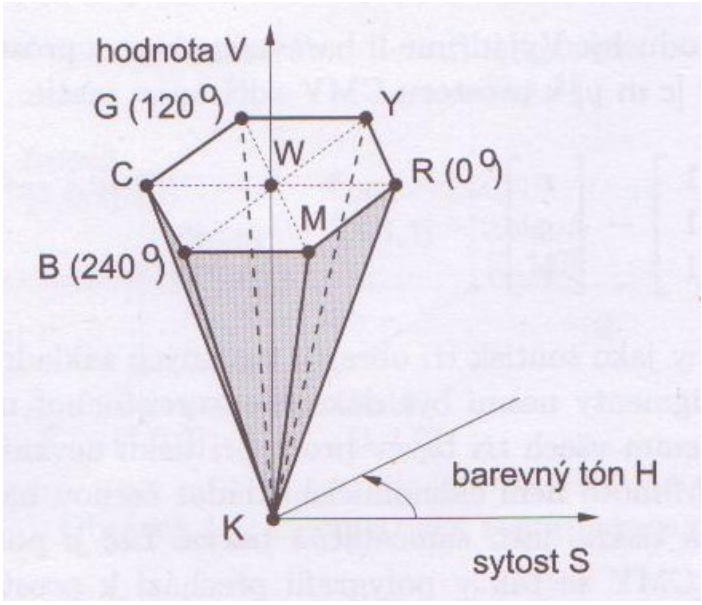
Bitmapový obrázek se skládá z pixelů, které obsahují barvu reprezentovanou v barevném systému. V této i dalších kapitolách se budu zmiňovat právě o barevných systémech RGB a HSB. Proto nejdříve uvedu jejich popis.

RGB barevný systém [ŽBSF2004] využívá faktu, že lze barvu reprezentovat ze tří základních barev r-červená, g-zelená, b-modrá. Hodnoty se většinou zapisují do barevného vektoru s hodnotami složek v rozmezí od 0 do 255, kde 0 znamená, že není barva zastoupena, a 255 naopak odpovídá plnému zastoupení barvy. Celý barevný prostor RGB lze reprezentovat pomocí krychle na obrázku 5.1.



Obrázek 5.1 – Geometrická reprezentace prostoru RGB.

Také HSB (nebo též HSV) barevný systém [ŽBSF2004] využívá tří složek, ale oproti RGB složky představují parametry barevný tón (H, hue), sytost (S, saturation) a jasová hodnota (B, brightness). Parametr H označuje převládající spektrální barvu (charakterizuje úhel od 0 do 360). Parametr S určuje příměs jiných barev (v rozmezí od 0 do 1). Parametr B reprezentuje množství bílého světla (v rozmezí od 0 do 1). Barevný prostor HSV lze tentokrát vyjádřit pomocí šestibokého jehlanu na obrázku 5.2.



Obrázek 5.2 – Geometrická reprezentace prostoru HSB.

Mezi oběma systémy lze hodnoty navzájem převádět většinou již pomocí funkcí ve vestavěných knihovnách programovacího jazyka.

Při vlastním načítání jsem jednotlivé pixely obrázku převáděl na vektory větrného pole pomocí HSB barevného systému. Jelikož jsou při načtení barvy pixelu známy pouze jeho RGB složky, převádím je pomocí vestavěné knihovny pro práci s obrázky a barvami na HSB složky. Po převodu jsem pak vybral následující variantu reprezentace jednotlivých složek. Hodnota H představuje úhel natočení vektoru (0 - 360), hodnota S určuje délku vektoru (0 – 1), informaci o složce B jsem kvůli použití pro 2D vektorové pole nevyužil. Hodnoty vektorů jsou uchovány v jednorozměrném poli o velikosti šířka krát výška načítaného obrázku.

5.2.3 Určení nového stavu

Za nejdůležitější funkci knihovny lze považovat určení nového stavu vektorového pole (používané například při další časové jednotce), přičemž jsem otestoval dvě provedení této funkce a to:

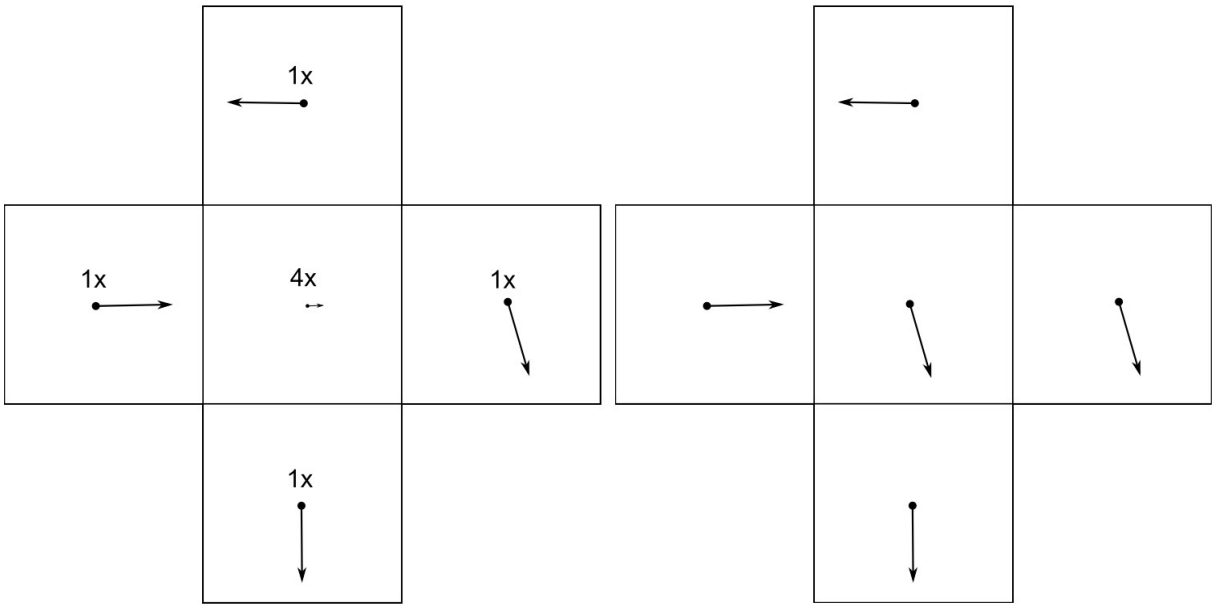
a) určení nového stavu ze 4-okolí

Pro každý vektor vektorového pole určím jeho nový stav pomocí jeho hodnoty a okolních hodnot ze 4-okolí. Vektory 4-okolí jsou sousední levý, pravý, horní a dolní vektor. Každou složku nového vektoru vypočtu jako podíl součtu okolních hodnot a čtyřnásobku původní hodnoty ku počtu sčítanců. Vzorec pro složku x pak vypadá takto:

$$x = (4 * x_{původní} + x_{horní} + x_{dolní} + x_{levý} + x_{pravý}) / 8$$

Vzorec 5.1 – Výpočet nové složky x vektoru větru z hodnot 4-okolí.

Jak proběhne určení nového vektoru, je vidět na následujícím obrázku 5.3. Levá část obrázku znázorňuje stav před výpočtem. Pravá část pak již znázorňuje nový stav pro počítaný vektor a pro názornost jsou v obrázku ponechány i předchozí stavy okolních sousedů.

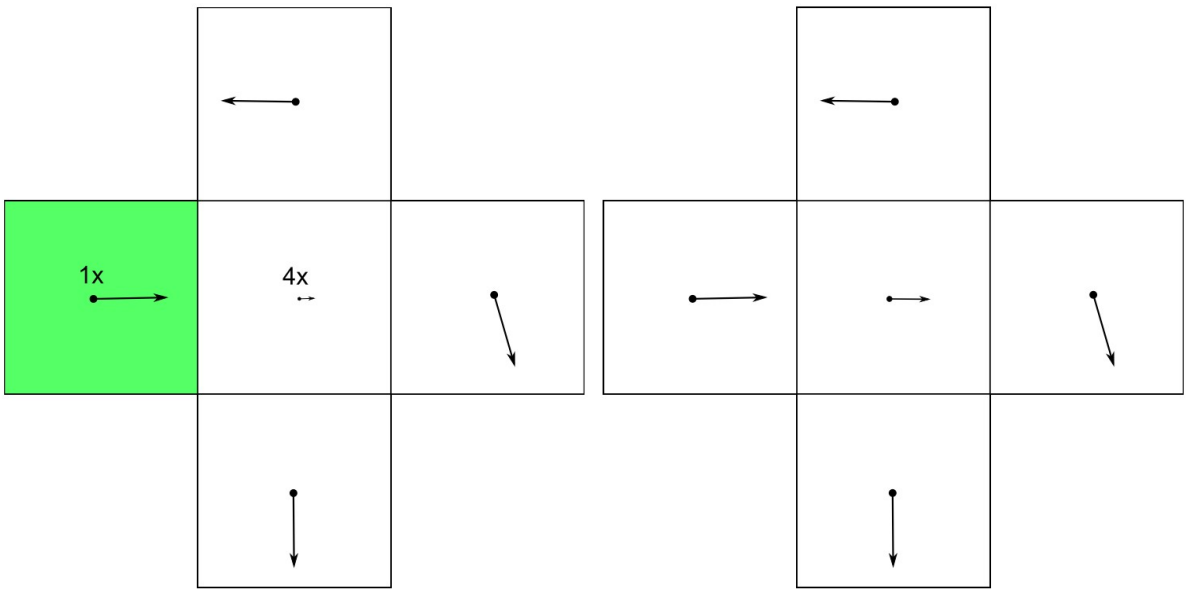


Obrázek 5.3 – Určení nového stavu vektoru na základě jeho 4-okolí.

b) 4-okolí s respektováním směru větru v okolních vektorech

Postup je obdobný postupu v bodě a), ale hodnoty sousedících vektorů jsou brány v úvahu jen pokud jejich směr směřuje k ovlivňovanému vektoru. Otázku, zda směřuje či nikoli, zodpoví skalární součin sousedního vektoru s vektorem, který k ovlivňované buňce pole směřuje (Například pro horní sousedící vektor bude tento vektor (0,-1).). Složky původního vektoru jsou opět brány v úvahu čtyřikrát a také dělitel odpovídá počtu sčítanců čitatele (Zde již ale není hodnota dělitele vždy konstantní jako v předchozím případě.).

Na obrázku 5.4 je opět zobrazen způsob určení nového stavu vektoru ve vektorovém poli. Na levé části obrázku je zobrazen stejný předchozí stav jako v případě a), kde jako jediný sousední vektor splňující podmínku je vektor v poli označený zelenou barvou (Vektor je do výpočtu zapojen, protože směřuje k ovlivňovanému vektoru.). Na obrázku vpravo je pak opět následující stav počítaného vektoru a předchozí stav sousedních vektorů.



Obrázek 5.4 – Určení nového stavu vektoru na základě vektorů k němu směřujících z jeho 4-okolí.

Pro zajištění správného výpočtu nového vektorového pole v obou postupech jsem vytvořil 2 pole vektorů, jejichž ukazatele se po výpočtu nového stavu vektorového pole zamění. První pole obsahuje aktuální hodnoty vektorů, z nichž se vypočítávají nové hodnoty do pole druhého. Na konci výpočtu se ukazatel na aktuální pole nastaví na druhé pole a naopak druhý pomocný ukazatel se nastaví na pole se starými hodnotami, které při dalším výpočtu nového stavu poslouží jako uložiště pro nové hodnoty. Tento postup je nutné dodržet, jinak by docházelo při výpočtu k chybám díky mísení nově vypočtených hodnot a původních hodnot připravených k výpočtu.

Při obou postupech dochází na okrajích pole ke čtení hodnot mimo rozsah vektorového pole. Tento nechtěný jev jsem odstranil pomocí čtení hodnot z protějšího okraje. Je-li například potřeba znát na levém okraji hodnoty levého sousedního vektoru, jsou tyto hodnoty přečteny jako hodnoty vektoru na tomtéž řádku z pravého okraje vektorového pole. Obdobně je tomu i u ostatních kritických případů. Tímto postupem dochází ke splynutí pravého a levého okraje a zároveň horního a dolního okraje. Proudění větru je pak díky tomu plynulejší a nedochází u okrajů k deformaci vektorů díky chybějícím hodnotám. Zároveň je tím částečně omezen požadavek na okrajové podmínky a navíc se více podobá reálnému světu, kde vítr může cirkulovat kolem planety.

5.2.4 Zjištění transformační matice pro volný objekt v daném místě

Na základě mapování lze zjistit index buňky vektorového pole. Z vektoru větru přenásobeného délkou jednoho metru mapy, na kterou provádíme namapování, vytvořím transformační matici posuvu, která je vrácena návratovou hodnotou.

5.2.5 Shrnutí vlastností

Knihovna umožňuje načíst vektorové pole z bitmapového souboru. Při výpočtu nového stavu je použito okolí každé buňky. Tento způsob výpočtu však bohužel směřuje k ustálení celého vektorového pole, neboť nejsou dodávány nové okrajové podmínky. Doba do neměnnosti vektorového pole proto bude předmětem testování (kapitola 6).

5.3 Implementace větru pomocí částicových systémů

Vzhledem k tendenci ustalovat se, kterou jeví první implementace pomocí buněčných automatů, jsem se rozhodl pracovat také na způsobu využívající částicového systému, přičemž bych chtěl co nejvíce zachovat rozhraní pro přístup ke knihovně (načtení výchozího stavu vektorového pole, určení následujícího stavu, získání transformační matice pro volný objekt), neboť je intuitivní a osvědčilo se.

Pohyb částic většinou vychází z fyzikálních rovnic, například přitažlivé síly či odstředivé síly a podobně. Naprosto zásadní hodnotou v oblasti simulace větru je atmosférický tlak. Rozdíly tlaku pak vyvolávají sílu o velikosti tlakového gradientu, která následně rozpohybuje částice vzduchu. Definovat a reálným způsobem poté měnit tlak v každém bodě mapy, si však nelze dost dobře představit, zejména kvůli množství hodnot, na kterých závisí. Bylo by tedy vhodné zjednodušit si situaci a převést tlak na veličinu, se kterou by práce mohla být snazší, jako je například teplota. Prozkoumejme tedy nyní vztahy mezi teplotou a tlakem (viz kapitola 2.2), které platí pro reálné proudění vzduchu. Ze stavové rovnice ideálního plynu (Vzorec 2.1) vyplývá, že pro konstantní tlak odpovídá nárůst teploty poklesu hustoty. Dále uvažujme pouze dvourozměrný případ simulace, čili výška bude konstantní. Potom z hydrostatické rovnice (Vzorec 2.2) plyne, že pokles hustoty zapříčiní pokles tlaku. Pro naše účely lze tedy předpokládat, že pokles tlaku vyvolá nárůst teploty a tedy nízká teplota odpovídá vysokému tlaku a tím nejnižší teplota tlakové výši a naopak nejvyšší teplota tlakové níži. Tuto úpravu si můžeme dovolit vzhledem k tomu, že se nesnažíme vytvořit matematický model proudění vzduchu, ale pouze jej napodobit, aby výsledek hlavně vizuálně odpovídal skutečnému větrnému proudění. Je také nutné zmínit, že zanedbáváme mnoho dalších faktorů jako vlhkost, nečistoty a další.

Nyní můžeme tvrdit, že rozdíl teplot odpovídá velikosti tlakového gradientu a tedy i rychlosti. Jak již bylo uvedeno (kapitola 4), lze si představit částice jako kvantum přenášeného vzduchu. Toto kvantum může mít definovanou teplotu. Pak se částice s určitou velikostí teploty budou pohybovat rychlostí odpovídající rozdílu teplot mezi přenášeným kvantem a tlakovou níží. Teploty tedy budou hrát velkou roli a bude zapotřebí pro každé místo v namapované oblasti teplotu definovat. Se známými hodnotami teplot pro libovolnou pozici je pak jednoduché určit pro částici s náhodnou počáteční pozicí její teplotu a tedy i rychlost pohybu. Z pozice částice pak lze ještě určit její směr pohybu, neboť základní pohyb vzduchu je ve směru tlakového gradientu a tedy k tlakové níži, což je v našem případě střed buňky s nejvyšší teplotou.

Opět jsem vycházel z představy reprezentovat vítr v každém bodě pomocí vektorového pole, jehož výchozí stav může odpovídat tlakovým gradientům, které je nyní možné nastavit díky známým hodnotám teploty. Z těchto úvah vyplývá první rozdíl ve využívání knihovny oproti implementaci s buněčnými automaty a to, že se budou načítat hodnoty teplot namísto hodnot vektorového pole.

Nyní jsme dosáhli rozpohybování částic a nastavení vektorového pole. Zatím ale částice nijak neovlivňují vektorové pole. Základní myšlenkou celého návrhu totiž je, že částice při svém pohybu nastavují vektory větru v buňkách vektorového pole tam, kde se právě nacházejí. Díky tomu, že jsou částice vytvářeny s různou teplotou a tím i rychlostí pohybu, bude se rychlost v buňkách vektorového pole v čase měnit a to předem nepředvídatelným způsobem, což odpovídá našemu cíli. Dokonce i dvě sousední buňky nemusí mít ve stejnou dobu stejnou hodnotu rychlosti větru. Tím je splněn cíl proměnlivosti rychlosti, ale směr větru zůstává přibližně stejný. Směr větru přímo závisí na pozicích tlakové níže a tlakové výše, neboť v blízkosti tlakové výše bude vítr směřovat směrem od výše a k níži. Je tedy zapotřebí měnit jejich pozice, což ve svém důsledku znamená měnit teploty. Pokud předpokládáme, že částice v částicovém systému přenáší určité kvantum hlavně studených částic vzduchu, lze si představit, že na pozici, kde částice skončí, dojde k ochlazení

(Případně k oteplení u částice teplejší než místo, kde skončily.). Když navíc definujeme život částice náhodně jako u klasického použití částicového systému, bude vývoj teplot značně proměnlivý a budou se tedy proměnlivě měnit i pozice tlakových útvarů. Další významnou silou je Coriolisova síla. Protože neznáme sílu tlakového gradientu ani hmotnost částic, nemá smysl Coriolisovu sílu počítat, a protože si fyzikálně neodpovídají, ani skládat s vektorem rychlosti. Její velikost je proto vypočtena jako pětina vektoru rychlosti a její směr je kolmý na vektor rychlosti. Opět je to ústupek, ale vzhledem k tomu, že nelze vypočítat ani vliv ageostrofického větru, který má také nemalý vliv, je přípustný. Pro větší podobnost s reálným světem lze navíc řídit pohyb částic uvnitř tlakových útvarů tak, aby se pohybovaly ve vírech, jak je popsáno v kapitole o tlakových útvarech (kapitola 2.3). Směr pohybu v tlakové níži a směr působení Coriolisovy síly závisí, na jaké polokouli planety se nacházíme. Já jsem si zvolil severní polokouli kvůli zeměpisné pozici České republiky.

Dále je nutné zmínit, že jsme naše úvahy omezili pouze na proudění větru poblíž zemského povrchu a na konstantní nadmořskou výšku i zeměpisnou šířku, neboť víme, že hodnoty tlaku závisí na prostředí (polární oblasti, oceány, hory a podobně).

Pro reprezentaci částic jsem použil dvě třídy. První obsahuje vykreslení částice. To je zavedeno pro možnost vykreslení všech částic tedy jakýsi debug (ladící) mód pro kontrolu všech pohybů. Vzhledem k velkému počtu vykreslovaných částic jsem namísto složitých tvarů jako koule a podobně zvolil tetrahedron jako prostorový objekt s pouze čtyřmi vrcholy a i čtyřmi stěnami. Druhá třída obsahuje veškerá data potřebná pro naše účely použití částice. Kromě typických vlastností jako délka života, pozice, vektor rychlosti pohybu obsahuje i výše zmíněnou teplotu v místě vytvoření pro výpočet rychlosti a distribuce teploty. Dále obsahuje hlavně metodu pro výpočet pohybu částice. Z parametrů vektor od tlakové výše, vektor k tlakové níži je spolu se současným vektorem rychlosti složen nový výsledný vektor, který se přenásobí rychlostí. Rychlost je předána v parametru a je určena na základě rozdílu teplot a to následujícím způsobem. Z tabulky naměřených tlaků vzduchu (Tabulka 2.1) plyne, že střed rozsahu hodnot je menší než normální atmosférický tlak, který je definován pro teplotu 20°C. Pro jednoduchost jsem zvolil jako střed intervalu teplot hodnotu 0°C a s ohledem na tabulku 2.2 jako maximální odchylku 40. Možné hodnoty teplot jsou tedy z intervalu <-40°C,+40°C>. Maximální možný rozdíl teplot pak odpovídá podle Beufortovy tabulky rychlostí větru (Tabulka 2.3) nejvyšší rychlosti větru, kterou je rychlost 30m/s. Na základě nového vektoru rychlosti a časové diference mezi dvěma posledními voláními metody move pro danou částici se aktualizuje pozice částice.

5.3.1 Volba datových struktur

V aplikaci jsou hodnoty teplot, vektorového pole i dalších pomocných polí uloženy v jednorozměrném poli. Důvodem je snadný průchod přes všechny hodnoty pole, který je vykonáván při každém volání metody pro určení nového stavu. Pro průchod polem totiž stačí pouze inkrementovat index a tím je potlačena nevýhoda reprezentace pomocí jednorozměrného pole, kterou je nutnost přepočítávání indexu ze sloupce a řádku. Toto provedení navíc může přinést podstatné urychlení díky postupnému čtení hodnot uložených za sebou v paměti počítače, neboť jsou z paměti hodnoty čteny po větších částech a pak následně efektivně používány díky cache pamětím procesoru. Dále je zapotřebí uchovávat částice. Protože bude často docházet k odstraňování částic a také k přidávání, zvolil jsem pro jejich uložení flexibilnější variantu a to seznam.

5.3.2 Mapování vektorového pole na terén

Princip mapování se shoduje s první implementací a je popsán výše v kapitole 5.2.1.

5.3.3 Načtení výchozího stavu

Obdobně jako při mé první implementaci jsem pro uložení pole hodnot zvolil bitmapový soubor kvůli snadné přípravě pomocí bitmapového editoru a přehlednosti. Jednotlivé pixely jsou pak při načtení interpretovány jako teploty z důvodů uvedených na začátku kapitoly o výsledné implementaci (kapitola 5.3). Hodnota modré složky barvy je chápána jako záporná teplota. Červená složka má pak význam kladné teploty. Například barva s pouze maximální červenou složkou odpovídá maximální teplotě. Pokud pixel obsahuje jak červenou tak modrou složku barvy, je výsledná teplota určena jako jejich součet. Při stejné velikosti obou složek tak vznikne nulová teplota. Pro zelenou složku při načtení zatím není využití, a tak je její velikost ignorována. Pro správnou funkci knihovny je zapotřebí určit i počáteční stav vektorového pole, který je možné určit na základě teplot, pokud známe pozici minima a maxima, což lze zjistit již při samotném načítání. Posledním krokem v rámci této funkce je vytvořit počáteční množství částic, které pak bude možné při určení nového stavu přesunout na nové pozice. Počáteční pozice částic je vždy generována náhodně v rámci celé namapovávané oblasti. Stejně tak i délka života částic se určí náhodně v daném rozmezí. Naopak vektor rychlosti je již vypočten a to opět na základě pozic minima a maxima teploty. Poslední důležitou hodnotou je teplota částice, která se, jak již víme, určuje na základě pozice vzniku částice. Pozici je možné opět převést na index v poli teplot pomocí principu namapování uvedeném v předcházející podkapitole.

5.3.4 Určení nového stavu vektorového pole

Před výpočtem nových vektorů větru je nutné nejprve pohnout částicemi a určit jejich vliv. Tuto funkčnost jsem umístil do jediné metody pro oddělení kroků. Metoda obsahuje průchod všemi částicemi, které jsou umístěny v datové struktuře seznam. U každé částice pak dojde k posunu částice pomocí jejich metody pro pohyb, která vyžaduje rychlost určenou v našem případě na základě teploty a směru od tlakové výše a k tlakové níži. Všechny parametry jsme schopni určit díky globálním proměnným pro uchování pozice obou tlakových útvarů. Po přesunu částice dojde k připočtení jejího vektoru rychlosti do buňky pomocného pole, jejíž index je opět určen z nové aktuální pozice částice. Také dojde k inkrementaci počtu dělitelů na stejném indexu v dalším pomocném poli, aby bylo pak možné správně určit průměrný vektor od všech částic ve stejné buňce pole.

Přenášením tepla z místa zrození částice na místo jejího zániku má za následek vývoj teplot v celém poli teplot. Bohužel se zde negativně projevuje náhodnost vzniku částic. Při uvažování dokonalého generátoru čísel a při velkém počtu volání docílíme rovnoměrného rozložení částic. My ale nechceme, aby v průběhu času měly dvě sousední buňky znatelně rozdílnou teplotu. Je proto nutné do programu zahrnout i vliv lokálního mísení teplot. Pole teplot si lze ale také představit jako bitmapový obrázek s jedním kanálem barvy, což je právě teplota. Potom se přenášení tepla projevuje jako vnášení šumu do obrázku. Tento šum může potencionálně vytvářet ony nerovnoměrnosti sousedních buněk, proto jsem se jej snažil redukovat odpovídající metodou pro úpravu bitmapových obrázků.

Odstranění šumu nebo také filtrace šumu je postup založený na vyhodnocení okolí pixelu a zněj se pak usuzuje, zda jde o šum či ne. Za šum se považují i u bitmapových obrázků velké změny hodnot sousedních pixelů. Filtry odstraňující šum z jediného obrazu pracují buď na principu konvoluce (matematická operace mezi obrázkem a filtrem) nebo lokální statistiky okolí. Vzhledem k tomu, že se snažíme o lokální mísení teplot, lze použít jeden z nejzákladnějších postupů a tím je průměrování neboli rozmazávání obrázku, které dobře odpovídá i opravdovému mísení teplot v reálném světě.

Tuto techniku lze také najít pod názvem obyčejné průměrování [ŽBSF2004] a jako nejjednodušší filtrovací metoda pracuje na principu výpočtu hodnoty pixelu jako průměr z

jeho hodnoty a zároveň hodnot z jeho okolí. Díky tomu pak zmizí vysoké frekvence a ostré změny v obraze jako například hrany či šum se tím rozmažou. Pokud bude filtr opakovaně aplikován na obraz, dojde ke stále většímu rozmazávání, až dokonce může v limitě vést k obrazu, který má ve všech pixelech jednu barvu, která je průměrem ze všech hodnot. Hodnotu průměru lze vypočíst s různými vahami uvažovaných pixelů. Pro zápis vah se používá takzvané konvoluční jádro, které se definuje většinou pro okolí 3 x 3, přičemž obvykle nabývá jednoho z následujících tvarů:

$$h = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Vzorec 5.2 - Konvoluční jádro.

Snížením vah směrem od středu získáme konvoluční jádro (Vzorec 5.3), které snižuje Gaussův šum:

$$h = \frac{1}{16} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Vzorec 5.3 - Konvoluční jádro pro eliminaci Gaussova šumu.

Součet všech hodnot v konvolučním jádře vynásobený koeficientem, který je uveden před maticí, musí být roven jedné jinak by se výsledný obraz zesvětloval nebo naopak ztmavoval.

Ve svém programu jsem pro výpočet použil první konvoluční jádro (Vzorec 5.2), avšak nezávisle na volbě jádra dochází během několika desítek kroků ke zprůměrování na jedinou hodnotu v celém obrazu, což známe z první implementace, kde se místo teplot průměrovaly vektory větru. Abych předešel zmíněnému nechtěnému efektu, provádí se mísení vzduchu vždy pouze po nastaveném počtu aktualizací. Tato hodnota je také jedním z klíčových faktorů ovlivňující celkové chování systému, a proto je možné ji libovolně nastavit díky veřejnému atributu knihovny.

Poslední funkčnost, která je vytvořena v rámci této funkce, je korekce vypočtené rychlosti. Po vložení tyčí na povrch terénu, které se podle větru naklánějí, vynikl další nechtěný jev, neboť při vstupu částice do buňky pole je okamžitě ovlivněna rychlost v buňce, což na pohled působit trhaně. Bylo by vhodné nějakým způsobem korigovat rychlost. Po delším uvažování jsem jako nejlepší možnost zvolil spojení rychlosti s časem a v podstatě je vynést do kartézské soustavy souřadnic. Body pak lze proložit křivku, díky níž je možné určit mezilehlé body. Tento přístup je nejlepší zejména z hledisek jednoduchosti a zároveň přesnosti výpočtu, neboť existují algoritmy pro výpočet bodu ležícího na křivce pouze z řídících bodů.

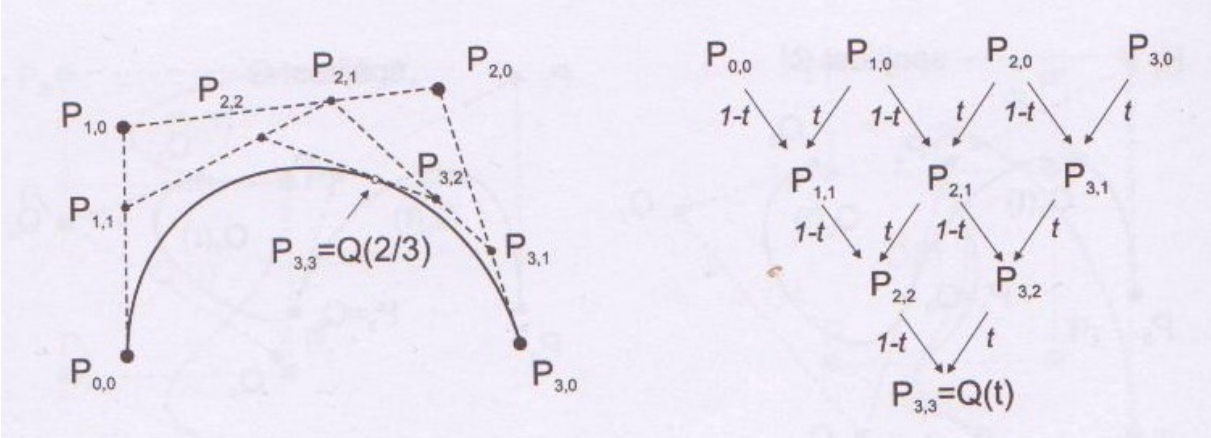
Jedním z možných řešení, jak vypočítat bod křivky dané řídícími body, je algoritmus de Casteljou [ŽBSF2004]. Jedná se o metodu založenou na rekurentním vztahu, který je schopen vyčíslit pro parametr jdoucí od 0 do 1 každý bod křivky (pro hodnotu 0 vypočte první bod a naopak pro hodnotu 1 poslední bod na křivce). Díky dalším vlastnostem tohoto přístupu je metoda schopna vykreslit celou křivku, ale pro naše účely plně postačuje schopnost vypočítat bod ležící na křivce podle následujícího vztahu.

$$P_{j,i}(t) = (1-t)\,P_{j-1,i-1} + tP_{j,i-1}$$

Vzorec 5.4 – Výpočet nového bodu algoritmu de Casteljou.

Do vzorce 5.4 se pak za i dosazují hodnoty {1,2,…,n} a za j hodnoty {i,i+1,…,n}. Jako vstup do algoritmu se kromě zmíněného parametru t použijí hodnoty řídících bodů

křivky, která je definována většinou čtyřmi body. Vstupní body P_i tak vlastně představují ve vzorci hodnoty $P_{i,0}$. Výsledkem pak je $P_{n,n}(t)$, což je bod ležící přímo na křivce pro požadovaný parametr. Jaký je princip postupu a jak se postupně dosazují body do vzorce, je vidět na obrázku 5.5.



Obrázek 5.5 – Princip algoritmu de Casteljou

Použití tohoto algoritmu přináší další významnou hodnotu, tedy nastavení parametru křivky. Podle provedených simulací jsem jej nastavil malý (0,05), aby dostatečně redukoval vysoké výkyvy. Dále jsem z hlediska použité paměti zvolil uchovávání pouze jediné hodnoty navíc a to předchozí vypočtené rychlosti pro každou buňku. Body, kterými bude křivka proložena, jsou proto aktuální rychlost, předchozí vypočtená rychlost a nově vypočtená rychlost.

Na konci metody ještě dochází k přesouvání tlakových útvarů, pokud je to zapotřebí, a k vytváření nových částic. Již při procházení polem vektorů (teplot) jsou nalezeny oba indexy extrémů teplot. Nová pozice extrému ale není určena z indexu nového maxima (minima), nýbrž z pozice staré, z které se přesune odpovídající rychlostí směrem k novému extrému. I když díky tomuto řešení může dojít k situaci, kdy mezi oběma pozicemi bude extrémně jiná teplota, která ovlivní rychlost, dochází tak hlavně k odstranění jevu, který vznikne, pokud jsou teploty v určité oblasti stejné, čímž dojde v téměř každém momentu ke změně pozice extrému a tím ke změně všech směrů. Tento efekt není ani vizuálně přípustný, protože opět působí trhaně, a ani neodpovídá situaci v přírodě, kde se přesouvá tlakový útvar jako celek, popřípadě zanikne.

Částice jsou opět vytvářeny náhodně a je jich vytvořeno tolik, aby doplnily maximální počet. Některé částice (5%) jsou stvořeny jako „rychlé“. Jsou totiž vytvořeny s rychlostí odpovídající rozdílu teplot tlakové výše a níže. Tyto částice vnášejí při průletu maximální možnou rychlost pro aktuální extrémní teploty, čímž lze dosáhnout vyšších rychlostí tak, aby více odpovídaly právě rozdílům v extrémních teplotách. Tato úprava má vliv především na celkovou reálnost simulace větru.

5.3.5 Zjištění vektoru větru v daném místě

Jak bylo výše uvedeno, lze díky mapování určit relativní pozicí vůči rohu mapy. Relativní pozicí se rozumí odečtení od souřadnic rohu od hledaného bodu a vydělení hodnoty x šířkou a hodnoty z výškou. Jednotlivé hodnoty relativní souřadnice (z intervalu <0,1>) vynásobíme počtem řádků/sloupců a tím zjistíme potřebné hodnoty pro určení indexu ve vektorovém poli. Protože jsem vektorové pole implementoval jako jednorozměrné pole, je ještě nutné index řádku přenásobit počtem sloupců a přičíst k indexu sloupce. Tímto postupem dostaneme hledaný index ve vektorovém poli. Vektor větru na tomto indexu pak je návratovou hodnotou funkce.

5.4 Ukázková aplikace

Pro ukázku funkce knihovny jsem vytvořil aplikaci umožňující procházet terénem na vymyšlené planetě.

5.4.1 Funkce ukázkové aplikace

Terén je zadaný výškovou mapou (polem 128x128) a je vykreslen pomocí pravidelné trojúhelníkové sítě. Výšková mapa je vždy načtena z binárního souboru, kde každý jeden byte reprezentuje jednu výšku mapy.

Terénem lze procházet pomocí klávesnice (pohyby vpřed, vzad, úkroky doprava a doleva) zároveň se může pozorovatel pomocí myši otáčet. Výška terénu je mezi body zadanými výškovou mapou interpolována, aby byl pohyb po terénu plynulý. Kvůli reálnosti pohybu se pozorovatel pohybuje rovnoměrnou rychlostí vzhledem ke skutečnému času. Pohyb je rovnoměrný i při pohybu z/do kopce.

Pro přehlednost je možné kromě pohledu pozorovatele pohybujícího se po planetě využít i pohled shora. Tento mód je přehlednější především při vizualizaci proudění pomocí obarvení vrcholů terénu nebo při vykreslování částic (viz níže).

Pro snazší provádění testů na zatížení se v aplikaci přímo na průmětnu vykresluje aktuální FPS (počet snímků za sekundu), neboť je z ukazatele jasně zřejmé, jaká zátěž (velikost vektorového pole, počet částic) už působí problémy s dodržením požadavku na plynulý běh v reálném čase.

Dále jsem pro možnosti interaktivní komunikace s aplikací implementoval konzoli podobnou té z počítačových her. Podobně jako ukazatel FPS se vykresluje přímo na průmětnu. Při zapnutí konzole jí je předáno řízení reakcí na vstup z klávesnice. Umožňuje několik příkazů s využitím především při testování zátěže jako nastavení koeficientu počtu částic v systému nebo zvýšení počtu tyčí (viz následující kapitola) pro vizualizaci proudění. Kromě toho umožňuje zadat požadavek na načtení výchozího stavu ze souboru, který je uveden v parametru příkazu. Poslední z významných funkcí, je nastavení příznaku pro umístění unášeného předmětu zpět na výchozí pozici. Všechny výše uvedené příkazy nastavují veřejné příznaky a atributy, které lze v příslušných třídách a jejich metodách vhodně vyhodnotit.

Z pohledu ověření funkcí knihovny pro modelování větru je nejdůležitější funkce pro vizualizaci větru. Možné varianty zobrazení větru a jejich použití jsou popsány v následující podkapitole.

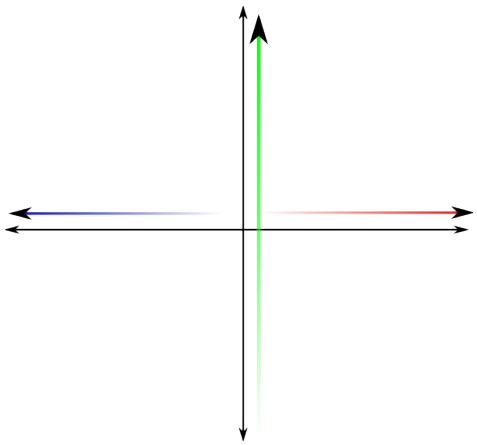
5.4.2 Vizualizace větru

Pro vizualizaci větru jsem využil hned několik postupů, aby vynikly všechny vlastnosti modelování větru. Postupy využívají metody knihovny a to buď pro získání teploty nebo vektoru větru. Asi nejjednodušší variantou je obarvení vrcholů terénu. Pro zobrazování směru a zároveň velikosti větru na základě obarvení jsem zvolil dva následující způsoby:

a) statické obarvení vrcholů terénu

Pro každý bod terénu je z výše popisované knihovní metody pro určení posuvu získána transformace (posun), která by byla aplikována na volný předmět. Z transformační matice lze určit vektor posuvu a podle něj je pak provedeno obarvení. Obarvení jsem zvolil takto: složka z určuje velikost zelené složky v barvě reprezentované pomocí RGB. Kladné hodnoty složky x určují velkost červené barvy a záporné hodnoty složky x reprezentují velikost modré barvy. Tyto složky pak dávají výslednou barvu v RGB barevném systému.

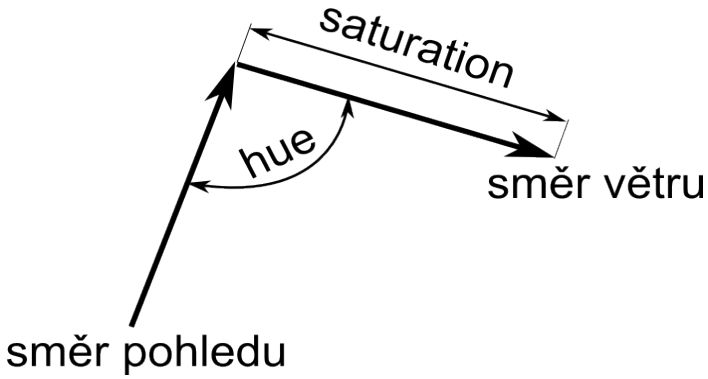
Na obrázku 5.6 je vidět, jak jednotlivé složky vektoru ovlivňují barvu. Vodorovná osa je osa X. Svislá osa je osou Z.



Obrázek 5.6 – Ovlivnění barevných složek podle směru větru při statickém obarvení.

b) dynamické obarvení vrcholů terénu vůči natočení pozorovatele

Vektor posuvu je získán stejným postupem jako v případě statického obarvení, ale informace z vektoru je využita jiným způsobem. Barva vrcholu je určena pomocí HSB barevného systému. Složka H (hue) je dána úhlem, který svírá vektor posuvu s vektorem pozorovatele. Složka S (saturation) je velikost vektoru posuvu a složka B (brightness) je trvale nastavena na hodnotu jedna. Z popisu je vidět, že obarvení vrcholu je závislé na natočení pozorovatele. Názorně situaci ukazuje obrázek 5.7.



Obrázek 5.7 – Ovlivnění HSB složek barvy podle směru větru a pohledu při dynamickém obarvení.

Oba výše uvedené způsoby jsou použity u první implementace pomocí buněčných automatů.

Obdobně jako u první implementace i u varianty s částicovými systémy dobře poslouží postup s obarvením vrcholů, ale tentokrát se vrcholy obarvují podle teploty, která je významným ukazatelem chování knihovny, či rychlosti větru v daném místě. Mezi oběma obarveními se lze snadno přepínat. Pro jejich odlišení se pro teplotu používá modrá složka (záporné teploty), červená složka (kladné teploty) a pro zobrazení absolutní velikosti vektoru větru pak zbývá zelená složka.

Dále jsem u druhé implementace použil i již zmiňovanou variantu se zobrazením objektů v terénu, které se naklánějí podle směru a rychlosti větru. Terén jsem tedy osadil několika stovkami dřevěných tyčí. Vzhled dřevěných tyček zajišťuje třída StaffVisual. Třída, která obsahuje pozici tyče a odkaz na třídu se vzhledem, je Staff. Tyč jako takovou představuje desetistěn, jehož výška je v rozsahu od 0 do 1 tak, aby s ním bylo možné

jednoduše rotovat a transformovat jej na velikosti odpovídající rozměrům mapy. Při vykreslení se kromě translace na cílové umístění uplatní právě i rotace kolem osy y pro otočení do směru větru a rotace kolem osy z pro určení sklonu představující sílu větru.

Také jsem dle požadavku v zadání demonstroval vítr na vzduchem unášeném předmětu. Předmětem je v aplikaci koule, jejíž pozice se mění na základě výpočtu velikosti posuvu z rychlosti větru a časové diference mezi dvěma posledními voláními metody pro vykreslení.

Nakonec ještě zmíním, že lze volat debug (ladící) funkci u knihovny využívající částicové systémy. Tato funkce má za následek vykreslení částic větru, což jsem v aplikaci použil při pohledu shora, kde může pohyb i chování částic vyniknout.

6 Testy

6.1 Testy implementace pomocí buněčných automatů

Abych mohl zhodnotit první způsob provedení knihovny větru, provedl jsem test na počet aktualizací, při kterých dochází ke změnám vektorového pole. Obě provedení určení nového stavu totiž po určitém čase vedou na triviální případy proudění. První způsob výpočtu z celého 4-okolí vede k bezvětří nebo minimální velikosti vektorů celého vektorového pole (dojde vlastně ke zprůměrování na jedinou hodnotu v celém poli). Druhý způsob využívající jen vektory 4-okolí, které směřují k počítanému vektoru, nemusí vždy nutně skončit v bezvětří, ale může skončit například ve stavu stálého proudění větru jedním směrem, který se již nezmění. U obou tedy po čase dojde k tomu, že každé další volání metody pro určení nového stavu nijak nezmění stav předchozí. Počet kroků pro různé velikosti vektorového pole, které jsem naměřil při testu, znázorňuje tabulka 6.1.

Velikost vektorového pole	50 x 50		100 x 100		200 x 200	
Verze Implementace	4-okolí	4-okolí + směr	4-okolí	4-okolí + směr	4-okolí	4-okolí + směr
FPS	47,7	46,61	41,44	37,75	30,81	28,62
Počet aktualizací	656	2094	2562	18161	10513	82290
Časová výdrž [s]	21,87	69,8	85,4	605,37	350,43	2743

Tabulka 6.1 – Porovnání výdrže obou implementací určení nového stavu vektorového pole.

Položka Časová výdrž představuje dobu, jakou by byla schopna implementace vydržet, než dojde k ustálení při volání funkce 30krát do vteřiny (minimální hodnota pro běh v reálném čase je 25krát do vteřiny + rezerva). Z tabulky vyplývá, že časový úsek mezi načtením a ustálením stavu přímo závisí na velikosti vektorového pole. Zvětšování vektorového pole ale vede ke zvětšení doby výpočtu, jak ukazuje FPS (počet snímků za sekundu). Větší vektorová pole by tedy při současné verzi implementace už nesplňovala požadavek na běh v reálném čase. Dále je nutné podotknout, že výše uvedené časové úseky jsou doby do úplného ustálení. Při subjektivním pozorování totiž dochází k neměnnosti vektorového pole mnohem dříve, často dokonce již před polovinou výše uvedeného času.

Pro obarvení terénu jsem využil statické a dynamické obarvení, kde první varianta je založena na obarvení vrcholů terénu podle systému RGB. V tomto provedení je přibližně vidět, kam směřují jednotlivé vektory větru. Naproti tomu dynamické obarvení vůči natočení pozorovatele je provedeno díky barevnému systému HSB. Při druhém způsobu pak vyniknou hlavně přechody mezi jinými směry proudění větru.

6.2 Testy implementace pomocí částicových systémů

V průběhu popisu druhé implementace jsem se zmínil o několika důležitých parametrech, které výrazně ovlivňují chování celého systému. Asi nejdůležitějším faktorem je celkové množství částic v systému. Z hlediska různých možných velikostí vektorového pole lze usuzovat, že počet částic by měl velikosti odpovídat. V knihovně je proto zaveden veřejný atribut, kterým je přenásoben jeden z rozměrů vektorového pole (počet řádků nebo sloupců). Pokud má pole různé rozměry, pak je rozměr vypočten jako odmocnina z jejich součinu. Při použití tak stačí nastavovat pouze onen veřejný atribut pro přenásobení. Můj první test jsem

tedy zaměřil právě na hodnoty koeficientu.

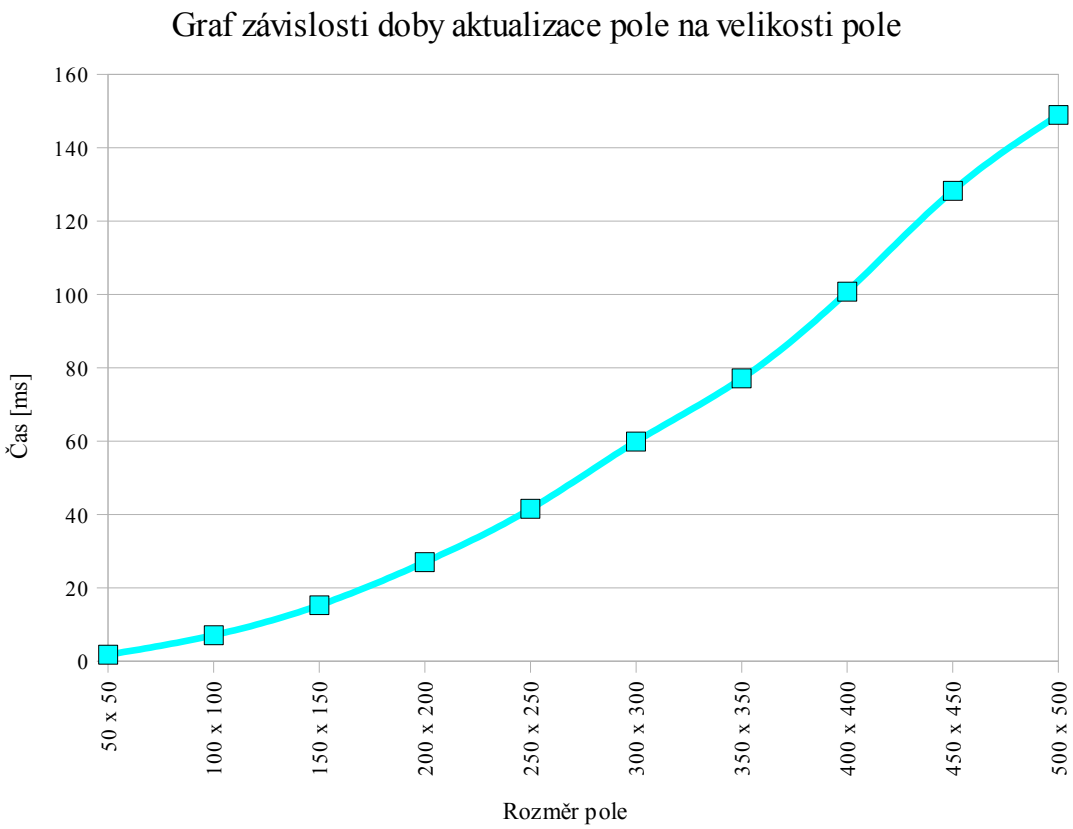
V příloze 1 jsou uvedeny obrázky vývoje teplot pro koeficienty 1, 5, 10, 20. Pro koeficient 1 jsou přiloženy ukázky pro 0, 2, 4, 6 a 9 hodin. Vyplývá z nich, že pro tak nízkou hodnotu je vývoj teplot velmi pozvolný a ani po 9 hodinách nedochází k ustálení. Bohužel se ale vytváří tak málo částic, že nestačí vhodným způsobem aktualizovat vektorové pole větru, jak je vidět na obrázku 7 (příloha 1). Dokonce ani výsledný dojem nepůsobí celistvě, což zobrazuje obrázek 6 (příloha 1). Pro koeficienty 5 a 10 jsou uvedeny obrázky v časech 0, 2, 4, 6 h. Z vývoje teplot je vidět, jak postupně dochází k ustálení. Pro hodnotu 10 jsou teploty v čase 6 h již téměř vyrovnané, ale k úplnému ustálení dojít nemusí. Z pozorování také usuzuji, že s rostoucím koeficientem se systém chová dynamičtěji a dochází častěji k posuvům tlakových útvarů. Posledním testovaným koeficientem byla hodnota 20. Jak opět znázorňují obrázky, systém s takto vysokou hodnotou se stává již velmi brzo nestabilním. Nestabilním ve smyslu nereálného vývoje teplot, neboť mnoho částic přenáší tolik chladu, že mohou ochladit tlakovou níži na hodnotu nižší než tlaková výše, čímž dojde k obrácení směru proudění. Tento nežádoucí jev zobrazuje obrázek 17 (příloha 1). Již po jedné hodině a asi 30 minutách dochází ke zkolabování, a jak ukazují obrázky 18 a 19 (oba příloha 1), k nesmyslným směrům proudění. Z testů tedy vyplývá, že čím nižší hodnota, tím je systém stabilnější, trvanlivější a se statickým chováním. Naopak čím vyšší hodnota, tím více se stává nestabilním, zkracuje se doba do ustálení, ale získává více na dynamičnosti. Podle mého názoru se jeví jako optimální hodnoty mezi 5 a 10.

Druhý test se zaměřuje na zjištění času výpočtu pro různé velikosti vektorového pole. Díky tomuto testu lze také stanovit maximální velikost vektorového pole při respektování požadavku na simulaci v reálném čase. Naměřené hodnoty prezentuje tabulka 6.2.

Rozměr vektorového pole	Čas pro vykreslení terénu [ms]	Čas pro výpočet nového stavu [ms]	Čas pro vykreslení tyčí podle větru [ms]
50 x 50	12,06	1,77	5,8
100 x 100	13,18	7,11	6,99
150 x 150	13,57	15,26	7
200 x 200	13,29	27,02	7,16
250 x 250	14,39	41,52	6,66
300 x 300	14,45	59,86	6,97
350 x 350	13,89	77,08	6,34
400 x 400	13,19	100,76	6
450 x 450	13,23	128,25	6,58
500 x 500	13,29	148,93	6,65

Tabulka 6.2 – Naměřené časy jednotlivých částí programu v závislosti na velikosti pole

Tabulka poukazuje na skutečnost, že s rostoucí velikostí pole se prakticky nemění čas pro vykreslení terénu (kde se aktualizují barvy vrcholů terénu), ani čas potřebný pro aktualizaci poloh tyčí. Co se naopak mění, je čas pro určení následujícího stavu vektorového pole, jak také ukazuje graf 6.1.



Graf 6.1 – Závislost doby aktualizace pole na velikosti pole

Z grafu pak lze určit, že přibližně vektorové pole 250 x 250 buněk ještě splňuje limit 40ms (25 aktualizací do vteřiny) pro simulaci v reálném čase.

Předchozí testy se specializovaly především na zátěž a stabilitu. Dalším vhodným testem je ověření kvality. Jelikož ale není k dispozici žádná implementace proudění větru v reálném čase, se kterou by bylo možné funkce programu porovnat, provedl jsem pouze subjektivní posouzení kvality. Dle mého názoru a názoru vedoucího práce je chování větru dostatečně věrohodné.

7 Závěr

Po prozkoumání chování reálného větru a způsobů pro jeho modelování jsem usoudil, že pro napodobení větru s ohledem na požadavek běhu v reálném čase přichází v úvahu modelování pomocí buněčných automatů nebo částicových systémů. Poté, kdy jsem naimplementoval první způsob, jsem z testování a pozorování běhu zjistil, že je schopen pracovat v reálném čase, ale až na triviální konfigurace a minimální požadavky na reálné chování nelze toto provedení pro účely samočinného vývoje větru použít. Systém velmi brzo skončí v jednom stavu, chová se předvídatelně a „nežije vlastním životem“, i když druhé provedení implementace s buněčnými automaty jeví z počátku běhu tendence nepředvídaného chování. Je nutné jim dodávat okrajové podmínky, tedy vlastně vítr nebo jeho zdroj, a pak by dokázaly vhodně reagovat. Cílem práce ale bylo ve své podstatě takovéto podmínky vytvářet, a proto jsem přistoupil k implementaci pomocí částicových systémů. Vývoj větrného pole pak vychází z vývoje teplot, ze základních fyzikálních principů a z reálných hodnot rychlostí větru a teplot. Při samotné realizaci jsem neuvažoval mnoho dalších faktorů jako vlhkost, periodické změny tlaku a podobně. Řešení navíc nevychází z přesného vývoje teplot se zachováním celkového množství teploty. Z těchto důvodů nelze v žádném případě řešení považovat za přesný matematický model proudění, což ale ani nebylo cílem práce.

V rámci vlastní implementace jsem vytvořil knihovnu s rozhraním, která umožňuje aplikacím volat tři základní funkce a to: načtení pole teplot z bitmapy, určení nového stavu vektorového pole a určení vektoru větru, případně teploty na souřadnicích. Pro správnou funkčnost všech těchto funkcí je díky informacím při inicializaci provedeno namapování vektorového pole na terén. Dále jsem vytvořil testovací aplikaci, která ze souboru s údaji výškové mapy dokáže vytvořit pravidelnou trojúhelníkovou síť představující terén. Po terénu je možné se volně procházet. Aplikace především umožňuje zobrazit vektorové pole několika způsoby. Kromě dřevěných tyček rozmístěných po terénu, které se dle větru naklánějí, jsem použil i obarvení terénu na základě směru i rychlosti větru (první implementace) a teploty nebo rychlosti (druhá implementace).

Naopak podle vývoje teplot i zobrazení větru pomocí tyčí lze říci, že použití částicových systémů splňuje cíle: samočinný vývoj i reálné chování. Při vhodném nastavení nemusí vývoj bez dalšího zásahu skončit ani po několika hodinách (i mnohem více jak 10 hodinách) na rozdíl od implementace pomocí buněčných automatů, kde vývoj končil maximálně po několika desítkách minut. Zároveň je dobré si uvědomit, že i když se po několika hodinách zdá pohyb tyčí nevýrazný, i tak vývoj větru nekončí a může se nadále rozvíjet. Maximální možný sklon tyčí je definován pro rychlosti 30 m/s, což v realitě odpovídá síle orkánu. Z toho plyne, že i minimální pohyby tyčí v mé aplikaci mohou odpovídat rychlostem řádově jednotek metrů za sekundu, což jsou běžné hodnoty větru. Z popisu implementace ale plyne, že vývoj by měl po určitém čase skončit, zejména díky průměrování (mísení vzduchu). Pro zabránění tohoto jevu by musel být pravděpodobně přidán nějaký dodatečný zdroj tepla, což by odpovídalo i reálnému světu, kde by planeta Země také vychladla nebýt Slunce. Z důvodu lepšího vizuálního efektu jsem navíc zahrnul i korekci rychlosti větru pro plynulejší přechody při skokových změnách rychlosti. Při testech byly zjištěny, kromě již zmiňované doby vývoje, i velikosti vektorových polí, pro která pracuje aplikace ještě v reálném čase bez problémů (počet snímků za sekundu je větší jak 25).

Druhý přístup tedy splňuje všechny cíle vytyčené v úvodu a je možné jej použít pro pohyb mraků či vegetace v aplikacích virtuální reality, což bylo důvodem vzniku této práce.

Použité zdroje

[CD1998] Chen S., Doolen G. D., Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics* [online]. 1998, vol. 30, no. 1, [cit. 2009-04-19]. Dostupné z http://dx.doi.org/10.1146/annurev.fluid.30.1.329

[CHMI2009] Český hydrometeorologický ústav. *Analýza synoptické situace Termin: 16.04.2009 06 UTC.* [online], Dostupné z http://www.chmi.cz/meteo/om/inform/asyne.html, poslední úpravy 16.4.2009, [cit. 2009-04-16]

[FHP1986] Frisch U., Hasslacher B., Pomeau Y., Lattice-gas automata for the Navier-Stokes equations. *Physical Review Letters.* [online], April 1986, vol. 56, [cit. 2009-04-19]. Dostupné z http://www.iop.org/EJ/article/1402-4896/40/3/027/physscr_40_3_027.pdf

[HPP1976] Hardy J.,Pazzis O., Pomeau Y., Molecular dynamics of a classical lattice gas: transport properties and time correlation functions. *Physical Review Letters.* [online], May 1976, vol. 13, [cit. 2009-04-19]. Dostupné z http://adsabs.harvard.edu/abs/1976PhRvA..13.1949H

[Hei2008] Heinemann Detlev. *Wind Flow Modeling, The Basis Resource Assessment and Wind Power Forecasting*. [online], Dostupné z http://www.forwind.de/events/files/heinemann_vortrag.pdf, poslední úpravy 11.9.2008, [cit. 2009-04-17]

[KB2005] Kopáček Jaroslav, Bednář Jan. *Jak vzniká počasí*. 1. vydání. Praha: Karolinum, 2005, 226 s., ISBN 80-246-1002-7

[LYD2001] Lesieur M., Yaglom A., David F. *New trends in turbulence : école de physique des Houches-UJF & INPG-Grenoble, a NATO Advanced Study Institute, Les Houches, Session LXXIV, 31 July-1 September 2000 = Turbulence nouveaux aspects.* Berlin: Springer, 2001, 554 s., ISBN 3-540-42978-6

[PS2008] Pidwirny Michael, Slanina Sjaak. *Wind*. [online], Dostupné z http://www.eoearth.org/article/Wind, poslední úpravy 16.3.2008, [cit. 2009-04-18]

[Rev1983] Reeves W. T., Particle system-a technique for modeling a class of fuzzy objects. *ACM Transaction on Graphics.* [online], April 1983, vol. 2, [cit. 2009-04-19]. Dostupné z http://graphics.stanford.edu/courses/cs448-01-spring/papers/reeves.pdf

[Ur2007] Uruba Václav. *Dynamika přechodových vrstev.* [online], Dostupné z http://cvut.cz/pracoviste/odbor-rozvoje/dokumenty/hab_inaug/hp/hp2007/habilitacni_prednaska.pdf, poslední úpravy 12.3.2007, [cit. 2009-04-18]

[Vaš2008] Vašíček J. Beufortova stupnice. [online], Dostupné z http://www.chmi.cz/meteo/olm/Let_met/beaufort/Beaufortova_stupnice.htm, poslední úpravy 7.2.2008, [cit. 2009-04-16]

[WLMK2003] Wei Xiaoming, Li Wei, Mueller Klaus, Kaufman Arie. *The Lattice-Boltzmann*

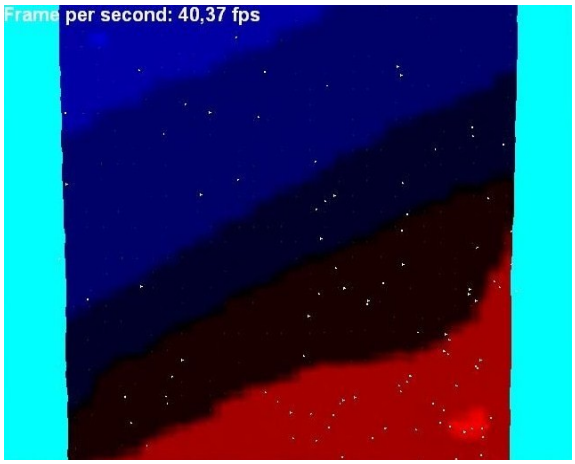
Method for Gaseous Phenomena. [online], Dostupné z http://www.cs.sunysb.edu/~vislab/projects/amorphous/WXMWebsite/smoke.pdf, poslední úpravy 11.3.2003, [cit. 2009-02-22]

[ŽBSF2004] Žára Jiří, Beneš Bedřich, Sochor Jiří, Felkel Petr. *Moderní počítačová grafika*. 2.Vydání. Brno: Computer Press, 2004, 612 s., ISBN 80-251-0454-0

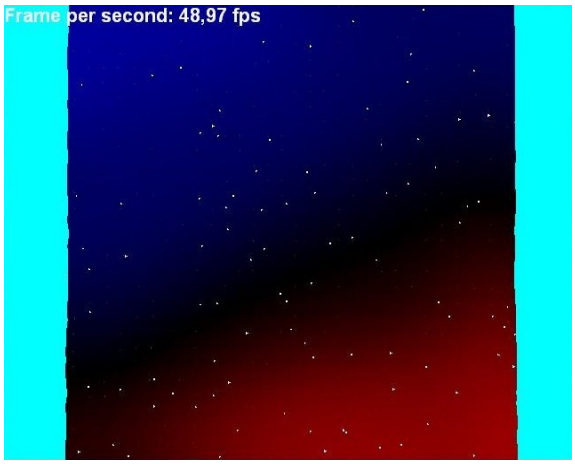
Příloha 1 – ukázky z aplikace

Implementace pomocí částicových systémů

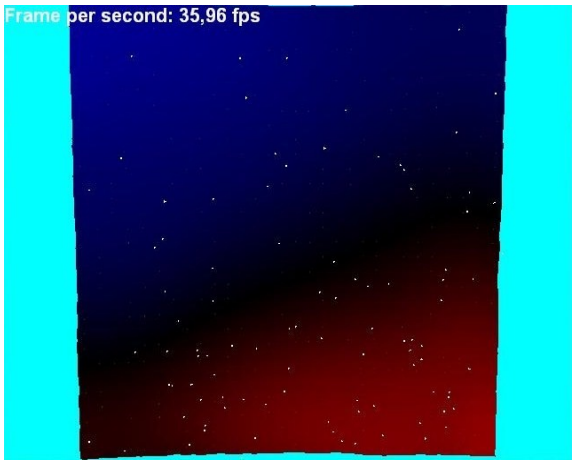
Koeficient 1



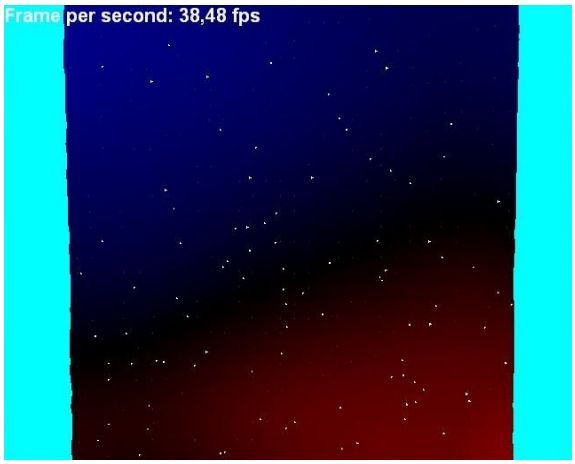
Obrázek 1 - teploty v čase 0h



Obrázek 2 - teploty v čase 2h



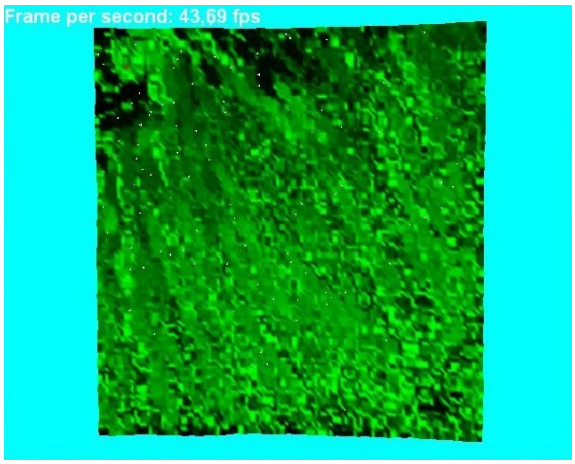
Obrázek 3 - teploty v čase 4h



Obrázek 4 - teploty v čase 6h

Obrázek 5 - teploty v čase 9h

Obrázek 6 - nestejný pohyb tyčí



Obrázek 7 - nedostatečná aktualizace rychlosti

Koeficient 5

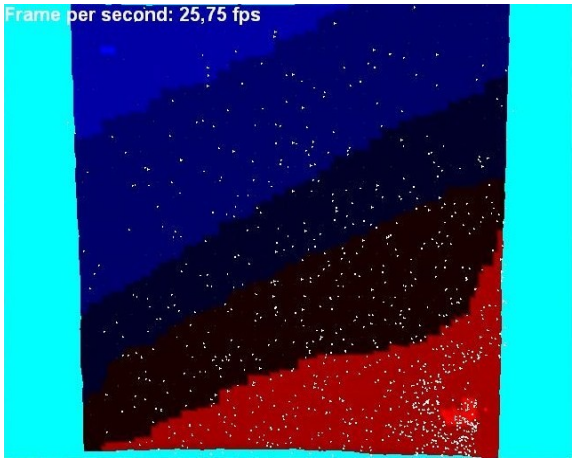
Obrázek 8 - teploty v čase 0h

Obrázek 9 - teploty v čase 2h

Obrázek 10 - teploty v čase 4h

Obrázek 11 - teploty v čase 6h

Koeficient 10



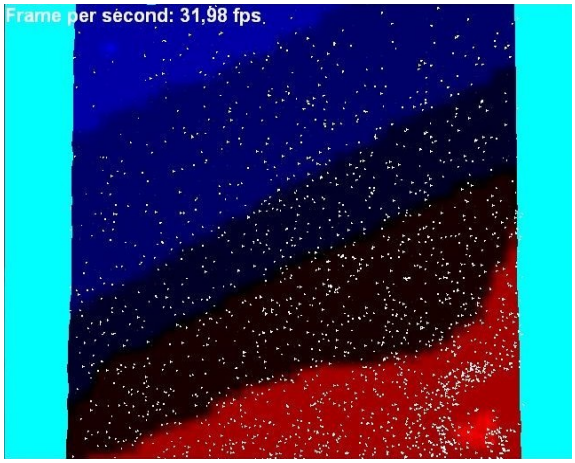
Obrázek 12 - teploty v čase 0h

Obrázek 13 - teploty v čase 2h

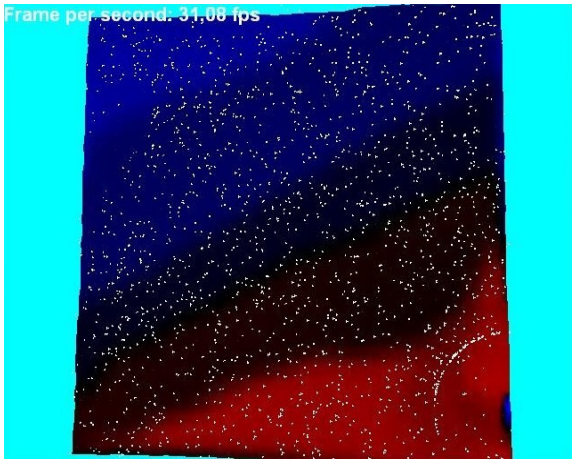
Obrázek 14 - teploty v čase 4h

Obrázek 15 - teploty v čase 6h

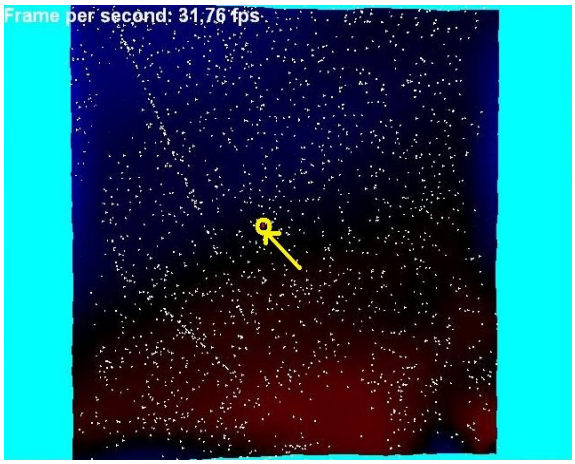
Koeficient 20



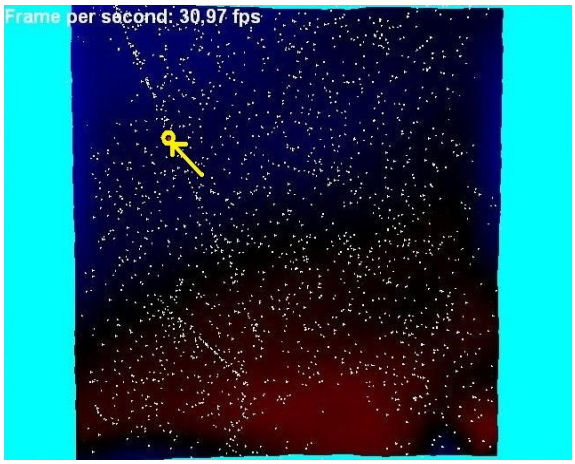
Obrázek 16 - teploty v čase 0h



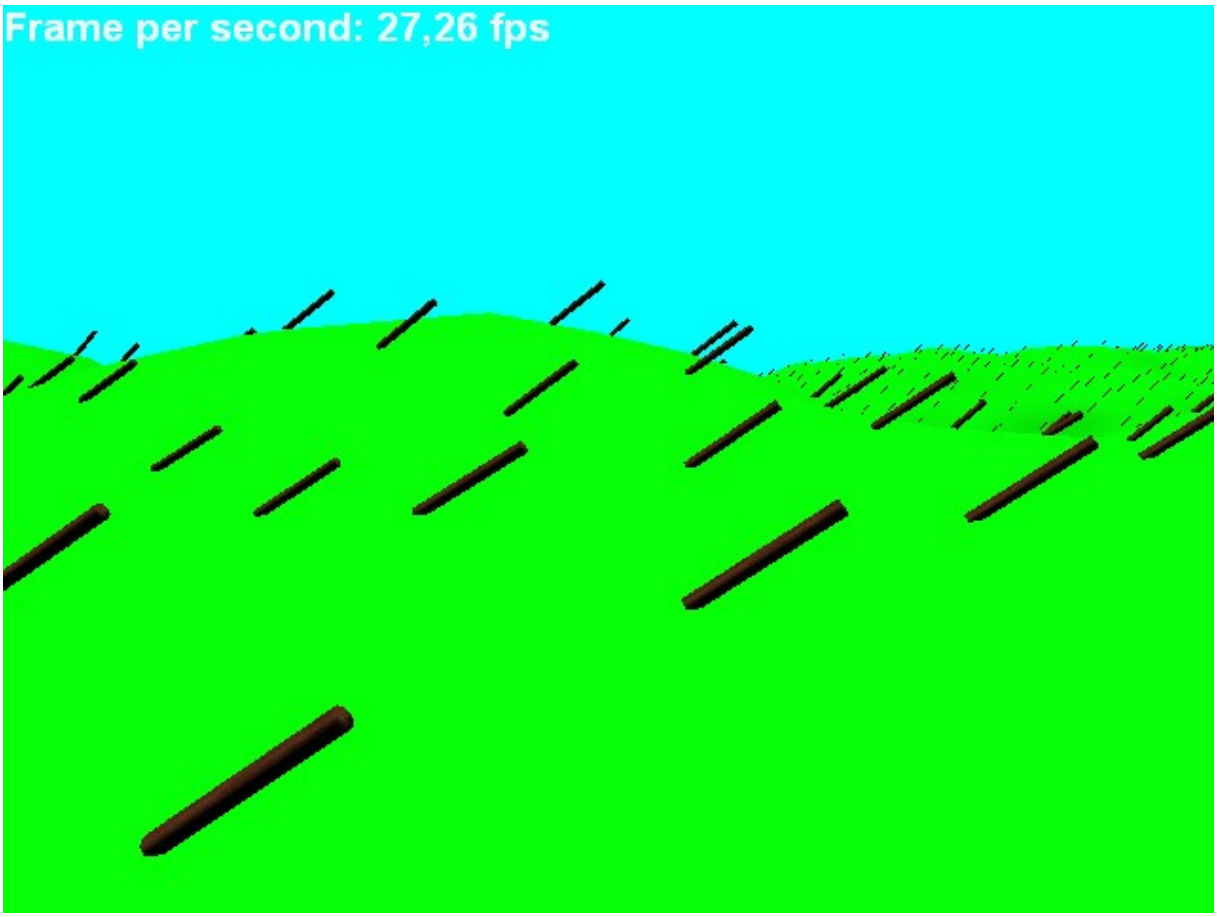
Obrázek 17 - první nestabilní stav v čase 0h 20min



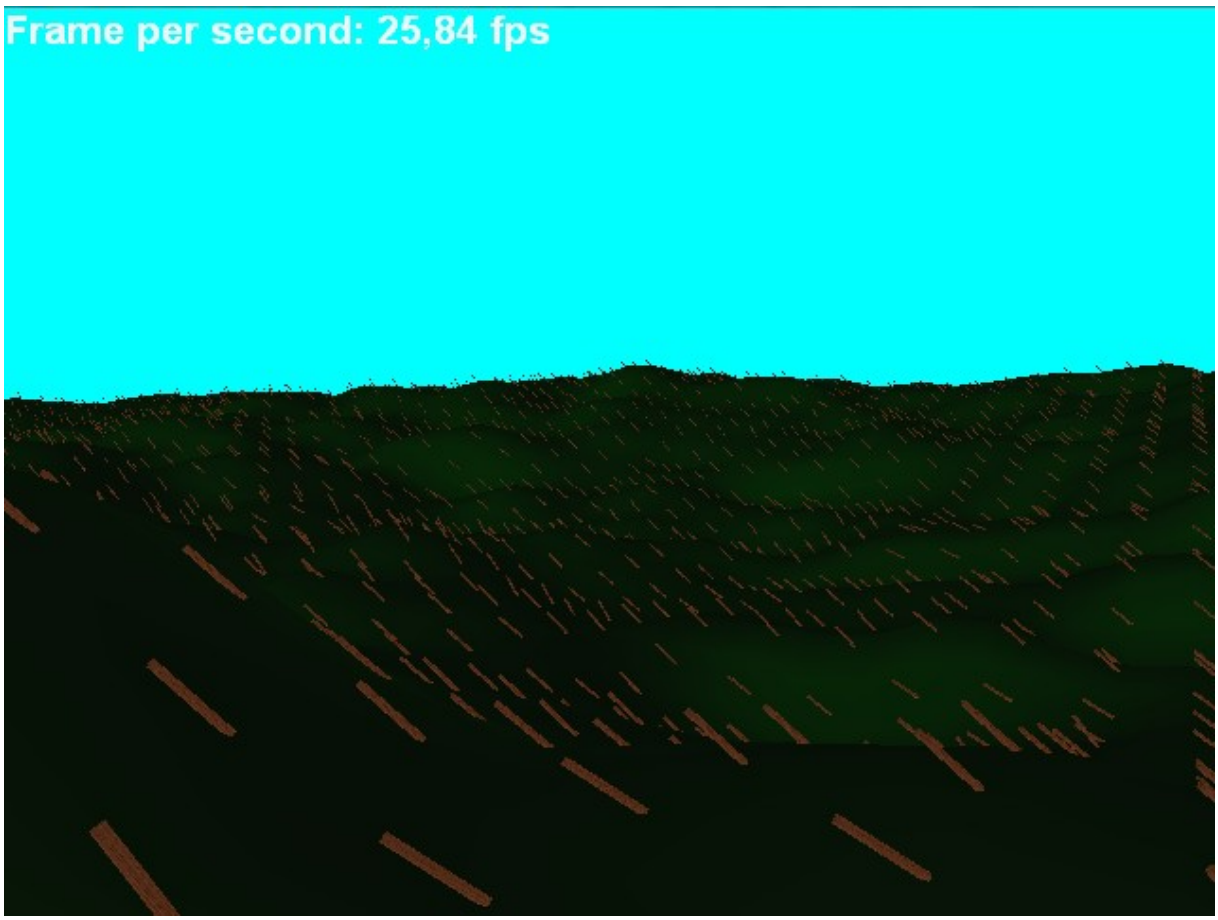
Obrázek 18 - nestabilní stav v čase 1h 30min s umístěním koule do středu



Obrázek 19 - nestabilní stav v čase 1h 30min posuv koule špatným směrem

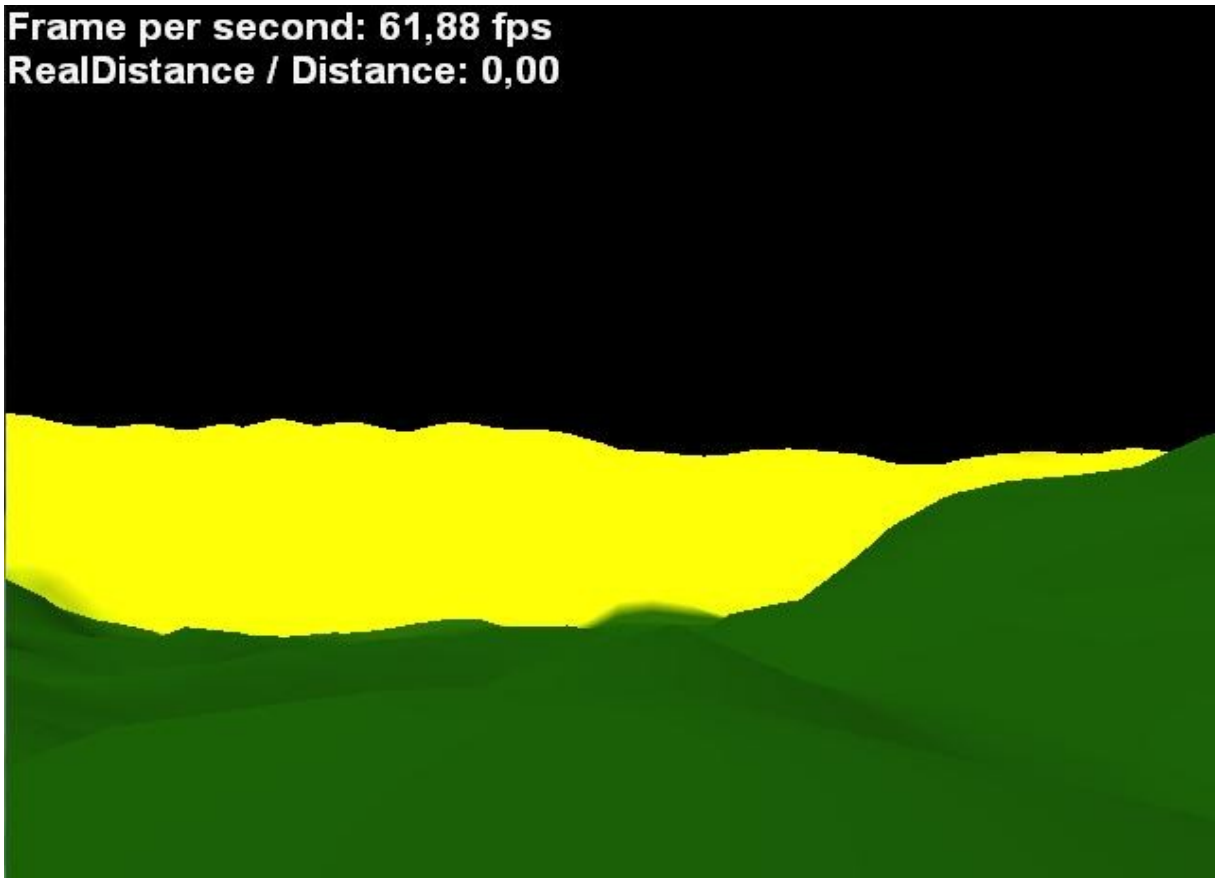


Obrázek 20 – ukázka z testovací aplikace při pohledu z povrchu

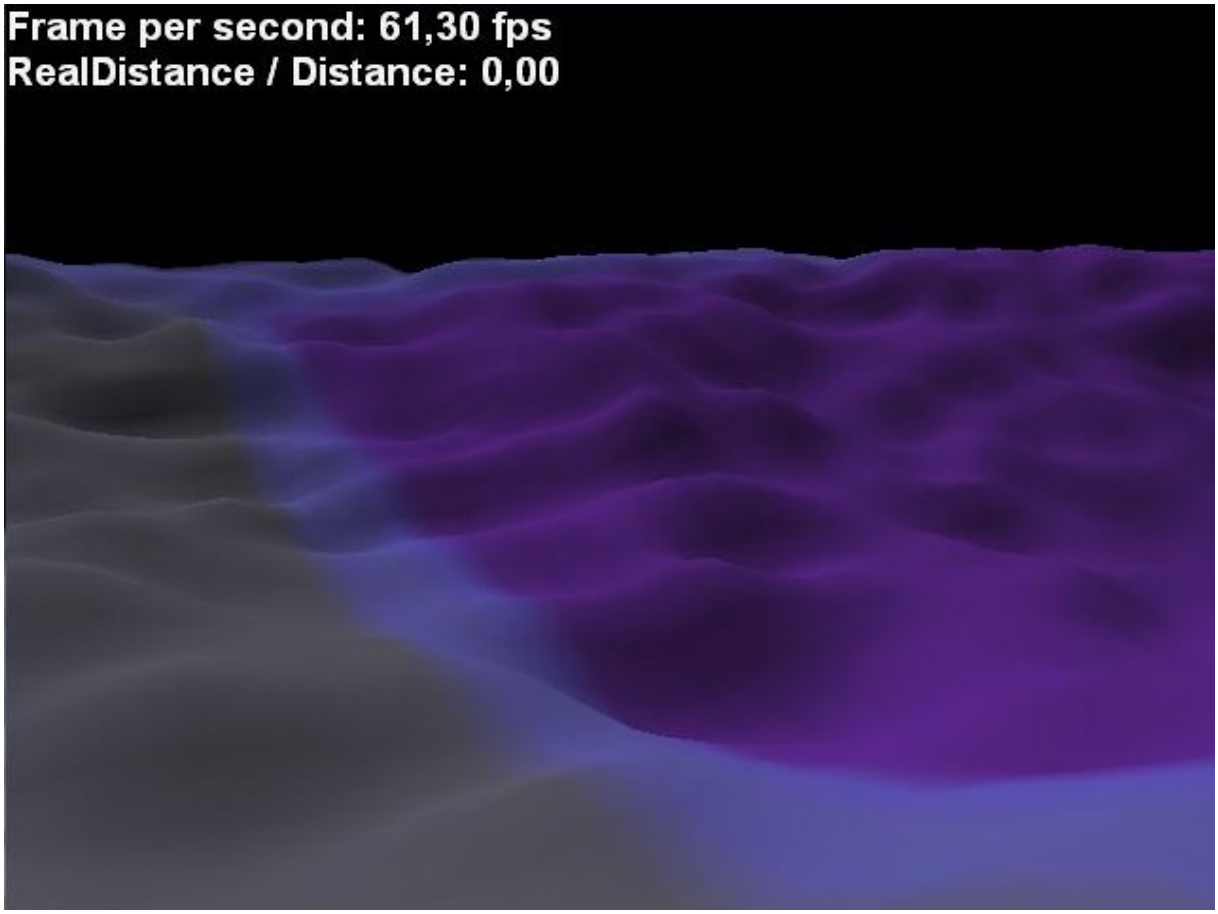


Obrázek 21 – druhá ukázka z testovací aplikace při pohledu z povrchu

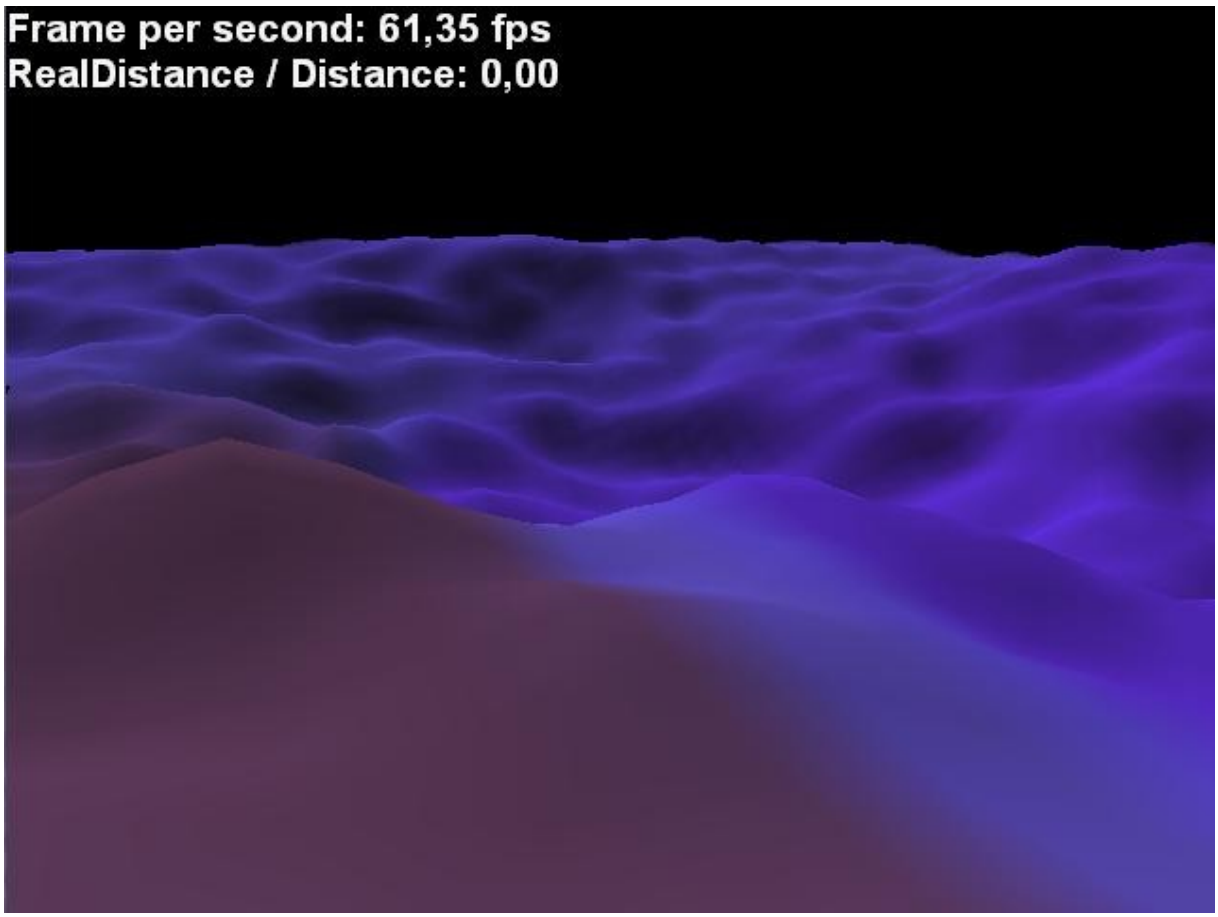
Implementace pomocí buněčných automatů



Obrázek 22 – ukázka z první implementace se statickým obarvením terénu



Obrázek 23 – ukázka z první implementace při dynamickém obarvení terénu



Obrázek 24 – stejná situace jako obrázek 23, ale jiné natočení

Příloha 2 – uživatelská příručka

a) použití knihovny

Instance větru se vytváří pomocí následujícího konstruktoru s 6 parametry, kde IWind je rozhraní knihovny.

```
IWind wind = new WindImplParticles(graphicsDevice,
                                   terain.xMin,
                                   terain.xMax,
                                   terain.zMin,
                                   terain.zMax,
                                   terain.meter);
```

Prvním parametrem je zařízení (device), na kterém se provádí vykreslení. Další 4 parametry slouží pro předání souřadnic levého dolního rohu a pravého horního rohu mapy, které jsou v ukázce uloženy v instanci třídy terain a slouží pro namapování. Posledním parametrem je velikost jednoho metru mapy, kvůli správným výpočtům rychlostí a tím i přesunů.

Pro správnou funkčnost je nutné po vytvoření instance použít její metodu pro načtení teplot ze souboru.

```
wind.LoadBaseStatusFromFile(@"temperature.bmp");
```

Jediným parametrem metody je název souboru, který musí být bitmapovým obrázkem, jehož pixely představují hodnoty teplot ve vektorovém poli. Červená složka barvy pixelu odpovídá velikosti kladné teploty a naopak modrá složka odpovídá velikosti záporné teploty (například: barva (255,0,0) = maximální teplota, barva (0,0,255) = minimální teplota, barva (100,0,100) = 0 ºC apod.).

Při každém průběhu hlavního cyklu (například v metodě pro vykreslení) je vhodné volat metodu pro výpočet nového stavu pole.

```
while(form.Created)
{
        form.Render();//v Render pak volání wind.CountNextStatus();

        Application.DoEvents();
}
```

Pro získání vektoru větru na pozici z namapované oblasti stačí použít metodu rozhraní getWindVector s parametrem, kterým je právě zmiňovaná pozice (například pozice přesouvané koule).

```
Vector3 windVector = wind.GetWindVector(ball.position);
```

Obdobně lze získat i velikost teploty.

```
float temperature = wind.GetTemperature(ball.position);
```

b) ovládání testovací aplikace

Ovládání testovací aplikace lze shrnout do následující tabulky.

vstup	**význam**
myš	otáčení vlevo/vpravo, rozhlížení nahoru/dolů
"W"	pohyb vpřed ve směru pohledu pozorovatele
"S"	pohyb vzad vůči směru pohledu pozorovatele
"A"	úkrok vlevo vůči směru pohledu pozorovatele
"D"	úkrok vpravo vůči směru pohledu pozorovatele
"U"	zapnutí/vypnutí invertace myši při rozhlížení nahoru/dolů
levý SHIFT	dvojnásobná rychlost pohybu
"F1"	zapnutí konzole
"F2"	přepnutí pohledu pozorovatel/shora
"F3"	při pohledu shora lze přepnout obarvení vrcholů podle teploty/rychlosti

Pro jednodušší interaktivní práci s programem je naimplementována herní konzole. Po zapnutí konzole klávesou F1, dojde v hlavním programu k ignorování vstupu z klávesnice. Veškeré vstupy z klávesnice jsou čteny pouze ve třídě GameConsole, přičemž se reaguje pouze na šipky, enter, backspace a písmena s číslicemi bez vlivu shift, ctrl i alt. V konzoli je zavedena historie, kterou se lze pohybovat pomocí šipek nahoru/dolů. Na aktuálním řádku je možné pohybovat kurzorem pomocí šipek vlevo/vpravo. Konzole podporuje tyto příkazy:

- lightning on případně lightning off znamená zapnutí/vypnutí osvětlování
- quit zapříčiní vypnutí konzole (konzoli lze také vypnout klávesou F1)
- exit vypnutí programu
- loadw příkaz umožňuje načíst pole teplot ze souboru, který je uveden v parametru (např. loadw nazev_souboru.bmp)
- partcount nastavuje koeficient počtu částic (např. partcount 10)
- staffcount nastavuje počet tyčí v řádce i sloupci (např. staffcount 50 znamená, že v obrázku bude 50 x 50 tyčí)