VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

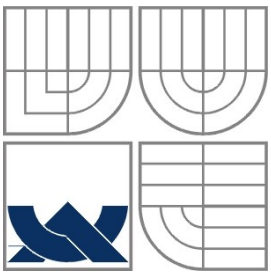
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTERS GRAPHICS AND MULTIMEDIA

ZOBRAZOVÁNÍ MRAKŮ V REÁLNÉM ČASE

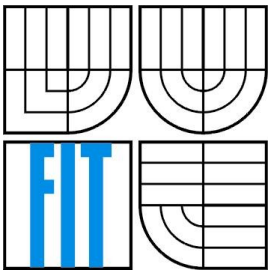
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Vít Kučera
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTERS GRAPHICS AND MULTIMEDIA

ZOBRAZOVÁNÍ MRAKŮ V REÁLNÉM ČASE

REAL TIME CLOUD RENDERING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Vít Kučera
AUTHOR

VEDOUCÍ PRÁCE — Ing. Adam Herout, Ph.D.
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Kučera Vít**
Obor: Výpočetní technika a informatika
Téma: **Zobrazování animovaných oblaků v reálném čase**
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte problematiku zobrazování oblaků v reálném čase s pomocí programovatelného grafického HW; shromážděte dostupné zdroje.
2. Popište algoritmy používané při zobrazování mraků v reálném čase.
3. Vyberte algoritmus vhodný pro zobrazování pomocí programovatelného grafického HW a implementujte jej.
4. Vyhodnoťte vlastnosti algoritmu, navrhněte vlastní vylepšení.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 -2, dílčí kroky směřující k řešení bodu 3.

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

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

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

Vedoucí: **Herout Adam, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 22. května 2007

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

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Vít Kučera**
Id studenta: 22393
Bytem: Prokofjevova 25, 623 00 Brno
Narozen: 23. 06. 1982, Velké Meziříčí
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Zobrazování animovaných oblaků v reálném čase
Vedoucí/školitel VŠKP: Herout Adam, Ing., Ph.D.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....
Nabyvatel

.....
Autor

Abstrakt

Mraky můžeme spatřit kdykoliv se podíváme na oblohu. Jsou nedílnou součástí počasí na Zemi, součástí koloběhu vody v přírodě a také významným indikátorem změn počasí. Oblaka jsou významnou částí vytváření realisticky vypadajících venkovních scén. Real-timové zobrazení dynamicky měnící se oblohy je velmi náročné a její vzhled ovlivňuje mnoho faktorů. Každý uživatel leteckých simulátorů by chtěl prolétávat kolem realisticky vypadajících oblak. Práce popisuje dva významné směry zobrazení. Ukazuje na tvorbu realisticky vypadajících oblak využívající Perlinova šumu, ale především směru, který využívá fluidní dynamiky pro realistické modelování a chování mraků.

Klíčová slova

Dynamika mraků, Perlinův šum, CML, GPU, OpenGL, Impostory, Fluidní dynamika, Rovnice Navier- Stokes, Volume rendering.

Abstract

Clouds are ubiquitous and ever-changing feature of the outdoors. They are integral factor in Earth's weather systems, component of water circulation in the nature and a strong indicator of weather patterns and changes. Clouds are important part of the visual simulation of any outdoor scene, but the complexity of cloud formation, dynamics, and light interaction makes cloud simulation and rendering difficult in real time. In an interactive flight simulation, users would like to fly around realistic and volumetric clouds. I present two main ways of clouds representation in computer graphics, where one way is modelling with Perlin noise and second one is modelling by fluid dynamic system.

Keywords

Cloud dynamic, Perlin noise, CML, GPU, OpenGL, Impostor, Fluid dynamic, Navier- Stokes equation, volume rendering .

Citace

Vít Kučera: Zobrazování mraků v reálném čase, diplomová práce, Brno, FIT VUT v Brně, 2007

Zobrazování mraků v reálném čase

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Rád bych poděkoval svému vedoucímu, kterým byl pan Ing. Adam Herout, Ph.D., že mi při práci a zpracování dat na projektu pomohl.

Obsah

1 Úvod

Při vytváření přírodních scén v počítačové grafice je důležité reálné zobrazení oblohy a také důležité při vytvářeni leteckých simulátorů. Při pozorování oblohy, zjistíme, že je složena z velkého množství kupovitých objektů, kterým říkáme mraky. Můžeme spatřit, jak jednotlivě mění svou velikost, tvar a také barvu v závislosti na poloze Slunce a výšce mraku.

Druhá kapitola popisuje mraky, které pozorujeme v přírodě. Mraky jsou ve své podstatě viditelné obrovské množství malých vodních kapiček nebo ledových krystalů (záleží na výšce mraku). Podle výšky se mraky dělí do třech základních skupin. Fyzikální dynamika mraků je popsána ve třetí kapitole. Je zde popsán zjednodušený matematický model popisující fyzikální děje, které probíhají a mění vzhled mraku. Neustálé probíhající změny přinášejí rozmanité množství tvarů. Barva mraků je určena procházejícím světlem, slunečními paprsky odraženými od hladiny moře, povrchu a jiných drobných částic.

V minulosti bylo vyvinuto několik matematických modelů, které umožňují převést tento fenomén na monitor. Matematické metody umožňují nejen zobrazování statických scén, ale také dynamicky se měnící mraky. Tyto metody umožňují animaci vzniku, pohybu a zániku mraků. Existují dva základní přístupy k zobrazení mraků. Jeden směr se snaží především o vizualizaci a opomíjí fyzikální podstatu, druhý směr využívá fyzikální podstaty k zobrazení. Ve čtvrté kapitole jsou podrobně popsány oba směry, včetně ukázek jejich výsledků. Práce je zaměřená na animaci modelu, který se opírá o fyzikální model. Pro výpočet simulačního modelu byla použita fluidní dynamika, kdy implementace se odráží od práce Stable Fluids [2]. O jednotlivých částech implementace fluidní dynamiky pojednává sedmá kapitola.

Osmá kapitola popisuje zobrazení simulačního modelu, pomocí metody CML. Pro samotné zobrazení je využita metoda billboarding, jednotlivé částice jsou natáčeny k uživateli a tím vytváří dojem 3D objektu.

V závěru práce byly provedeny testy. Testování se provedlo na dvou zcela rozdílných počítačích, které ukázaly výpočetně časovou závislost jak na vyspělosti grafického procesoru tak procesoru (CPU).

2 Jak vznikají mraky

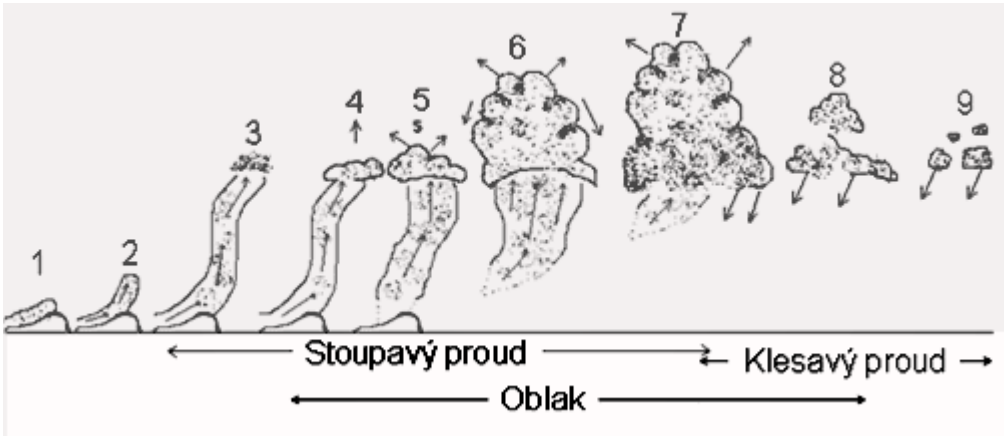
Při snaze o realistické zobrazení je důležité pochopit fyzikální principy, které zapříčiňují životní cyklus. V této kapitole pouze nastíním danou problematiku a v následující kapitole ji popíši podrobněji.

2.1 Vznik kupovité oblačnosti

Kupovitá oblaka také nazýváme oblaka s vertikálním vývojem. Je to proto, že jejich vznik je bezprostředně vázán na vertikální pohyb v atmosféře, který nazýváme termickou konvekcí. Jiným způsobem kupovitá oblačnost nevzniká.

Sluneční paprsky dopadající na zem ohřívají vzduch, který proudí po povrchu. Tento ohřátý vzduch je nasycen vodní parou. Jestliže se proud vzduchu ohřeje na určitou teplotu a relativní rozdíl teplot je velký, vznikne "komín", kterým se začne odsávat celá zásoba ohřátého přízemního vzduchu nahoru. Následně začne studený vzduch z vrchu nahrazovat odsátý teplý vzduch. Takto vzniklým prouděním mohou vzniknout víry a jestliže množství teplého vzduchu je větší, mohou vzniknout i tornáda. Celý proces je velmi urychlován nerovnostmi zemského povrchu. Aby celý proces odsávání mohl začít, musí vniknout impuls, který zapříčiní rozdíl teplot. Tímto impulsem jsou právě nerovnosti na zemském povrchu, jakými jsou například hory.

Tento odsátý teplý vzduch stoupá a dostává se do tlakových hladin s nižším a nižším atmosférickým tlakem. Díky tomu se vystupující vzduch rozpíná a ochlazuje. Relativní vlhkost uvnitř stoupavého proudu neustále vzrůstá a teplota vzduchu se pomalu přibližuje k teplotě rosného bodu. Jakmile se obě teploty ztotožní, nastává kondenzace vodní páry, tedy přeměna z jejího plynného stavu do stavu kapalného, malé kapičky vody vytváří oblačnost, kterou můžeme spatřit lidským okem.



Ilustrace 2.1.: Popisuje vývoj vzniku kupovité oblačnosti.

1. Hromadění teplého vzduchu při zemském povrchu.
2. Teplý vzduch se začne odtrhávat od povrchu a začne stoupat.
3. Začíná se vytvářet kolmý stoupavý proud. Jestliže je zásoba teplého vzduchu při zemi malá, stoupavý proud se přerušuje a oblačnost se nevytvoří.
4. Jestliže vrchol stoupavého proudu dosáhne kondenzační hladiny, vytvoří se jemná mlhovina.
5. Oblak se začíná vytvářet v nepravidelných kapkách, které rostou a navzájem se spojují.
6. Oblak má stále výraznější a ostřejší okraje. V místech kde je proud nejsilnější, stoupá mrak nejvýše, má hladké bílé okraje a odspodu je základna nejtmavší.
7. Oblak roste dokud je dostačující zásoba teplého vzduchu při zemi. Pokud se tato zásoba vyprázdní, začne se oblačnost rozpadávat.
8. Obrysy oblaku se začínají rozpouštět; oblak se po částech rozpadává.
9. V sestupných pohybech vzduchu se oblak začíná od vrchu rozpouštět.

2.2 Druhy mraků

Mraky dělíme podle specifického tvaru a výšky. Můžeme najít na 10 základních typů, které dělíme do třech základních pater: vysokého (6 až 10 km předpona cirro), středního (od 2 do 6 km předpona alto) nebo nízkého patra (do 2 km předpona cirro).

1. Cirrus je oblak vysokého patra, což znamená, že se běžně nachází ve výškách 7-10 km. Je složen z ledových krystalů, má vláknitý vzhled a často hedvábný lesk. Nepadají z něho srážky a jeho výskyt na obloze bývá často příznakem blízkosti atmosférické fronty, nebo-li přechodné pásmo mezi dvěma frontami.

Ilustrace 2.2: Mrak typu Cirrus, výška 7-10 km .

2. Cirrocumulus má podobu tenkých, menších nebo větších skupin nebo vrstev bílých oblaků bez vlastního stínu, složených z velmi malých částí v podobě zrnek nebo vlnek apod. Tyto jednotlivé části mohou být buď navzájem oddělené, nebo mohou spolu souviset a jsou více méně pravidelně uspořádány.

Ilustrace 2.3: Mrak typu Cirrocumulus, výška 7-10 km.

3. Cirostratus je průsvitný bělavý závoj oblaků. Může být vzhledu vláknitého nebo hladkého, který úplně nebo částečně zakrývá oblohu.

Ilustrace 2.4: Mrak typu Cirostratus, výška 7 - 10 km.

4. Altocumulus jsou menší nebo větší skupiny nebo vrstvy oblaků, barvy bílé nebo šedé, popř. obojí, mající vlastní stíny. Skládají se z malých oblačných částí podoby vln, oblázků nebo valounů apod., které mohou být buď navzájem oddělené, nebo mohou spolu souviset. Mnohdy mají částečně vláknitý nebo rozplývavý vzhled.

Ilustrace 2.5: Mrak typu Altocumulus výška 6-2 km.

5. Altostratus mají vzhled šedavé nebo modravé oblačné plochy nebo vrstvy se strukturou vláknitou nebo žebrovitou nebo také bez patrné struktury, pokrývající úplně nebo částečně oblohu. Je tak tenká, že místy jsou patrné alespoň obrysy Slunce jako za matným sklem. U altostratu se halové jevy nevyskytují.

Ilustrace 2.6: Mrak typu Altostratus, výška 6 - 2 km.

6. Nimbostratus má podobu šedé, často tmavé oblačné vrstvy, která vlivem vypadávání trvalých dešťových nebo sněhových srážek má matný vzhled. Srážky většinou dosahují Země. Vrstva je všude tak hustá, že poloha Slunce patrná není.

Ilustrace 2.7: Mrak typu Nimbostratus, výška 4-2 km.

7. Stratocumulus mají vzhled jako šedé nebo bělavé, popř. obojí barvy, menší nebo větší skupiny nebo vrstvy oblaků, které téměř vždy mají tmavá místa. Oblak se skládá z částí podobných dlaždicím, oblázkům, valounům apod., nemívá vláknitý vzhled (s výjimkou zvláštního případu virga). Jednotlivé části oblaku buď spolu souvisí, nebo mohou být oddělené.

Ilustrace 2.8: Mrak typu Stratocumulus.

8. Stratus je oblačná vrstva, obvykle šedá, s celkem jednotvárnou základnou, z níž vypadává mrholení, ledové jehličky nebo sněhové vločky. Prosvítá-li vrstvou stratu Slunce, jsou jeho obrysy zřetelně patrné a nikoliv rozplizlé jako v případě altostratu.

Ilustrace 2.9: Mrak typu Stratus.

9. Kumulus je osamocenou zářivě bílou až našedlou kupou s ostře ohraničenými obrysy ve fázi rozvoje a se stále měnící se tvary ve stádiu rozpadu. Kumulus se rozvíjí směrem vzhůru a formuje se do podoby kup a věží, přičemž rostoucí části mají podobu květáku. Kumulus prochází několika stádii rozvoje. V počátcích svého vzniku má podobu neuspořádaných chomáčů s dosud ne zcela srovnanou spodní základnou. Za příznivých podmínek se tato oblaka dále rozvíjí do tvarů s ostře ohraničenými obrysy a původně plochý oblak získává jasně definovaný vertikální rozměr. Oblak má oslnivě bílou barvu a má tendenci spojovat se do větších celků.

Ilustrace 2.10: Mrak typu Kumulus.

10. Kumulonimbus bouřkový oblak. Mohutný, hustý, často hrozivě vyhlížející kupovitý oblak velkých horizontálních a hlavně vertikálních rozměrů. Vrchol se často zplošťuje a rozlévá do podoby vějíře, závoje, nebo kovadliny. Tento jev je způsoben výškovým prouděním a prorůstáním vrcholů kumulonimbů až do spodních vrstev stratosféry. Spodní základna se obvykle nachází ve spodním patře, avšak vrcholky nezřídka dosahují vysokého patra a nacházejí se ve výškách 7-9 km.

Ilustrace 2.11: Mrak typu Kumulonimbus.

3 Dynamika a radiosita mraku

3.1 Dynamika

Realistická vizuální simulace mraků vyžaduje dva rozdílné aspekty mraků v přírodě: dynamika a radiometrie. Dynamika mraků obsahuje pohyb vzduchu v atmosféře, kondenzaci a odpařování vody, které se různě mění v závislosti na teplotě a jiných fyzikálních aspektech. Radiometrie mraků je studie světla a jeho vzájemná reakce s mraky. Oba tyto pohledy jsou natolik rozsáhlé, že je nemožné je efektivně simulovat bez použití zjednodušeného pohledu. V této kapitole je matematicky popsána dynamika mraku a několik možných způsobů, jak reálně a efektivně se dají využít ke zjednodušenému modelu mraku. Jsou vynechány přílišné detaily, které nejsou nutné, nebo jejich vliv na zobrazení je zanedbatelný.

3.1.1 Dynamika mraků

Dynamika tvaru mraků je komplex růstu, pohybu a rozptýlení. Pro vytvoření efektivní simulace je nutné pochopit dynamiku a na jejím základě vytvořit model, který bude aproximovat fyzikální podstatu, ale nebude jí detailně popisovat. Výsledkem tohoto modelu je několik rovnic, které řeší každý časový krok.

Základní veličiny, které jsou nezbytné pro simulaci mraků: rychlost $\vec{u}$ = (u, v, w), tlak vzduchu p, teplota T, vodní pára q a kondenzace vodních par q_c. Viditelná část mraků je tam, kde je dostatečná hodnota q_c. Simulace mraků vyžaduje systém rovnic popisující dynamiku života mraků, mezi hlavní rovnice patří rovnice pohybu, termodynamiky.

3.1.2 Rovnice pohybu

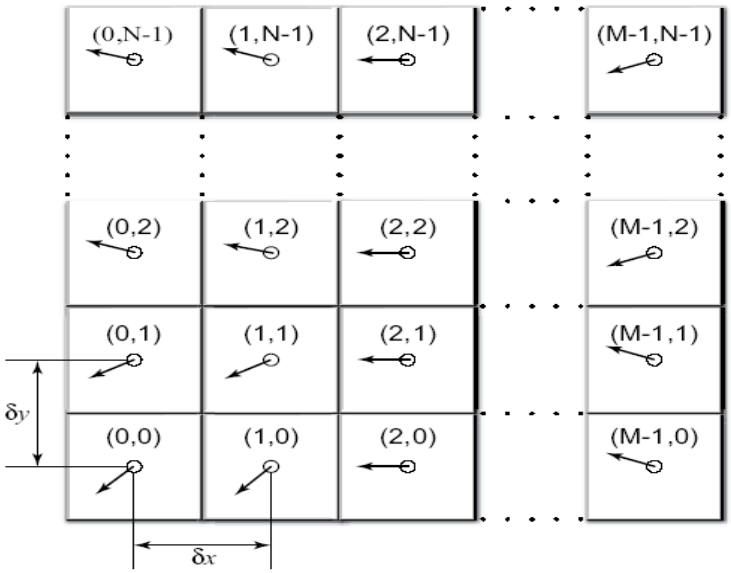
Rovnice pohybu vychází z faktu, že vzduch je součástí atmosféry (včetně mraků) a celek tvoří nestlačitelnou látku. Nestlačitelná látka je látka, jejíž jednotlivé částice jsou neměnné v čase. Látka je homogenní, jestliže hustota ρ je konstantní v celém prostoru. Kombinace homogenity a nestlačitelnosti znamená, že hustota je konstantní, jak v čase tak i v prostoru. Víme, že vzduch je stlačitelný. Jakkoliv je ale vzduch stlačen, nemění se jeho možnost pohybu v uzavřeném prostoru (v našem případě atmosféře), proto můžeme celou problematiku převést na dynamiku kapalin.

Klíč k dynamice látek je v časovém kroku, kdy je nutné v každém kroku stanovit správné pole rychlostí. Z mého pohledu nejvhodnější metodou řešení je použití Navier-Stokasovy rovnice pro nestlačitelné látky. Jestliže zjistíme rychlost a tlak pro každý časový okamžik, potom jsme schopni rozpohybovat cokoliv, co se dá popsat látkovou dynamikou jako je kouř, voda a jiné.

Mraky můžeme simulovat jako kapaliny na regulární karteziánské mřížce pomocí prostorových souřadnic $\vec{x}=(x,y,z)$ a proměnné času t. Kapalina je reprezentována rychlostí pole $\vec{u}(x,t)=(dx/dt, dy/dt, dz/dt) = (u,(\vec{x})),(v,(\vec{x})),(w,(\vec{x}))$ a tlaku $p(\vec{x},t)$. Jestliže známe rychlost a tlak pro počáteční čas, potom můžeme popsat stav látky pomocí Navier – Stokesovy rovnice pro nestlačitelné kapaliny. Tato rovnice popisují závislost změny tlakového a rychlostního pole v závislosti na čase v každém bodě zkoumaného prostoru. Pokud známe hodnoty rychlostí a tlaků, pak má rovnice následující tvar:

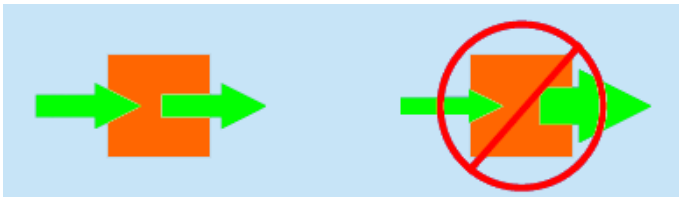
$$\frac{\partial \vec{u}}{\partial l}=-(\vec{u}\cdot\nabla)\cdot\vec{u}-\frac{1}{\rho}\nabla p+v\nabla^2\vec{u}+\vec{F} \tag{2.1}$$

$$\xi\cdot\vec{u}=0 \tag{2.2}$$



Ilustrace 3.1: Reprezentuje místo simulace, kdy každá buňka má svůj vektor rychlosti (šipka z bodu).

Kde i a j je umístění buňky v mřížce a δx je vzdálenost jednotlivých buněk viz. Ilustrace 3.1. Pro zjednodušení budeme počítat, že každá buňka je stejná a má tvar čtverce $\delta x=\delta y$, ρ je (konstantní) hustota, v je kinematika viskozity a $\vec{F}=(f_x, f_y, f_z)$ reprezentuje externí sílu. Rovnice (2.1) se nazývá rovnice zákonu zachování hybnosti a Rovnice (2.2) je rovnice kontinuity, nebo-li zákon zachování hmoty. Hmota nevzniká z ničeho a nezaniká v nic. Symbol "·" v rovnicích (2.1) a (2.2) představuje skalární součin vektorů.



Ilustrace 3.2.: Obrázek popisuje rovnici kontinuity, vyjadřuje zákon zachování hmoty, hmota nevzniká z ničeho a nezaniká v nic.

Levá strana rovnice hybnosti obsahuje čtyři části, které reprezentují zrychlení. Jedná se o:

1. **Posun vodorovným prouděním**

Rychlost je zapříčiněna pohybem částí v dané kapalině. Vstříknuté barvivo do pohybující se tekutiny se dá do pohybu ve směru proudu kapaliny. První část vzorce (2.1) na pravé straně představuje právě tento posun ve vodorovném směru.

2. **Tlak**

Molekuly tekutiny se mohou pohybovat kolem sebe, a proto mohou jedna druhou "postrčit". Jestliže aplikujeme sílu na vstupní objem, tak jednotlivé molekuly daného objemu se začnou pohybovat. Podle druhého Newtonova zákona, čím větší silou působíme na objekt, tím získá větší zrychlení. Druhá část, nazývaná tlak, reprezentuje tuto akceleraci.

3. **Šíření**

Některé tekutiny mají různou viskozitu, nebo-li proudí jinou rychlostí. Představme si olej a benzín. Jestliže rozlijeme stejné množství oleje a benzínu na stůl, benzín nám vytvoří mnohem větší louži než olej. Říkáme, že řídké kapaliny (alkohol) mají velkou viskozitu, tím pádem „tečou" rychleji. Pomocí viskozity můžeme určit jak snadno se daná kapalina šíří.

4. **Externí síla**

Představuje externí sílu, která působí na objem mraku. Síla může být znázorněna např. větrákem, který fouká (tlačí) na tělo mraku, nebo působením gravitace.

3.1.3 Eulerovy rovnice

Vzduch na zemi má velmi nízkou viskozitu, což je pro nás velmi důležité a tím pádem problematiku pohybu ve vzduchu můžeme popsat jednodušeji pomocí Eulerovy rovnice na nestlačitelné látky.

$$\frac{\partial \vec{u}}{\partial l} = -(\vec{u}\cdot\nabla)\cdot\vec{u} - \frac{1}{\rho}\nabla p + B\vec{k} + \vec{F} \qquad (2.3)$$

$$\nabla\cdot\vec{u} = 0 \qquad (2.4)$$

V tomto příkladu je oddělena externí síla $\vec{F}$ od síly $B\vec{k}$, která vyjadřuje sílu udržující mrak v jedné hladině a působící opačně vůči gravitaci.

3.1.4 Ideální plyn

Ideální plyn dokonale stlačitelný, bez vnitřního tření, dále při stejné teplotě a tlaku nemění svůj objem. Částice takového plynu musejí splňovat následující podmínky. Rozměry částic jsou zanedbatelné vzhledem ke vzdálenostem mezi nimi, kromě srážek na sebe částice jinak nepůsobí a jejich celková kinematická energie se při vzájemných srážkách nemění. Skutečné plyny téměř vyhovují podmínkám ideálního plynu.

Vzduch na naší planetě je složen převážně z dusíku a kyslíku, mimo tyto dva základní plyny obsahuje taky vodní páry, ozon a oxid uhličitý. Tyto plyny ovlivňují počasí na naší planetě, mimo jiné vodní pára je to co vidíme, při pozorování mraků. Všechny výše zmíněné plyny budeme považovat za ideální plyny, protože se příliš neliší a zjednoduší nám to práci.

$$p = \rho RT \qquad (2.5)$$

Kde ρ hustota plynu, R je plynná konstanta, pro suchý vzduch $R = 287\, J\, kg^{1} K^{1}$. Kdy obecně je vzduch brán jako směs suchého vzduchu a vodní páry.

$$p = \rho RT_v \quad kde \quad T_v \approx T(1 + 0{,}61\, q_v) \qquad (2.6)$$

Kde T_v je virtuální teplota, která je důležitá pro zobrazení vodních par q_v . Je definována jako teplota suchého vzduchu, jestliže její tlak a hustota jsou shodné.

3.1.5 Parcely a Potencionální teplota

Konceptuálně se využívají ke studii dynamiky atmosféry parcely vzduchu, což si můžeme představit jako malý úsek (krychli) vzduchu. Tyto úseky jsou určitou aproximací skutečnosti a jsou jednodušší pro vytvoření matematické simulace mraků. Představme si úsek ohřátého vzduchu, který má konstantní tlak. Tím, že je teplejší a má větší tlak, expanduje do okolí především vertikálně a mění

svoji nadmořskou výšku. Probíhá zde adiabatický jev, kdy plyn neodevzdává teplo do okolí, ale vykonává práci tím, jak se rozrůstá. Vykonává práci a tím klesá jeho teplota. Vlastnosti vzduchu (tlak a teplota) se mění s výškou, tím se mění i teplota a tlak daného úseku. Na základě zákona o ideálním plynu a zákona termodynamiky byla odvozena Poissonova rovnice pro adiabatické jevy. Ta ukazuje souvislost mezi teplotou a tlakem při adiabatických jevech.

$$\left(\frac{T}{T_o}\right)=\left(\frac{p}{p_o}\right)^{k} \qquad k=0.286 \tag{2.7}$$

Kde T_o a p_o jsou vstupní hodnoty tepoty a tlaku a T a p jsou hodnoty po adiabatických jevech. K je konstanta, která definuje tepelnou kapacitu suchého vzduchu při konstantním tlaku a objemu.

Pro adiabatické změny v teplotě a tlaku se používá konceptu potencionální teploty. Potencionální teplota θ může být definována jako konečná, jestliže se díky adiabatický jevům tlak p a teplota T změní v tlak $\hat{p}$

$$\theta=\frac{T}{\Pi}=T\left(\frac{\hat{p}}{p}\right)^{k} \tag{2.8}$$

Π se nazývá Exnerova funkce a $\hat{p}$ je standardní tlak 100 kPa. Potencionální teplota se používá z důvodu zjednodušení výpočtu, je konstantní v průběhu adiabatických jevů, na rozdíl od absolutní teploty.

3.1.6 Nadnášející síla

Jedná se o sílu, která udržuje danou parcelu (mrak) v určité nadmořské výšce. Jestliže v daný okamžik má nižší hustotu než jeho okolí, tak tato síla roste. V případě rostoucí hustoty síla klesá, protože stoupá gravitační síla a tím daný úsek klesá. Dalším zjednodušením při matematickém modelování mraků je vliv lokálního tlaku na hustotu, který je popsán v následující rovnici.

$$B=g\left(\frac{\theta_v}{\theta_{vo}}-q_H\right) \tag{2.9}$$

Zde g představuje gravitační zrychlení, q_H je poměr nasycení vodních par, který obsahuje všechny formy vody jiné než vodní páry. θ_{vo} je odhad virtuální potencionální teplota, někde mezi 290 a300K. θ_v je virtuální potencionální teplota, která se zavádí pro vliv vodních par na možnou teplotu okolí. Virtuální potencionální teplota je přibližně $\theta(1+0.61q_e)$

3.1.7 Teplotní gradient

Atmosféra na zemi je v rovnováze. Hydrostatická rovnováha je vytvářena silou gravitační a tlakem vzduchu, který klesá s výškou.

$$p(z) = p_o\left(1 - \frac{z\Gamma}{T_o}\right)^{g/(\Gamma R_d)} \tag{2.10}$$

z udává nadmořskou výšku, g je gravitační zrychlení a p_o (100 kPa) a T_o (290 K) jsou tlak a teplota v základní nadmořské výšce. Kdy teplotní gradient Γ udává pokles teploty z nadmořskou výškou. Největší teplota vzduchu na Zemi je v nadmořské výšce moře a potom postupně lineárně klesá každých 100 m o jeden stupeň až do výšky 15 km (tropopauza). Pro zjednodušení budeme předpokládat, že teplotní gradient je konstantní 1°C. Dosazením do rovnic (2.8) a (2.10) vypočítáme teplotu a tlak pro danou nadmořskou výšku, které využijeme pro zjištění bodu nasycení.

3.1.8 Nasycení a relativní vlhkost

Oblaky páry neustále mění svou fázi z tekutého do plynného a naopak. Jestliže míra kondenzace a odpařování je stejná, vzduch se nasytí a vodní páry se začnou stlačovat. Toto nasycení je označováno $e_s(T)$ a nazývá se tlakem nasycených par. Jestliže vodní pára překoná hranici nasycenosti, stane se s ní super-nasycená.

$$e_s(T) = 611.2\exp\left(\frac{17.67\mathrm{T}}{T + 243.5}\right) \tag{2.11}$$

Kde teplota je udávána ve ˚C a tlak nasycených par v Pa. Tento vzorec slouží pro naplnění meteorologické tabulky. Obsah vodních par ve vzduchu se nazývá relativní vlhkost, kde jsou vodní páry stlačovány. Kombinaci vzorce (2.11) a RΠ = q_v / q_{vs} získáváme vzorec pro nasycené vodní páry.

$$q_s(T) = \frac{380.16}{p}\exp\left(\frac{17.67\mathrm{T}}{T + 213.5}\right) \tag{2.12}$$

3.1.9 Kontinuita

Bulk Water Continuity model popisuje vývoj a poměr směšování vodních par q_v a poměr směšování kondenzace vody v mracích q_c. Vodu obsaženou v mracích tvoří malé kapičky, které se ale nemůžou neustále zvětšovat, dochází tak k neustálým změnám ve vývoji mraku (velikosti , fázi). V modelu musí být poměr vypařování a kondenzace v rovnováze.

$$\frac{\zeta g_c}{\zeta t} + (u\cdot\xi)q_c = -\left(\frac{\zeta g_c}{\zeta t} + (u\cdot\xi)q_c\right) = C \tag{2.13}$$

3.1.10 Termodynamické rovnice

První zákon termodynamiky se zabývá změnami teploty v daném úseku. Pro Ideální plyn platí rovnice o termodynamické energii.

$$c_v dT + pd\,\alpha = dQ \tag{2.15}$$

Kde dQ je teplotní poměr a α specifikuje objem ρ^{-1} . První část rovnice je míra vnitřní energie a druhá část ukazuje míru vykonané práce daným úsekem v daném prostředí. Jestliže použijeme rovnice (2.5) pα = R_dT dostaneme pdα + αdp = R_ddT a dále když použijeme c_p = R_d + c_p získáme rovnici.

$$c_p dT - \alpha\, dp = dQ \tag{2.16}$$

Jestliže adiabatické jevy jsou platně přizpůsobeny pro vzduch, který není nasycen vodní párou, potom potencionální teplota nasyceného vzduchu nemůže zůstat po celou dobu konstantní. Základní předpoklad pro modelování mraků je, že ochlazování a oteplování jsou jedinými neadiabatickými jevy. Proto dQ = - Ldq_v. Kde L je latentní teplo při vypařování vody 2.501 J kg^{-1} kdy teplota je 0°C. Proto může být použita následující termodynamická rovnice.

$$d\,\theta = \frac{-L}{c_p \Pi} dq_v \tag{2.17}$$

Protože jak teplota tak vodní pára přináší pohyb vzduchu, musí být vše zahrnutu do rovnice termodynamické energie, kterou jsem použil.

$$\frac{\partial \theta}{\partial t} + (u \cdot \nabla)\,\theta = \frac{-L}{c_p \Pi}\left(\frac{\partial q_v}{\partial t} + (u \cdot \nabla)\, q_v\right) \tag{2.18}$$

konstanta	popis	hodnota
$\hat{p}$	Základní tlak u hladiny moře	100 kPa
g	Gravitační zrychlení	9.81 $m.s^{-2}$
R_d	Ideální plyn pro suchý vzduch	287 J $kg^{-1}K^{-1}$
L	Latentní teplo při vypařování vody	2.501 J kg^{1}
c_p	Teplotní kapacita	1005 $kg^{-1}K^{-1}$

3.2 Radiosita

Jestliže chceme zobrazit reálně vypadající oblačnost, je nezbytné simulovat absorpci a rozptyl světla při průchodu mrakem. Existují dva přístupy. První cesta využívá jednonásobného rozptylu světla, tzv. simuluje rozptyl světla pouze v jednom směru a to k pozorovateli. Tato metoda počítá pouze z rozptylem způsobeným v jednom směru. Druhá metoda je mnohem přesnější a dosahuje realističtějších výsledků, protože počítá s vícenásobným rozptylem světla přicházejícího přímo ze směru slunce a také světla odraženého od povrchu Země a jiných částic obsažených v atmosféře.

3.2.1 Základní pojmy

Pro pochopení metod je nutné popsat základní děje. Světlo při průchodu atmosférou reaguje s různými částicemi různé velikosti, mezi základní interakce patří absorpce, rozptyl a odraz.

3.2.1.1 Absorpce

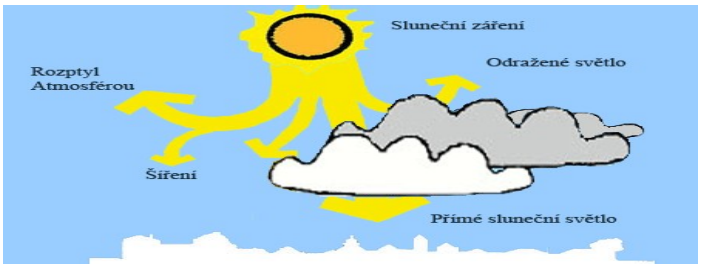
Absorpce je proces, při kterém částice přeměňují část své energie na jinou formu (teplo). Celkové ztráty způsobené absorpcí na jednotku délky ve směru k pozorovateli jsou vyjádřeny objemovým koeficientem (volume absorption coeficient) σ_a .

3.2.1.2 Rozptyl

Rozptyl je proces, při kterém je část energie absorbována a poté znovu vyzářena. Částice se začne chovat jako bodový zdroj světla, ztráty světelné energie způsobené rozptylem jsou vyjádřeny pomocí objemového koeficientu rozptylu (volume scattering coefficient) σ_s .

3.2.1.3 Odraz

Odraz závisí jak na vlnové délce dopadajícího světla a také na velikosti částice od které se odráží. Podle velikosti částice se používají dvě základní metody. Pro částice o velikosti zrnek prachu se používá komplexnějšího přístupu jako je např. Mileova teorie rozptylu. Pro částice velikosti molekul je možné použít jednodušší metodu Rexleighovy teorie rozptylu.



Ilustrace 3.3: Šíření slunečních paprsků

Intenzita světla $l(\lambda,\theta)$ je vyzařována v daném úhlu θ . Je vyjádřena pomocí rovnice

$$I(\lambda,\theta)=\frac{(I_o K \rho F_r(\theta))}{\lambda^4} \tag{3.1}$$

$$K=2\Pi^2\frac{(n^2-1)^2}{3\mathrm{N}_s} \tag{3.2}$$

$$\rho(h)=\exp(\frac{-h}{H_0}) \tag{3.3}$$

$$F_r(\theta)=\frac{3}{4}(1+\cos^2(\theta)) \tag{3.4}$$

I_0 je intenzita dopadajícího světla, K je konstanta udávající molekulární hustotu v nulové nadmořské výšce, $\rho(h)$ je relativní hustota částic v atmosféře, která je závislá na nadmořské výšce. $F_r(\theta)$ je fázová funkce udávající směrovou charakteristiku rozptylu, θ je úhel rozptylu, λ je vlnová délka dopadajícího světla, n je index lomu prostředí, N molekulární hustota atmosféry. Ho je konstanta odpovídající tloušťce atmosféry a v případě malých částic má velikost 7994m.

Jestliže se podrobněji podíváme na rovnici (3.1), zjistíme, že intenzita rozptylu je přímo úměrná čtvrté mocnině vlnové délky. Z tohoto důvodu jsou také kratší vlnové délky při průchodu atmosférou utlumeny více a naopak větší vlnové délky nejsou utlumeny skoro vůbec. Tento jev můžeme pozorovat na denní obloze, kdy vidíme modré zbarvení (modrá barva má nejdelší vlnovou délku). Situace je ale různá v ranních nebo pozdních večerních hodinách, kdy cesta paprsku se začne prodlužovat a barva oblohy z důvodu zvýšeného rozptylu kratších vlnových délek zbarvuje oblačnost do červena. Pro vyjádření velikosti útlumu světla na jednotku délky se zavádí koeficient útlumu β , který je roven součtu koeficientu absorpce a rozptylu. Pro vyjádření tohoto koeficientu slouží rovnice.

$$\beta=\frac{(4\Pi^3)}{\lambda^4} \tag{3.5}$$

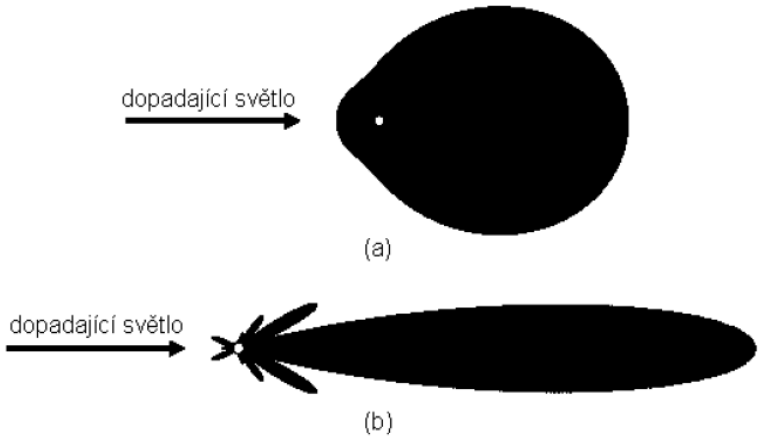
V případě rozptylu světla částicemi větších rozměrů je nutné počítat s jinou konstantou tloušťky (Ho = 1200), kdy fázová funkce se spočte pomocí rovnice (3.6).

$$F(\theta,g)=\frac{(3(1-g^2)(1+\cos^2\theta))}{(2(2+g^2)(1+g^2-2g\cos\theta)^{(3/2)})} \tag{3.6}$$

$$g = \frac{5}{9}u - (\frac{4}{3} - \frac{25}{81}u^2)x^{(-1/3)} + x^{(1/3)} \tag{3.7}$$

$$x = \frac{5}{9}u + \frac{125}{729}u^3 + (\frac{64}{27} - \frac{325}{243}u^2 + \frac{1250}{2187}u^4)^{(1/2)} \tag{3.8}$$

Zde u je hodnota, která závisí na atmosférických podmínkách a vlnové délce, většinou se používá hodnota v rozmezí 0.7 – 0.85.



3.3 Zobrazení

3.3.1 Zobrazení založené na jednonásobném rozptylu

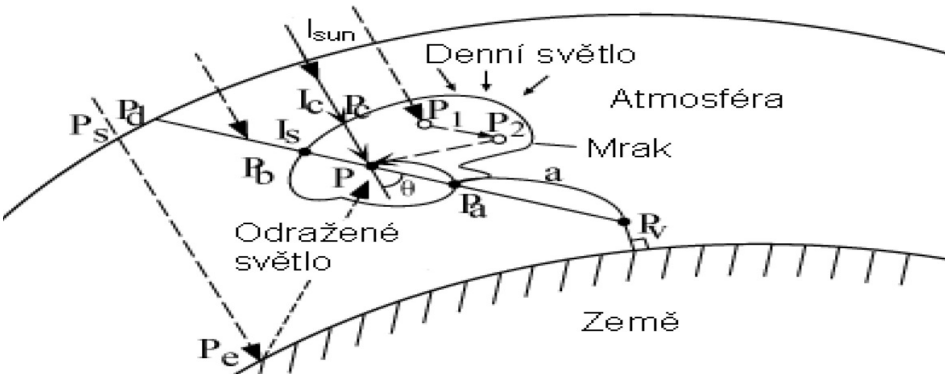
Oblačnost je zobrazována pomocí metaballů (2D texturou), v nichž každý je určen poloměrem a hustotou ve svém středu. Hustota s jakou je definován určuje jeho projekci na rovinu a tím hodnotu útlumu procházejícího světla. Intenzitu v místě obsahující oblačnost získáme jako vážený součet intenzit jednotlivých metaballů.

$$\rho(x,t) = \sum_{(i,j,k)\varepsilon\Omega(P,R)}^{N} q(i,j,k,t) f(|x - x_{(i,j,k)}|) \tag{3.9}$$

N značí počet voxelů, které jsou ve vzdálenosti R od bodu P, daná rovnice může být jednoduše implementována na grafický procesor s použitím metody míchání barev. Její popis bude podrobně popsán v kapitole implementace.

3.3.2 Zobrazení založené na vícenásobném rozptylu světla

Tato metoda je založena na vícenásobném rozptylu světla a předpokládá následující faktory. Simulační model zobrazuje dynamiku mraků, které obsahují částice větších rozměrů. Pro počítání je využit silný dopředný Milův rozptyl. Částice obsažené v mracích mají vysokou odrazivost, a proto zde dochází k významnému odrazu světla mezi jednotlivými částmi. Proto světlo nedopadá jen ze směru Slunce, ale především během dne odrazem od zemského povrchu. Sluneční paprsky jsou rozptýleny každou částicí ležící ve směru od Slunce k pozorovateli.



Ilustrace 3.4: Částice je osvětlena přímým slunečním světlem I sun, rozptýleným světlem (P1,P2), odraženým světlem (Ps,Pe) .

Simulační model je složen z voxelů, které nahrazují objemové částice. Mezi jednotlivými elementy je počítána energie, která z nich v průběhu simulace vystupuje a naopak vstupuje do jiných částic. Intenzita $I(\rho,\omega)$ v bodě P závisí na fázové funkci (form factor), která udává množství energie rozptýlené, emitované a absorbované částicí, spočítá se pomocí rovnice.

$$I(P,\omega)=I(P_e,w)\exp(-\tau(PP_s)) + \int_{(z=0)}\left[(\beta\rho(s)\exp(-\tau)(P(s)P)\frac{1}{(4\Pi)}\int F(\theta)I(P,\omega)d\omega)\right]ds \qquad (3.10)$$

Pro výpočet form factoru se místo metody počítání energie mezi každým párem částic vytvoří podprostor se střední hodnotou hustoty a pouze v dané části se pak počítají příspěvky intenzity rozptylu od sousedních částic k voxelu umístěným ve středu definovaného podprostoru. Pro rychlejší výpočet se používá pouze faktorů vymezených úhlem, definovaným silně dopředným rozptylem velkých částic, kde významná část této energie se vyskytuje v malém úhlu. Takto vytvořený podprostor se nazývá referenční zdroj a slouží jako filtr. Postup metody je následující:

Prvně je vybrán bod, do kterého je umístěn střed referenčního vzoru, následně jsou postupně načítány intenzity rozptylu světla od ostatních voxelů. Poté jsou vybrány voxely s intenzitou větší než

prahová hodnota. Konfigurační faktory mezi jednotlivými voxely jsou uloženy v tabulce spolu s referenčním vzorem a násobeny s hodnotami fázových funkcí.

V simulačním modelu můžeme simulovat druhý a třetí rozptyl pomocí posunu referenčních vzorů z voxelu do voxelu a uložením hodnot. Intenzita prvního stupně rozptylu v bodě P je spočtena s využitím míry útlumu, uložena v každém voxelu a intenzita druhého a třetího stupně z hodnot několika voxelů.

4 Modelování mraků

Obecně lze metody pro modelování mraků rozdělit do dvou základních směrů. Na jedné straně jsou metody založené na fyzikální podstatě, využívající meteorologických a fyzikálních procesů. Na druhé straně je vytvářen směr ontogenický, který se snaží zachytit vizuální podstatu a neřeší výpočtově a tím časově náročné operace některých jevů, které mají na vzhled někdy nepatrný vliv. Tento směr je tlačen především počítačovými hrami a leteckými simulátory. Oba směry mají své kouzlo a každý má své klady a zápory. Práce popisuje oba směry. Implementace využívá fyzikální podstaty pro vytvoření simulace vzniku mraků a samotné mraky jsou složeny z částic, tvořené metodou Perlinova šumu.

První směr tvorby mraků je směr ontogenický, který se především vyvíjel v minulosti, ale dál se úspěšně používá. Snaží se přenést vizuální model pomocí jednoduchých matematických modelů, které definují tvar a vzhled mraků. Kladem těchto modelů je většinou poměrně jednoduchá aplikace a výpočetně nenáročné zobrazení. Některé scény vznikají vytvořením mraků a následným importováním a rozmístěním ve scéně.

Obecně tyto metody poskytují dostatečné výsledky, ale neposkytují možnost dynamiky na základě měnícího se prostředí. Pro zjednodušení uvedu příklad. Mějme vlnící se hladinu vody ve sklenici, která bude tvořena Perlinovým šumem. Takto vytvořená hladina bude vypadat dobře, ale zkusme do sklenice s vodou hodit železnou kouli. Tento jev není možné modelovat pomocí perlinova šumu tak, abychom dosáhli realistické reakce koule s vodou.

4.1 Modelování pomocí procedurálního šumu

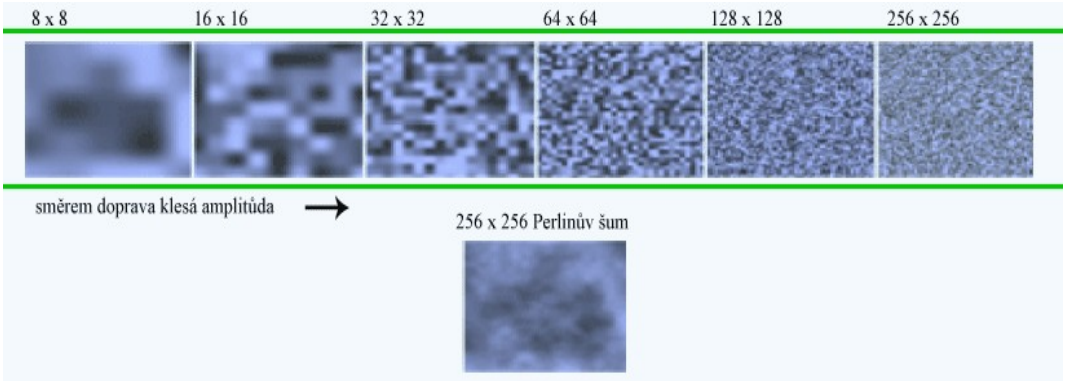
Procedurální šum (Procedural noise) je metoda patřící mezi významné přístupy při tvorbě mraků. Využívá šumu jako základu, na který jsou aplikovány další metody pro vznik celistvého tvaru, jež ve výsledku připomíná model mraku.
Mnoho tvůrců při snaze vytvořit nějaký přírodní fenomén, který nemá vždy zcela předvídatelný tvar, využívá náhodných čísel. Účelem je přiblížit se přírodnímu vzhledu, který je pozorovateli mnohem bližší než přesně definované tvary. Využívá se metody Perlinova šumu. Pro jednodušší pochopení, celou problematiku vysvětlím na dvourozměrném šumu a následně ukáži jeho využití k tvorbě 2D oblaků.

Ken Perlin je tvůrcem myšlenky Perlinova šumu (perlin noise). Velmi rychle se stal základem, bez kterého se v dnešní grafice neobejdeme, především pro jeho jednoduchost a velmi příjemný výsledek.

4.1.1 Tvorba Perlinova šumu

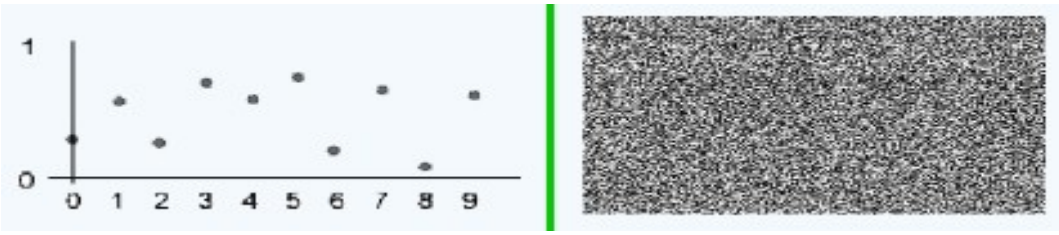
Perlinův šum napodobuje přírodní fenomén pomocí spojování několika šumů, které vznikly s různou amplitudou a frekvencí. Pro nejlepší pochopení uvádím příklad skalnatých hor. Jestliže se na pohoří díváme z velké dálky, je tvořeno jednoduchou křivkou bez větších detailů (v našem případě jedna vrstva šumu), ale pokud se podíváme na dané pohoří blíže, spatříme spoustu menších detailů. S každým dalším přiblížením se tato množina zvětší a zjemní. Tato myšlenka je základem Perlinova šumu.

Na ilustracích zobrazených níže jsou jednotlivé šumy vytvořené rozdílnou frekvencí (počet opakování na reálné číslo) a amplitudou (výškou ”vlny”). První snímek nám udává základní tvar výsledného mraku, postupně další snímky nám přidávají detaily. Spodní obrázek je výsledkem součtu všech předchozích, a když jednotlivé obrázky interpolujeme, získáme Perlinův šum.



Ilustrace 4.1.: Obrázek popisuje vznik Perlinova šumu, jednotlivé obrázky vznikají s různou frekvencí, amplitudou a jsou interpolovány na stejné rozlišení. Perlinův šum vznikne složením obrázků, zdroj [5].

Jádro Perlinova šumu je matematická funkce, která na základě parametrů v podobě celých nezáporných čísel generuje pseudonáhodná čísla v rozmezí -1 až 1. Jestliže do této funkce vložíme stejné parametry, dostaneme vždy stejně “náhodné” číslo. Takto vytvořené pole hodnot je jedna vrstva šumu, nebo-li jedna oktáva. Pro vytvoření oblak je nutné vytvořit několik oktáv s různou frekvencí a amplitudou. Nejpoužívanější vstupní hodnotou je frekvence rovna jedné, kterou postupně zdvojnásobujeme. Počet oktáv záleží na uživateli, obecně stačí čtyři.



Ilustrace 4.2.: Jádro Perlinova šumu je generátor pseudonáhodných čísel. Levá část obrázku ukazuje jednotlivé hodnoty, na pravé straně její interpretace v odstínech šedi.

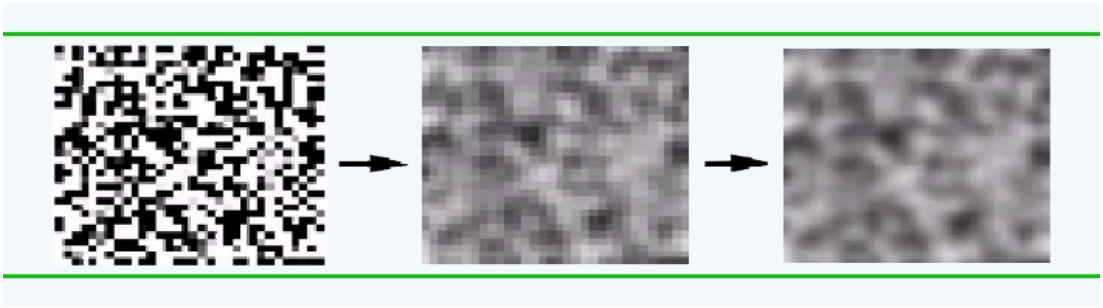
4.1.1.1 Persistence

Nyní víme jak vytvořit čistý šum, ale pro vhodný výsledek je nutné použít persistenci (zlomková hodnota). Používá se z důvodu postupného přičítání oktáv, čím větší oktáva, tím má na výsledek menší vliv.

4.1.1.2 Vyhlazení šumu

Čistý šum vyjadřuje náhodné hodnoty bez jakýchkoliv přechodů, můžeme jej přirovnat k diskrétnímu signálu, který je na pohled příliš vzdálený mlhavé podstatě mraků. Mezi jednotlivými pixely (sousedy) není žádný vztah, každý vzniká náhodně. Pro vytvoření mlhavých tvarů oblaku je nutné propojit jednotlivé pixely.

Metoda interpolace zkoumá vztah sousedů tak, že počítá vážený průměr nejbližších sousedů a tím získá výslednou hodnotu pixelu, která je závislá na svém okolí.



Ilustrace 4.3.: Popisuje převod čistě vygenerovaného šumu, zjemnění a následnou bilineární interpolaci.

4.1.2 Tvorba mraků využívající Perlinův šum

Existuje několik variant, jak využít Perlinův šum k tvorbě mraků. Především záleží na tom, jaký typ mraku chceme modelovat a pro jaký účel. Pomocí jednotlivých šumů můžeme vytvořit jednotlivé mraky a ty následně zobrazit ve scéně a nebo celé nebe pokrýt jednou texturou šumu a vytvořit tak dojem nekonečné oblačnosti.

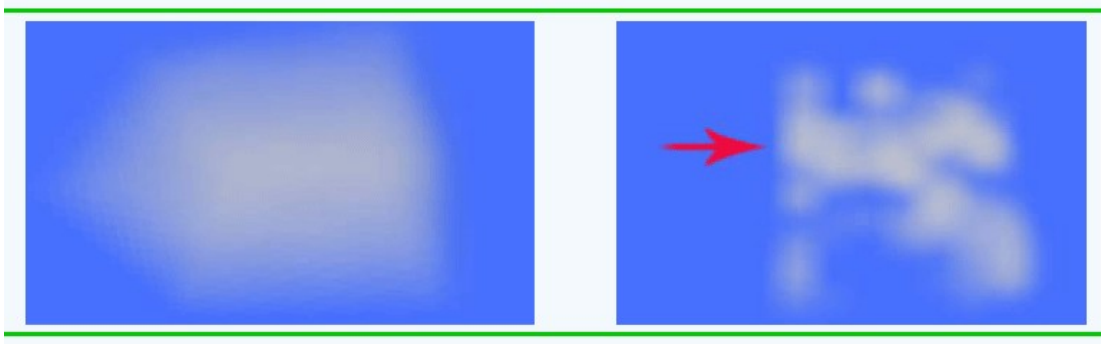
Textura mraku je dvojrozměrné pole vytvořené výše popsanou metodou. Každý prvek pole obsahuje hodnotu intenzity (průhlednosti – 0 pixel není vidět, 255-pixel je zcela neprůhledný).

Po zobrazení takto vzniklého pole (viz. Ilustrace 5.4 levá část), je výsledek nepřijatelný z důvodu nespecifikovaného tvaru mraku. Z takto vzniklé mlhoviny je nutné určitt tvar mraku.

Filtr šumu

Pro vytvoření je vhodné použít exponenciální filtr, který na základě několika proměnných udává tvar mraku, nebo- li kolik částic z dané mlhoviny bude tvořit náš mrak.

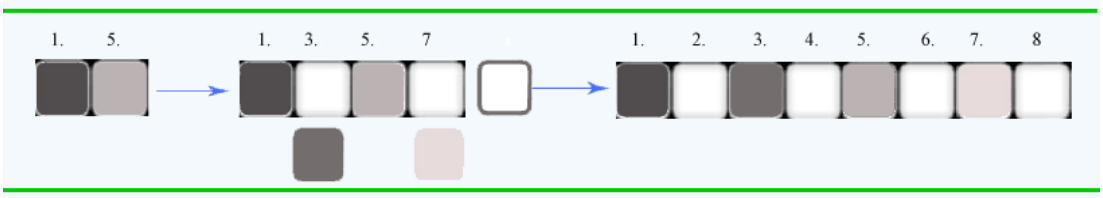
```
cloudCover- jak velký mrak chceme mít
cloudSharpness - hrana mraku (ostrost)
c = (noise64[x][y]*255) - cloudCover;
if (c < 0) c=0;
cloudDensity = 255 - ((pow(cloudSharpness, c)) * 255);
```



Ilustrace 4.4.: Obrázek vlevo popisuje zobrazuje interpolované 2D pole bez použití filtru. Obrázek vpravo zobrazuje totéž pole po použití exponenciální filtru. Červená šipka nám ukazuje hranici pole, ze kterého byla textura vyfiltrována.

4.1.3 Problém s interpolací

Jestliže zobrazíme náhodně vygenerovaný mrak (pole), zjistíme závadu s hranicemi, které definují mrak. Problém vzniká při interpolaci na větší rozlišení, viz obrázek níže. První pixel, který se náhodně vygeneruje, se postupnou interpolací na větší rozlišení přenáší směrem doprava. Interpolujeme (zjemňujeme) pixely směrem zleva doprava, ale krajní levý pixel se nemá kam zjemňovat. Tento nepříjemný jev se dá odstranit několika způsoby.



Ilustrace 4.5.: Obrázek popisuje problém s interpolací. První pixel se přenáší a nezjemňuje.

Jedna z metod využívá průběhu funkce SINUS. Díky jejímu průběhu snížíme intenzitu hustoty barvy pixelu směrem k hranicím pole. Metoda odstraňuje artefakt hranic, ale s použitím metody vzniká problém podobnosti všech mraků, protože všechny oblačnosti jsou na středu tmavé a k hranicím jejich intenzita klesá. Proto byla pro odstranění problému hranic zvolena jiná metoda. Tato metoda je založena na principu zvětšení základního pole na každé straně o jeden. Hraniční prvky jsou nastaveny na hodnotu nula. První pixel šumu má v tomto případě možnost interpolovat do všech stran. Jednotlivé další oktávy šumu se nepřičítají k hranicím, které musí zůstat nulové. Tímto vytvoříme kompaktní objekt ve tvaru oblaku.

Ilustrace 4.6.: Zobrazení mraků, při jejichž tvorbě jsem využil Perlinova šumu.

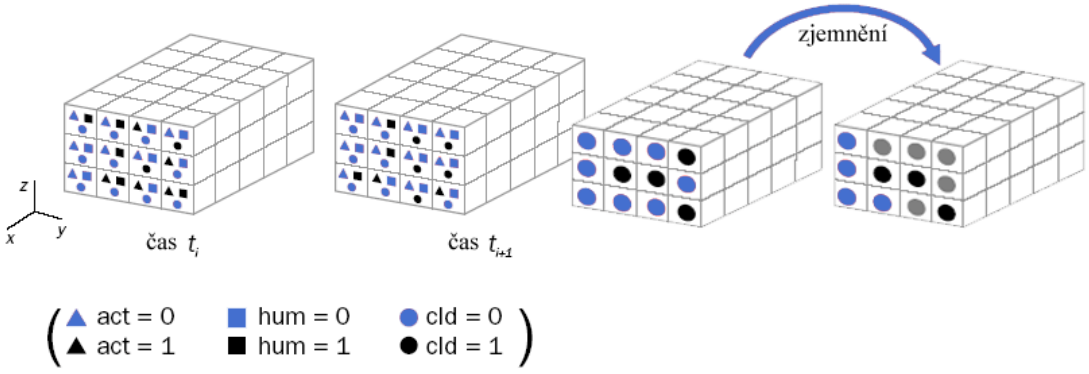
4.2 Fyzikálně založené metody

Fyzikální procesy vzniku a zániku mraku jsou popsány podrobně ve druhé kapitole (Dynamika mraků). Ve zkratce, mraky vznikají z vodních par, které se vypařují na povrchu. Vodní pára stoupá do míst kde expanduje, zde nastane fázová proměna a změní se v kapičky vody, které představují mrak.

4.2.1 Modelování mraků pomocí buněčných automatů

Buněčné automaty jsou ve světě počítačových simulací hojně používané, ale mají také vztah k teorii chaosu a evoluční biologii. V zásadě jde o mapu tvořenou jednotlivými políčky a sadu pravidel, které popisují, jak se budou políčka zbarvovat.

Metoda není příliš vhodná pro simulování založeném na fyzikální podstatě chování mraků. Procesy jsou zde značně zjednodušeny na několik základních pravidel. Výhodou buněčného automatu je snadná implementace, rychlost a použitelné vizuální výsledky.



Ilustrace 4.7.: Simulovaný prostor je rozdělen do voxelů, levá část ukazuje mřížku v čase t a v čase t+1, kdy každý voxel nese informace, jestli je mrak, jestli je vlhkost a zda může proběhnout fázová proměna. Pravá část ukazuje rozšíření v podobě zjemnění mraku.

Prostor, ve kterém probíhá simulace, se rozdělí na voxely. Voxel představuje jednu buňku modelu, obsahující základní údaje jako je vlhkost (hum), mrak (cd) a fázovou funkci, která říká, zda v dané buňce může proběhnout změna (act). Každý údaj je buď TRUE nebo FALSE, a proto jde každou informaci uložit do jednoho bitu. Simulace potom probíhá na základě jednoduchých pravidel, která udávají životní cyklus mraku (simulace). Informace jsou jednobitové, lze úspěšně používat Booleovy algebry, a proto je daná aplikace velmi rychlá. Protože buňka mraku nabývá hodnot pouze 1 nebo 0, je nutné je upravit použitím zjemnění, které dané zobrazení převede do reálnější podoby.

K simulaci se daný prostor rozdělí na množství buněk ve směrech souřadných os (x, y, z). Jednotlivé buňky jsou na začátku naplněny náhodnými hodnotami reprezentující vlhkost a možnost fázové proměny. Výskyt mraku je nastaven na nulu ve všech buňkách. Jestliže buňka obsahuje vlhkost (hum = 1), značí, že je zde dostatek vlhkosti pro vznik mraku. Dynamiku simulace popisují následující pravidla.

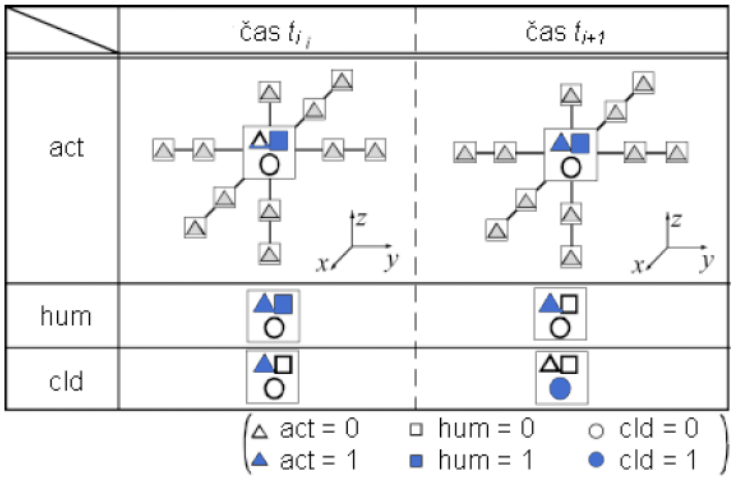
$$act(i,j,k,t_{(i+1)}) = \neg act(i,j,k,t_i) \wedge humt(i,j,k,t_i) \wedge f_{act}(i,j,k)$$

$$\begin{aligned} f_{act}(i,j,k) = {} & act(i+1,j,k,t_i) \vee act(i,j+1,k,t_i) \vee act(i,j,k+1,t_i) \\ & \vee act(i-1,j,k,t_i) \vee act(i,j-1,k,t_i) \vee act(i,j,k-1,t_i) \\ & \vee act(i-2,j,k,t_i) \vee act(i,j-2,k,t_i) \vee act(i,j,k-2,t_i) \\ & \vee act(i+2,j,k,t_i) \vee act(i,j+2,k,t_i) \end{aligned}$$

$$hum(i,j,k,t_{(i+1)}) = hum(i,j,k,t_i) \wedge \neg act(i,j,k,t_i)$$

$$cld(i,j,k,t_{(i+1)}) = cld(i,j,k,t_i) \vee act(i,j,k,t_i)$$

t specifikuje čas a i, j ,k jsou souřadnice dané buňky. Na každou buňku se postupně aplikují jednotlivá pravidla. Mimo simulovaný prostor se předpokládá nulová hodnota všech proměnných.



Ilustrace 4.8.: Aplikace jednotlivých pravidel na každou buňku simuluje dynamiku mraků.

4.2.2 Rozšíření metody

Jestliže se podrobně podíváme na pravidla tohoto buněčného automatu, zjistíme, že daná buňka mraku (cld = 1) nemůže žádnou cestou zaniknout. Proto se využívá možnosti rozšíření této metody o pravděpodobnost zániku p_{ext}, která je pro každou buňku nastavena na jinou hodnotu. Tímto způsobem může být přibližně definován tvar mraku, neboli určitá šablona výsledného vzhledu. Zavedení tohoto pravidla nás nutí použít další funkce, které popisují jeho chování. V momentě zániku mraku není možnost jeho znovu vzniknutí, proto je nutné vytvořit pravděpodobnost vzniku vlhkosti p_{hum} a pravděpodobnost fázové změny p_{act}. Nastavením těchto pravděpodobností jde kontrolovat tvorbu mraků. Pro každou pravděpodobnost generujeme náhodnou hodnotu, jestliže je číslo menší než definovaná pravděpodobnost, uloží se do dané buňky stav 1 (true). Takto definovanými pravidly kontrolujeme tvar výsledného mraku. Jednou z metod je využití elipsoidu, kde je pravděpodobnost p_{hum} a $p_{act\ nejvyšší}$ ve středu nejvyšší a dále k okrajům klesá, opačně je nastavena pravděpodobnost zániku.

Nakonec se provede zjemnění mezi jednotlivými buňkami pomocí následujícího vzorce:

$$q(i,j,k,t_i)=\frac{1}{((2i_o+1)(2j_o+1)(2k_o+1)(2t_o+1))}$$

$$\sum_{t_0}^{t'=-t_0}\sum_{i_0}^{i'=-i_0}\sum_{j_0}^{j'=-j_0}\sum_{k_0}^{k'=-k_0} w(i',j',k')\,cld\,(i+i',\ j+j',k+k',t+t')$$

w je váhová funkce pro kterou se používá vzdálenost středu voxelů od okolních.

5 Mraky pomocí fluidní dynamiky

5.1 Modelování využívající fluidní dynamiku

Mraky jsou jeden příklad z mnoha případů použití fluidní dynamiky, další jsou tekoucí voda, kouř, oheň, pára. Všechny tyto přírodní fenomény se snažíme realisticky zobrazit v grafických aplikacích. Pomocí fluidní dynamiky se nám daří realisticky přenést tyto jevy a zobrazit je v grafických aplikacích.

Metoda simuluje vytváření mraků s využitím numerického modelu. Zobrazení je založené na metodě Coupled Map Lattice (CML), která je postavena na základě buněčného automatu. Celý prostor simulace je rozdělen do buněk a na jednotlivé částice jsou aplikována pravidla.

5.1.1 Použití Navier – Stokasovy rovnice

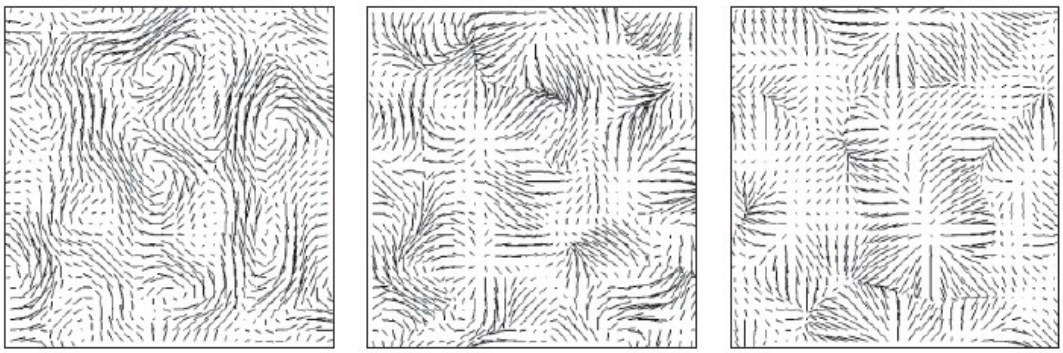
Analytické řešení rovnice je natolik výpočetně náročné, že i při použití dnešních vyspělých výpočetních technik bychom nedosáhli použitelné rychlosti aplikace. V počítačové grafice jsme naopak ochotni obětovat určitou přesnost pro zrychlení výpočtu, pokud výrazně neutrpí vizuální dojem. V následující kapitole je popsána metoda založená na metodě charakteristik.

$$\frac{\partial \vec{u}}{\partial l} = -(\vec{u}\cdot\nabla)\cdot\vec{u} - \frac{1}{\rho}\nabla p + v\nabla^2\vec{u} + \vec{F} \tag{4.1}$$

$$\xi\cdot\vec{u} = 0 \tag{4.2}$$

Pro řešení rovnice jsem se rozhodl použít metodu charakteristik, která je založena na technice Stable fluids [2]. Cílem autora bylo navrhnout metodu, která bude za všech okolností stabilní, tedy nebude omezena časovým krokem simulace. Metoda byla navrhnuta pro modelování chování plynů.

Řešení rovnice vyžaduje tři aktualizace rychlosti pro každý časový okamžik (posun vodorovným směrem, šíření, externí sílu), výsledkem je pak pole rychlostí (w). Po spočítání tohoto pole můžeme získat tlak a nakonec aktualizovat vektor rychlosti $\vec{u}$, při použití Poissonovy rovnice tlaku $\nabla^2 p = \nabla\cdot w$.



Ilustrace 5.1: Helmholtz-Hodgeova dekompozice a = b−c: a) konzervativní pole, b) pole vzniklé aplikací difúze a zpětným sledováním částic, c) pole gradientů.

Soustavu rovnic (4.1) a (4.2) lze sloučit do jedné rovnice. Využije se vztahu známého jako Helmholtz-Hodgeova dekompozice, která říká, že každé vektorové pole w se dá rozložit na potenciální a gradientní složku.

$$w = \vec{u} + \nabla q \tag{4.3}$$

Kde $\vec{u}$ má nulovou divergenci $\nabla \cdot u = 0$ a q je skalární pole. Tento výsledek nás opravňuje k použití operátoru P, který promítne libovolné vektorové pole w do jeho potenciální složky. Po aplikaci tohoto operátoru dostaneme rovnici obsahující pouze rychlost.

$$\frac{\partial u}{\partial t} = P(-(u \cdot \nabla)u + v\nabla^2 u + F) \tag{4.4}$$

Použitím této rovnice ztrácíme informaci o rozložení tlaku, proto je nutné zavést ještě jednu rovnici, která popisuje vývoj tlaku.

$$\frac{\partial \rho}{\partial t} = (-(u \cdot \nabla)\rho + \kappa\nabla^2 \rho + S) \tag{4.5}$$

Prvně popíší rovnici pro tlak, která je jednodušší na pochopení. Začnu prvním členem zleva. Pohyb $-(u \cdot \nabla)\rho$ popisuje, jak hustota sleduje rychlost. Druhý člen, difúze říká, jak hustota prostupuje rychlostí. S reprezentuje vstup nebo výstup do simulačního modelu.

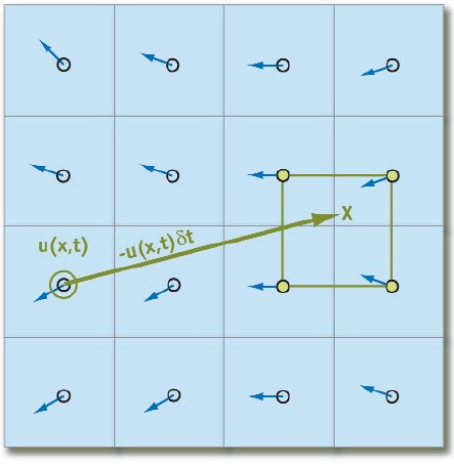
- **Difúzi** - si lze představit jako vzájemnou výměnu hustoty okolních buněk (vyrovnávání tlaku). Rozdíl hustot sousedních buněk je přímo úměrný předané hustotě. Zde může dojít k problému, hustota se nesmí rozšířit dále než do vzdálenosti sousední buňky v jednom časovém kroku, aby byla simulace stabilní. Pro každou buňku v systému je potřeba zaručit, že během daného časového intervalu nemůže dojít k přenosu změn v rychlostním poli na vzdálenost větší než je velikost jedné buňky. Z toho vyplývá podmínka svazující vzájemně

maximální rychlost v systému s velikostí buňky a časovým krokem. Daný problém se dá řešit zpětně, kde se pokusíme nalézt takové rozložení hustoty pro čas $t+\Delta t$, na které když aplikujeme krok $-\Delta t$, získáme požadované rozložení hodnotu v čase t. Tímto odstraníme nestabilitu, ale na druhou stranu musíme řešit soustavu lineárních rovnic rovné počtu buněk. Tuto soustavu lze řešit pomocí iterativních metod.

- **Pohyb-** jestliže chceme spočítat posun látky, musíme spočítat posun částic, které tuto látku tvoří, každá částice je představována buňkou. Nejprve spočítáme posun těchto částic. Použijeme metodu sledování trajektorii částice (hustoty) v čase, pomocí jednoduché aproximace pohybu po křivce ve vektorovém poli nám pomůže vypátrat pozici částice v čase $t-\Delta t$. Tato křivka značí proud dané látky, parametry na této křivce jsou konstantní. Položíme hustotu ve zkoumaném bodě rovnu hustotě odkud daná buňka přišla. Konkrétní hodnotu získáme interpolací ze sousedních bodů.. Metoda je stabilní, výsledný parametr nemůže nabýt větší hodnoty než některé z předchozích. Pro sledování trajektorie použijeme následující rovnici.

$$q(x,t+\delta)=q(x-u(x,t)\delta t,t) \tag{4.7}$$

Nyní popíši druhou rovnici. Druhý člen představuje viskozitu, obecně difúzi. Systém výpočtu je stejný jako v předchozím příkladě. Prvním členem pravé strany je opět pohyb, ve srovnání s výpočtem posunu v předchozí rovnici se zde vyskytuje nelinearita (rychlost je tu dvakrát), která zapříčiňuje, že je nutné každou složku vektoru rychlosti počítat zvlášť.



lustrace 5.2: Krok sledující zpětný posun v poli rychlosti , kdy bod v čase se získá interpolací bodů v čase t-1, místa částice.

Nyní potřebujeme vyřešit dvě rovnice. Jedna rovnice tlaku a druhá rovnice viskozity. Použijeme rovnici, která každým řešením iteruje k výsledku. Poissonova rovnice je rovnicí matice ve formě Ax = b, kde x je vektor hodnot, pro které hledáme řešení (v našem případě p nebo u), b je konstantní vektor a A je matice. V našem případě A je reprezentováno operátorem ∇^2 . Iterativní metoda, kterou používáme začíná s dosazením odhadnutého řešení $x^{(0)}$ v každém kroku se blížíme k přesnému řešení. Nejednodušší vhodnou iterační metodou je Jacobyho iterace, tato metoda je použita pro jednoduchost a také pro snadnou implementaci. Můžeme také použít Gaus Seidlovu iterační metodu, která rovněž poskytuje dobré výsledky.

5.1.2 Inicializace a ohraničení

Je nutné nastavit vstupní hodnoty. Pro naši simulaci nastavíme vstupní hodnoty rychlosti a tlaku ve všech místech rovnu nule. Protože naše simulace probíhá v uzavřeném místě čtverci a nemůžeme počítat vliv buněk mimo vymezený prostor, dále ani buňky nemůžou opustit daný prostor simulace. Je nutné nastavit podmínky ohraničení. Pro rychlost nastavíme podmínku, že rychlost směrem k hranicím klesá k nule a na hranicích je nula.

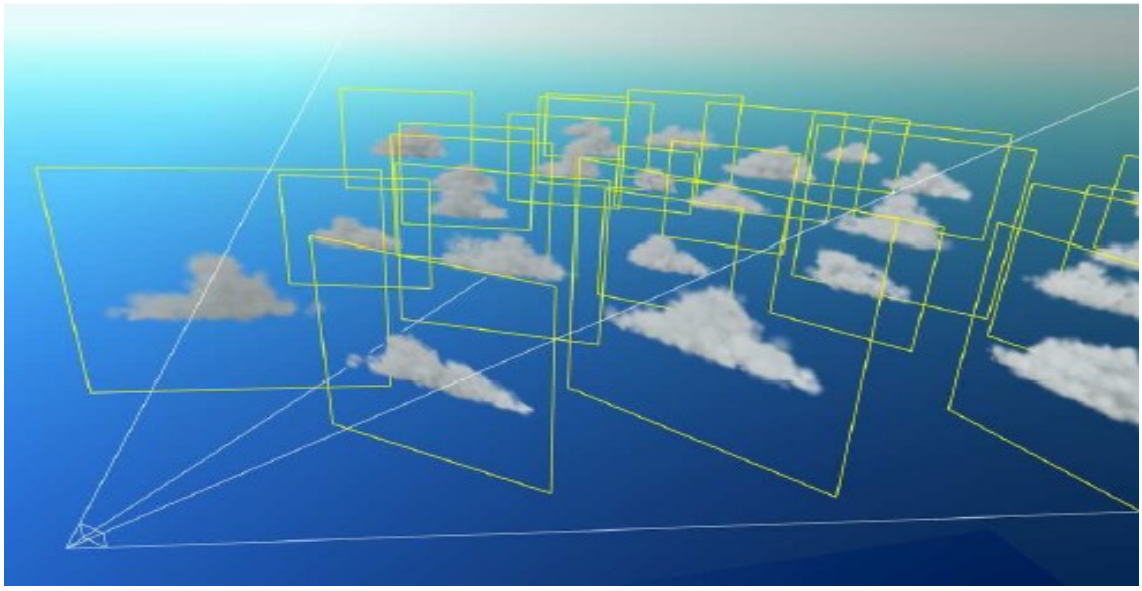
6 Impostory

6.1 Popis impostorů

V minulosti, ale i v současnosti jsou impostory používané převážně k náhradě geometrických modelů. Slouží k zobrazování měst, která se skládají z obrovského množství objektů. Vzdálené objekty jsou nahrazovány impostory. Při zobrazení ostrých hran (domy) vzniká problém geometrických nepřesností. Především u blízkých objektů, kde se jednotlivé hrany při použití impostorů na sebe příliš nelícují a vzniká hromada nepříliš dobře vypadajících artefaktů, které se složitě odstraňují. Mraky mají tu výhodu že u nich ostré hrany budeme těžko hledat, a proto není nutné používat ani tak velké rozlišení.

Mraky jsou většinou jednou z mnoha součástí vykreslované scény, proto zde musí být zachován výpočetní výkon pro zbylé součásti scény. Při snaze o vytvoření komplexní scény s velkým množstvím mraků, které se vykreslují do značné dálky ve směru pohledu pozorovatele, rychlost výrazně klesá.

Renderování velkého množství částic vyžaduje překreslování mnoha pixelů. Mrak má podstatnou komplexní hloubku, vyžadující blending, který dělá renderování pro velké množství mraků časově náročným i pro nejrychlejší grafické karty. Jestliže se pozorovatel (view point) blíží k mraku, tak se zvyšuje velikost překreslované částice, která je promítána, a tím dále zvyšuje množství překreslovaných pixelů. Čím je tedy pozorovatel blíže, tím více se snižuje rychlost zobrazení.

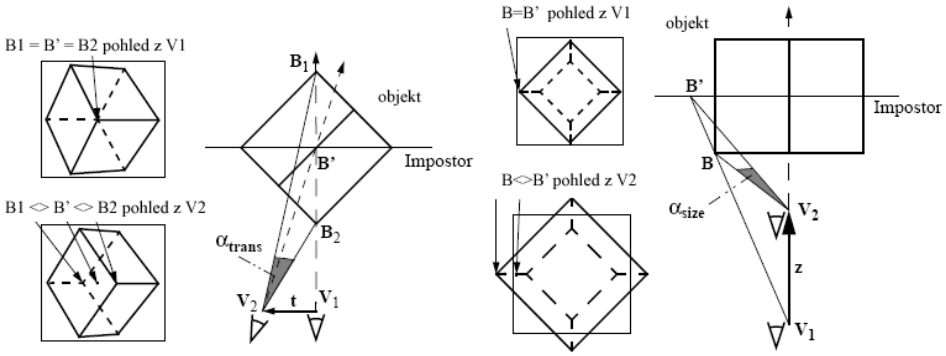


Ilustrace 6.1: Dynamický generované impostory (representovány žlutým ohraničením), které jsou generovány vůči pohledu pozorovatele, zdroj [11].

Jestliže chceme zobrazovat scény, které jsou složeny z velkého počtu částic při zachování vysokých rychlostí, existují dvě možnosti řešení. První možnost je redukce částic ve scéně a druhou možností je snížení překreslovaných pixelů. Metoda použití dynamických impostorů řeší oba dva problémy.

Impostor nahrazuje trojrozměrný objekt ve scéně dvojrozměrným poloprůhledným polygonem s namapovanou texturou nahrazeného objektu. Obraz objektu získáme přečtením framebufferu vyrenderovaným z pohledu V, který je platný (s určitou tolerancí) pro pohledy z blízkého okolí. Tolerance určuje, zda pro daný pohled musí být generován nový, nebo můžeme ponechat stávající. Nabízí se zde také možnost dopředu si vygenerovat množinu impostorů z různých pohledů a potom je zpětně použít. Tato metoda je paměťově náročná a dochází ke generování impostorů, které nebudou použity. Druhá metoda se nazývá *Dynamic generated impostors* .

Pro tvorbu oblohy je spíše vhodný druhý způsob, pohledový objem je nastaven z místa, odkud budou pozorovány mraky (capture point), a těsně sleduje bounding box objektu.

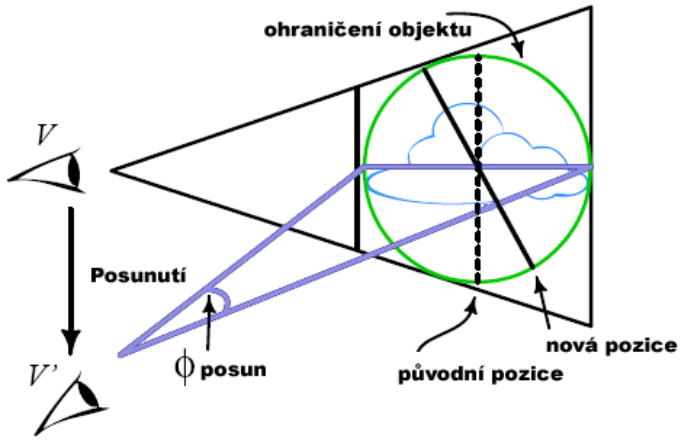


Ilustrace 6.2: Na levé straně je popsán způsob jakým dochází k translation error, na pravé straně je popsán způsob jakým dochází k zoom error.

Impostory zvyšují rychlost aplikace, protože využívají koherence v jednotlivých snímcích, které se zobrazují na monitoru. Relativní pohyb objektu ve scéně se zmenšuje s rostoucí vzdáleností od pozorovatele. U vzdálených objektů uživatel nedokáže rozeznat tolik detailů. Této vlastnosti a nepatrné změny objektu po několik po sobě jdoucích snímcích můžeme využít k znovu použití již vytvořeného obrázku (impostoru). Přibližnou chybu impostoru lze jednoduše vypočítat a tím zjistit vhodný okamžik pro aktualizaci impostoru pro aktuální pozici. Existuji dva důvody pro znovu vytvoření impostoru. První se nazývá chyba posunu (translation error), která vzniká posunutím pozorovatele o daný úhel, odkud byl daný impostor generován. Za druhé je to chyba při zvětšení (zoom error). Ta vzniká při pohybu přímo k objektu.

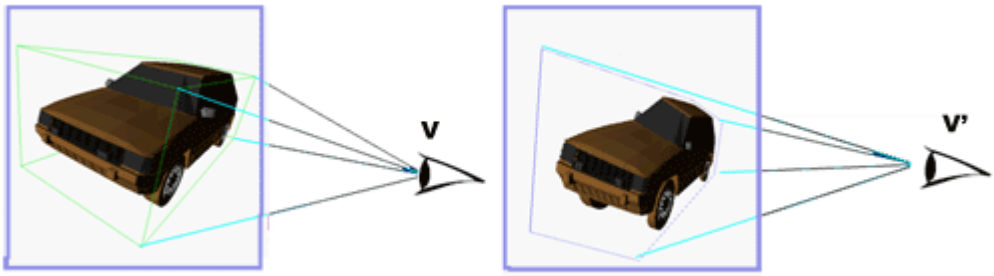
Pro výpočet chyby při posunu se vypočte úhel α vyvolaný relativním posunem vzhledem k místu odkud byl obrázek impostoru generován (capture point). Chyba při zvětšení porovnává současné rozlišení textury impostoru s rozlišením požadovaným, vypočítaným dle vzorce.

$$rozlišení_{textura} = rozlišení_{impostor} \frac{velikostObj}{vzdálenostObj} \qquad (5.1)$$



Ilustrace 6.3: Vytvoření impostoru a chyba při posunu.

Pokud chyba při posunu je větší než stanovená úhlová tolerance (0.15º). Nebo současné rozlišení obrázku je příliš malé nebo v případě kdy nastanou obě chyby najednou, musí být impostor znovu vygenerován ze současné pozice a nahradit dosavadní.



Ilustrace 6.4: Chyba při posunu pohledu a zobrazení stejného impostoru ze dvou pohledů. Obrázky ukazují posun o jeden stupeň.

6.2 Umístění pozorovatele v oblacích

Impostory poskytují velké zrychlení i v případě pozorovatele umístěného uvnitř prostoru obsahující impostory, kdy k překreslování dochází po několika snímcích. Tvar a konzistence (vzhled) mraku způsobuje obtížnost uživatele rozeznat artefakty, které impostory přináší. Pro generování impostorů je využit pohledový objem pro zobrazení celé scény.

6.2.1 Objekty v oblacích

Mraky jsou především používané v leteckých simulátorech. Pro realističnost musíme umožnit letadlům nebo jiným objektům možnost mraky prolétávat.

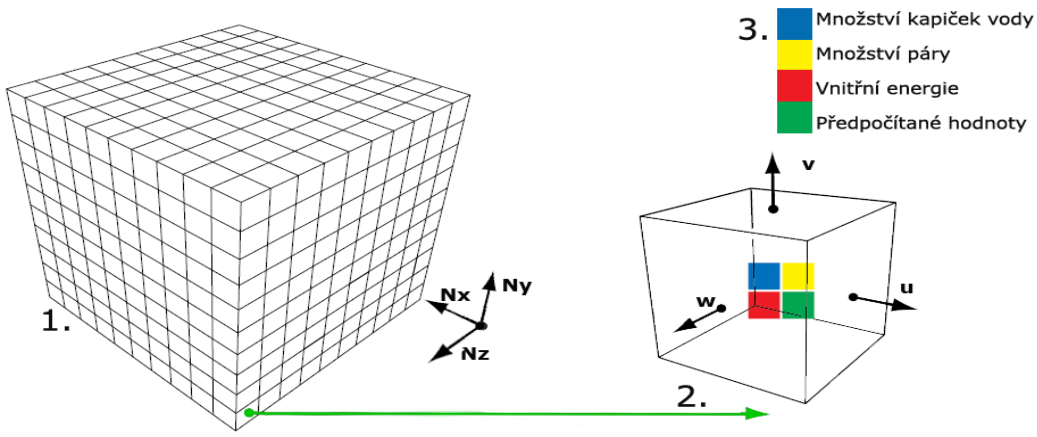
Zde je vnesen problém, že impostory jako takové jsou pouze dvojrozměrné a tím neumožňují průchod objektem. Původní myšlenka impostoru byla náhrada za geometrický tvar a nikdo nepředpokládal, že by měl umožňovat průchod skrz objekt. Jestliže necháme určitý objekt proletět mrakem, tak objekt vypadá, jako když letí skrz obrázky a ne skrz oblačnost v prostoru. Jedním z mnoha způsobů jak danou situaci řešit, je detekovat mraky, které obsahují objekty a zobrazovat jejich částice přímo do frame bufferu. Tím se ale ztrácí výhody impostorů. Dalším a lepším řešením je detekovat objekty procházejí hraniční objem mraku a rozdělit impostor, reprezentující mrak na několik vrstev. Je důležité, kolik objektů se v daném mraku nachází. Pokud se vyskytuje pouze jeden objekt v určitém mraku, pak mrak je zobrazen ze dvou vrstev. Jeden impostor mraku, který leží přibližně za objektem (tj. myslíme dále od pozorovatele) a druhý impostor, který leží před objektem. Pokud se v mraku vyskytuje více objektů než jeden, pak se mrak dělí pro dva objekty na tři vrstvy, atd. Toto rozdělení má za následek množinu střídajících se vrstev. Množina je vykreslována ze zadu směrem k pozorovateli se zapnutým z-bufferem pro objekty a vypnutým pro impostory.

Ilustrace 6.5: Letadlo v mracích, vlevo je přímo vyrenderovaný obrázek s viditelnými artefakty. Vpravo s použitím dělených impostorů, které tento artefakt odstraňují.

7 Implementace

V této kapitole popíši implementaci dynamiky a postupně popíši jednotlivé kroky a faktory, které ovlivňují průběh simulace dynamiky mraků. V první kapitole jsem popsal základní faktory, které zapříčiňují vznik oblačnosti v přírodě. Jen pro zopakování, mraky vznikají v důsledku stoupavých proudů teplého vzduchu, který sebou nese i vodní páru. Stoupá do té doby, než se jeho teplota vyrovná s okolní teplotou. V průběhu stoupání v závislosti na několika faktorech jako je: nadmořská výška (tlak), koeficient šíření tepla, páry, atd., dochází k fázové přeměně, kdy se vodní pára mění v kapičky vody, které můžeme spatřit.

Pro zobrazení mraků v reálném čase se využívá metody nazývané Coupled map lattice (dále jen CML). Prostor, ve kterém probíhá simulace je rozdělen do mřížky (Nx, Ny, Nz). Důležité je také říct, že daná simulace musí mít hraniční roviny, za které se simulace nesmí dostat. Z tohoto důvodu je rozlišení zvětšeno o jeden prvek pole v každém směru osy. Každý bod mřížky má specifické souřadnice (x, y, z), dále nese v sobě proměnné informace charakterizující jeho vlastnosti, jako jsou: vektor rychlosti $\vec{v}=(u,v,w)$, vnitřní energie, množství páry, množství vodních kapek a hodnoty, které jsou předem nastaveny jako konstanty pro urychlení simulace. Konstantní hodnoty jsou popsány ke konci této kapitoly.



Ilustrace 7.1: 1. Ukazuje pole, které definuje celý prostor simulace. Je definováno počtem voxelů v každé ose (Nx, Ny, Nz). 2. Vyobrazení jednoho voxelu simulace. Pohyb částice v simulaci je definován pomocí vektoru rychlosti . 3. Každý voxel nese v sobě jemu charakteristické informace.

Nyní když jsme popsali základní řešení problematiky dynamiky v reálném čase, můžeme přistoupit k jeho implementaci. Celé řešení dynamiky je tvořeno samostatným objektem **Fluid3d.** Každý krok (snímek) simulace je složen z jednotlivých kroků, které se stále opakují, a nepřetržitým přepočítáváním simulujeme vznik oblačnosti. Jednotlivé kroky simulace jdou volat postupně, jestliže nepotřebujeme scénu zobrazit v každém snímku, ale jen jednou za daný časový okamžik.

7.1 Simulace dynamiky (jeden krok)

1. *add(aV, e);*

2. *swap(u, uOld); swap(v, vOld); swap(w, wOld);*
   VorticityConfinement(u, uOld, v, vOld, w, wOld);

3. *swap(v, vOld);*
   addForce(v, vOld, e, c_vX);

4. *swap(e, eOld); swap(aV, aVOld);*
   project(u, v, w, uOld, vOld, aV, aVOld, e, eOld);

5. *swap(e, eOld); swap(aV, aVOld); swap(aL, aLOld);*
   swap(u, uOld); swap(v, vOld); swap(w, wOld);
   advect(u, uOld, v, vOld, w, wOld, aV, aVOld, aL, aLOld, e, eOld);

6. *swap(e, eOld); swap(aV, aVOld); swap(aL, aLOld)*
   update(aV, aVOld, aL, aLOld, e, eOld);

7.1.1 Zdroj vodní páry, síly (add)

Začneme prvním krokem, který je nejednoduší a zároveň nejdůležitější a to je zdroj vodní páry a její energie. Tyto faktory jsou základním kamenem celé simulace. Ovlivňují typ vznikajících mraků a do jisté míry také místo vzniku. Plocha (Nx * Nz) představuje podstavu simulace a také zdroj těchto dvou hodnot. Tyto hodnoty se v každém kroku přičítají ke spodní rovině prostoru simulace a představují tak zdroje vodní páry **aV** a rychlost stoupajících proudů **e** v přírodě. Tyto proudy vznikají z mnoha různých příčin a je nemožné všechny simulovat, proto je na uživateli, aby sám specifikoval, kde se budou vyskytovat v každém kroku nebo se o dané zdroje stará herní engine, který tyto místa může libovolně měnit a tím docílit požadovaných výsledků.

Ilustrace 7.2: Simulace vzniku oblačnosti s jedním zdrojem vodní páry (šipka ukazuje přibližně místo).

Ilustrace 7.3: Simulace vzniku oblačnosti s více zdroji vodní páry (šipky ukazují přibližná místa zdrojů).

Zdroje vodních par mohou být jednoduše zadané body, nebo pro lepší dosažení výsledků je vhodné využít šumové funkce a tím nerovnoměrně rozložit zdroj a dosáhnout tak více realističtějších výsledků. Důležitým aspektem je také doba, po jakou zdroj zásobuje simulaci vodní párou a energií.

7.1.2 Vířivost (VorticityConfinement)

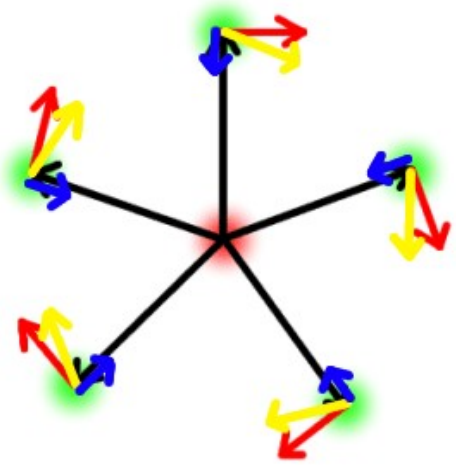
Obecně kouř nebo v našem případě stoupavé proudy horkého vzduchu obsahují velké množství prostorových odchylek od základního vektoru rychlosti, který definuje posun. Tyto odchylky vytvářejí rotace a turbulence plynu. Existuje několik možností, jak tuto vlastnost simulovatak. Jednou z cest je vytvoření a přidání pseudo-náhodné odchylky malé velikosti. Metoda tohoto přístupu je generace odchylek využívající Kolmogorovo spektrum. Takto vytvořené turbulence neodpovídají fyzikálnímu chování plynu. Přidávají detaily tam, kde se rychlost plynu ztrácí a tím znovu oživují pohyb plynu a na druhé straně přidávají malé odchylky v místech, kde dosahuje plyn velké rychlosti a tím v těchto místech dochází k malé pravděpodobnosti vířivosti. Klíč k vytvoření animací plynů spočívá ve vytvoření odchylek, které rovnoměrně odpovídají vývoji simulace a dají se simulovat v uzavřeném prostoru a jsou časově stabilní.

Metoda VorticityConfinement byla vyvinuta pro počítání turbulencí proudů okolo vrtulníků. Zde nebylo možné vytvořit dostačující rozlišení simulačního modelu, tak aby poskytl použitelné výsledky při zkoumání proudů vznikajících okolo rotoru.

Prvním krokem je generování malých detailů, které definují směr pohybu plynu. V nestlačitelné tekutině poskytují malé měřítko velikosti.

$$w = \nabla \times u \tag{6.1}$$

Každou část přírůstku si můžeme představit jako lopatku kola, které nám následně roztáčí simulaci částicového systému ve všech osách. Dále popíši jednotlivé kroky jak jednoduše metodu použít.



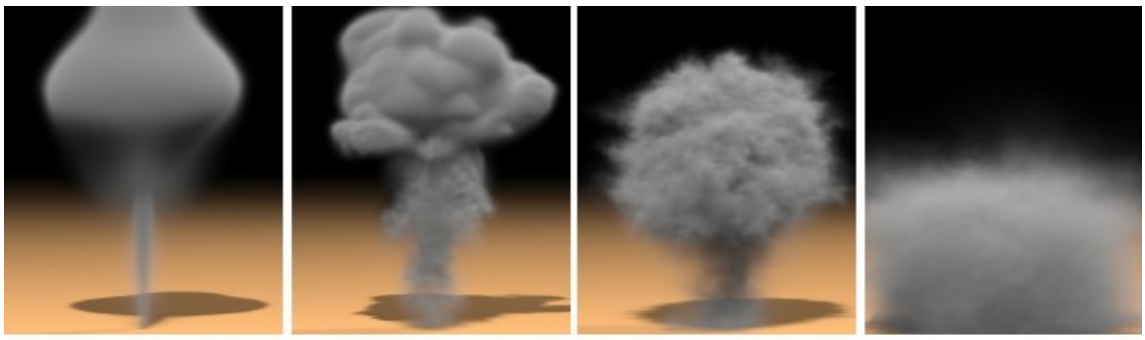
Ilustrace 7.4: Vorticity Confinement - definování vířivosti jedné částice, kdy každý vektor představuje určitou lopatku, která pak víří simulací.

Prvním krokem je nutné provést normalizaci vektoru v daném místě tak, aby směřoval z nižšího místa přírůstku rychlosti do míst s větší koncentrací rychlosti. Dále je nutné spočítat velikost "lopatky", která danou simulaci bude roztáčet. Síla a velikost se spočítá pomocí níže popsané rovnice.

$$N = \frac{\eta}{(|\eta|)} \qquad \eta = \nabla |w| \tag{6.2}$$

$$f_{conf} = \epsilon\, h(N \times w) \tag{6.3}$$

Kde $\epsilon > 0$, je konstanta určující množství detailu, které budou přidány zpětně do pole vektorů rychlostí a h je závislost. Zaručuje, že do simulace bude přidáno mnoho detailů a bude zachován fyzikální směr simulace. Tato metoda se také úspěšně používá pro složité komplexní modely.



Ilustrace 7.5: Popisuje vliv koeficientu na průběh dynamiky, zdroj [3].

7.1.2.1 Implementace Vorticityconfirment

Vířivost řešíme také v rámci uzavřeného prostoru, který nám vymezuje simulaci. Teplota (vnitřní energie), množství vodní páry a vody popisuje ilustrace 8.1. Tyto hodnoty jsou obsaženy uprostřed každého voxelu.

Řešení vychází z rovnic, které byly popsány výše. Vyžaduje jedno pole (curl) navíc. Toto pole má stejný rozměr jako ostatní, obsahuje hodnotu charakterizující okolí daného voxelu, pro názornost vypíši pseudokód. Před přidáním je nutné v každém snímku toto pole přepočítat, aby jsme dostali aktuální hodnoty.

```
float curlMethod(i,j,k){
        float du_x = (u[iX(i,j,k+1)]) – u[iX(i,j,k-1)]) * 0.5f;
        float dv_x = (u[iX(i+1,j,k)]) – u[iX(i-1,j,k)]) * 0.5f;
        float dw_x = (u[iX(i,j+1,k)]) – u[iX(i,j-1,k)]) * 0.5f;
        return du_x – dv_y -  dw_z;
}
for each voxel do curl[iX(i,j,k)] = math.abs(curlMethod(i,j,k));
```

Dále musíme zjistit velikost přírůstku v každé ose. pro tento výpočet použijeme vztahu $\eta = \nabla |w|$.

```
dn_x = (curl(i+1,j,k) – curl(i-1,j,k) ) * 0.5f;
dn_x = (curl(i,j+1,k) – curl(i,j-1,k) ) * 0.5f;
dn_x = (curl(i,j,k+1) – curl(i,j,k-1) ) * 0.5f;
```

Z takto získaných hodnot jednoduše zjistíme normálový vektor $|\eta|$, kterým jednotlivé hodnoty podělíme a získáme hodnotu N, a pak stačí doplnit získané hodnoty do vztahu a tím docílíme požadovaného výsledku. Pro lepší výsledek doporučuji změnit znaménko ve složce vektoru rychlosti

směřujícího vzhůru (v). Jestliže bychom znaménko ponechali, podařilo by se nám pouze zesílit vztlakovou silu (buyonci), ale nám jde především o rozvíření, a proto daný vektor otočíme.

V mé implementaci jsem vyzkoušel i zjednodušenou metodu, která vychází z výše popsané metody, jen je výpočetně méně náročná a její výsledky jsou srovnatelné. Tato metoda je schopna vypočítat odchylky rychlosti vektorového pole v jednom cyklu a nepotřebuje žádné pomocné pole. V konečné fázi simulace bylo použito zjednodušené metody.

Tato metoda ke každé ose rychlosti připočítává odchylku a spolu s koeficientem vířivosti získává velikost v každém směru osy. Pro názornost vyjádřím pseudokód výpočet jedné ze složek, ostatní složky je pak už jednoduché odvodit.

```
v[iX(i, j, k)] = vOld[iX(i, j, k)] + (c_e * dt) *
    ((vOld[iX(i, j+1, k)] + vOld[iX(i, j-1, k)] - 2 * vOld[iX(i, j, k)])/2+ (uOld[iX(i+1, j+1, k)]
    -uOld[iX(i-1, j+1, k)] - uOld[iX(i+1, j-1, k)] +uOld[iX(i-1, j-1, k)] +wOld[iX(i, j+1, k+1)]
     -wOld[iX(i, j+1, k-1)]-wOld[iX(i, j-1, k+1)]+wOld[iX(i, j-1, k-1)]) / 4);
```

7.1.3 Vztlak (addForce)

V tomto kroku je na základě zjednodušených fyzikálních zákonů přidáván vektor rychlosti v celém rozsahu simulace.

Tento vektor rychlosti může být tvořen určitým externím zdrojem např. foukající vítr, který naráží ze strany naší simulace a nebo vnitřní zdroj energie.

Části oblačnosti, které mají větší vnitřní energii než je energie jejich sousedů, mají tendenci vůči svým sousedům stoupat. Naopak jestliže vnitřní energie částice je menší než průměrná energie jeho sousedů, potom částice klesá a pak se spíše jedná o působení gravitace na částici. Tento vztlak gravitace ovlivňuje celé vektorové pole rychlosti. Sílu, jakou daná částice bude stoupat nebo klesat vůči svému okolí, ovlivňují jeho sousedé v horizontální výšce. Sílu vztlaku můžeme vyjádřit jednoduchou rovnicí a jednoduše převést do našeho prostoru simulace.

```
For each voxel(i,j,k) do {
        v[iX(i, j, k)] = (vOld[iX(i, j, k)] + dt * c_buy *
        (4*e[iX(i, j,k)] - e[iX(i+1, j, k)] - e[iX(i-1, j, k)] - e[iX(i, j, k+1)]-e[iX(i, j,k-1)])) *
coefVelocity;}
```

c_buy je koeficient vyjadřující míru působení vztlaku na částici. Chování stoupavých proudů je také ovlivněno nadmořskou výškou (coefVelocity). Předpokládejme stejný rozdíl energie částice a jeho okolí v celé vertikální ose systému, přesto rychlost vztlaku částice s nadmořskou výškou klesá, což je způsobeno klesajícím tlakem s nadmořskou výškou. Pro tento jev byla předpočítána tabulka koeficientů (pomocí funkce sinus), která je definována pro každou výšku simulačního modelu a

pomocí těchto hodnot můžeme simulovat chování rychlosti vztlakové síly působící na mrak v každé nadmořské výšce.

7.1.4 Projekce (project)

Projekce je hlavní částí simulace, která je časově nejnáročnější. Řeší část dynamiky a tou je spočítání výsledného pole vektorů rychlostí pro každý časový okamžik a na základě dosažených výsledků může dojít k posunu hodnot vodní páry, kapiček vody. Samotný posun řeší další krok simulace (advect). Princip projekce byl vysvětlen ve 4 kapitole, proto zde shrnu jen princip. Simulace v každém kroku přepočítává vztlak a externí sílu atd. a na základě těchto změn je přepočítáváno pole rychlostí, čímž se simulační model stabilizuje. Díky projekci zůstává celá simulace jako nestlačitelné pole.

Prvně se vypočte rozdíl (divergence) rychlostí pro každý prvek, h je fyzická délka jednoho voxelu v metrech.

```
div[iX(i, j, k)]= -0.5 * h*
      (u[iX(i+1,j,k)] - u[iX(i-1,j,k)]  +  v[iX(i,j+1,k)] - v[iX(i,j-1,k)] +  w[iX(i,j,k+1)] -
w[iX(i,j,k-1)]);
```

Následně se spočítá tlak, který působí na jednotlivé částice. Pro spočítání soustavy rovnic o více neznámých je nutné použít jednu z iteračních metod. V mé implementaci jsem použil Gauss – Seidell iterační metodu. Prakticky se ukázalo že počet iterací je vhodné nastavit na hodnotu 15. Při výrazně nižším počtu iterací se vyskytuje v simulaci mnoho artefaktů, které jsou znakem nestability systému. S rostoucím počtem iterací dochází k stabilnějšímu řešení, ale musíme si uvědomit, že každá iterace navíc představuje ohromné množství operací.

```
Počet iterací {
  For each voxel(i, j, k){
     p[iX(i, j, k)] = (div[iX(i, j, k)] +
     p[iX(i-1, j,k)]+ p[iX(i+1, j, k)] + p[iX(i, j-1, k)]+
    p[iX(i, j+1, k)] + p[iX(i, j, k-1)] + p[iX(i, j, k+1)]) / 6;
  }}
```

V tomto kroku je také řešeno šíření energie a vodní páry. Nabízí se mnoho způsobů jak zabezpečit šíření těchto hodnot, ale v simulaci nastává problém, že málo částic obsahuje hodnotu nerovnající se nule, a tím by většina lineárních systému zapříčinila vzniku mnoho chyb. Aby i tato část simulace byla stabilní a nevznikaly v simulačním modelu různé nepříjemné vizuální jevy, je využito stejné iterační metody. Pro samostatný výpočet je použito následující rovnice.

```
aV[iX(i, j, k)] = (aVOld[iX(i, j, k)] + a_c_vapor *
        (aV[iX(i+1,j,k)] + aV[iX(i-1, j, k)] +
         aV[iX(i, j-1,k)]+ aV[iX(i, j+1, k)] +
        aV[iX(i, j, k-1)] + aV[iX(i, j, k+1)])) /a_c_vapor1;
```

Kde a_c_vapor je roven součinu času konstanty udávající pravděpodobnost šíření a velikost pole (dt*c_vapor*NX*NY*NZ), a_c_vapor1 je rovno 1+6 * a_c_vapor.

Rovnice šíření vnitřní energie je úplně stejná, stačí pouze dosadit na místo hodnot udávající vodní páru hodnoty energie.

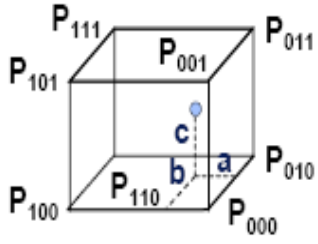
7.1.5 Posun (advect)

Podrobně tato část byla popsána ve čtvrté kapitole. V jednoduchosti sledujme pohyb částice v čase o krok zpět a na základě této hodnoty posuneme aktuální hodnotu. Jednotlivé části popíši pouze na x -ové složce a ostatní složky jsou jednoduše odvoditelné.

Prvním krokem je posunutí zpět v čase na pozici odkud hodnota přišla v čase t-1. Je důležité si uvědomit, že simulace má hraniční roviny, za které se v žádném kroku nesmíme dostat (druhá část rovnice). K řešení použijeme rovnice.

```
x = i - dtx * uOld[iX(i, j, k)]
if (x > (NX + 0.5)) x = NX + 0.5f;
if (x < 0.5)     x = 0.5f;
```

Při zpětném dívání se nám může stát, že vektor neukazuje na určitý voxel, a proto je nutná interpolace sousedů tak, abychom získali průměrnou hodnotu v daném místě. Z důvodu rychlosti jsem pro výpočet použil trilineární interpolaci, která poskytla dostačující výsledky.



$$P(a,b,c) = (1-a)\left\{\begin{matrix}(1-b)\left[(1-c)P_{000} + cP_{001}\right]+\\ +b\left[(1-c)P_{010} + cP_{011}\right]\end{matrix}\right\} + a\left\{\begin{matrix}(1-b)\left[(1-c)P_{100} + cP_{101}\right]+\\ +b\left[(1-c)P_{110} + cP_{111}\right]\end{matrix}\right\}$$

Ilustrace 7.6: Trilineární interpolace.

Pomocí popsané metody posuneme v naší mřížce všemi hodnotami, které charakterizují vlastnosti voxelu.

7.1.6 Update (update)

Tento krok popisuje změnu stavů jednotlivých proměnných. Dochází k přeměně vodní páry ve vodní kapičky a tím vzniku oblačnosti.

Množství vodních kapek vytvářených při fázové přeměně je úměrný rozdílu maximálního množství vodní páry v jednotlivých voxelech a množství vodní páry v jednotlivých vrcholech. Pro implementaci z důvodů řešení dynamiky jsem vypustil některé faktory, které tuto fázovou proměnu popisují a využil jsem rovnic 6.4 .

$$aL = aLold + \alpha(aVold - W_{max}) \tag{6.4}$$

$$aV = aVold - \alpha(aVold - W_{max})$$

$$E = E - Q(aVold - W_{max})$$

aL je množství vodních kapiček a aV je množství páry obsažené v daném voxelu, α koeficient udávají míru fázové přeměny v jednom kroku a Q je latentní teplo odevzdané př proměně páry ve vodní kapičky je rovno 2.5 kJ. T je teplota v kelvinech.

Wmax je předpočítané pole hodnot, které se inicializuje při vytvoření objektu. Je definováno rovnicí.

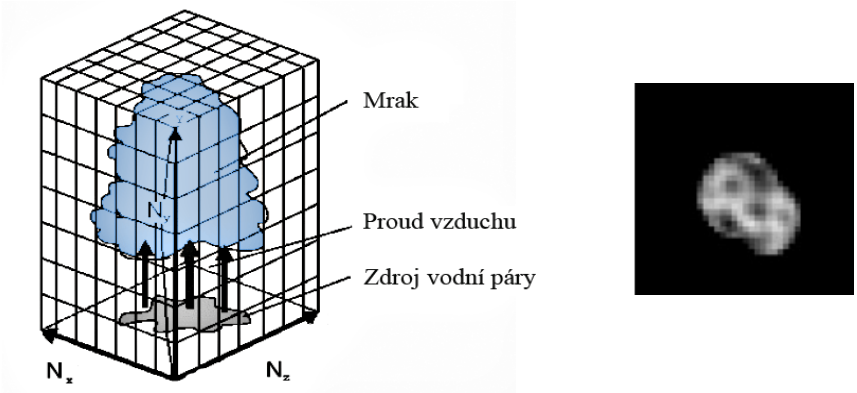
$$w_{max} = 217.0 \exp(19.482 - 4303.4/(T - 29.5))/T \tag{6.5}$$

Tato rovnice byla odvozena ze zkoumaní reality a výsledků. Byl záměrně vynechán rozdílný tlak v atmosféře a počítá se pouze s měnící se teplotou, která postupně klesá od hladiny moře (300 K). Průběh dané rovnice ukazuje graf níže. Graf (ilustrace 8.7) popisuje množství páry měnící se v kapičky vody v závislost na nadmořské výšce (teplotě T).

Ilustrace 7.7: Graf popisuje závislost teploty na velikosti fázové změny v simulaci.

7.2 Tvar výsledné oblačnosti

Simulace vzniku oblačnosti, tvaru a druhu mraků jsou v přírodě závislé na mnoha faktorech a pro jejich reálné simulování a je nutné definovat hodnoty, které jsou typické pro vznik dané oblačnosti. V dalších řádcích je popsán vliv jednotlivých faktorů, které definují druh a tedy i průběh vzniku mraků.



Ilustrace 7.8: Obrázek popisuje vznikající cumulus na základě definovaného šumu, jako zdroje energie. Pravá část definuje zdroj vodní páry, zdroj [3].

Pokud chce uživatel vytvořit mraky typu cumulus, musí nastavit zdroj vodní páry ve větším rozsahu a menší silou stoupavých proudů, pokud naopak chce vytvořit jiný známý druh mraků cumulusnymbus, musí specifikovat zdroj vodních par ve více zdrojích s malou velikostí a velkou energií stoupavých proudů.

Tabulka popisuje jednotlivé hodnoty, které jsou nastaveny před spuštěním simulace a také jakým způsobem ovlivňují výslednou oblačnost. V tabulce níže (ve sloupci „hodnota“) jsou zobrazeny hodnoty, které byly zadány jako defaultní na základě zkoumání a výpočtů. Tyto hodnoty lze opatrně měnit a pozorovat vliv jednotlivých parametrů na průběh simulace.

Název	Program	Popis	Hodnota
Šíření páry	c_vapor	Hladinu šíření vodní páry	0.00000044f
Šíření teploty	c_temp	Hladinu šíření teploty	0.000000541
Fázové přeměny	c_f	Koeficient přeměny páry v kapičky	0.5f
Vztlak	c_buy	Hladina síly způsobující vztlak	0.00925f
Vířivost	c_e	Vorticity confirement	0.95f
Výška modelu	H	Výšku simulačního modelu v metrech	15 000m

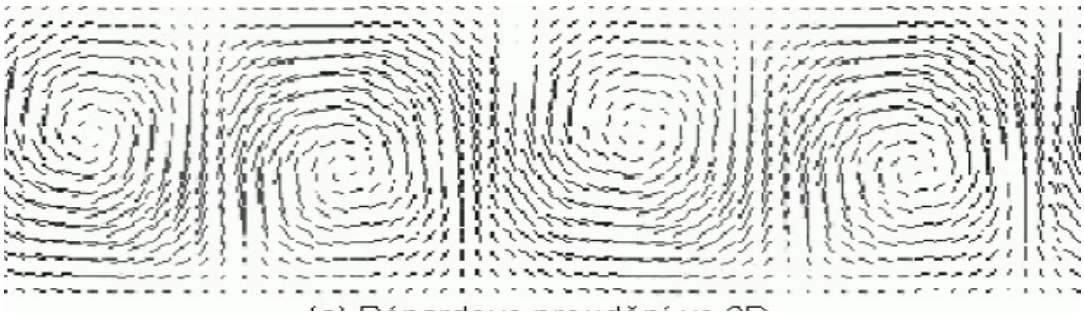
7.3 Modelování vybraných typů mraků

V simulaci je možné nastavit různé parametry a na základě těchto hodnot vytvořit určitý druh mraku. V mé implementaci se především jedná o druhy mraků cumulus, cumulusnymbus. Jestliže chceme modelovat některé ostatní oblačnosti je možnost model rozšířit o tzv. Benardovo proudění.

7.3.1 Benardovo proudění

Benardovo proudění lze pozorovat například v nádobě s vodou, kterou položíme na vařič a začneme zahřívat. Voda na dně nádoby začne stoupat vzhůru a naopak voda navrchu začne klesat. Tím se vytvoří cirkulace uvnitř nádoby, které nazýváme Benardovo proudění. Toto proudění vytvoří vzorky, které můžeme přirovnat k buňkám. Při tvorbě mraků, jako jsou například stratocumulus, altocumulus, se využívá Benardova proudění, v němž každá buňka značí jeden mrak.

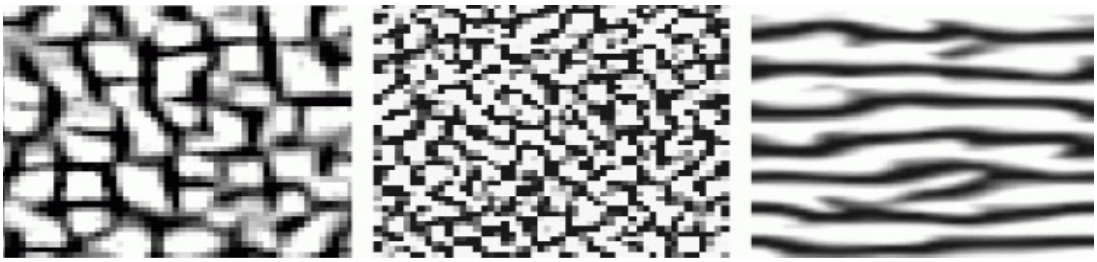
Problematiku Benardova proudění lze také řešit pomocí CML.



Ilustrace 7.9: Bernardovo proudění ve dvourozměrné mřížce, zdroj [3].

Při využití Benardových buněk nemusíme specifikovat stoupavé proudy, které jsou vyjádřeny těmito buňkami.

Potom jen záleží na uživateli, jaký typ buněk specifikuje. Pokud jsou buňky větší, vznikne stratocumulus, pokud jsou menší, může vzniknout cirrocumulus. Můžeme je také definovat horizontálně a tím nám vzniknou válcovité buňky a typ mraku altrostatus.



Ilustrace 7.10: Různé druhy Bénardových buněk, , zdroj [3].

8 Vizualizace

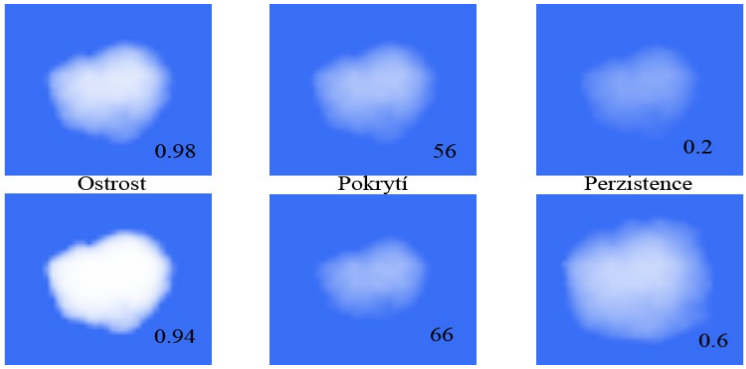
8.1 Zobrazení oblačnosti

Dnešní výpočetní výkon stolních počítačů je stále nedostačující pro simulování dynamiky v reálném čase, proto musíme simulovat dynamiku s malým rozlišením (počtem voxelů). Abychom získali přijatelné výsledky i při nižším rozlišení, je nutné řešit vzhled samotného voxelu.

Zobrazení voxelu je řešeno ve dvou krocích. První krok zobrazí textury s definovaným rozlišením. Částice je dvojrozměrná textura šumu, která je vynásobena obsahem vodních par nebo vodních kapiček v závislosti na uživateli. Rozhodující je, zda chce pozorovat dynamiku mraku nebo dynamiku páry. Následně je dvojrozměrná textura natočena k pozorovateli a jako impostor zobrazena, takto zobrazené textury z dálky vytváří dojem 3D textury.

8.1.1 Reprezentace voxelu jako 2D textury

Textura je řešena za pomocí objektu (**vaporIn**), pro její inicializaci stačí zadat velikost textury, která musí být násobkem dvou. V programu je použito rozlišení 32 * 32 pixelů, které poskytuje velmi realistické výsledky.



Ilustrace 8.1: Porovnání vytvořených textur pro jeden voxel s různými parametry.

Podrobnosti vygenerování textury jsou popsané v kapitole 3.1**.** Definováním jednotlivých parametrů určíme, jak výsledná textura bude vypadat. Hodnoty může uživatel měnit. Vliv těchto konstant na výslednou simulaci můžete vidět na ilustraci 41.

Hustota voxelu mraku je reprezentována alpha složkou textury. Pro nastavení hodnoty alpha se používá funkce v OpenGL **glColor4ub(colorR,colorG,colorB,density)**. V této metodě color definuje barvu textury a density určuje množství oblačnosti v daném bodě. Vzniklá textura pak odpovídá zobrazení pro danou částici.

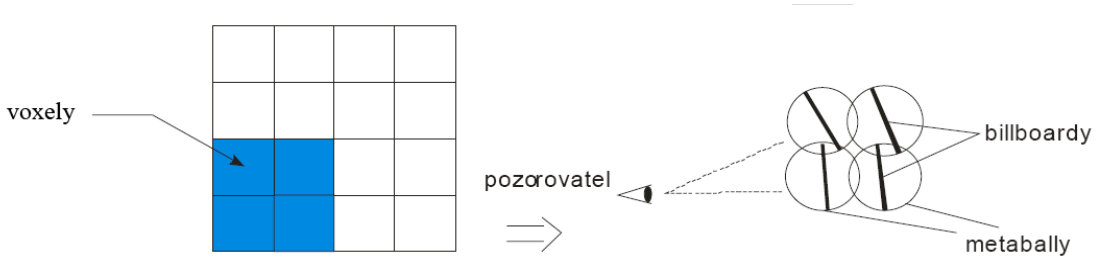
8.2 Zobrazení voxelu

Jednotlivé voxely (metabally) jsou nahrazeny dvourozměrnou texturou. Textura má přibližně kruhový tvar a její intenzita ke středu vzrůstá, takto vytvořená textura reprezentuje částici oblačnosti.

Aby tyto textury mohly reprezentovat 3D objekt, je nutné je z každého místa pozorovatele natáčet tak, aby byl zachován pravý úhel mezi pozorovatelem a zobrazovanou ploškou. Tato metoda reprezentování 3D scén se nazývá billboarding.

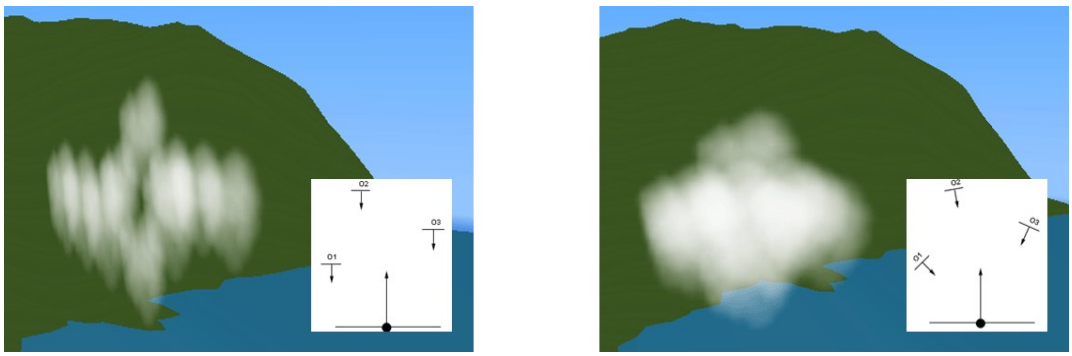
8.2.1 Billboarding

Metoda se především používá v počítačových hrách, v nichž programátoři řeší neustále kompromis mezi vizuálním dojmem a výkonnostními nároky. Výborným příkladem pro popsání metody je zobrazení stromu. Strom hlavně z jara obsahuje obrovské množství listů. Každý list je tvořen složitě definovaným 3D objektem, a proto by nebylo reálné v každém snímku přepočítávat celý 3D objekt. Programátor ví, že daný strom bude ve scéně zobrazen, a proto si může předgenerovat sadu 2D textur, které reprezentují strom z různých úhlů. Následně může aplikovat tyto textury na místo objektu, kdy se budou jednotlivé textury měnit v závislosti na pohledu.



Ilustrace 8.2: Popisuje metodu billboardu a její využíti pro zobrazení CML, vlevo řez 3D voxelovou mřížkou s voxely, které jsou převedeny na metabally a následně zobrazeny jako billboardy.

Více o zobrazení impostoru naleznete v kapitole 5. Výsledek dává iluzi 3D objektu s nesrovnatelně menší časovou náročností a porovnatelným vzhledem (více o technice bilboardingu včetně její implementace [19])



Ilustrace 8.3: 3D objekt je složen s impostorů. Vlevo vypnutá metoda natáčení, vpravo zapnutá. Bílý obrázek ukazuje směr texturek (malé plošky) a směr pohledu kamery (velká šipka nahoru).

8.3 Výpočet intenzity světla

Abychom dokázali co nejvěrněji zobrazit mrak, musíme do naší simulace také započíst zbarvení oblačnosti v závislosti na poloze Slunce. Existuje spousta metod, které řeší zobrazení oblačnosti, ale jsou časově velmi náročné (např. metody sledování paprsků), nebo jsou jejich výsledky špatně použitelné. V implementaci bylo využito metody výpočtu intenzity světla procházejícího skrz oblačnost.

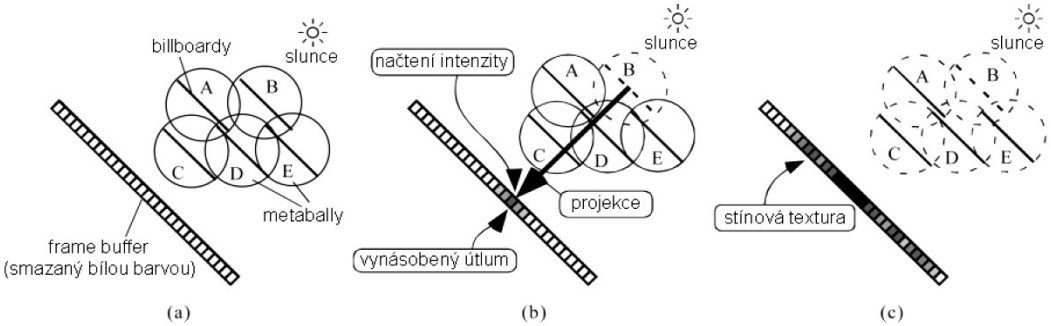
Metoda je složena ze dvou kroků, v prvním je vypočítána intenzita světla přicházející od slunce ke každému voxelu. V tomto kroku je také možné získat stínovou mapu, kterou lze použít jako vržený stín. V druhém kroku je vytvořen obraz z pohledu pozorovatele.

8.3.1 Inicializace

Princip prvního kroku spočívá ve výpočtu intenzity světla dopadajícího na střed mataballu. Tento algoritmus se snaží v plné míře využít možnosti grafického procesoru. Nejprve jsou seřazeny všechny voxely podle vzdálenosti od Slunce od nejbližšího. K řazení byl použit algoritmus QuickSort. Tento algoritmus byl vybrán pro jeho rychlost. Pozice pozorovatele se přesune na pozici Slunce s nastavenou rovnoběžnou projekcí. Vymažeme framebuffer bílou barvou a inicializujeme barvu pro kreslení na černou. Počáteční fáze: ilustrace 9.4 část (a).

Se zapnutým mícháním barev se postupně vykreslují jednotlivé voxely, kdy při každém vykreslení dalšího se zjišťuje útlum světla mezi sluncem a zobrazovanou částicí. Část (b), kde byla vykreslena částice C a její intenzita je pak čtena ze středu. Jestliže chceme spočítat útlum světla pro částice B, musím získat útlum vynásobením všech voxelů, které leží směrem ke slunci (C– D– A–E) a následně získáme hodnotu částice B. Na uvedené násobení lze efektivně použít metodu míchání barev, která je realizována na grafickém procesoru. Pro každou částici je vypočtena barva z

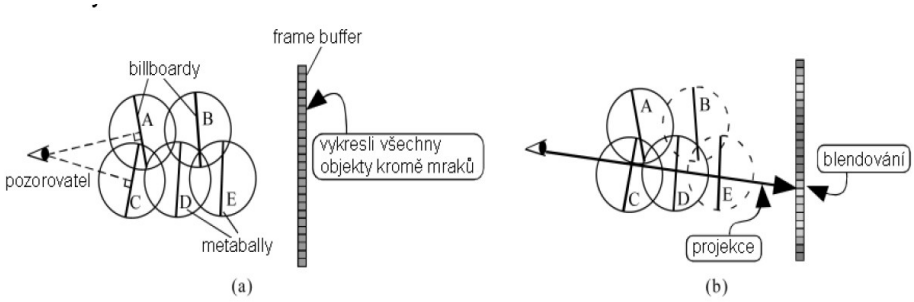
framebufferu z odpovídající pozice jeho středu. V okamžiku vykreslení všech částic, je možné uložit tuto texturu a využít jí jako stínovou mapu oblačnosti.



Ilustrace 8.4: (a) Metabally jsou nahrazeny texturami, které jsou seřazeny podle vzdálenosti od slunce. Frame buffer je inicializován bílou barvou. (b) Billboardy jsou promítnuty do framebufferu (do zobrazovací roviny). Hodnoty ve frame bufferu jsou vynásobeny útlumem v textuře billboardu (intenzita barvy určuje útlum). (c) Po vykreslení všech billboardu obdržíme texturu použitelnou pro vykreslení stínu vrženého mrakem. Zdroj obrázku [7].

8.3.2 Samotné zobrazení

V druhém kroku je nutné zobrazit všechny objekty kromě mraků a následně začít zobrazovat jednotlivé voxely, které musíme před vykreslením seřadit podle vzdáleností od nejvzdálenější částice od pozorovatele. Kamera je zpět přesunuta na místo pozorovatele a nastavena na perspektivní projekci. Následně jsou za pomocí funkce pro míchání barev postupně vykreslovány jednotlivé textury s barvou získanou v prvním kroku.



Ilustrace 8.5: Druhý krok výpočtu intenzity světla, a inicializace před vykreslováním, b postupné extendování při kreslení jednotlivých částic.

8.3.3 Provedené optimalizace

Stínování se ukázalo jako nejnáročnější část celé simulace, které je způsobeno velmi pomalou funkcí glReadPixel, která je rychlejší na grafických kartách společnosti Nvidia než kartách Ati. Rychlost je také ovlivněna neustálým řazením vzdáleností voxelů od Slunce a také od pozorovatele. Funkci pro čtení z framebufferu nedokážeme optimalizovat, ale druhou část lze. Aby nedocházelo při každém snímku ke spuštění řadícího algoritmu dvakrát, byla použita metoda spočívající v přepočítání a uložení vzdáleností. Částice je nutné seřadit pouze při změně polohy slunce nebo pozorovatele.

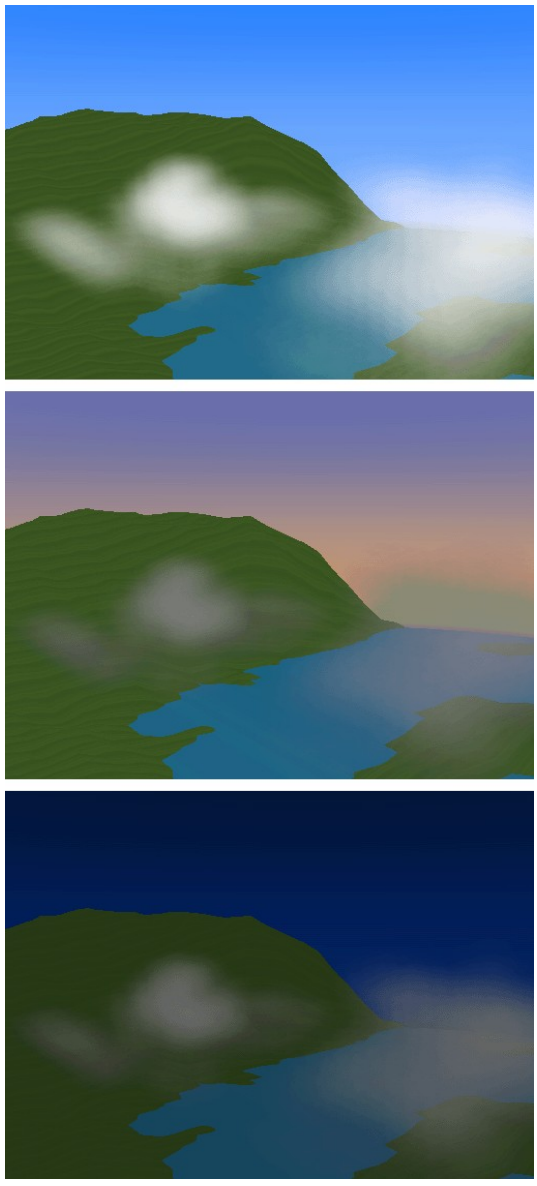
Metoda spočívá ve vytvoření dvou dalších polí o stejných rozměrech jako simulační model. Jednotlivá pole obsahují struktury, definující pozici vykreslení tak vzdálenost od pozorovatele a od Slunce.

8.4 Zbarvení mraků

Mraky během dne mění své zbarvení v závislosti na poloze slunce a vytváří tak nezapomenutelné scenérie.

Existuje mnoho metod, které dokáži spočítat intenzitu (barvu) jednotlivých částí mraku, podrobně popsány v kapitola 4.2 a 9.3). Tyto metody poskytují realistické výsledky, ale jsou výpočetně náročné.

Abychom mohli reálně a za cenu malé výpočetně časové ztráty zobrazit mraky v různých časových okamžicích, byla vytvořena jednoduchá funkce, která definuje hodnoty barvy celého mraku v závislosti na denním čase. Ilustrace 9.6., nám ukazuje zobrazení mraků ve třech časových okamžicích. Snímek nahoře nám ukazuje simulaci v čase 13:00 a zde můžeme spatřit zářivě bílé mraky. S přibývajícím časem (střední část 18:00, spodní část 21:00), můžeme spatřit postupné ztmavování mraků, které částečně odpovídá realitě.



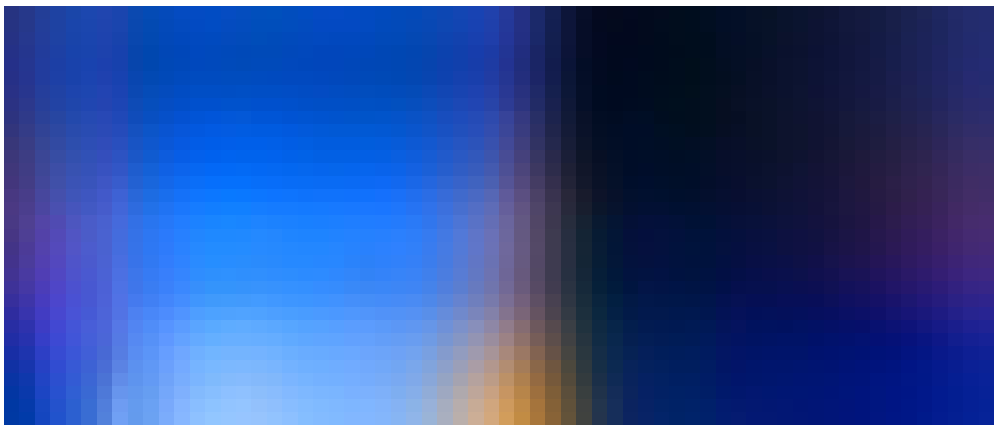
Ilustrace 8.6: Zbarvení mraků

8.5 Zobrazení oblohy

V mé simulaci jsem použil metodu, která dokáže velmi rychle zobrazovat oblohu a zároveň její vizuální výsledky jsou přijatelné. Metoda se nazývá Sky tones map. Je založena na vizuální podstatě oblohy. Využívá textůry závislé na denním čase (osa x) a výšce Slunce nad horizontem (osa y). Metoda byla použita z důvodu nízkého zatížení procesoru, který je dostatečně vytížen zobrazením dynamiky mraků.

8.5.1 Textura

Textura byla převzata z metody Sky tones map, kde autor tuto texturu získal na základě zkoumání barevného podání oblohy během dne.



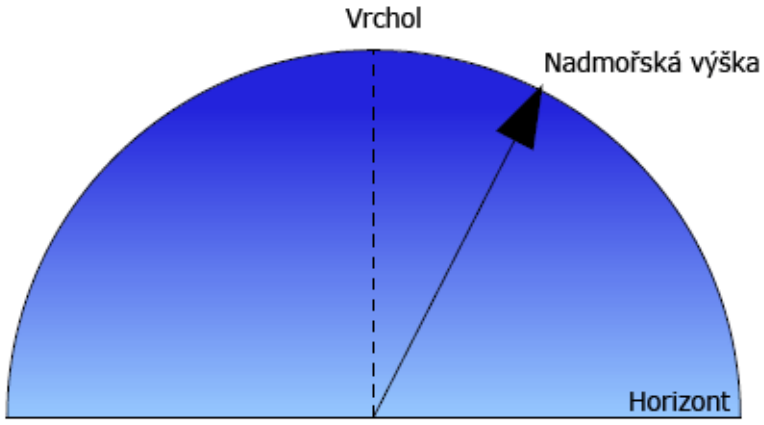
Ilustrace 8.7: Textura pro mapování oblohy.

Pro mapování textury je použita funkce

$$f(a,t)=(r,g,b) \tag{8.1}$$

Kde a je nadmořská výška, která se mění a pohybuje v rozmezí 0 až 180 stupňů, t určuje denní čas v rozsahu 0 až 1. Tato funkce nám pak vrací danou barvu r, g, b.

8.5.2 Implementace oblohy



Ilustrace 8.8: Mapování textury na polokouli.

Prvně je nutné vytvořit polokouli, na kterou se postupně nanáší textura. V některých aplikacích je lepší zvolit kouli, abychom nemuseli hlídat zobrazovanou krajinu. V mé aplikaci jsem použil polokouli z důvodu rychlosti.

Byla použita textura s rozlišením (128 * 64) pixelů, která nám dává jemné přechody v různých časových okamžicích simulace. Je nutné také použít lineární filtr textury, který nám daný sloupec (časové pásmo) mapy plynule rozvrhne po polokouli.

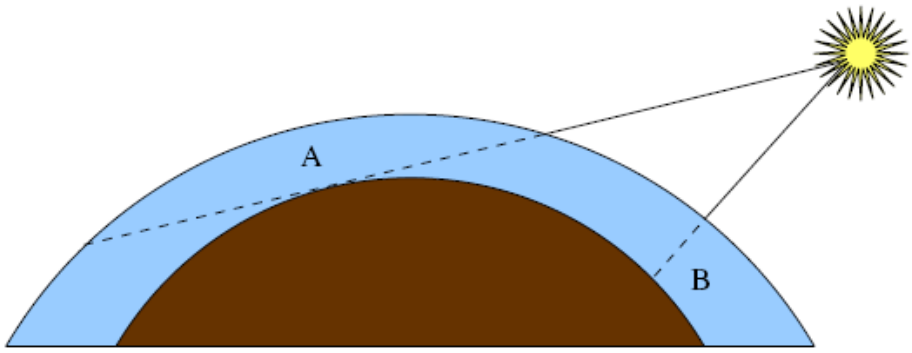
8.5.3 Mapováni

Texturovací souřadnice jsou v rozmezí (O, 1), hodnotu pro osu x získáme spočítáním úhlu dráhy slunce. V simulaci je nastaven čas na 6:00 jako východ slunce a proto je úhel Slunce roven nule (texturovací souřadnice x=0). V každém okamžiku posunu denního času simulace musí být celá polokoule překreslena.

8.5.4 Možnost dalšího rozšíření

Moje implementace oblohy nepočítá s žádnou výpočtem absorpce světla a jiných fenoménů, které určitým způsobem ovlivňují vzhled oblohy.

Atmosféra absorbuje nerovnoměrně sluneční paprsky, tuto nelinearitu způsobuje mlha, voda, mraky. Jestliže je Slunce vysoko nad obzorem, nejsou tyto rozdíly tolik patrné, čím níže je Slunce horizontu, tím jsou rozdíly větší a způsobují, že strana přivrácená ke slunci je mnohem světlejší než strana odvrácená.



Ilustrace 8.9: Různá délka dopadajících slunečních paprsků na zem.

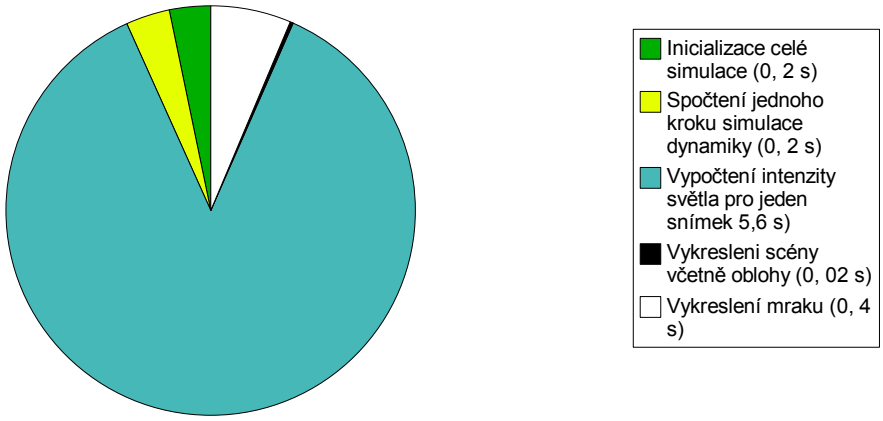
Existuje mnoho použitelných cest jak implementovat tuto operaci. Jedna z nich je implementace shaderů, které nám budou přepočítávat zbarvení pixelu na základě vzdálenosti od slunce, nebo variantou, která také využívá GPU, metodou multitexturingu.

9 Testování simulace

9.1 Časová náročnost aplikace

Pro výpočet časové náročnosti aplikace jsem provedl mnoho testů, které poskytly zajímavé výsledky. Simulace byla vyvíjena na laptopu Pentium 4 centrino 1.73 GHz, s 1024 DDR2 RAM, který byl osazen grafickou kartou ATI X600, který byl zdrojem prvních testovacích výsledků.

Pro testování jsem použil knihovnu **time.h.** Jejím využitím jsem měřil délku každého kroku pomocí metody FPS (frame per second). První fáze testování popisuje časovou náročnost jednotlivých částí celého simulačního modelu. První graf ukazuje náročnost celé aplikace, v níž simulační mřížka byla rozměru 64 * 32 * 32 částic, tedy obsahovala 65536 voxelů.



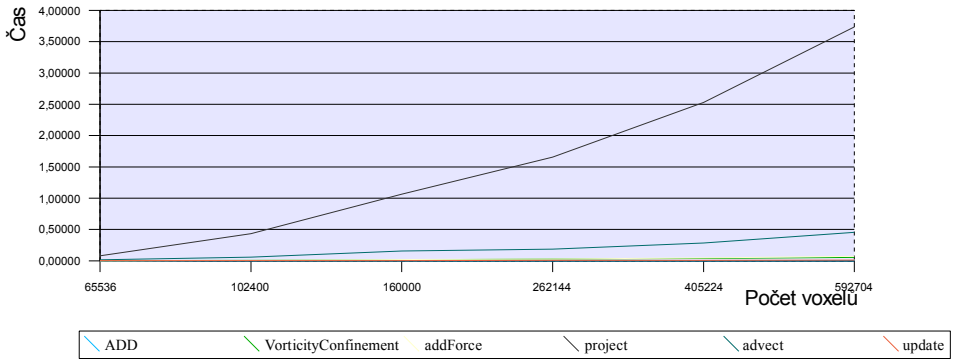
Ilustrace 9.1: Časová náročnost jednotlivých kroků.

Z obrázku je patrná obrovská časová náročnost výpočtu intenzity světla procházejícího skrz mrak. Tento čas je způsoben počtem vykreslovaných částic, a především obrovskou časovou náročností funkce glReadPixel.

9.1.1 Testovaní samotné dynamiky mraku

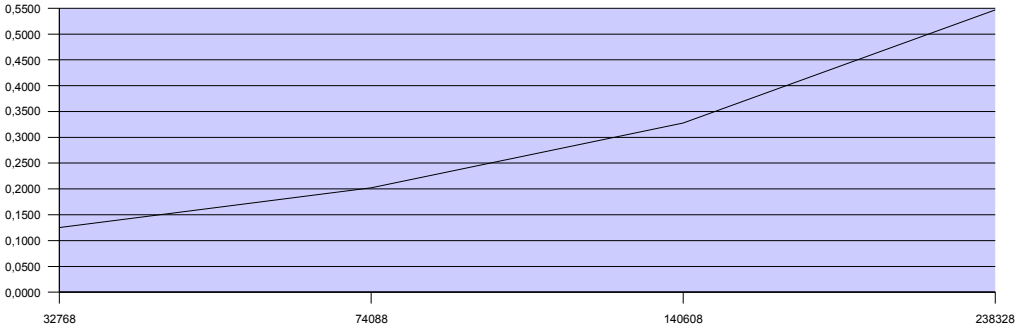
Simulace dynamiky mraku je složena z jednotlivých částí, které lze spočítat jednotlivě a následně zobrazit. Ilustrace 10.2., nám ukazuje časovou náročnost jednotlivých etap výpočtu. Z grafu můžeme jednoduše vyčíst, že čas potřebný pro výpočet simulace je převážně odvozen od času potřebného k výpočtu projekce. Projekce je časově nejnáročnější částí, je zde počítána iterační metoda, kterou řešíme výpočet tlaku a šíření částic v simulaci, dále část advect, která řeší posunutí částic pomocí výpočtu trilineární interpolace. Ostatní části jsou méně náročné a jejich náročnost nestoupá tak výrazně s rostoucím počtem zobrazovaných voxelů.

	64*40*40	64*40*40	64*50*50	64*64*64	74*74*74	84*84*84
Add	0,0001	0,0005	0,0006	0,001	0,0015	0,002
VorticityConfirment	0,0016	0,0094	0,0156	0,0282	0,0375	0,0562
AddForce	0,0012	0,0095	0,0203	0,0296	0,0469	0,0703
Project	0,082	0,435	1,063	1,657	2,531	3,735
Advect	0,00	0,0575	0,156	0,1875	0,28280	0,456
Update	0,0017	0,0015	0,0047	0,0062	0,0094	0,0014



Ilustrace 9.2: Časová náročnost jednotlivých kroků simulace dynamiky.

9.1.2 Zobrazení mraku

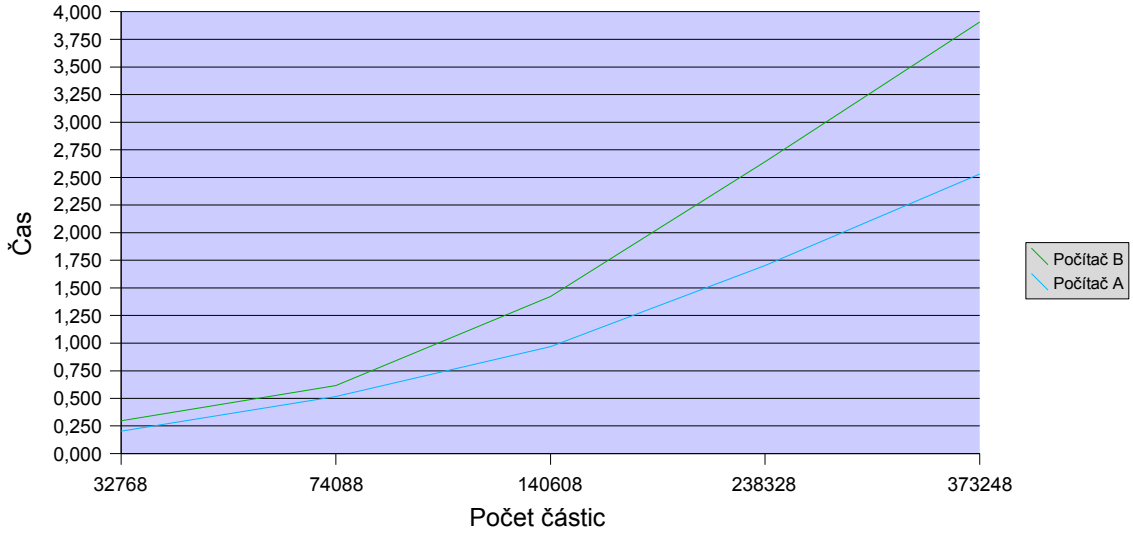


Ilustrace 9.3: Časová náročnost zobrazení mraku.

Ilustrace 52., ukazuje časovou náročnost zobrazení částic. Z grafu je patrná takřka lineární závislost mezi počtem vykreslovaných částic a potřebného času pro jeho vykreslení.

9.2 Porovnání výkonnosti

Aplikace byla dále testována na školním počítači AMD Athlon (64), procesor 3200 +, grafická karta NVIDIA 6600 (128). Tento počítač byl vybrán z několika důvodů. Jedná se o stolní počítač, jeho taktovací rychlost je výrazně vyšší a také z důvodu porovnání dvou výrobců grafických karet (NVIDIA a ATI).

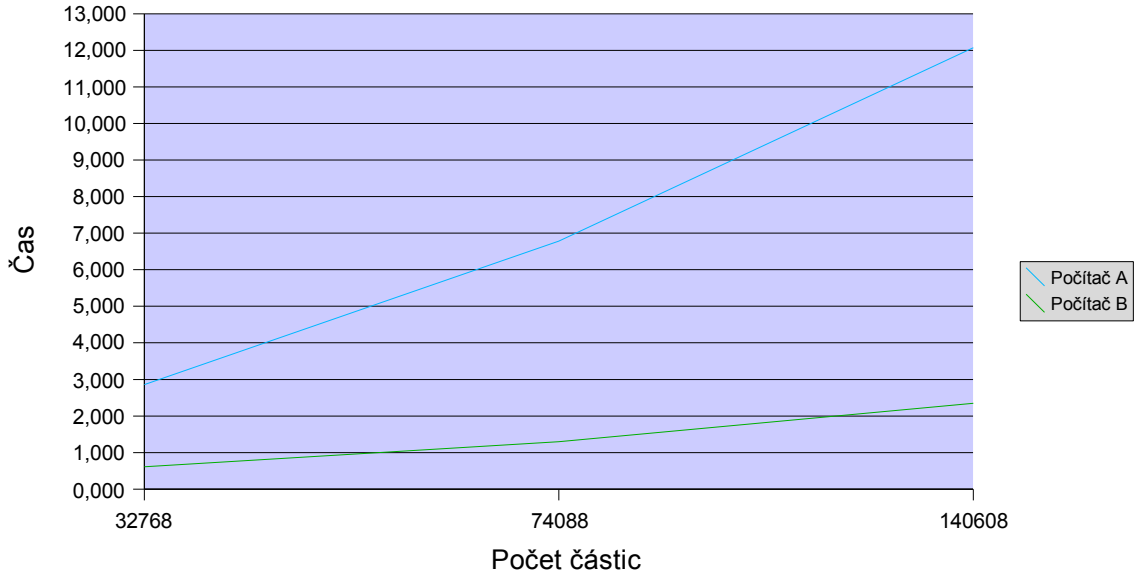


Ilustrace 9.4: Počítač A - Centino 1,73 GHz, Počítač B - AMD Athlon 3200+

První porovnání bylo zaměřeno na výpočetní výkon. Byla porovnávána časová náročnost výpočtu dynamiky na vrůstajícím počtu částic. Výsledky prvního testu jsou znázorněny na ilustraci 9.4. Z grafu je patrné zmenšení časového kroku potřebného k výpočtu dynamiky u počítače z větší taktovací rychlostí.

9.2.1 Výpočet intenzity

Další test byl zaměřen na problematiku výpočtu intenzity světla a zobrazení mraků. Test ukázal, obrovský rozdíl v potřebném času pro vykreslení a vypočtu intenzity světla jednotlivých částic. Tento rozdíl byl natolik veliký, že jsem se snažil nalézt příčinu. Test jsem několikrát opakoval, ale vždy jsem získal stejné výsledky. Tento rozdíl je způsoben výpočetně vyspělejší grafickou kartou NVIDIA. Z internetových diskuzí jsem také zjistil, že obě společnosti používají rozdílnou implementací funkce glReadPixel. Funkce je výrazně pomalejší na kartách společnosti ATI, i na dvou stejně vyspělých grafických kartách. Proto bych pro další použití aplikace a také další vývoj doporučil grafické karty společnosti NVIDIA.



Ilustrace 9.5: Počítač A - Centino 1,73 GHz, ATI - X600, Počítač B - AMD Athlon 3200+, NVIDIA 6600

10 Závěr

Účelem této práce bylo jak teoretické pochopení a nastudování problematiky vzniku mraků v reálném čase, tak samotné zobrazení. Podařilo se mi popsat všechny důležité metody zobrazující oblačnost včetně popisu implementace. Samostatný vývoj směřoval dvěma směry. Na jedné straně ukázal výsledky, které je možně získat metodami založených na vizuální podstatě a na druhé straně použití metody dynamiky kapalin pro vytvoření oblačnosti založené na fyzikálních zákonech.

Při implementaci jsem se zaměřil na druhý směr využívající fyzikální zákony směřující ke vzniku oblačnosti. Pro dynamiku mraků byla použita fluidní dynamika, její výhodou je velká variabilita, kdy uživatel může pomocí nastavení potřebných parametrů použít vytvořený objekt simulačního modelu v jakékoli aplikaci. Nevýhodou je poměrně vysoká výpočetní náročnost. Aplikace je plynule spustitelná jen na výkonných strojích při vysokém rozlišení mřížky, proto bylo využito i Perlinova šumu pro vytvoření textur, které nahradily jednotlivé částice a tím získal simulační model i v malém rozlišení přijatelný vzhled.

Směry, kterými se může tato diplomová práce dále rozvíjet, je několik. Jednou z možností je doplnění simulačního modelu větrem ve tvaru benardových buněk, a tím bychom mohli dosáhnout větší variability zobrazované oblačnosti (více druhů mraků). Dalším směrem je pokusit se o zvýšení rychlosti výpočtu intenzity světla, které se ukázalo výpočetně nejnáročnější. Složitost spočívá především ve čtení daného pixelu z framebufferu. K dalšímu zrychlení by také mohlo dojít při přenesení některých výpočtů na grafickou kartu. Toto přenesení by obnášelo převést jednotlivé složky pole rychlostí na jednotlivé barevné kanály textury, následný výpočet jednotlivých iterací pro jednotlivé složky rychlostí by se mohl počítat paralelně. Tato optimalizace by mohla do určité míry urychlit nejpomalejší část simulačního modelu.

11 Literatura

[1] N. Foster and R. Fedkiw. Practical Animation of Liquids. In Proceedings of SIGGRAPH ’01, volume 26(4) of Annual Conference Series.

[2] J. Stam. Stable Fluids. In Proceedings of SIGGRAPH ’99, volume 25(4) of Annual Conference Series

[3] Y. Dobashi, K. Kaneda, T. Okita, T. Nishita, A Simple, Efficient Method for Realistic Animation of Clouds. Proc of SIGGRAPH 2000

[4] M. J. Harris, A. Lastra. Real-Time Cloud Rendering. Eurographics 2001, 20(3):76-84, 2001.

[5] K. Perlin. An Image Synthesizer. Proc of SIGGRAPH’85, pages 287-296, 1985.

[6] R. Miyazaki, S. Yoshida, Y. Dobashi, T. Nishita, A Method for Modeling Clouds based on Atmospheric Fluid Dynamics, Pacific Graphics 2001, pages 363-373,2001.

[7] Fedkiw, R., Stam, J. and Jensen, H.W., "Visual Simulation of Smoke", SIGGRAPH 2001, 23-30 (2001).

[8] N. Foster and D. Metaxas. Realistic Animation of Liquids.Graphical Models and Image Processing, 58(5):471–483, 1996.

[9] Michal Poneš. Modelování a renderování mraků. ČVUT Katedra počítačů , 2003

[10] Mark J. Harris. Implementation of a CML Boiling Simulation using Graphics Hardware

[11] Mark J. Harris, Greg Coombe, Thorsten Scheuermann, Anselmo Lastra. Physically-Based Visual Simulation on Graphics Hardware

[12] R. Miyazaki, Y. Dobashi, T. Nishita. Simulation of Cumuliform Clouds Based on Computational Fluid Dynamics

[13] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, James F. O’Brien. Fluid Animation with Dynamic Meshes

[14] Sky and Atmospheric Rendering. Anglicky.
Dostupný z : <http://www.vterrain.org/Atmosphere/index.html>

[15] Jesús Alonso Abad, Simple model for fast and realistic sky dome.
Dostupný z : <http://www.geocities.com/ngdash/whitepapers/skydomecolor.html>

[16] Paul Bourke, Parametric Equation of a Sphere and Texture Mapping. Anglicky.
Dostupný z : <http://local.wasp.uwa.edu.au/~pbourke/texture_colour/spheremap/>

[17] Xavier Decoret, Gernot Schaufler, François Sillion, Julie Dorsey. Multi-layered impostors for accelerated rendering

[18] NVIDIA Corporation, *GPU Gems : Programming Techniques,Tips, and Tricks for Real-Time Graphics*. Dostupný z : < http://developer.nvidia.com>

[19] Jiří Sochor, FI MU Brno. Přímé zobrazování objemových dat, 2003.

[20] Billboarding (přiklápění polygonů ke kameře). Anglicky.
Dostupný z : <http://nehe.ceskehry.cz/cl_gl_billboard.php>

[21] Billboarding – Billboard tutorial. Česky.
Dostupný z : <http://www.lighthouse3d.com/opengl/billboarding/index.php3?billCyl>

[22] NeHe, OpenGL – NeHe OpenGL. Česky.
Dostupný z : <http://nehe.ceskehry.cz/tut_obsah.php>

[23] ROOT.CZ , Grafická knihovna OpenGL . Česky.
Dostupný z : <http://www.root.cz/clanky/graficka-knihovna-opengl-1/>

12 Přílohy

12.1 Ovládání programu

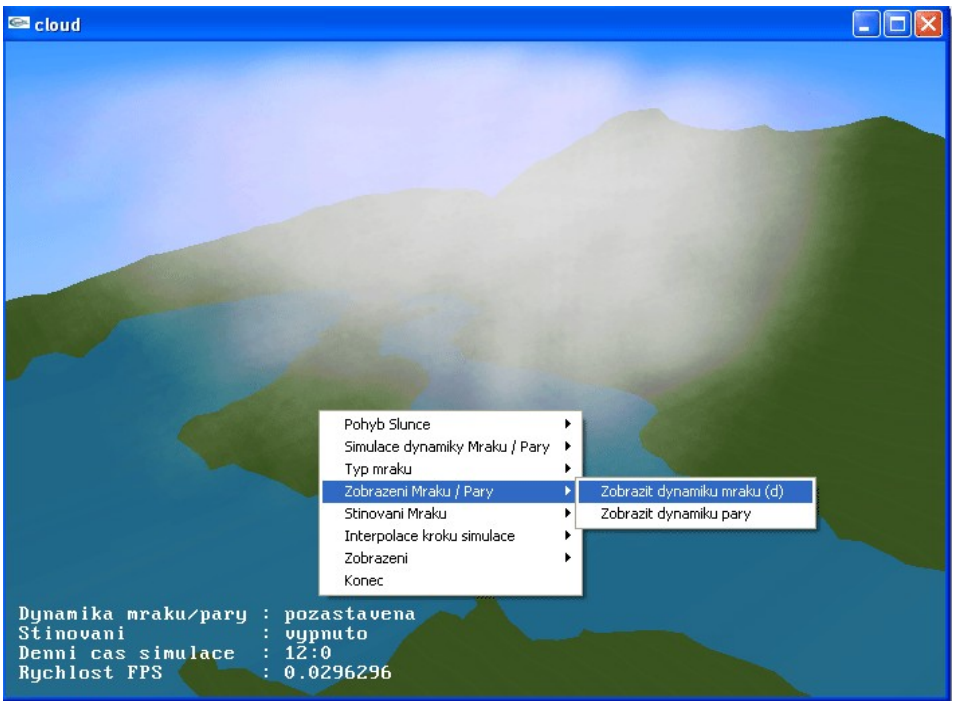
Celá vizualizace byla vytvořena jako ukázka funkčnosti jednotlivých tříd, proto není možné měnit všechny parametry simulace dynamiky, které kontrolují vzhled a vývoj simulace. Předpokládá se další možné použití a to jak třídy (**Fluid3d**), která se stará o dynamiku simulace, tak také třídy (**vaporIn**), sloužící pro vytvoření částic (2D textur). Nastavením jednotlivých parametrů u zmíněných tříd může uživatel použít dané třídy podle svých představ.

Hardwarové nároky jsou velké. Při experimentech bylo zjištěno, že minimum pro spuštění aplikace je počítač s taktovací rychlostí 1,5 GHz, 512 MB RAM, operační systém Windows 2000 a výše. Pro pohodlné prohlížení zobrazované scény je nutná třítlačítková myš. Na počítačích se slabším výkonem bude simulace spuštěna s velmi nízkým číslem FPS a nebude působit plynulým dojmem. Pro spuštění aplikace doporučuji procesor s taktovací rychlostí 2GHz, na kterém bude možnost přepnout aplikaci do zobrazení na celý monitor (klávesa f).

Spuštění aplikace

Pro spuštění programu dochází k inicializaci všech potřebných hodnot, proto spuštění může zabrat delší časový okamžik. Aplikace zobrazí scénu (ilustrace 12.1). V případě, že nedošlo ke spuštění, zkontrolujte, zda na vašem počítači byla nainstalována knihovna GLUT a GLU, které jsou nezbytné.

Ve spodním levém rohu může uživatel vidět základní informace o běžící aplikaci. Aplikace je spuštěna předem definovaným nastavením. Dynamika je zapnuta, stínování vypnuté, nastavení denního času na 9:00.



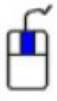
Ilustrace 12.1: Základní vzhled spuštěné aplikace.

Ovládání pomocí myši

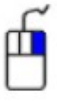
Myš slouží jako hlavní ovládací prvek pohybu po scéně, jak již bylo řečeno v hardwarových požadavcích, je nutná třítlačítková myš. S jinou myší není možně využít plné funkčnosti.

Funkčnost jednotlivých tlačítek myši.

Při zmáčknutém levém tlačítku, můžeme natáčet pohled na zobrazovanou scénu ve všech osách, okolo středu zobrazení.



Při zmáčknutém prostředním tlačítku, je možné se přibližovat nebo oddalovat vůči středu zobrazení. Této funkce je možné využít pouze s použitím třítlačítkové myši.



Po zmáčknutí krajního pravého tlačítka se zobrazí jednoduché pop-up menu, které dává možnost uživateli definovat některé parametry zobrazované scény.

Ovládání pomocí klávesnice

Klávesnice slouží pro přepínání jednotlivých stavů zobrazované aplikace.

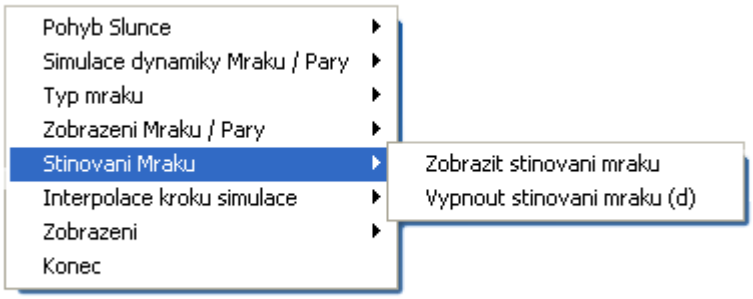
ESC Ukončí aplikaci.

F Přepne zobrazení na celou obrazovku, zde musím upozornit, výpočetní náročnost aplikace výrazně vzroste.

W Přepne simulaci zpět do původního okna s rozlišením 700 * 500.

Ovládání pomocí zobrazeného pop-up menu

Menu dává možnost uživateli nastavit jednotlivé parametry zobrazené scény. Každá část menu umožňuje vybrat požadované chování modelu. Předem definované nastavení je označeno malým písmenem d.



Ilustrace 12.2: Menu, umožnující měnit jednotlivé vlastnosti aplikace.

12.2 Obsah přiloženého CD

Toto CD obsahuje jednotlivé knihovny, které byly vytvořeny pro podporu aplikace, jsou umístěny v adresáři knihovny.

- Fluid3d.h – obsahuje třídu Fluid3d, která se stára o dynamiku simulace.
- MyMath.h – obsahuje jednotlivé matematické funkce, které byly vytvořeny pro podporu simulace.
- TextureNoise.h – obsahuje třídu vaporIn, která se stará o vytvoření textury částice a funkce pro načítání textur.
- DrawFunction – funkce, které se podílejí na vykreslování jednotlivých částí scény.
- ProjectionTexture – funkce starající se o zobrazování textu na obrazovce uživatele v OpenGL.

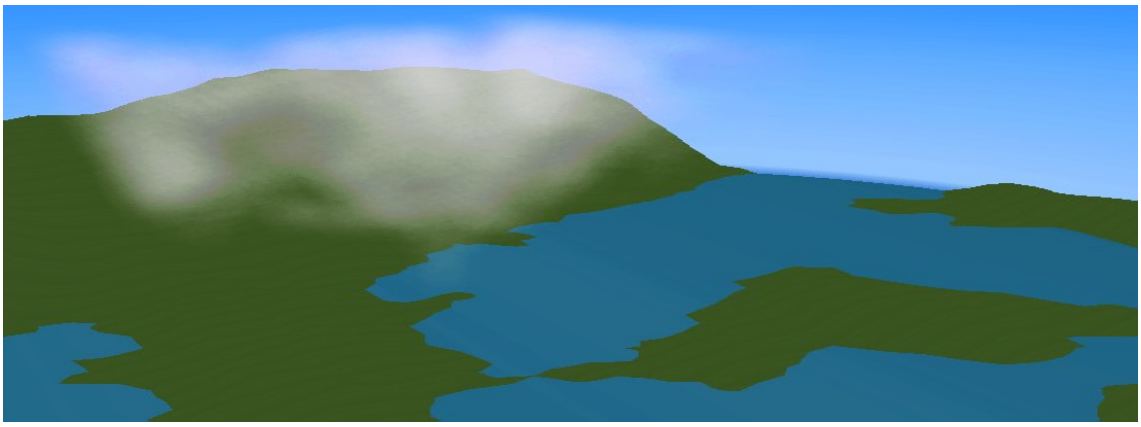
CD obsahuje všechny potřebné textury pro spuštění scény. Tyto textury jsou uloženy v adresáři Data.

- Sky.bmp – textura oblohy
- teren.raw – bump mapa, zdroj pro vytvoření krajiny
- zeme.bmp – textura pro texturovaní krajiny
- wave.bmp – textura pro texturovaní vodních ploch

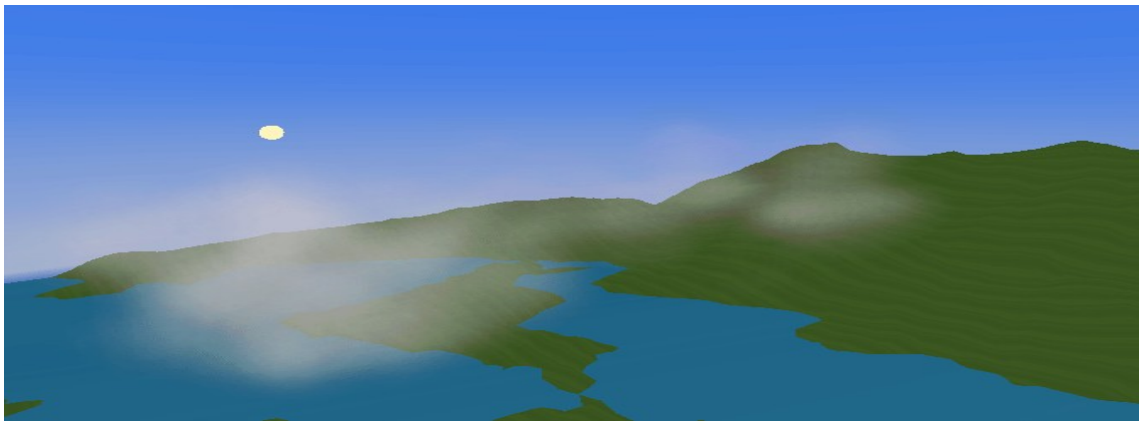
Většina literatury zabývající se touto tématikou byla nahrána do adresáře Literatura.

12.3 Obrazové výsledky simulace

Ilustrace 12.3: Jeden zdroj v deset hodin večer.



Ilustrace 12.4: Jeden zdroj v pravé poledne.



Ilustrace 12.5: Ukazuje výsledek simulace při nastavení několika vodních zdrojů v pět hodin odpoledne.