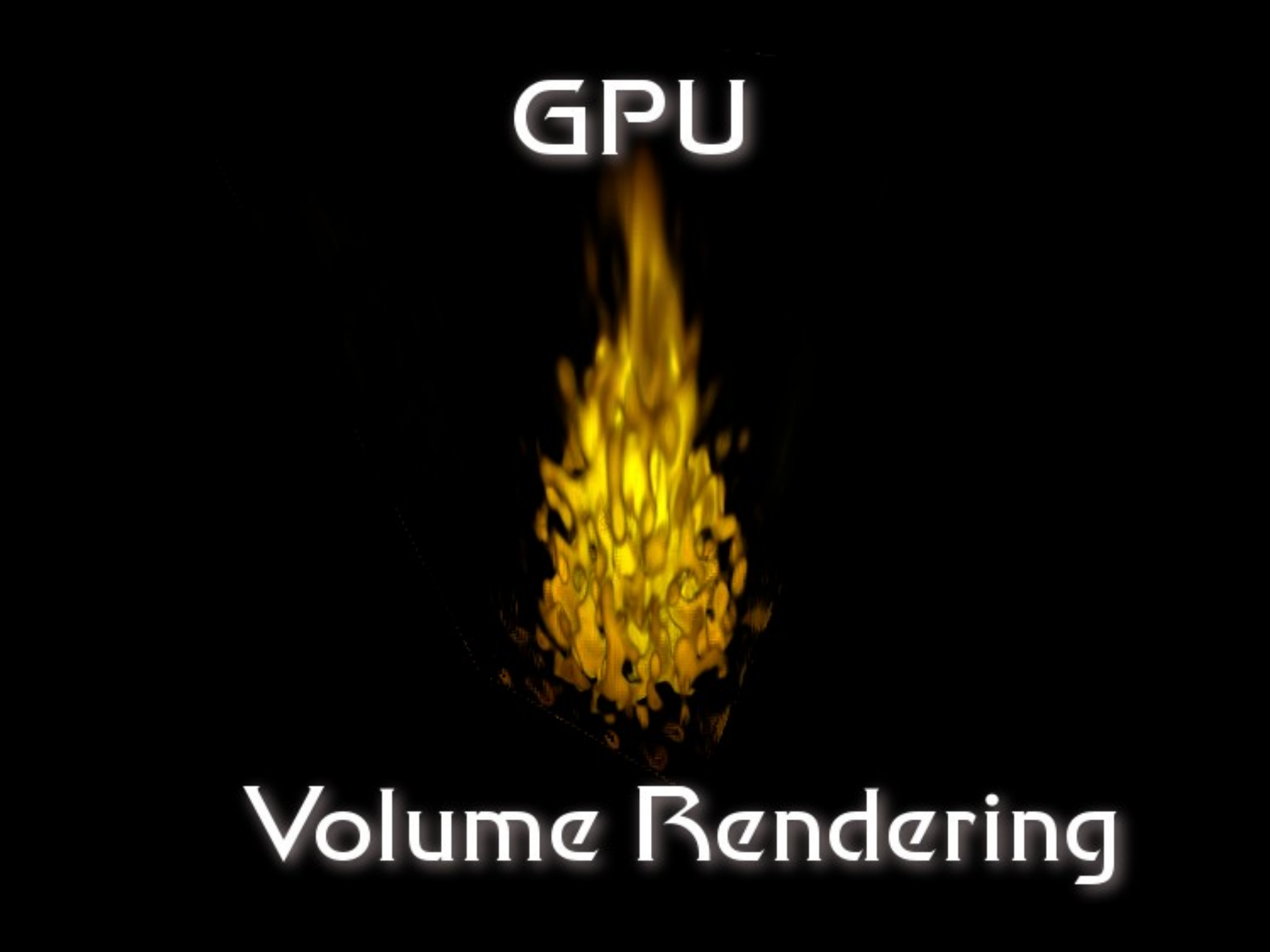
# GPU



# Volume Rendering

# Základ přednášky

- **V**ýsledky diplomové práce na MFF UK
- **Z**obrazování objemových dat pomocí GPU
  - Zobrazení dat zadaných funkcí $R^3(R^4)$ --> $R^n$
- **C**íle práce
  - Minimální zatížení CPU
  - Interaktivní rychlosti
  - Omezení na moderní GPU

# Obsah

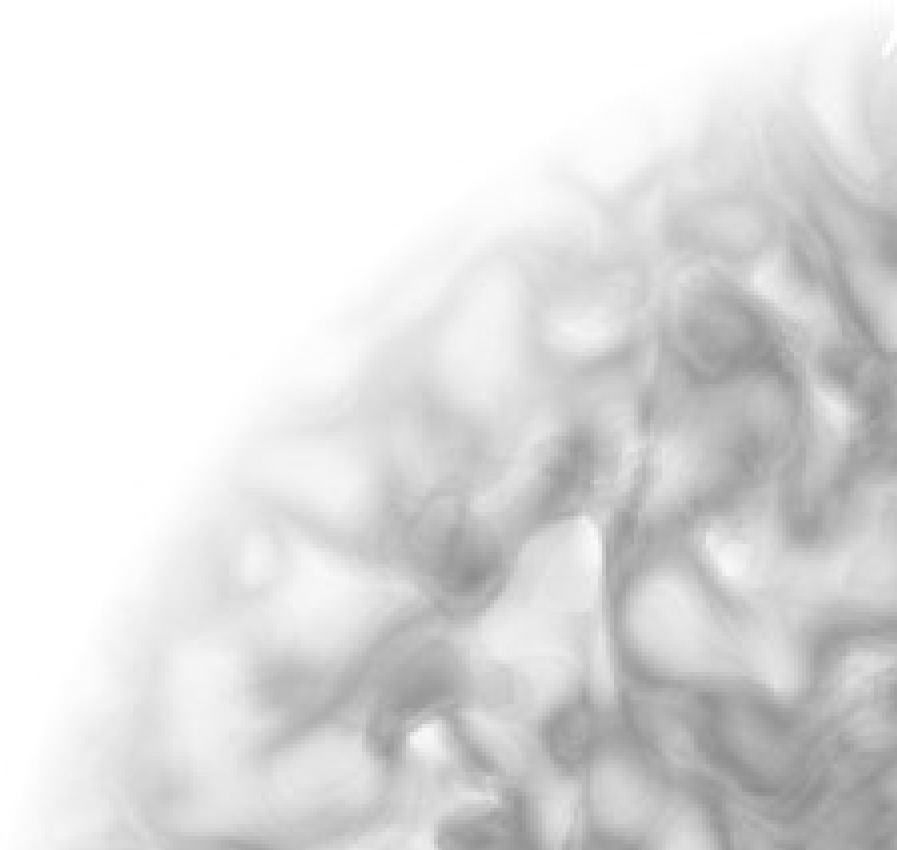
- **V**ýběr metody pro implementaci
- **U**kázky výsledků
- **D**etaily metody
- **S**hrnutí a užitečnost
- **A**ktuální práce
- **O**dkazy
- **D**iskuse nad ukázkami

# Vývojové prostředí

- **P**ůvodní záměr byl použít přímo produkční prostředí
  - Příliš neflexibilní
- **F**X Composer 1.6
  - Přechod na HLSL/DirectX
  - Mladé ale ideální prostředí
- **G**eForce 6 Series
  - Nutnost použití Shader Model 3.0

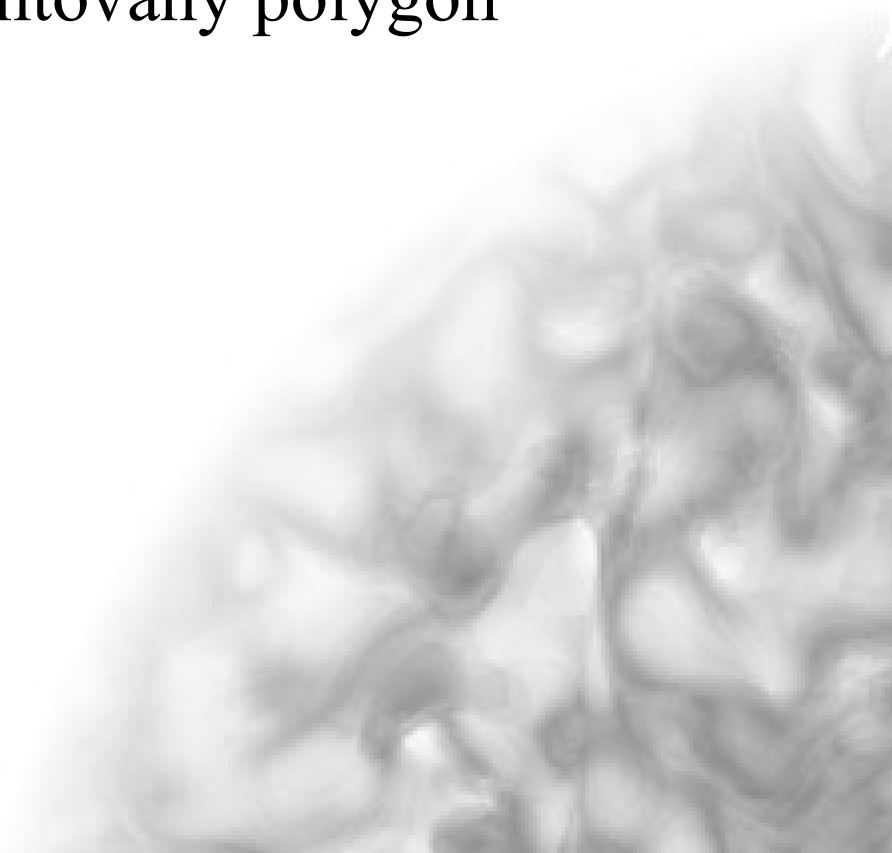
# Předstartovní příprava

- **O**ffline / CPU algoritmy
  - Billboardy, Slicing
  - Hledání povrchu
  - Přímé zobrazovací metody
- **E**xistující GPU algoritmy
  - Billboardy
  - Částicové systémy



# Billboardy

- **V**ytvoření 2D reprezentace fenoménu
- **N**amapování textury na orientovaný polygon
- **P**roblémy
  - Ztráta dimenze
  - Interakce s ostatnímy objekty
  - Omezené možnosti
- **P**ozitiva
  - Rychlost
  - Rozšířenost



# Slicing

- **N**ěkolik sad orientovaných průřezů
- **K**ompozice průřezů pomocí alfa kanálu
- **P**roblémy
  - Generování průřezů
  - Přechod mezi sadami
  - Omezené možnosti
- **P**ozitiva
  - Rychlost
  - Možnost předgenerovat průřezy

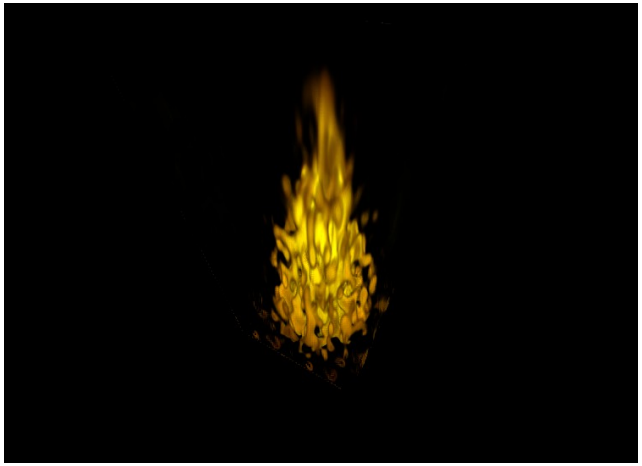
# Hledání povrchu

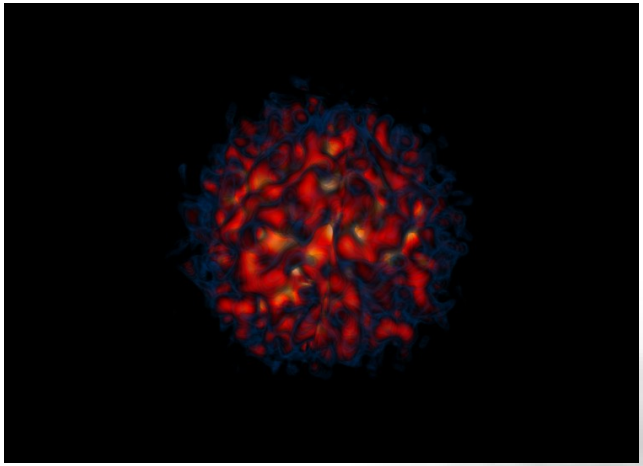
- **N**ejznámější zástupce Marching Cubes
- **K**onstrukce polygonálního povrchu z izoploch
  - Podle polohy izoplochy v buňce se generuje sada trojůhelníků
- **N**epříliš vhodná pro GPU implementaci
  - „Volume driven“
  - Nutnost uložení velkého počtu degenerovaných trojúhelníků
  - Povrch se jednou spočítá, pak zobrazuje
  - Nepříliš vhodné po průsvitné a průhledné povrchy

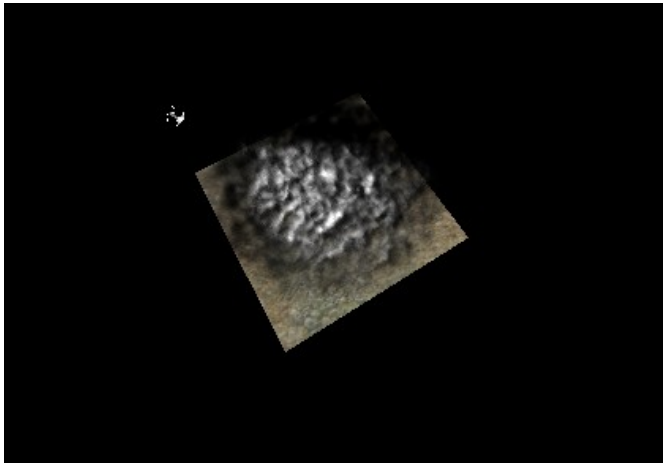
# Přímé zobrazování

- **S**platting
  - Opět nepříliš vhodné pro GPU implementaci
  - „Volume driven“
- **V**rhání paprsku
  - „Pixel driven“
  - Velká flexibilita v poměru: náročnost vs. Kvalita
  - Fyzikálně nejreálnější
  - Minimum dodatečné geometrie

# Trocha motivace





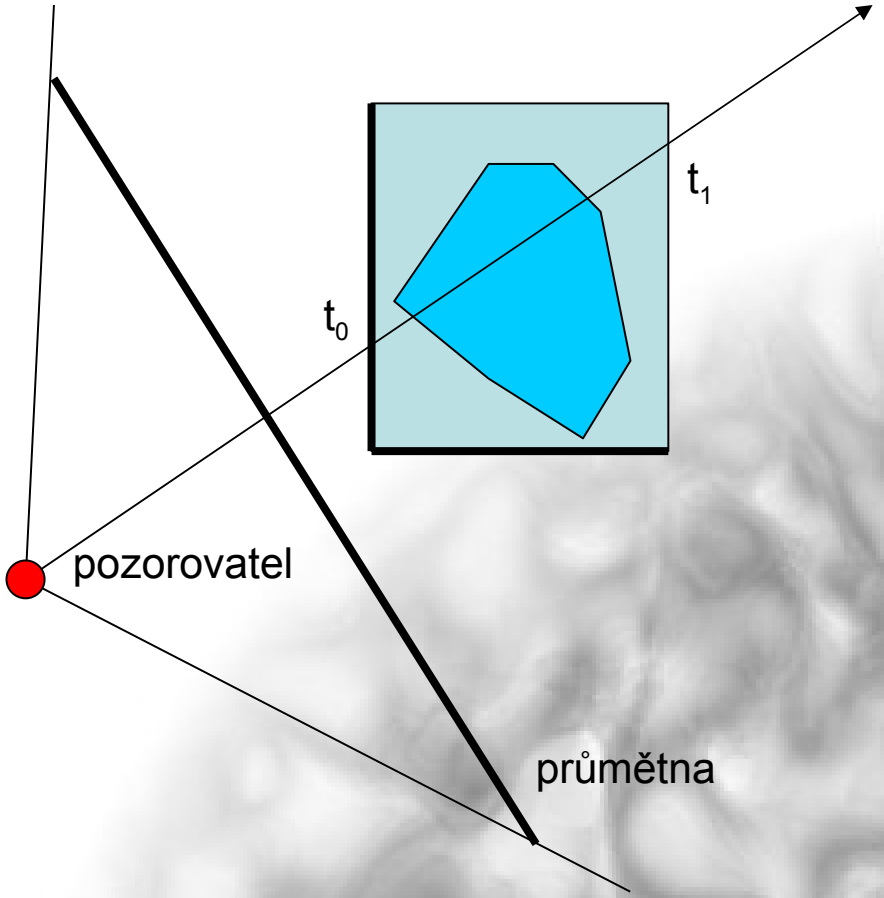


# Princip metody

- **Z**obrazování pomocí vrhání paprsku
- **V** objemu aproximace integrálu pro výpočet průchodu záření materiálem
- **L**ow-albedo model propagace světla
- **D**ata uložena jako 3D textura

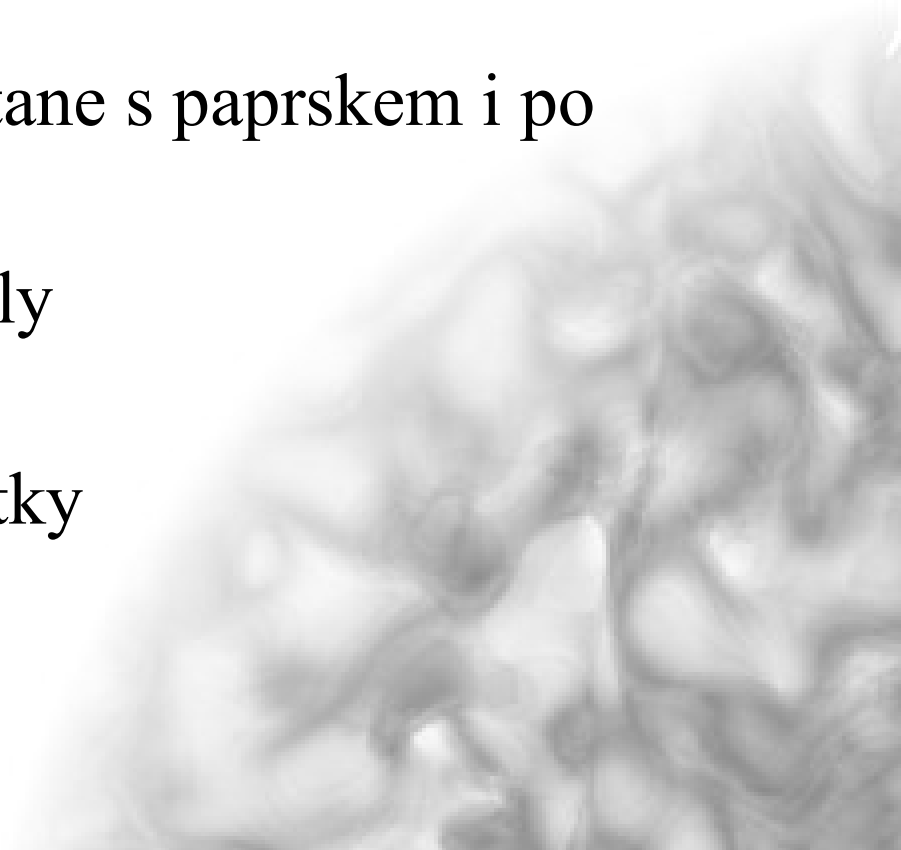
# Vrhání paprsků

- **J**ak generovat paprsky
  - Proxy geometrie
  - Velké nároky na rychlost zpracování pixelů
- **K**de paprsky vzorkovat
  - Přidat jeden průchod
  - Offscreen FP buffer



# Low-albedo model

- **P**ředpokládá, že sekundární a další odrazy jsou nedůležité
- **H**igh-albedo počítá, co se stane s paprskem i po primárním odrazu
- **V**hodný pro difusní materiály
  - Mlha, kouř, mraky
- **N**evhodný pro kontrastní látky
  - Sprška vody



# 3D textury

- **P**řirozený způsob, jak objem uložit
- **J**iž několik generací karet HW podpora
- **J**ak je vytvářet
  - CPU
    - dds formát umí „Volume textures“
    - D3DX je umí pohodlně zapisovat a pracovat s nimi
  - GPU
    - Generování přímo z HLSL
    - „FillFunction“

# Osvětlení

- **D**osud $R^3(R^4)$ --> R
- **P**ro nevyzařující materiály chceme $R^3(R^4)$->$R^n$
- **A**ktuálně
  - Gouradův model
  - Generování normál
  - Nebere v úvahu útlum na dráze od světla
- **S**tínovací paprsky

# Shrnutí metody

- **V**ygeneruj nebo načti objemová data
- **V**ykresli odvrácené polygony a paprsky zapisuj do textury
- **V**ykresli přivrácené polygony,
  - Vytvoř paprsek a urči $t_0$ a $t_1$
  - Pro každý vzorek na tomto paprsku
    - Urči hustotu a barvu v tomto bodě
    - Přičti k výsledku a hustotu akumuluj
  - Výslednou barvu zapiš do pixelu

# K čemu se to hodí

- **H**ry
  - Další odebrání práce z CPU
  - Další zvýšení realističnosti
- **E**fektivní vývoj procedurálních textur
- **V**ědecká vizualizace

# Aktuální práce I

- Testy parametrické obálky
  - Vyplatí se ztráta obecné geometrie ?
- Stínovací paprsky a rozptyl světla
  - Myšlenkově snadné
  - Technické problémy
    - Zápis do 3D textury jen skrze „FillFunction“
    - Nebo přejít k parametrické obálce
- Interakce s ostatnímy objekty

# Aktuální práce II

- **H**igh - albedo model a další triky paprskové optiky
  - Práce se směrem paprsku uvnitř objemu
  - Myšlenkově snadné, boj o rychlost
- **O**ptimalizace a měření

# Postřehy na závěr

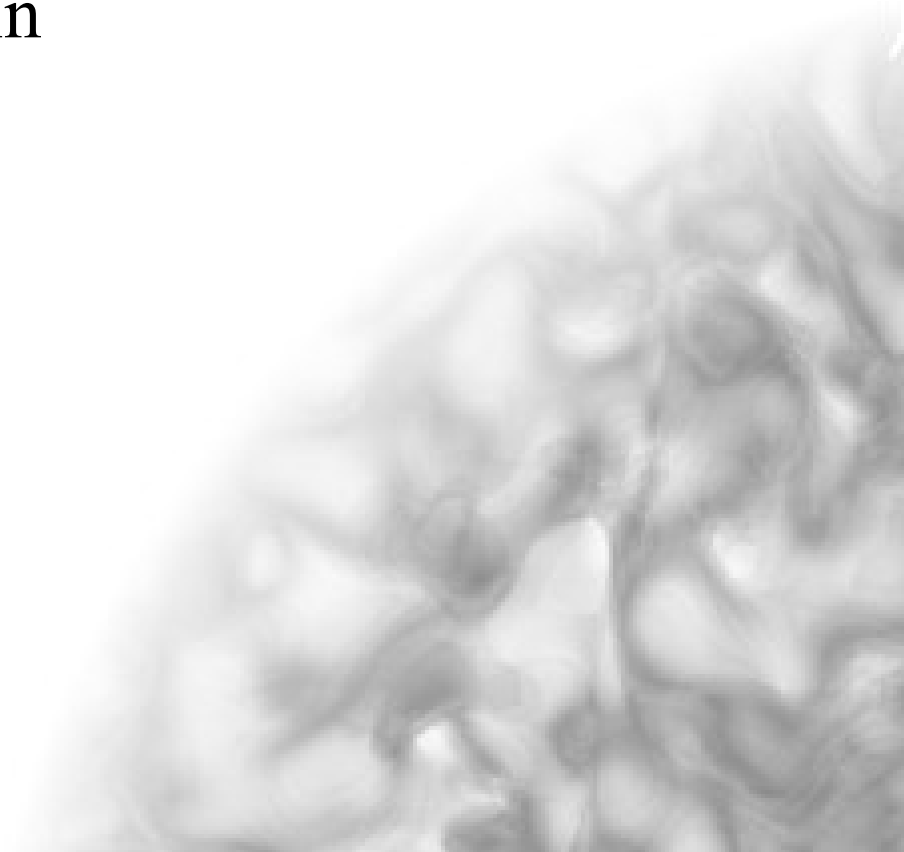
- **H**LSL funkce noise() přístupná jen ve „FillFunctions“

# Zdroje a reference I

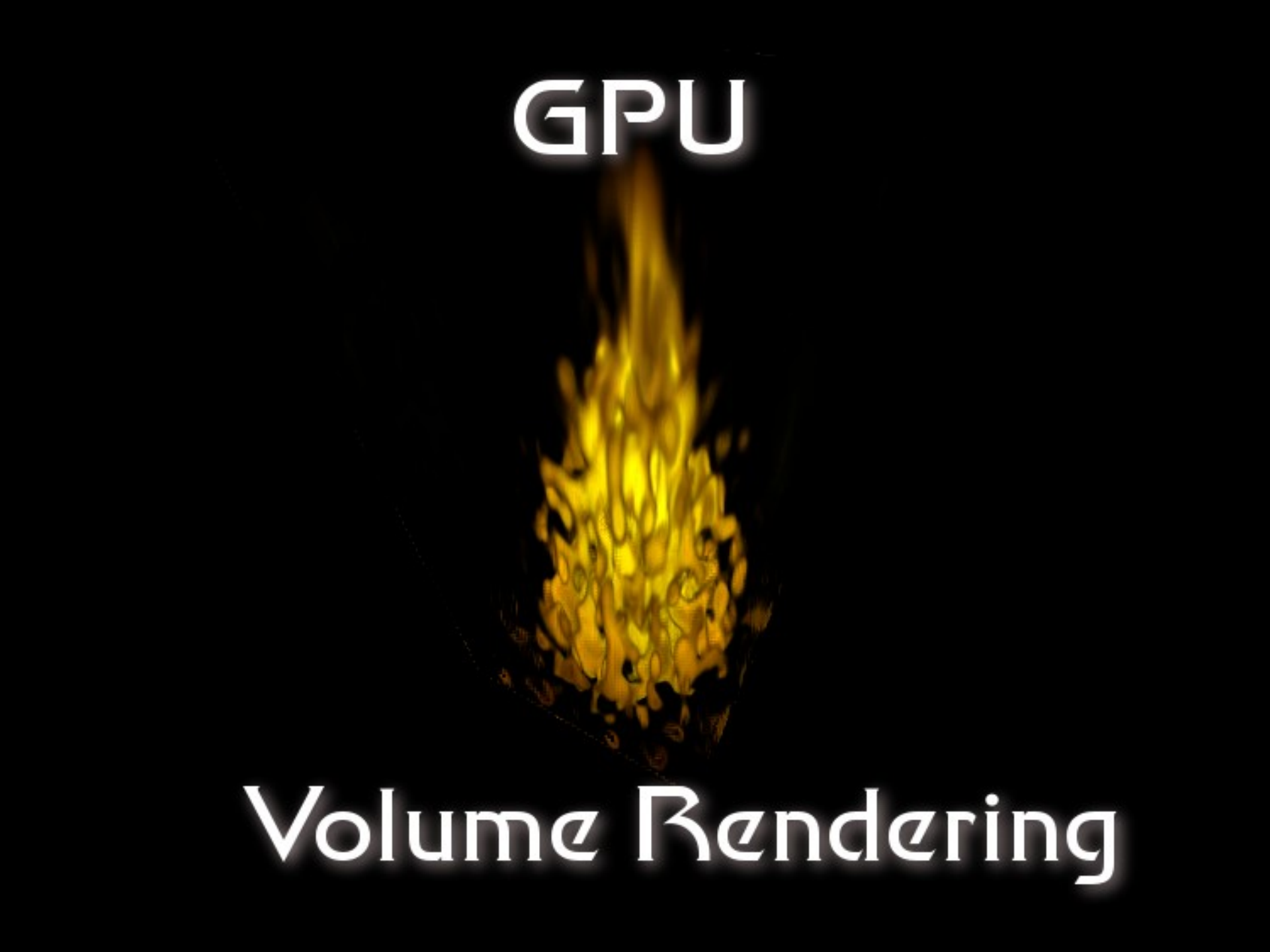
- **d**eveloper.nvidia.com
- **w**ww.noisemachine.com
- **T**exturing & Modeling: A Procedural Approach, Ebert, Musgrave,Perlin,Worley, 3rd Edition
- **T**he Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics, Randima Fernado, Mark J. Kilgard
- **C**ourse notes: Interactive Visualization of Volumetric Data on Consumer PC Hardware, Daniel Weiskop

# Zdroje a reference II

- **G**DC2005 Presentation by Nvidia
- **I**mproving Noise, Ken Perlin



# GPU



# Volume Rendering