Paralelné programovanie Počítačová grafika

Róbert Špir

Výpočtovo náročné úlohy

- CFD – computational fluid dynamics, modely turbulencií, modely šírenia požiarov
- modelovanie globálnych zmien počasia
- supravodivosť, jadrové a termojadrové reakcie
- biológia, spracovanie genómov, tzv. protein folding
- chémia, modelovanie chemických väzieb
- medicína, spracovanie medicínskych dát
- správa veľkých databáz
- obsluha veľkého počtu transakcií
- ...

⇒ snaha zefektívniť **časovo/pamäťovo náročné výpočty**
⇒ využitie paralelných počítačov (súbežné vykonanie elementárnych úkonov)

Architektúra paralelných počítačov

Príklady paralelných architektúr:

- **Multiprocesory** (spoločné adresné pole)

- nesymetrické multiprocesory
- symetrické multiprocesory

- **Klaster** (nemá spoločné adresné pole)
- **Grid**

-niekoľko uzlov (aj bežné PC)

- **GPU** – graphical processing unit

- 100-3000 PU (nVidia, AMD)

Multiprocesorové architektúry

Zloženie:

- niekoľko plnohodnotných procesorov
- zdieľané pamäte (rovnaké adresy u dvoch rôznych CPU ukazujú na rovnaké miesto v pamäti)
- nezdieľané pamäte (nezávislý adresový priestor u rôznych CPU)

Delenie:

- podľa typu pamäte (zdieľaná alebo distribuovaná)
- podľa „zdieľateľnosti“ adresového priestoru

Table 1. Categorization of Parallel Architectures

	Shared Address Space	**Individual Address Space**
Centralized memory	SMP (Symmetric Multiprocessor)	N/A
Distributed memory	NUMA (Non-Uniform Memory Access)	MPP (Massively Parallel Processors)

Klaster – SAV

- Príklad: klaster na SAV (slovenská akadémia vied)

IBM Power 775
- Konfigurácia: 1 stojan, 12 políc, 98 uzlov, 3072 jadier

92 výpočtových uzlov
- 32 jadier na uzol

- Pamäť : 256 GB na výpočtový uzol

- Celková pamäť 24 TB

- Úložný priestor 600 TB

Cena: 13 miliónov eur

Klaster – SAV

- Dostupný pre vedcov na celom Slovensku
  - Vysoký výkon
  - Dostupné veľké objemy pamäte
  - Dlhé čakacie doby
  - Malá kontrola behu a spúšťania programov
  - Obmedzená softvérová výbava
  - Nutnosť platiť za procesorový čas

Klaster – KMaDG

- Klaster na katedre matematiky SvF STU:

AMD Opteron, OS Linux
- Konfigurácia: 6 uzlov, 160 jadier

- 32 jadier na uzol – 4 uzly
- 16 jadier na uzol – 2 uzly

- Pamäť : 8 GB na jadro
- výpočtové uzly s 256 a 128 GB

- Celková pamäť 1.28 TB

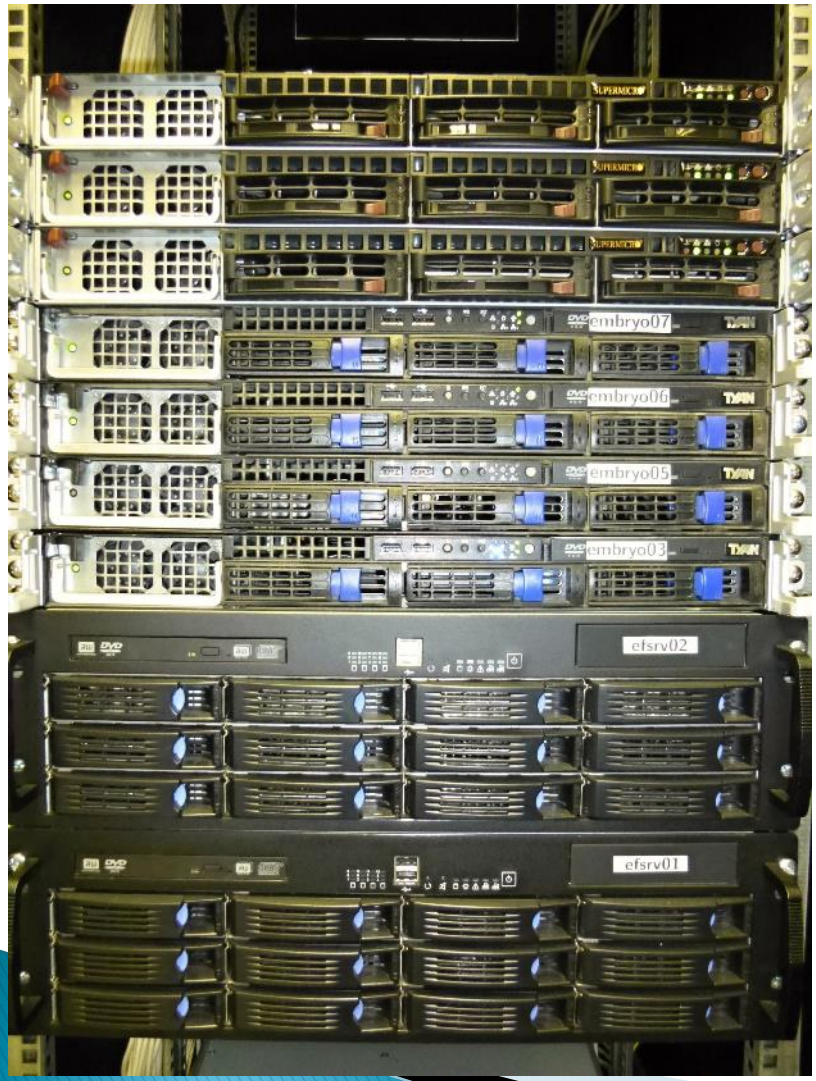
- Úložný priestor 40 TB

Cena: cca 50000 eur

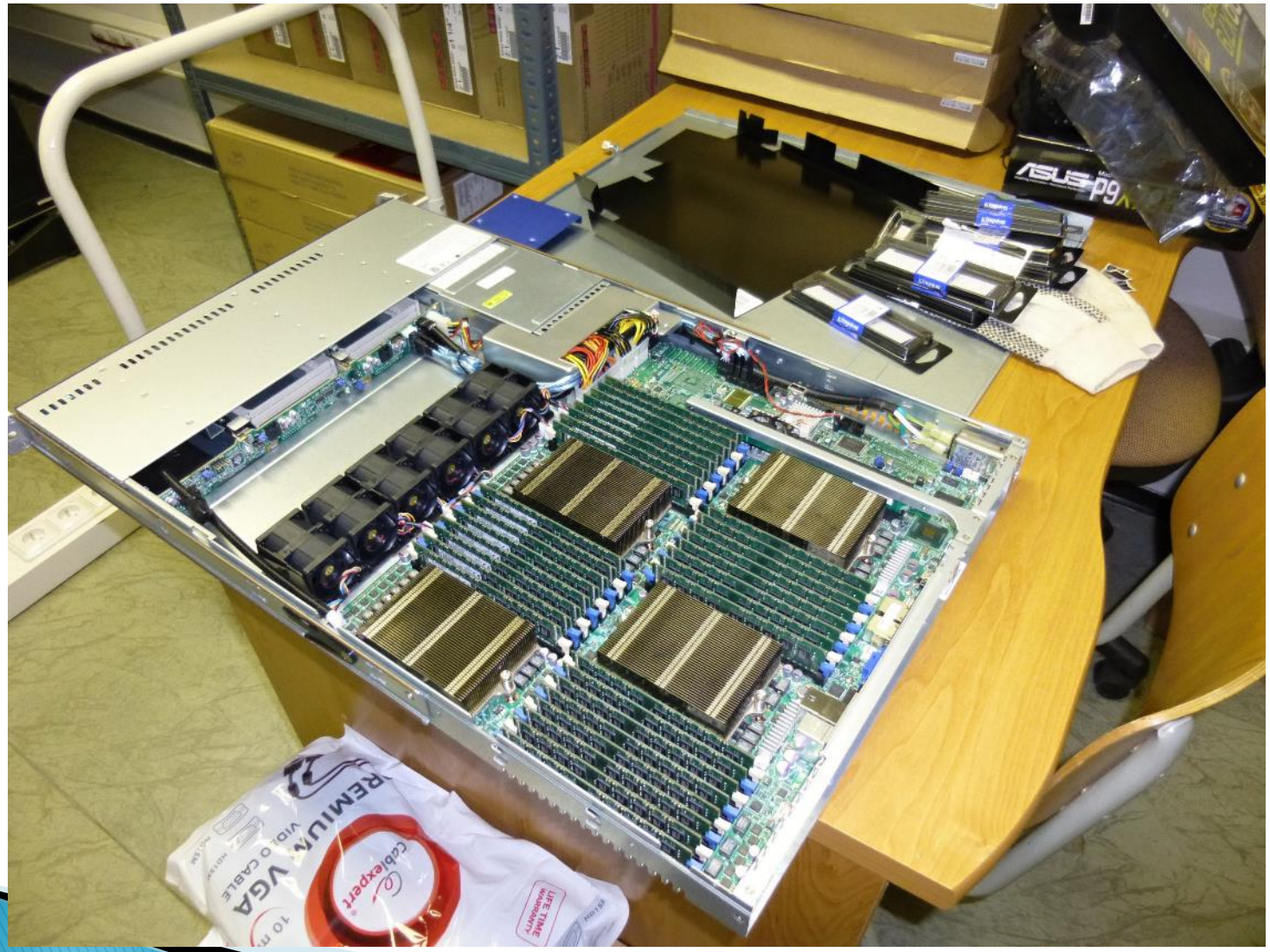
Klaster – KMaDG

- Vyhradený iba pre potreby katedry
  - Možnosť využitia aj študentami MPM pri bakalárskych, diplomových prácach...
  - Žiadne čakacie doby (v rámci dohody kto a kedy bude počítať na katedre ☺)
  - Plná kontrola behu a spúšťania programov
  - Možnosť doinštalovania rôznych softvérov podľa potreby a požiadaviek
- 2x6-jadrový core i7 s 64GB RAM
  - gauss a tesla

Klaster – KMaDG



Klaster – KMaDG



GPU computing

- výpočty na grafických kartách $\Rightarrow$ výrazné zrýchlenie (aj 10-20-krát)
- hrubý výkon jedného výpočtového modulu Nvidia Tesla K40
  - 5364 Gflops
- obmedzenie na pamäť (12 GB)

- hrubý výkon jedného procesora Intel Core i7-4770K
  - 474 Gflops

CUDA (Compute Unified Device Architecture)
- a parallel computing architecture developed by NVIDIA

OpenCL (Open Computing Language)
- otvorený multiplatformový štandard

DirectCompute
- API vytvorené spoločnosťou Microsoft

Modely paralelného programovania

Cieľ paralelného programovania:

- využiť dostupné procesory a znížiť výpočtový čas
- distribuovať výpočty na rozsiahlych dátach na viacero pamäťových uzlov

Rôzne druhy architektúr $\Rightarrow$ rôzne modely paralelného programovania:

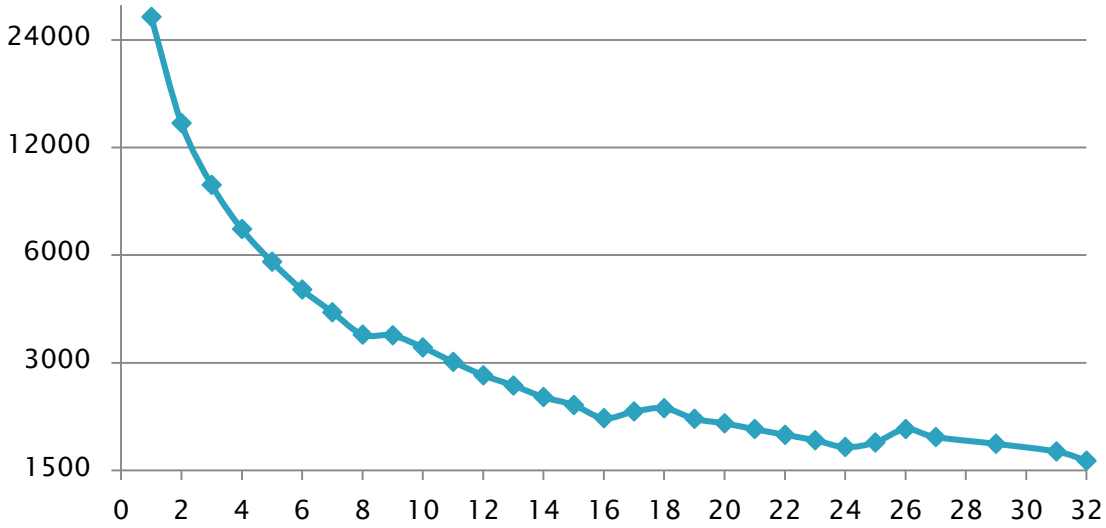
a) SMP Based
b) MPP Based on Uniprocessor Nodes (Simple MPP)
c) MPP Based on SMP Nodes (Hybrid MPP)
   - Multiple Single-Thread Processes pre Node
   - One Multi-Thread Process per Node

Modely paralelného programovania

- „multi-threaded“ programy sú najvhodnejšie pre SMP architektúru
  - Jednotlivé vlákna zdieľajú pamäť, nie je potrebná špeciálna komunikácia medzi nimi
    - OpenMP
- Ak adresový priestor nie je zdieľaný medzi uzlami
  - potreba prenosu dát cez prepojovaciu sieť
  - message-passing
    - MPI – message-passing interface
    - Rozhranie pre komunikáciu medzi procesmi a uzlami

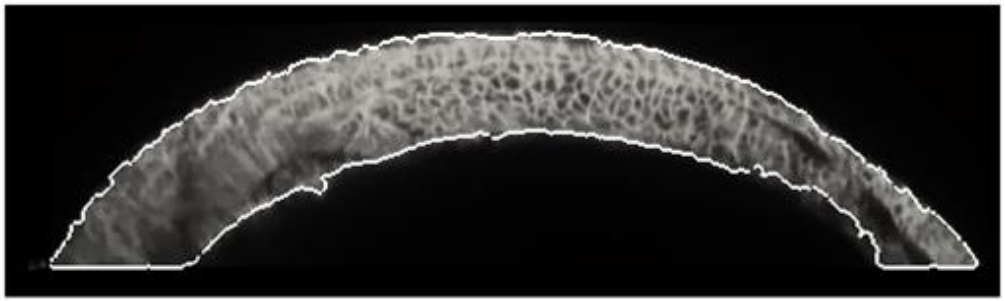
Paralelné výpočty

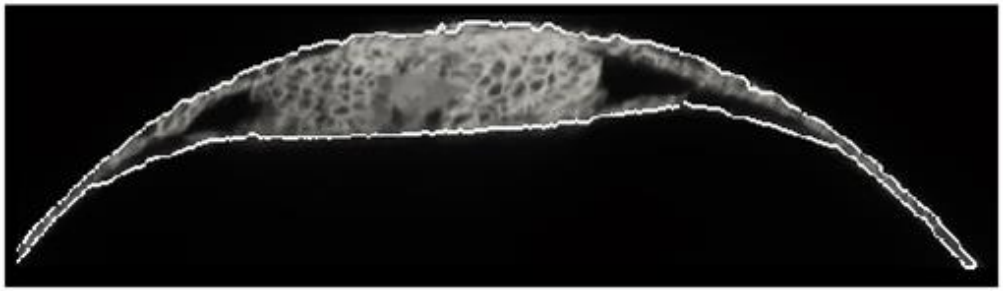
- Porovnanie zrýchlenia paralelného výpočtu
  - Paralelná segmentácia medicínskych dát

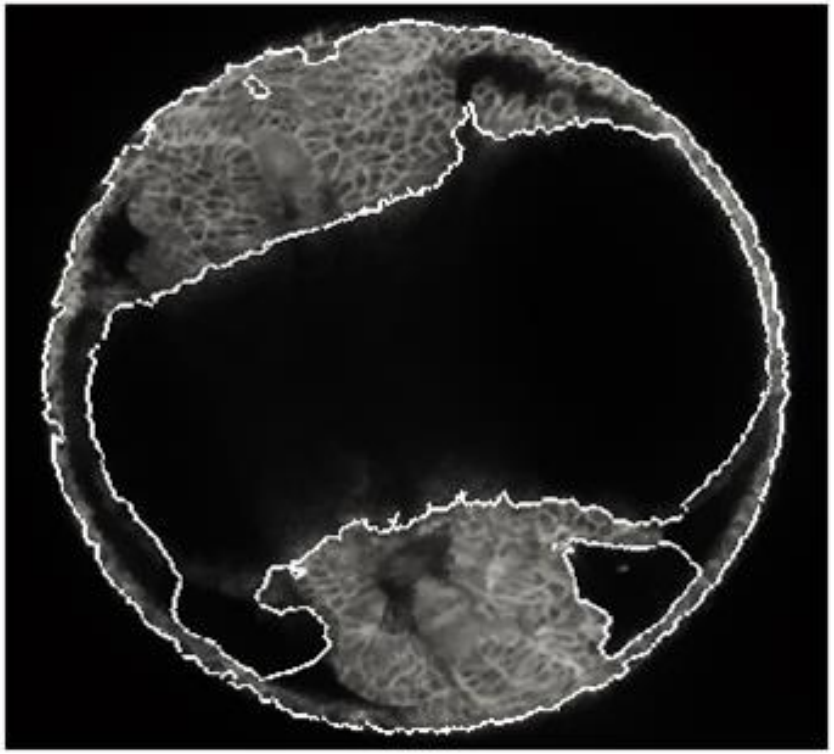


Počet procesov	1	2	4	8	16	32
Čas výpočtu [s]	27802	14036	7099	3596	2098	1597
Faktor zrýchlenia	-	1,98	3,91	7,73	13,25	17,41

Paralelné výpočty







Paralelné výpočty

- Porovnanie zrýchlenia paralelného výpočtu
  - Paralelný výpočet funkcie vzdialenosti v 4D priestore

Počet procesov	1	2	4	8	16	32
Čas výpočtu [s]	752	324	143	73	43	32
Faktor zrýchlenia	-	2,32	5,26	10,30	17,49	23,50

Paralelné výpočty

- Ďalšie paralelné výpočty na katedre:
  - Geodetické aplikácie
    - Výpočet geodetických veličín na zemskom povrchu
    - Modelovanie geoidu
    - Filtrácia geodetických dát
  - Medicínske aplikácie
    - Filtrácia a segmentácia medicínskych dát
    - Automatická detekcia bunkových centier v mikroskopických snímkach organizmov
    - Sledovanie pohybu a delenia buniek pri vývoji organizmov
    - Výpočet vzdialenosti medzi jednotlivými bunkovými centrami
  - Ostatné aplikácie
    - Modelovanie šírenia požiarov
    - Spracovanie obrazu
    - Vývoj geometrických plôch

Počítačová grafika

Počítačová grafika

- Projekcia 3D scény na 2D obrazovku
  - V reálnom čase
  - Pred-renderovanie
- Techniky
  - Ray-casting
    - Sledovanie svetelných lúčov z miesta pohľadu
  - Ray-tracing
    - Sledovanie svetelných lúčov zo zdroja svetla
  - Rasterizácia
    - Rýchly spôsob konverzie vektorovej scény na rasterový obrázok (zložený z pixelov)

Počítačová grafika

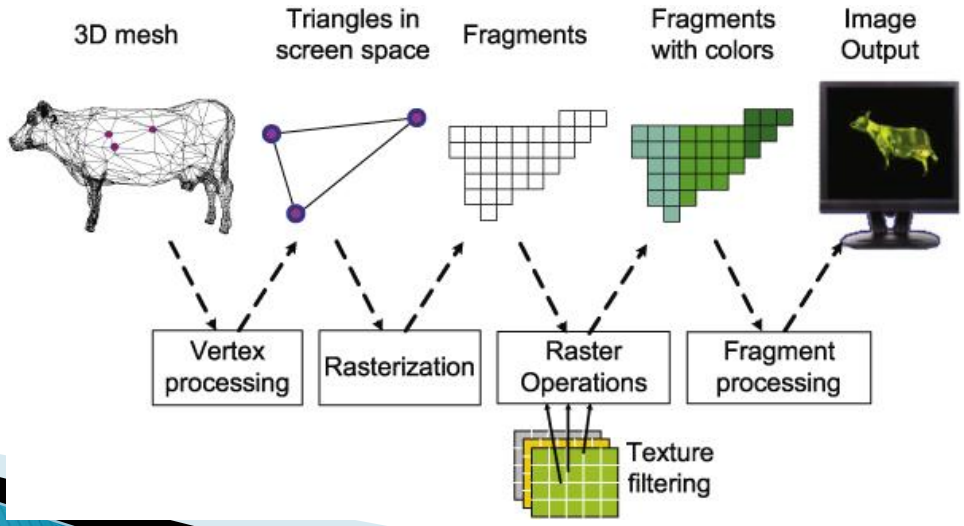
- Rendering:
  - Softvérový
    - Vykresľovanie vykonáva CPU počítača
    - Pomalé
    - Neobmedzené možnosti
  - Hardvérový
    - Vykresľovanie vykonáva špecializovaný čip, GPU (graphics processing unit)
    - Veľmi rýchle
    - Obmedzené možnosťami hardvéru

Softvérový rendering

- Možnosť využitia ľubovoľných matematických algoritmov pre vytvorenie a osvetlenie scény
  - Ray-casting
  - Ray-tracing
- Najčastejšie sa používa na pred-renderovanie scény
  - Vytvorenie jedného obrázka trvá niekoľko sekúnd až minút

Hardvérový rendering

- Obmedzené podporovanými technológiami GPU
  - Rasterizácia
  - Osvetlenie a efekty pomocou zjednodušených, rýchlych algoritmov
  - Primárne pre hry a pre vizualizačné softvéry zobrazujúce scénu v reálnom čase
  - 10-100 snímkov za sekundu



Hardvérový rendering

- 2 štandardné API (aplikačno-programovacie rozhranie):
  - DirectX
    - Kolekcia API pre prácu s multimédiami
    - Microsoft platforma
    - DirectX 11.2
  - OpenGL
    - API pre 2D a 3D rendering vektorovej grafiky
    - Zamerané na profesionálne využitie
    - Multiplatformové
    - OpenGL 4.5

Tvorba softvéru

- Motivácia:
  - Neexistuje softvér pre naše potreby
  - Existujúci softvér nepodporuje dáta v našom formáte
  - Softvér je drahý a my potrebujeme len niektoré funkcie
  - Softvér zvláda to, čo potrebujeme, ale jeho použitie je ťažkopádne
  - Softvér má priveľa funkcií a pre naše potreby je pomalý
    - Špecializovaná aplikácia s iba potrebnými funkciami môže byť výrazne rýchlejšia

Tvorba softvéru

- Výber jazyka:
- C#
  - Moderný vysokoúrovňový jazyk
  - .net Framework obsahuje veľké množstvo pomocných knižníc
  - Automatická správa pamäte
  - Prepracovaný systém výnimiek

  - Programátor sa môže sústrediť na dosiahnutie svojho cieľa a nemusí riešiť problémy, ktoré s tým priamo nesúvisia ako v C/C++

Tvorba softvéru

- Úloha programátora:
  - Definícia 3D scény
    - Uhol pohľadu
    - Osvetlenie
  - Vytvorenie geometrického 3D modelu
    - Pomocou súradníc vrcholov v 3D priestore a definovaním ich logického pospájania do trojuholníkov
    - Definovaním farby...
  - Transformácia modelu
    - Rotácia, translácia, škálovanie
  - Odoslanie dát GPU, ktorá vykoná rasterizáciu

Tvorba softvéru

- Vytvorenie 3D scény:
  - Nastavenie pohľadu

```
            Matrix viewMatrix = Matrix.LookAtRH(new Vector3(0, 0, -3.5f), new Vector3(0, -0.1f, 0), new
Vector3(0, 1, 0));
            Matrix projectionMatrix = Matrix.PerspectiveFovRH((float)Math.PI / 4f,
(float)device.Viewport.Width / device.Viewport.Height, 0.01f, 300.0f);
            device.SetTransform(TransformState.View, viewMatrix);
            device.SetTransform(TransformState.Projection, projectionMatrix);
```

Tvorba softvéru

- Osvetlenie 3D scény:

```
Light svetlo = new Light
{
    Type = LightType.Directional,
    Specular = Color.White,
    Direction = new Vector3(0.0f, 0.0f, 10.0f),
    Range = 10f,
    Falloff = 1f,
};
Light svetlo2 = new Light
{
    Type = LightType.Point,
    Specular = SharpDX.Color.White,
    Diffuse = SharpDX.Color.White,
    Ambient = SharpDX.Color.White,
    Position = new Vector3(posunX, posunY, -3.0f),
    Range = 15f,
};
device.SetLight(1, ref svetlo2);
device.SetLight(0, ref svetlo2);
device.SetRenderState(RenderState.Ambient, new SharpDX.Color(0.2f,0.2f,0.2f).ToRgba());
device.SetRenderState(RenderState.AmbientMaterialSource, ColorSource.Color1);
device.SetRenderState(RenderState.Lighting, true);
```

Tvorba softvéru

- Vykreslenie scény
  - Zobrazenie nášho 3D modelu

```
Matrix worldMatrix = Rotation * Matrix.Translation(posunX, posunY, 0f) * Matrix.Scaling(zoom, zoom, zoom);
device.SetTransform(TransformState.World, worldMatrix);
device.Clear(ClearFlags.Target | ClearFlags.ZBuffer, pozadie, 1.0f, 0);

device.BeginScene();

device.SetStreamSource(0, vb, 0, Marshal.SizeOf(typeof(Vertex)));
device.Indices = ib;
device.VertexDeclaration = vertexDecl;
device.DrawIndexedPrimitive(PrimitiveType.TriangleList, 0, 0, pocet_voxel/3, 0, pocet_voxel/3);

device.EndScene();
```

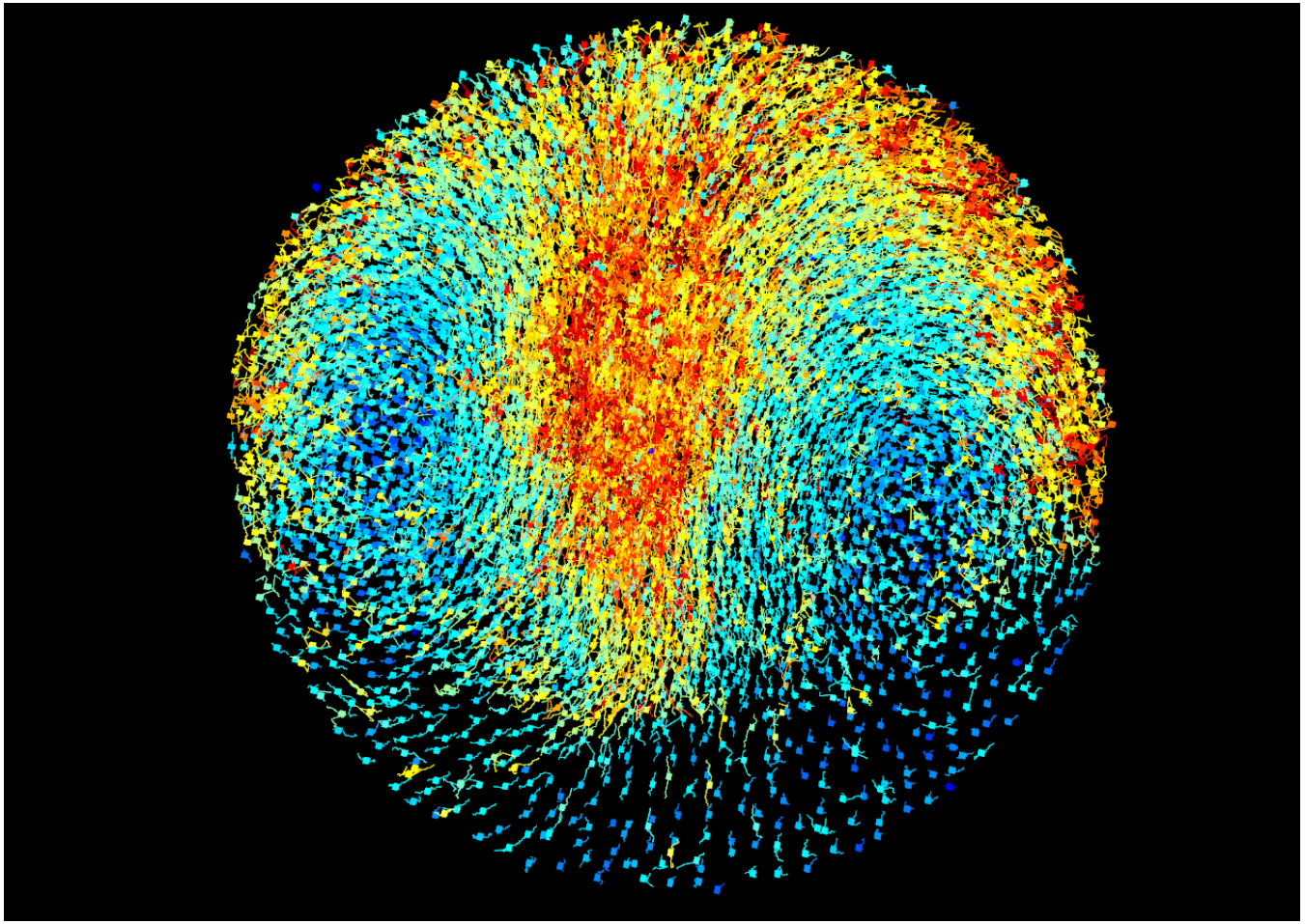
Vykreslenie scény

- 3D model
  - Každý 3D model sa skladá z trojuholníkov
  - Je definovaný pomocou série vrcholov
    - Môže byť definované ich vzájomné pospájanie do trojuholníkov
- Vertex buffer
  - Pamäťová štruktúra obsahujúca informácie o vrcholoch (vertexoch)
- Index buffer
  - Pamäťová štruktúra obsahujúca informácie, v akom poradí majú byť vrcholy poprepájané do trojuholníkov

Softvér

- Sledovanie pohybu buniek počas vývoja organizmov
  - Vstupné dáta: strom bunkového vývoja v čase
  - Výstup: animácia pohybu buniek aj s ich trajektóriami v čase
    - Možnosť zvýraznenia bunkového delenia a iných javov
  - Bakalárska práca, Natália Heliová

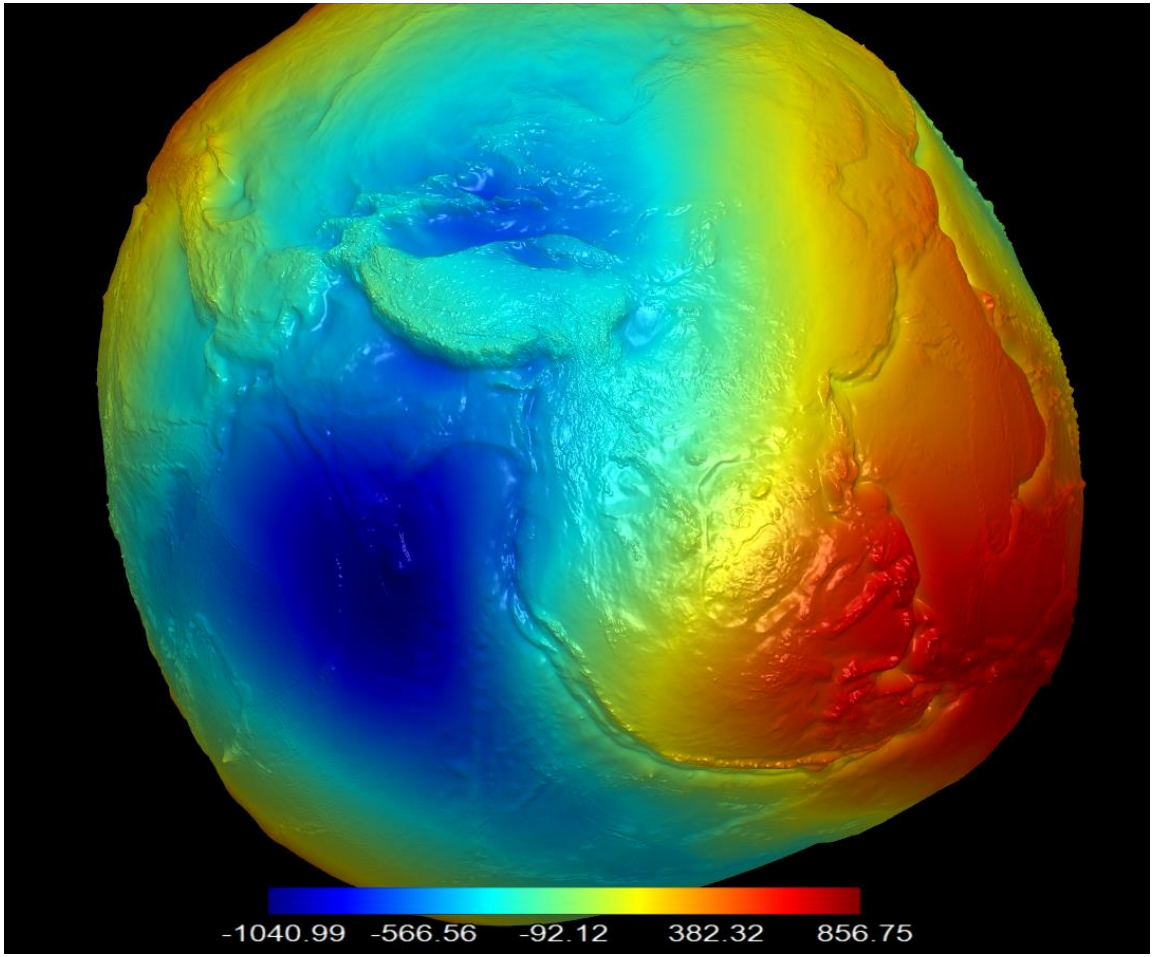
Sledovanie pohybu buniek počas vývoja organizmov



Softvér

- Zobrazenie geodetických dát na guľovej ploche
  - Vstupné dáta: geodetické súradnice a hodnoty veličiny (poruchový potenciál) v nich
  - Výsledný model: 3D reprezentácia veličiny na guľovej ploche, kde hodnota veličiny určuje „výšku“ nad guľou a tiež farbu
  - Súčasť mojej diplomovej práce

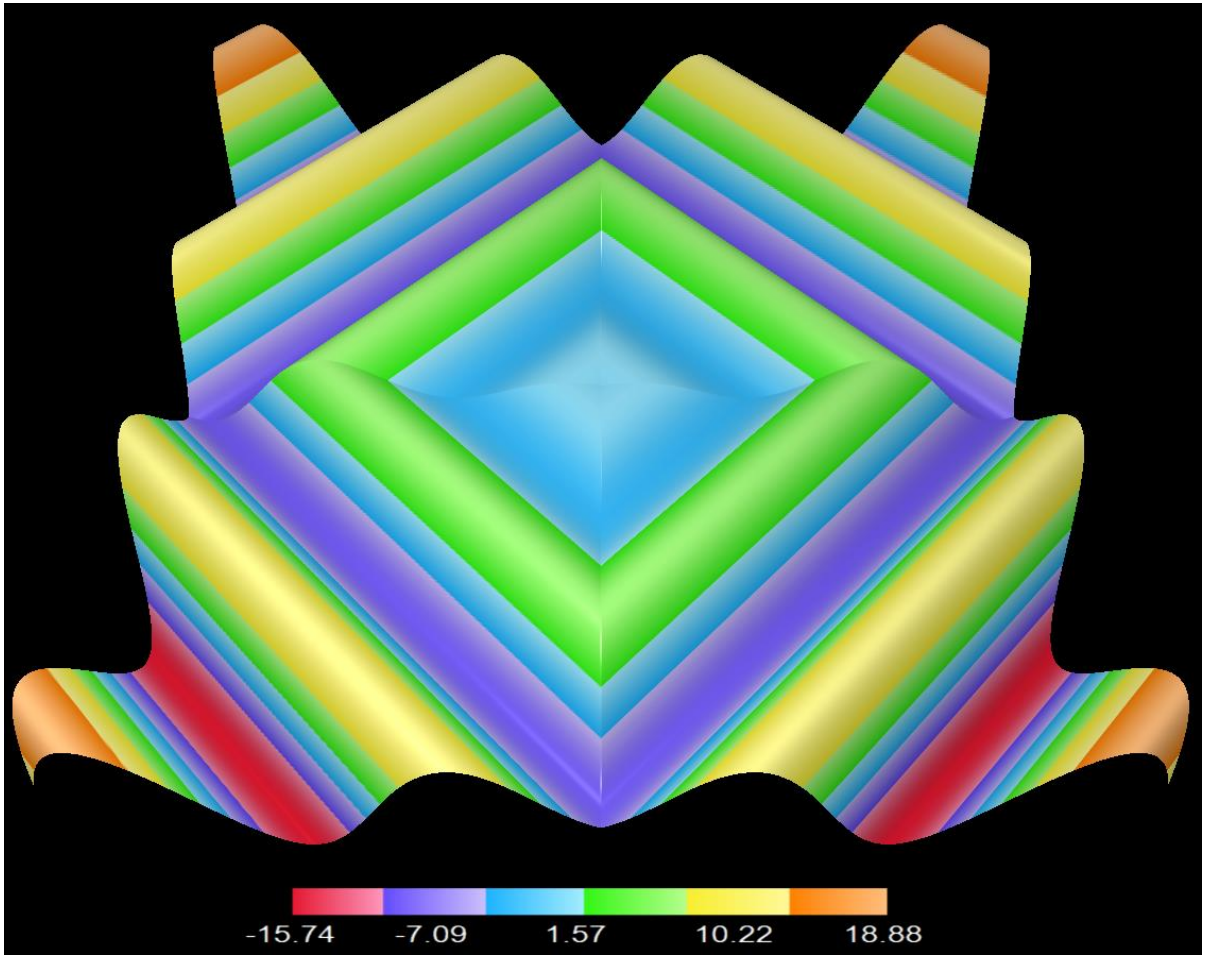
Zobrazenie geodetických dát na guľovej ploche



Softvér

- Zobrazenie dát na obdĺžnikovej ploche
  - Vstupné dáta: súradnice a hodnoty veličiny alebo funkcie na obdĺžnikovej ploche
  - Výsledný model: Zobrazenie dát na výškovej mape s možnosťou definovania vlastného zafarbenia
  - Bakalárska práca, Balázs Kósa

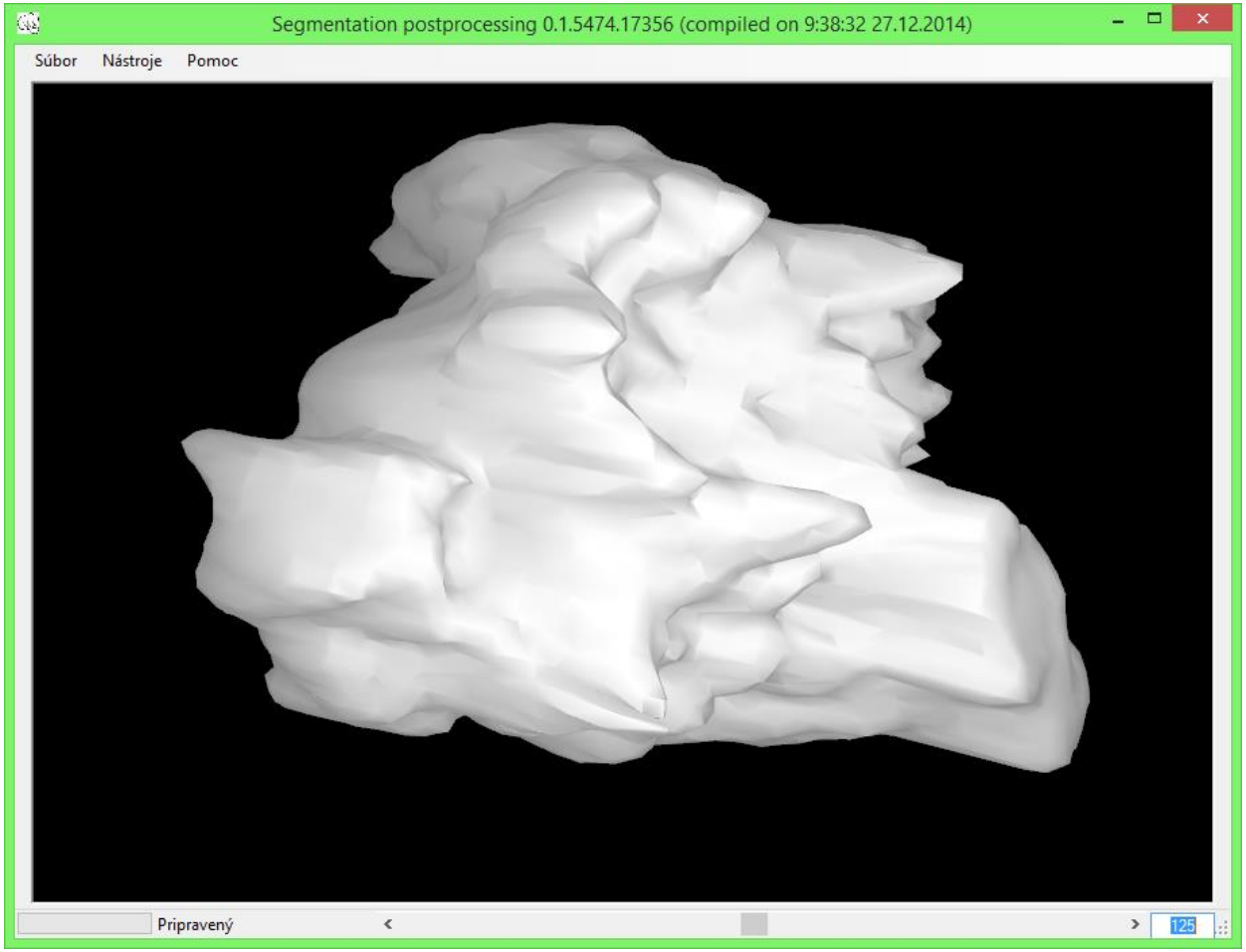
Zobrazenie dát na obdĺžnikovej ploche



Softvér

- Spracovanie výsledkov získaných automatickými segmentačnými algoritmami
  - Vstupné dáta: segmentácia vo formáte funkcie úrovňovej množiny (level-set function)
  - Výstup: zobrazenie segmentácie výberom izoplochy funkcie, manipulácia tejto segmentácie
  - Diplomová práca, Peter Jasaň

Spracovanie výsledkov získaných automatickými segmentačnými algoritmami



Ďakujem za pozornosť