Black hole orbiter (Orb)

Samuel Amrich

Dominik Pavlov

Lukáš Randuška

Filip Stempel

Any of the concentric spheres in old astronomy surrounding the earth and carrying the celestial bodies in their revolutions

WHY?

- Dostali sme to ako zadanie
- Kvalitných simulátorov

  GC je žalostne málo
- Je to výzva!




Introduction

Aneb jak tím A* k Orbu došiel

Tím

Team

Dominik Pavlov

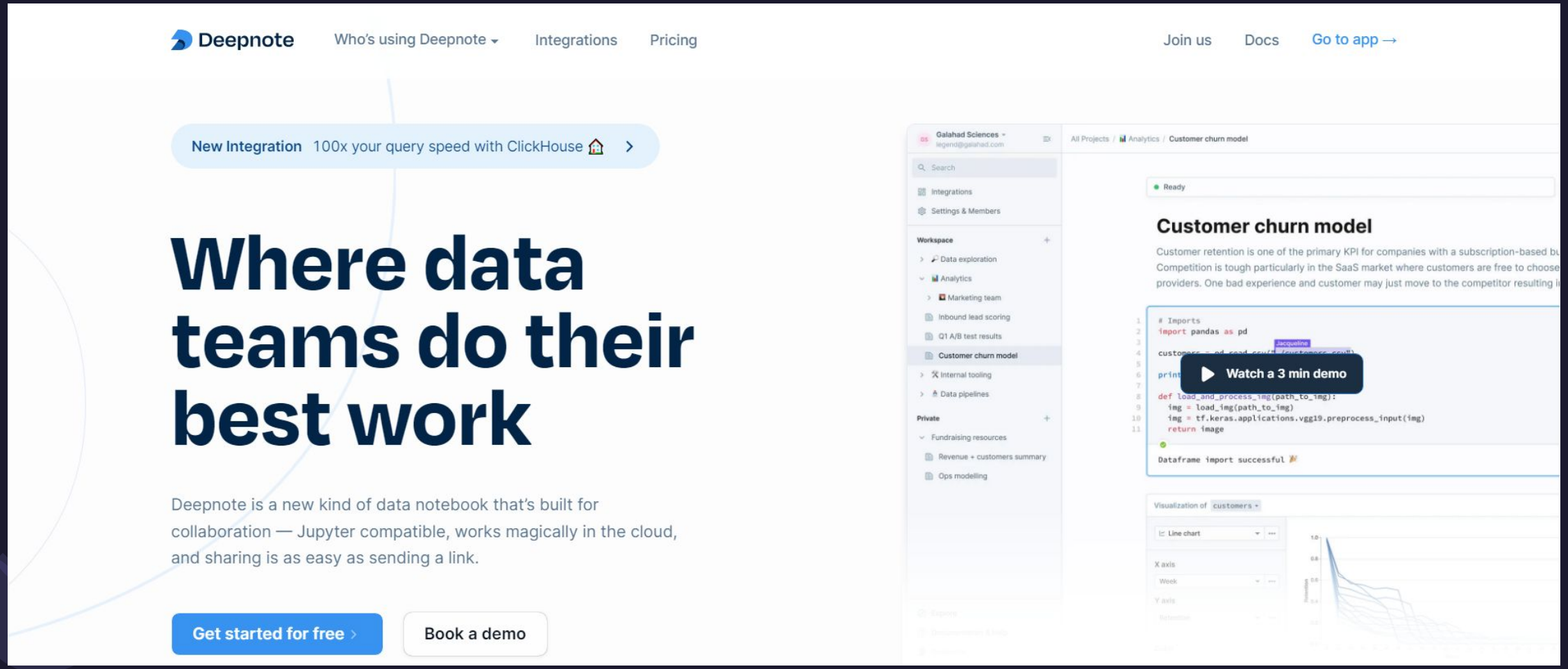
Samuel Amrich

Filip Stempel

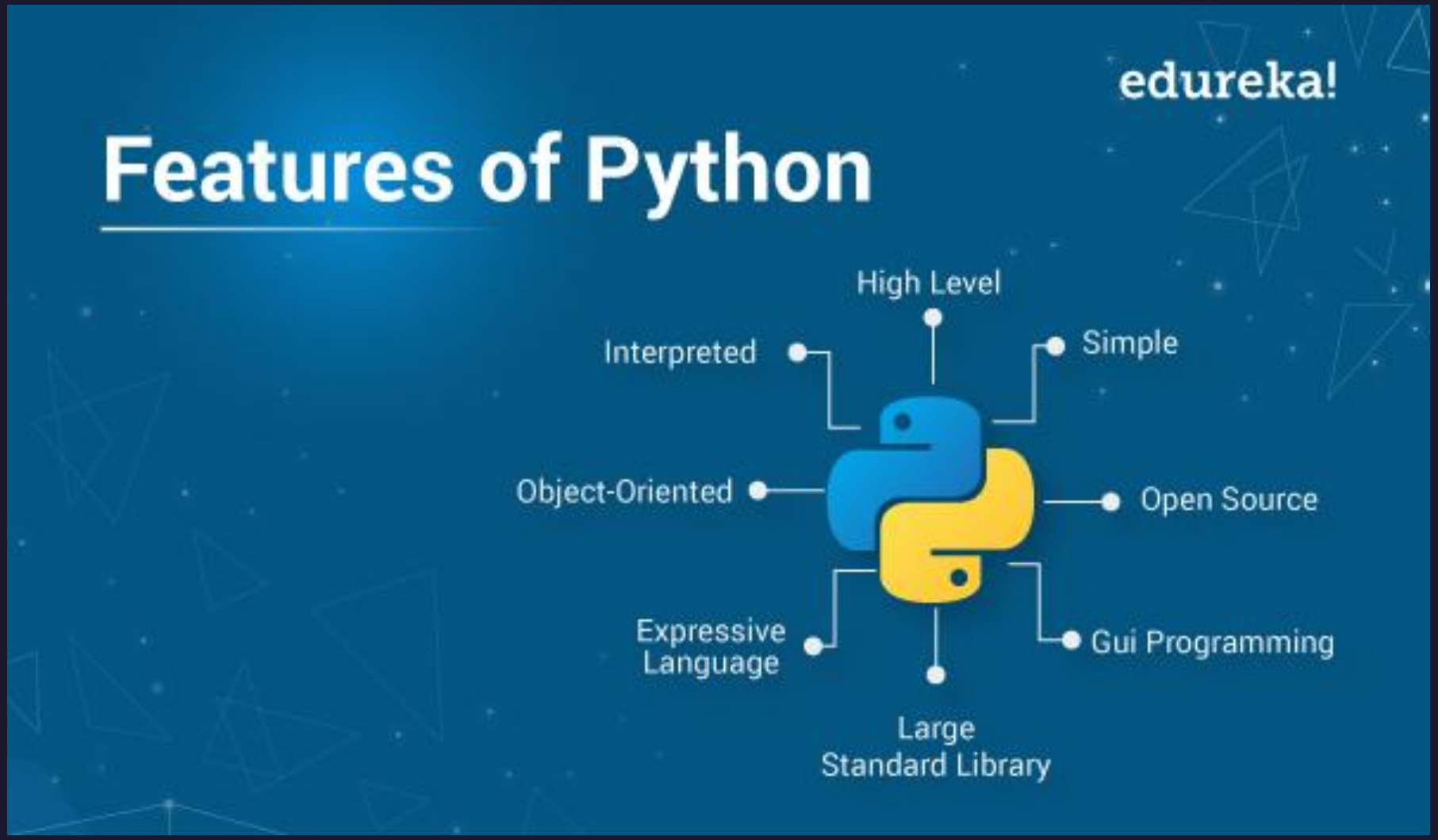
Lukáš Randuška

Prostredie

Deepnote


Deepnote
Who's using Deepnote
Integrations
Pricing
Join us
Docs
Go to app →
New Integration 100x your query speed with ClickHouse
Where data teams do their best work
Deepnote is a new kind of data notebook that's built for collaboration — Jupyter compatible, works magically in the cloud, and sharing is as easy as sending a link.
Get started for free
Book a demo
Galahad Sciences
All Projects / Analytics / Customer churn model
Search
Integrations
Settings & Members
Workspace
Data exploration
Analytics
Marketing team
Inbound lead scoring
Q1 A/B test results
Customer churn model
Internal tooling
Data pipelines
Private
Fundraising resources
Revenue + customers summary
Ops modelling
Ready
Customer churn model
Customer retention is one of the primary KPI for companies with a subscription-based bu
Competition is tough particularly in the SaaS market where customers are free to choose
providers. One bad experience and customer may just move to the competitor resulting i
Jacqueline
Watch a 3 min demo
# Imports
import pandas as pd
def load_and_process_img(path_to_img):
img = load_img(path_to_img)
img = tf.keras.applications.vgg19.preprocess_input(img)
return image
Dataframe import successful
Visualization of customers
Line chart
X axis
Week
Y axis

Python



Nie je to C++

Problém

Globular cluster (GC)

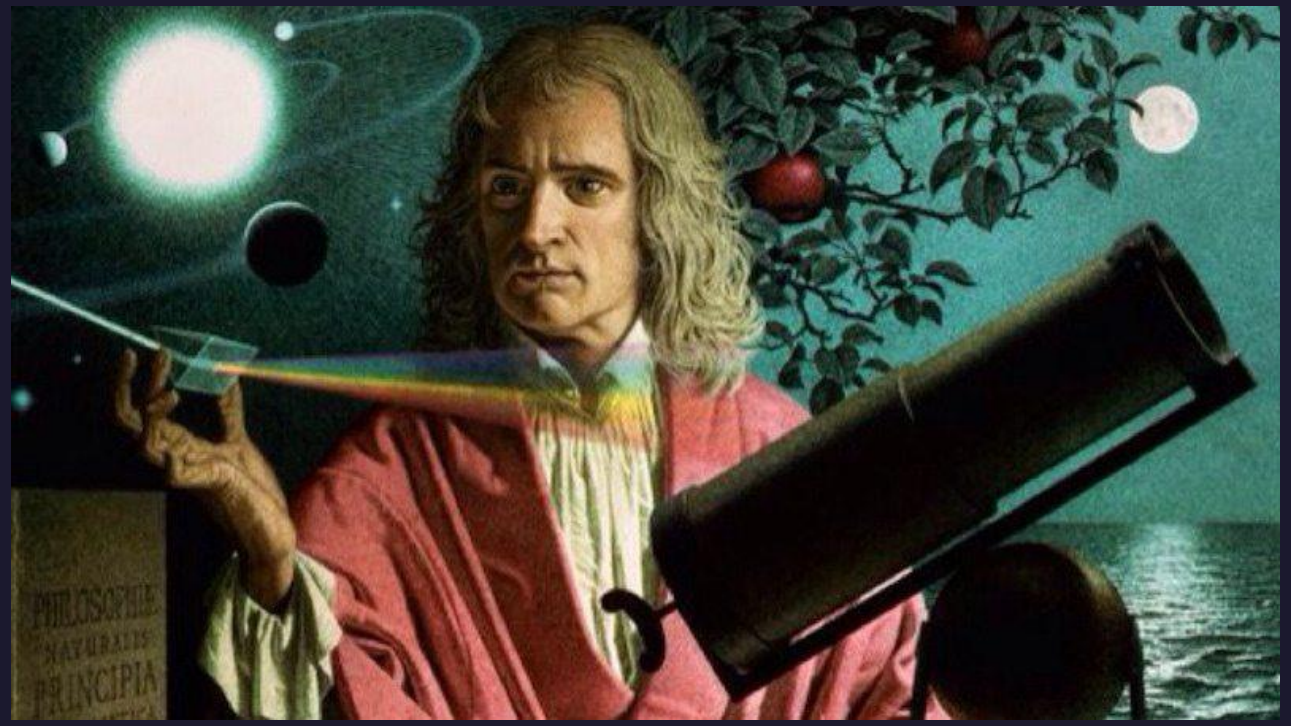
Newtonovská gravitácia

$$\vec{F} = -G\frac{M_1 \cdot M_2}{r^2}\overrightarrow{e_r}$$

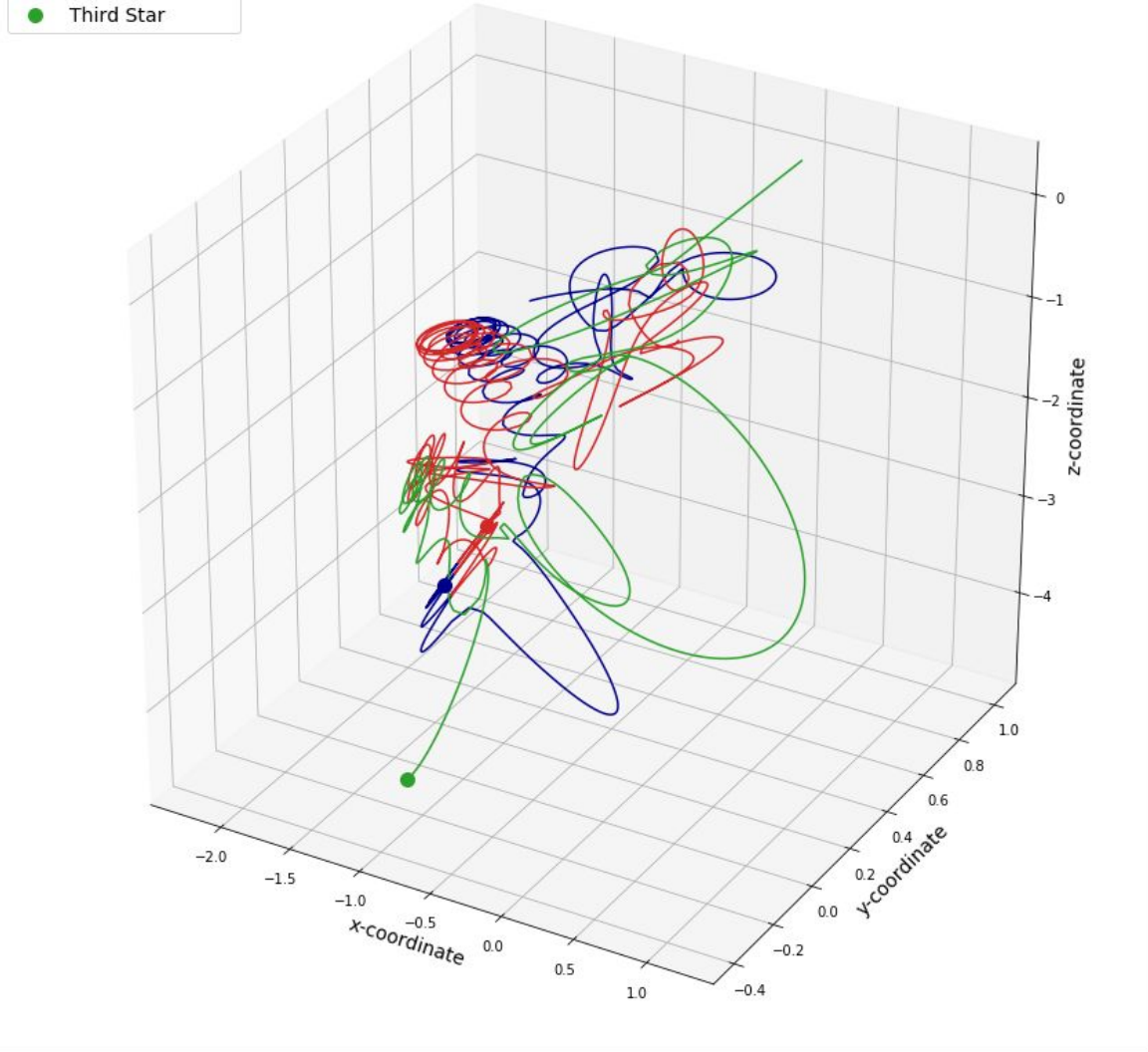
Okamžitý prenos informácie

Univerzálna sila

Jediná uvažovaná sila



N-body problem



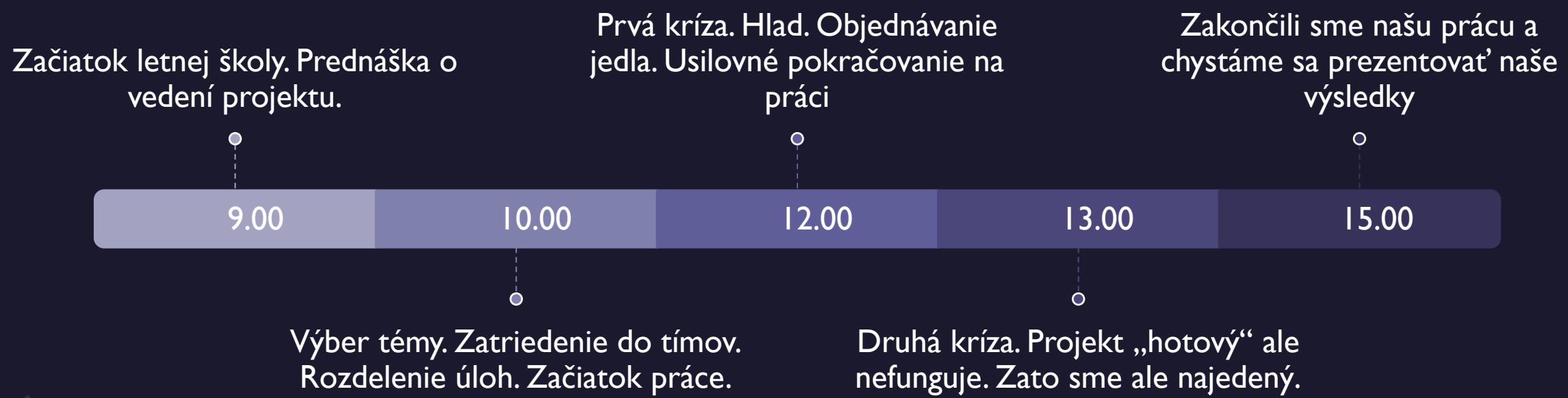
Analyticky neriešiteľné

Výpočetne náročné

Vedecky podstatné

Riešenie

Timeline


Začiatok letnej školy. Prednáška o vedení projektu.
Prvá kríza. Hlad. Objednávanie jedla. Usilovné pokračovanie na práci
Zakončili sme našu prácu a chystáme sa prezentovať naše výsledky
9.00
10.00
12.00
13.00
15.00
Výber témy. Zatriedenie do tímov. Rozdelenie úloh. Začiatok práce.
Druhá kríza. Projekt „hotový“ ale nefunguje. Zato sme ale najedený.

Počiatočné podmienky

- Hmotnosť $M = 2 \cdot 10^5\ M_\odot$
- Polomer $R = 50\ ly$
- Počet hviezd $N = 330\ 000$
- Zgranulovanie $gran = 1000$
- Doba simulácie $T = 1000\ 000$
- Časový krok $dt = 1000$

Rozdelenie hmotnosti a polôh

HUSTOTNÉ ROZDELENIE

$$\rho(r) = \frac{3M}{4\pi a^3 \left(1+\frac{r^2}{a^2}\right)^{\frac{5}{2}}}$$

PRIESTOROVÉ ROZDELENIE

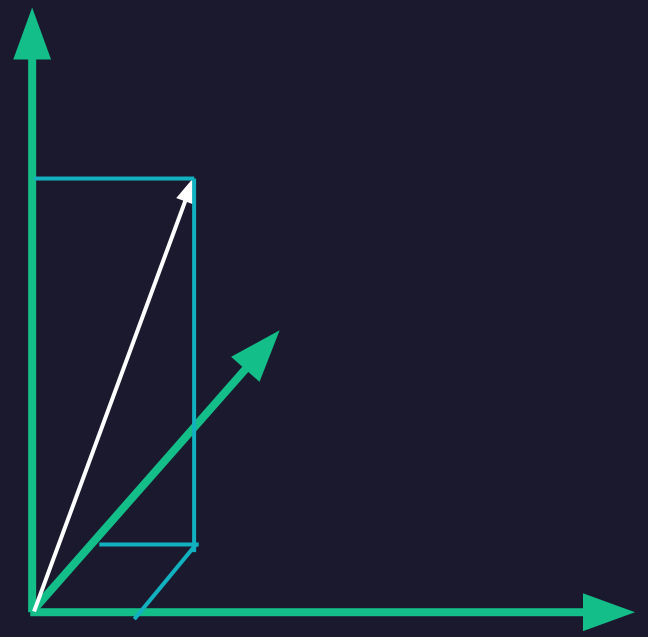
- Prechod od spojitého rozdelenia hustoty k diskrétnym hmotnostiam

Náhodné rozdelenie rýchlosti

ROZDELENIE VEĽKOSTI RÝCHLOSTI

- $v(r) = \left(1 - \frac{r}{R+0.5}\right) \cdot 15km/s$

NÁHODNÝ PRIEMET



Kód

```
import random as rnd
import numpy as np
import tensorflow as tf
```

Format dat

[[[x1, y1, z1] , [x2, y2, z2] , [x3, y3, z3] , ..., [x1000, y1000, z1000]], v case 0

[[x1, y1, z1] , [x2, y2, z2] , [x3, y3, z3] , ..., [x1000, y1000, z1000]], v case 1

...,

[[x1, y1, z1] , [x2, y2, z2] , [x3, y3, z3] , ..., [x1000, y1000, z1000]]] v case 100000

```
# Vygeneruje nahodne projekcie pre jednu zadanu rychlost
def vel_projection(vel):
    velx = rnd.randrange(np.floor(vel))
    vely = rnd.randrange(np.floor(np.sqrt(vel*vel - velx*velx)))
    velz = np.sqrt(vel*vel - velx*velx - vely*vely)
    return np.array([velx, vely, velz])
```

```
# Vygeneruje nahodne projekcie pre numpy array rychlosti
def vel_projections(vel):
    velx = np.random.random()*np.floor(vel)
    vely = np.random.random()*np.floor(np.sqrt(vel*vel - velx*velx))
    velz = np.sqrt(vel*vel - velx*velx - vely*vely)
    return velx, vely, velz
```

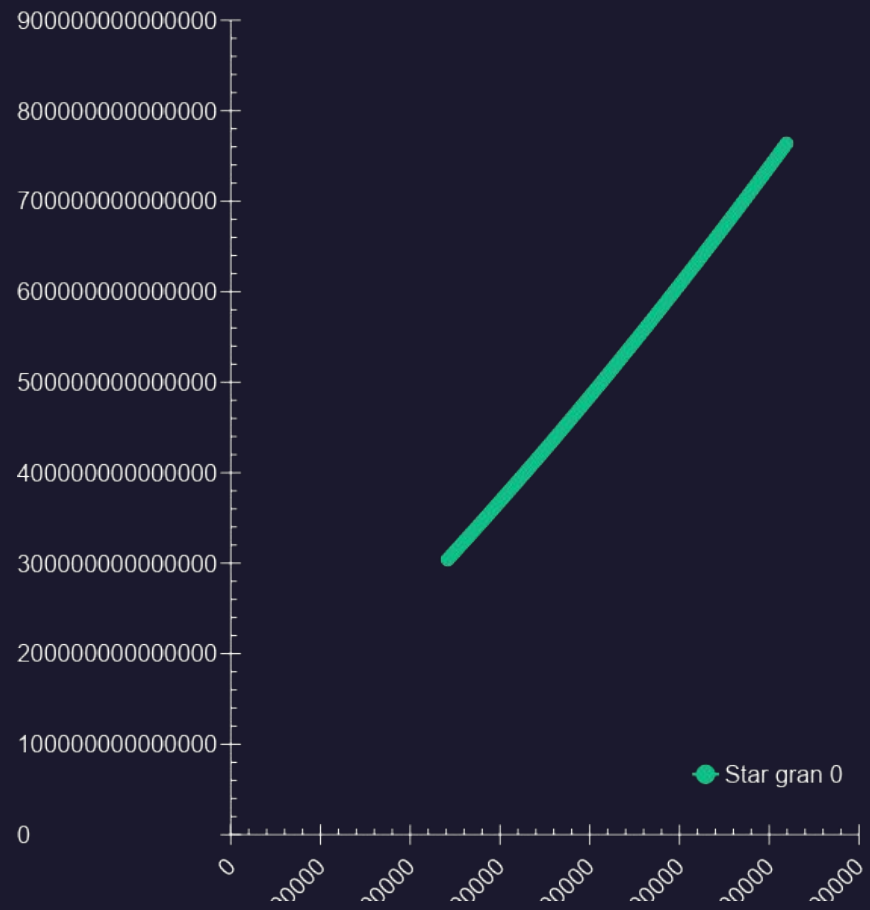
```
# Pociatocne podmienky M92

M = 4*(10**35) # Hmotnost gulovej hviezdokopy v kg
R = 5*(10**16) # Polomer hviezdokopy v m
N = 330000 # Pocet hviezd v hviezdokope
gran = 1000 # Kolko hviezd reprezentuje jeden bod (granula)
n = N//gran # Prvy odhad Pocet bodov do simulacie
m = M/gran # Hmotnost jednej granule v kg
G = 6.6743*10**(-11) # Gravitacna konstanta
a = 0.6*R # Plumerova sfera
dt = 1000*365*24*60*60 # Casovy krok
T = 1000000*365*24*60*60 # Doba simulacie
Q = T//dt # Pocet krokov simulacie
```

```
# Generovanie pociatocnich dat

pos, n = gen_pos(M=M, R=R, n=n, m=m, a=a)
vel = gen_vel(R=R, n=n, pos=pos)
vel = vel_vari(vel, delta=0.05)
```

```
# get acceleration based on gravitational pull and update sub-clusters (granule) postions

def tf_delete(tensor,index,row=True):

    if row:
        sub = list(range(tensor.shape[0]))
    else:
        sub = list(range(tensor.shape[1]))
    sub.pop(index)

    if row:
        return  tf.gather(tensor,sub)
    return tf.transpose(tf.gather(tf.transpose(tensor),sub))

def acc_by_G_pull(pos, m, G, dt, vel):
```

```
# get acceleration based on gravitational pull and update sub-clusters (granule) postions

def tf_delete(tensor,index,row=True):

    if row:
        sub = list(range(tensor.shape[0]))
    else:
        sub = list(range(tensor.shape[1]))
    sub.pop(index)

    if row:
        return  tf.gather(tensor,sub)
    return tf.transpose(tf.gather(tf.transpose(tensor),sub))

def acc_by_G_pull(pos, m, G, dt, vel):

    pos_tf = tf.convert_to_tensor(pos, dtype=tf.float64)
    trans_pos = tf.transpose(pos_tf)

    m_temp = tf.convert_to_tensor(m, dtype=tf.float64)
    m_tf = tf.fill(pos_tf.shape, m_temp)

    G_tf = tf.convert_to_tensor(G, dtype=tf.float64)
    vel_tf = tf.convert_to_tensor(vel, dtype=tf.float64)

    x = pos_tf[:, 0:1]
    y = pos_tf[:, 1:2]
    z = pos_tf[:, 2:3]

    dx = tf.transpose(x) - x
    dy = tf.transpose(y) - y
    dz = tf.transpose(z) - z

    inv_r3 = (dx ** 2 + dy ** 2 + dz ** 2 + 0.1 ** 2) ** (-1.5)

    ax = G_tf * (dx * inv_r3) @ m_tf
    ay = G_tf * (dy * inv_r3) @ m_tf
    az = G_tf * (dz * inv_r3) @ m_tf

    ax = tf_delete(ax, 1, False)
    ax = tf_delete(ax, 1, False)
    ay = tf_delete(ay, 1, False)
    ay = tf_delete(ay, 1, False)
    az = tf_delete(az, 1, False)
    az = tf_delete(az, 1, False)

    a = tf.concat([ax, ay, az], axis=1)


    vel += a.numpy() * dt / 2.0
    pos += vel * dt

    return vel, pos



acc_by_G_pull(pos, m, G, dt, vel)
```

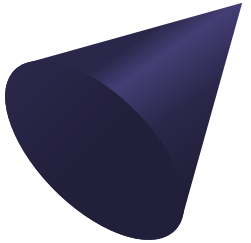
Semi výsledky

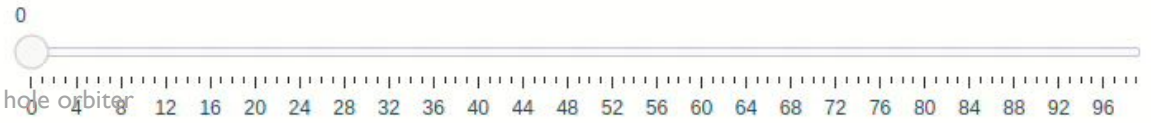


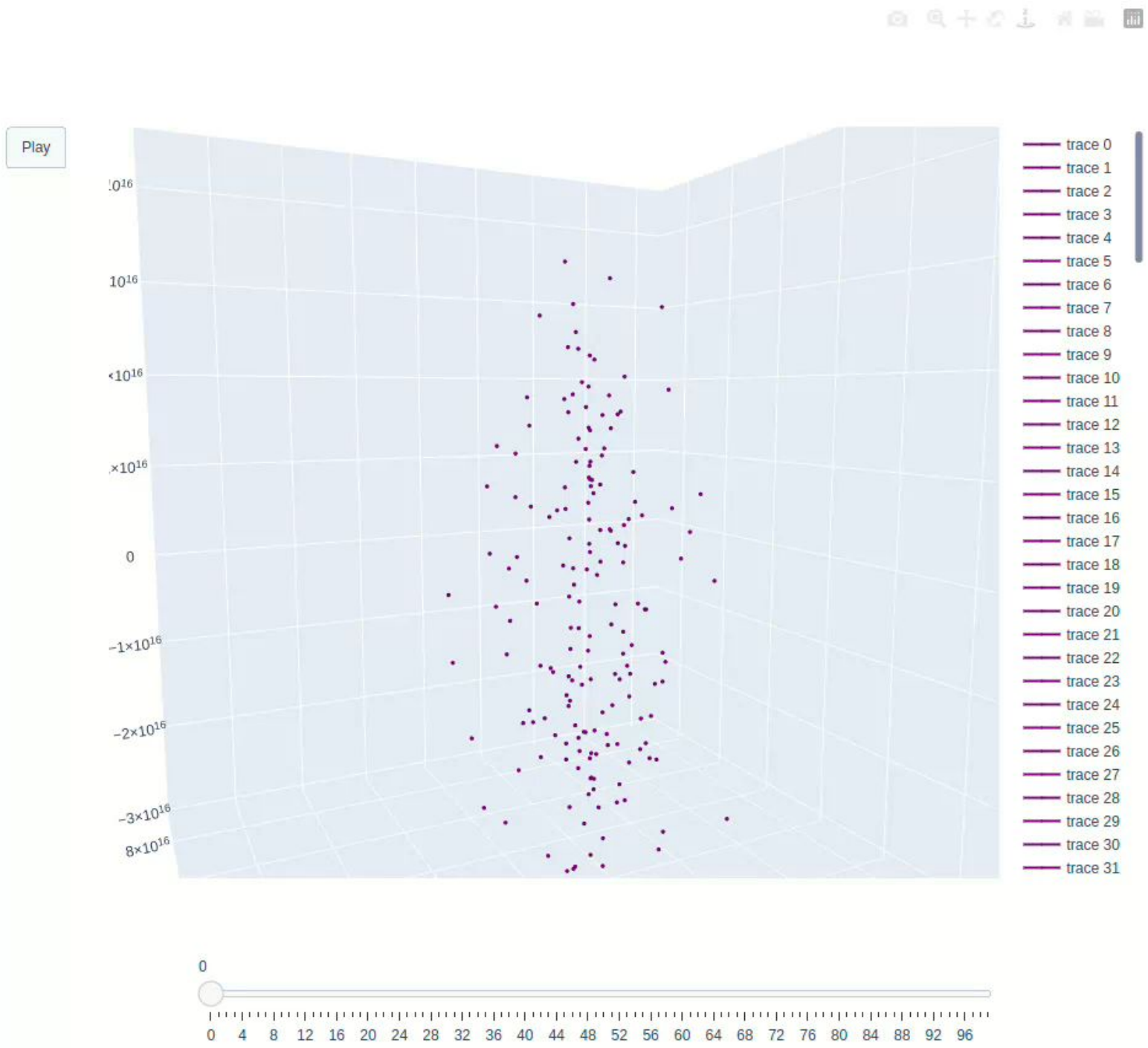
Lepšie výsledky


Play
0
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96


Play




0
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96


Play
.0^16
10^16
<10^16
×10^16
0
−1×10^16
−2×10^16
−3×10^16
8×10^16
trace 0
trace 1
trace 2
trace 3
trace 4
trace 5
trace 6
trace 7
trace 8
trace 9
trace 10
trace 11
trace 12
trace 13
trace 14
trace 15
trace 16
trace 17
trace 18
trace 19
trace 20
trace 21
trace 22
trace 23
trace 24
trace 25
trace 26
trace 27
trace 28
trace 29
trace 30
trace 31
0
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96

For the future

ABY SIMULÁCIA FUNGOVALA

✓

DASH VIZUALIZÁCIE

DODATOČNÉ VYLEPŠENIA

Záver

Ďakujem Vám za pozornosť