Programátorská dokumentace simulátoru Kozmot

Vojtěch Vlachovský - NPRG031 PA(9:00)

August 21, 2025

1 Úvod a architektura

Program ***Kozmot*** je implementován v jazyce C# společně s následujícími knihovnami

- .NET 9.0
- OpenTK (C# wrapper knihovny OpenGL)
- SixLabors-ImageSharp (Knihovna pro úpravy textur)

2 Struktura souborů a složek programu

- GUI
  - `ImGuiController.cs` Ovládácí program pro GUI
  - `ImGuiElementContainer.cs` Program ve kterém se implementuje samotné GUI
- ImageDependecies
  - Skybox
    - back.jpg
    - front.jpg
    - ⋮
    - Obsahuje 6 stran krychle která tvoří tzv. skybox (hvězdnou oblohu)
  - Textures
    - custom.jpg
    - desert.jpg
    - ⋮
    - textury pro každý zvolitelný typ planety

- Logs
  - `ConsoleOutputLogger.cs` skript pro zapisování konzolového výstupu do složek pro účely debuggování
  - `Known_Bugs.txt` Soubor pro jednoduché zapisování známých chyb pro práci v týmu
- Objects
  - `Camera.cs` Skript řídící kameru a veškerou její geometrii
  - `DirectionIndicator.cs` V případě že by se přecházelo z modelu matematického na model Newtonovský je zde možnost přidat na indikátor šipku která bude ukaazovat jakým směrem působí síla takzvaně in real time
  - `Grid.cs` Při běhu programu je viditelná mříž která slouží jako statický obět pro relativní pojem o rychlosti a míře rotace, tento skript slouží renderování této mříže na jejích vlastních bufferech a následném swappování do virtuální paměti grafické karty
  - `Indicator.cs` Ve výchozím nastavení polopropustná zelená koule která slouží jako ukazatel pozice a volených parametrů pro uživatele, neovlivňuje ji žádná simulace a je možno ji skrze GUI vypnout
  - `Object.cs` Při počátku vývoje měl tento skript sloužit jako rodičovský a všechny typy nebeských těles by z něj mohly dědit, v průběhu vývoje se ale z této myšlenky sešlo a bohužel neproběhl kompletní refaktoring kódu. Pro jakoukoliv změnu nebo přidání vlastnosti planet se mění konstruktor uvnitř tohoto skriptu. Tento skript je zároveň ten uvnitř kterého se provádí veškeré kalkulace pozic planet a to viz 1
  - `Skybox.cs` Tento skript slouží k vytvoření krychle doprostřed které je položena kamera tak aby byla vytvořena iluze hvězdného nekonečného nebe, ve své podstatě je skript hotový a efektivní a pro jakoukoliv změnu textury stačí vyměnit obrázky ve složce `Textures/Skybox`
- Shaders
  - `UI.frag`
  - `UI.vert`

    ⋮

    Soubor obsahující všechny shader kódy, tyto kódy jsou následně vlastním skriptem kompilovány a používány, pro zachování funkčnosti render pipeline musí každý nový shader být jak **vertexový** tak **fragmentový**

- Testing
  - `angularTest.cs` Soubor unit testů pokrývající velkou část programu a jeho funkčnosti a správnosti výpočtů

- `InputHandler.cs` Skript který pomocí event subscriptionu zachytává a zpracovává veškerý vstup z klávesnice nebo z myši, v případě přidání klávesové zkratky stačí pouze vytvořit novou funkci s logikou po stisknutí a následně ji přidat do logické větve buď myši nebo klávesnice

- `Program.cs` Vstupní bod celého programu, zde se inicializují veškeré předem nastavené hodnoty pro okno programu jako například : Velikost okna, full-screen, V-Sync, Kursor myši, dále se zde inicializují globální skripty které slouží jako předavači referencí a které musí být inicializovány mezi prvními jako třeba `Camera.cs`, `InputHandler.cs` a `Renderer.cs`. Jako poslední krok před spuštěním se spustí interní test pro OpenGL errory který následně vypíše například neplatnou cestu k obrázku nebo grafické chyby.

- `Renderer.cs` Nejkritičtější bod programu co se týče změn. Renderer zajišťuje celou Render Pipeline a její správný chod, pořadí ve kterém se jednotlivé funkce volají je kritické a nejlépe by nemělo být měněno jinak dojde k grafickým chybám které debugger neodhalí. Renderer neobsahuje žádnou logiku programu ale pouze vyměnuje zásobníky virtuálních bodů které postupně protékají render pipeline.

- `Shader.cs` Krátký program který usnadňuje vytvýření nového "shader programu". Jsou zde pouze dvě funkce a to funkce pro vytvoření programu z cesty k shader kódu a funkce pro zkompilování daného programu, ta se ale volá uvnitř první funkce při vytváření nového shaderu.

- `TextureLoader.cs` Tento kód slouží k namapování textur ze složky `Textures` na libovolný objekt, program prohledá složku uvedenou v cestě pro všechny soubory s koncovkou `png, jpeg, jpg` a při výběru planety tento kód vybere soubor podle typu dané planety, je tedy nutné aby každý obrázek měl jméno jako typ planety kterou reprezentuje aby byla tato automatizace zachována.

- `WindowManager.cs` Krátký kód který přihlásí funkce k daným eventům, kontroluje zda-li okno (hlavní agent programu) nehází OpenGL chyby a zajišťuje přepínání periferního vstupu při "překliknutí" z GUI na část se simulací a naopak.

- `Constants.cs` Soubor obsahující veškeré konstanty, je přístupný z každého jmenného prostoru nebo není v podsložce ale je v adresáři kořenovém.

3 Výpočet oběhu při simulaci

Model programu **Kozmot** je matematický heliocentrický model, tedy nesimuluje věrně gravitaci, ale pouze oběh ve viditelnější škále. Model je počítán pomocí úhlových rychlostí a velmi jednoduše popsáno, se jedná o soustavu jednotkových kružnic, ze kterých se bere jejich sinus a cosinus podle tělesa okolo kterého obíhají, viz obrázek.

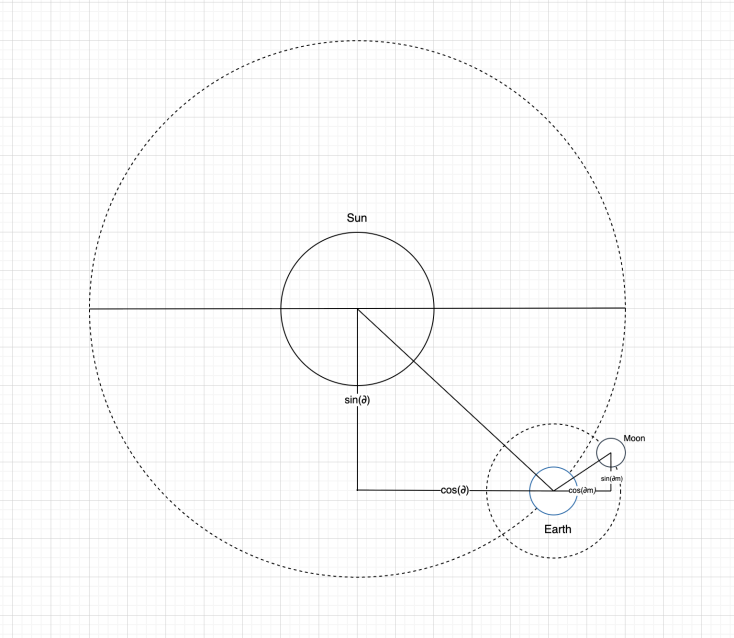

Figure 1: **Zjednodušený model výpočtu oběhů**

4 Závislosti

Program používá systém **NuGet** který automaticky podle přidaných balíčků upravuje `.csproj` soubor, ten při spuštění pomocí příkazů `dotnet restore` a `dotnet run` automaticky stáhne veškeré závislosti.

Program má pouze 2 hlavní knihovny a to `OpenTK` a knihovnu `StbImageSharp`. Knihovna OpenTK zároveň využívá 3 vlastní jmenné prostory a to `OpenTK.Windowing.Desktop , OpenTK.Graphics a OpenTK.Mathematics`. Knihovna ImageSharp slouží k transformacím obrátků a textur na bitmapy. Knihovna OpenTK zase usnadňuje a zprostředkovává komunikaci s grafickou kartou.

5 Testování

Program obsahuje celkem 7 testů a to následující

1. Generace planety na zadaných koordinacích
2. Generace planety s nulovým poloměrem využije výchozí poloměr
3. Test zda-li matice modelu zahrnuje všechny změny
4. Reference mezi rodičem a dítětem jsou správně nastaveny při generaci
5. Pozice pro rodičovské objekty se po změně projevuje správně dle kalkulací
6. Pozice pro dětské objekty se po změně projevuje správně dle kalkulací
7. Pozice dětského objektu se aktualizuje v závislosti na rodičovském objektu

testy se spouští pomocí vývojového prostředí skrze integrované testovací nástroje nebo přes příkaz `dotnet test OpenGL.Testing` spouštěný z kořenové složky

6 Možnosti pro další rozšíření

- Přepracování modelu z matematického na pravý newtonovský
- Přepracování z OpenTK na pravé OpenGL v jazyce C++
- Přidání dalších typů planet a simulace vícero slunečních soustav paralelně