SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-16605-115069

INTERAKTÍVNA APLIKÁCIA SIMULUJÚCA SLNEČNÚ SÚSTAVU OD JEJ VZNIKU PO ZÁNIK

BAKALÁRSKA PRÁCA

2024

Daniel Janík

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-16605-115069

INTERAKTÍVNA APLIKÁCIA SIMULUJÚCA SLNEČNÚ SÚSTAVU OD JEJ VZNIKU PO ZÁNIK

BAKALÁRSKA PRÁCA

Študijný program :	Aplikovaná informatika
Názov študijného odboru:	Informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	Ing. Dominik Janecký

2024

Daniel Janík

Slovenská technická univerzita v Bratislave
Ústav automobilovej mechatroniky

Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024
Evidenčné číslo: FEI-16605-115069

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Daniel Janík**
ID študenta:	115069
Študijný program:	aplikovaná informatika
Študijný odbor:	informatika
Vedúci práce:	Ing. Dominik Janecký
Vedúci pracoviska:	Ing. Ján Cigánek, PhD.

Názov práce: **Interaktívna aplikácia simulujúca Slnečnú sústavu od jej vzniku po zánik**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

V posledných rokoch je vesmírna tématika opäť veľmi zaujímavou a často rozoberanou témou. Základnou témou bakalárskej práce je návrh a vývoj edukačného virtuálneho prostredia v aplikácii, vytvorenej pomocou herného enginu Unity. Cieľom tejto bakalárskej práce je umožniť študentom a bežným ľuďom prístup k informáciám o histórii celej Slnečnej sústavy, ktoré doteraz neboli takto interaktívne spracované a verejnosti dostupné. Výsledná aplikácia prinesie zaujímavú skúsenosť, kde bude mať používateľ možnosť pohybovať sa vo voľnom vesmíre a simulovať životnosť celej Slnečnej sústavy. S jej pomocou môže vybrať z míľnikových epoch, aby sa zapojil do real-time simulácie. Aplikácia bude ponúkať možnosť pozastaviť čas, preskúmať informácie o hlavných telesách sústavy, ako sú planéty, mesiace, asteroidové pásma a ďalšie.

Úlohy:

1. Analyzovať existujúce edukačné hry a aplikácie týkajúce sa vesmíru a astronómie.
2. Navrhnúť koncept edukačnej aplikácie umožňujúcej interaktívne preskúmanie Slnečnej sústavy, vrátane jej hlavných telies.
3. Implementovať návrh aplikácie v hernom engine Unity.
4. Zhodnotiť výsledky vývoja a funkčnosti aplikácie, vypracovať dokumentáciu vrátane programovej dokumentácie a používateľskej príručky.
5. Popísať potenciálne využitie aplikácie vo vzdelávaní a ďalšie možnosti jej použitia alebo rozšírenia o iné moderné technológie.

Zoznam odbornej literatúry:

1. JANECKÝ, Dominik; KÓSA, Arpád. *Vývoj virtuálnej edukačnej hry o LED diódach.* Bakalárska práca. 2019. 45 s.
2. ONDREJOVIČ, Peter; KUČERA, Erik. *Edukačná hra vyvinutá v 3D engine pre oblasť informačno-komunikačných technológií.* Diplomová práca. Bratislava : 2020. 72 s.

Termín odovzdania bakalárskej práce:	31. 05. 2024
Dátum schválenia zadania bakalárskej práce:	11. 03. 2024
Zadanie bakalárskej práce schválil:	prof. Dr. rer. nat. Martin Drozda – garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program :	Aplikovaná informatika
Bakalárska práca:	Interaktívna aplikácia simulujúca Slnečnú sústavu od jej vzniku po zánik
Autor:	Daniel Janík
Vedúci záverečnej práce:	Ing. Dominik Janecký
Miesto a rok predloženia práce:	Bratislava 2024

Témou tejto bakalárskej práce je návrh a implementácia interaktívnej aplikácie v hernom engine Unity simulujúcej históriu a budúcnosť našej slnečnej sústavy. Cieľom tejto práce je vytvoriť aplikáciu, ktorá umožní prístup bežným ľudom k veľkému množstvu zaujímavých informácií o celej slnečnej sústave v jednom ucelenom celku. V prvej kapitole sa práca zaoberá analýzou dostupných technológii a ich voľbou. Druhá kapitola predstavuje porovnanie existujúcich riešení a vyvodenie ich nedostatkov. Nasledujúca kapitola sa venuje návrhu a implementácií interaktívnej aplikácie. Finálne riešenie poskytuje používateľovi možnosť výberu počiatočného roku spustenia simulácie, voľný pohyb po celej slnečnej sústave a množstvo informácií o každom vesmírnom telese nachádzajúcom sa v nej. V prílohe sa nachádza návod na inštaláciu a samotná aplikácia s príručkou obsluhy.

Kľúčové slová: Vzdelávacia aplikácia, Slnečná sústava, Unity, Vznik, Zánik, C#, Vesmír

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Bachelor Thesis:	An Interactive application simulating the Solar System from its formation to its demise
Autor:	Daniel Janík
Supervisor:	Ing. Dominik Janecký
Place and year of submission:	Bratislava 2024

The topic of this bachelor's thesis is the design and implementation of an interactive application made in the game engine Unity that simulates the history and future of our solar system. The goal of this thesis is to create an application that will give access to a huge amount of interesting information about the entire solar system in one comprehensive package for ordinary people. The first chapter deals with the analysis and selection of available technologies. The second chapter presents a comparison of existing solutions and identifies their shortcomings. The following chapter deals with the design and implementation of the interactive application. The final solution provides the user with the ability to select the start year of the simulation, free movement trough the solar system and a wealth of information about each celestial body. The appendix contains an installation guide and the application with a user manual.

Key words: Education application, Solar System, Unity, Formation, Demise, C#, Space

Vyhlásenie autora

Podpísaný Daniel Janík čestne vyhlasujem, že som Bakalársku prácu Interaktívna aplikácia simulujúca Slnečnú sústavu od jej vzniku po zánik vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci. Uvedenú prácu som vypracoval pod vedením Ing. Dominika Janeckého.

V Bratislave dňa 31.05.2024

.....

podpis autora

Poďakovanie

Ďakujem môjmu vedúcemu práce Ing. Dominikovi Janeckému za poskytnutú podporu a odbornú pomoc. Taktiež by som sa chcel poďakovať svojej rodine a kamarátom za podporu počas štúdia.

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek a značiek

UE5 – Unreal Engine 5

IDE – Integrated Development Environment

VS – Visual Studio

VS Code – Visual Studio Code

MS – Microsoft

CAD – Computer Aided Design

GUI – Graphical User Interface

GIMP – GNU Image Manipulation Program

PS – Photoshop

AI – Artificial intelligence

NASA – Národný úrad pre letectvo a vesmír

UX – User Experience

UML – Unified Modeling Language

Úvod

V posledných rokoch záujem o prieskum vesmíru dosahuje svoje historické maximum, čo sa prejavuje čoraz častejšími štartmi rakiet smerujúcich k obežnej dráhe mesiacu a aj jeho povrchu v rámci rôznych národných vesmírnych programov. Tento fenomén je často označovaný ako druhé vesmírne preteky, tentoraz medzi USA a Čínou, ktorých cieľom je opätovné pristátie človeka na mesiac [1].

Tento zvýšený záujem o vesmír je poháňaný technologickým pokrokom v oblasti výpočtovej techniky a internetu, ktorý umožňuje bežným používateľom jednoduchý prístup k širokému spektru informácií [2]. Súčasne sa zlepšuje grafické rozhranie, čo umožňuje realistickejšie vizualizácie vesmíru. Dnes sa najčastejšie stretávame s vizualizáciou vesmíru prostredníctvom kníh, dokumentárnych filmov a vedeckých článkov. Avšak obsah týchto médií je často obmedzený a poskytuje iba čiastočný pohľad na problematiku. Okrem toho, tieto zdroje neposkytujú možnosť interakcie. V dôsledku toho sa môže stať, že uvedené tradičné formy prezentácie vesmíru neuspokoja narastajúci dopyt po interaktívnych a prehľadných zdrojoch informácií.

Ako efektívnu alternatívu, ktorá by prekonala tieto nedostatky, možno považovať interaktívne aplikácie [3]. Tieto umožňujú používateľom aktívne skúmať vesmírne koncepty, poskytujú širšie množstvo informácií a umožňujú interakciu. Návrh a implementácia interaktívnych aplikácií sú kľúčové pre vytváranie pútavých vzdelávacích nástrojov, ktoré nielen dopĺňajú tradičné médiá, ale aj otvárajú cestu k lepšiemu pochopeniu a interaktívnemu prieskumu vesmíru [4].

Cieľom tejto bakalárskej práce je navrhnúť a implementovať interaktívnu aplikáciu, ktorá zobrazuje vznik a zánik našej slnečnej sústavy, pričom zjednocuje širokú škálu dostupných informácií a sprostredkúva komplexné javy bežným používateľom. Aplikácia bude následne sprístupnená verejnosti s cieľom zvýšiť záujem o vesmír a podporiť širšie pochopenie jeho fungovania. Zdrojový kód aplikácie je navrhnutý tak, aby bol jednoducho aktualizovateľný, čo umožní rýchlu implementáciu nových poznatkov o vesmírnych telesách alebo na najnovšie objavy v menej preskúmaných častiach slnečnej sústavy. Týmto zabezpečíme, že aplikácia zostane relevantná aj s väčším odstupom času.

1 Výber technológií

Návrh a vývoj počítačovej hry sa od vývoju bežnej aplikácie nelíši vo veľkom. V oboch prípadoch si vieme implementáciu zefektívniť a uľahčiť použitím dostupných knižníc a frameworkov pre daný programovací jazyk. Herný framework obsahuje kolekciu knižníc a nástrojov, ktoré vedia zjednodušiť vývoj. Herný engine naopak je softvér, ktorý ponúka vývojárovi grafické rozhranie spolu so sadou nástrojov pre grafiku, vstup, zvuk a iné. V prípade vývoju komplexných hier preferujeme herný engine, nakoľko ponúka voči frameworku oveľa viac funkcionality [5].

1.1 Herný Engine

Herný Engine je softvérové vývojové prostredie, s mnohými nástrojmi, pre optimalizáciu a zjednodušenie vývoju hier v rôznych programovacích jazykoch. Herný engine zahŕňa 2D alebo 3D grafické rozhranie, kompatibilné s rôznymi formátmi ako sú fyzikálne rozhranie, umelú inteligenciu, ktorá automaticky reaguje na vstup hráča, zvukové rozhranie a animácie a nespočetne veľa ďalších nástrojov. V súčasnej dobe je nepredstaviteľné vyvíjať hry bez použitia herného engine a ďalších vhodných nástrojov. Tieto nástroje sa z odstupom času stali široko dostupné verejnosti [6].

1.1.1 Unreal

Je voľne dostupný herný engine vyvíjaný spoločnosťou Epic Games. Najnovšia dostupná verzia je UE5, ktorá dovoľuje vývojárom vytvárať tie najrealistickejšie objekty a herné scény spomedzi všetkých dostupných engine na dnešnom trhu. Vďaka tomu je UE5 je ideálnou voľbou pre graficky a výpočtovo náročné projekty, vrátane filmov a animácií [7].
Výhody UE5:

- Rozsiahla sada nástrojov pre tvorbu 3D objektov, prostredia a herných mechaník
- Vhodné obchodné podmienky, účtovanie začína od dosiahnutí 1 milióna USD s tým, že vývojár má poplatok 5% z hrubého príjmu.
- Výkonný engine napísaný v C++, zvláda výpočtovo náročne úlohy voči konkurenčným engine.
- Svetelné efekty, voči Unity má efektívnejší a optimalizovanejšie efekty.

Nevýhody UE5:

- Programovací jazyk C++, narozdiel od konkurencií, ktoré využívajú vysokoúrovňové jazyky ako napr. C#, UE5 vyžaduje znalosti komplexnejšieho jazyka C++.
- Menšia komunita. V porovnaní s konkurenčnými má Unreal menšiu komunitu vývojárov vytvárajúcich návody a nástroje na zlepšenie workflowu [8].

1.1.2 Godot

Godot engine je multiplatformový 2D a 3D open-source herný engine. Vhodný pre začiatočníkov na tvorbu jednoduchých hier [9]. Voči ostatným herným engine sa líši v programovacom jazyku GDScript. Tento jazyk je dynamický so syntaxou podobnou k Pythonu. Avšak Godot podporuje aj C# a technológiu na konverziu na iné jazyky ako C a C++ [10].

Výhody Godotu:

- Open-source podporovaný a aktualizovaný hlavne jeho komunitou s vysokou možnosťou úprav aj samotného jadra kódu.
- Nenákladný a efektívny na rôznych platformách vhodný pre hry aj simulácie a multimédiové aplikácie.
- Jednoduché používateľské rozhranie s vizuálnym skriptovaním, bez väčšieho programovania vhodný pre programovacích začiatočníkov.

Nevýhody Godotu:

- Menšia knižnica assetov voči konkurencií. Vývojár si musí mnohokrát vytvoriť vlastné assety od základov.
- Nevhodný pre výpočtovo náročné úlohy a komplexnejšie projekty veľkého rozmedzia.
- Nedostatočná dokumentácia pre vývojárov [11].

1.1.3 Hazel

Je interaktívna 3D aplikačná vývojová platforma, inak nazývané ako herný engine. Vyvíjaná malým štúdiom Cherno od roku 2018. Engine začínal ako návod na vývoj herných engine na platforme YouTube, neskôr vďaka komunite sa z neho stal plne funkčný engine, v ktorom sa už dnes dajú vytvárať hry. Štúdio Cherno má v budúcnosti v pláne vydať bezplatnú verziu pre všetky platformy [12].

Výhody Hazel:

- Naprogramovaný v C++, čím sa zaručuje podpora pre Windows a Linux.
- C# a .NET ako skriptovací jazyk (podobne ako pri Unity).
- Použitie technológie Vulkan pre vykreslovanie [13].
- „Hot reloading“ pre assety.

Nevýhody Hazel:

- Pomalý vývoj, štúdio Cherno sa skladá iba z piatich ľudí.
- Pomerne nový a neuhladený engine, môže prísť k veľkým chybám samostatnej aplikácie počas vývoju.
- Dostupný iba pre predplatiteľov služby patreon [12].

1.1.4 Unity

Unity Engine je real-time 2D a 3D multiplatformový vývojársky engine pre umelcov, dizajnérov a vývojárov, ktorý umožňuje vytvárať interaktívne aplikácie. Unity sa využíva taktiež v filmovom, hernom priemysle a je vhodný pre tvorbu mobilných hier, ale taktiež aj pre robustné a komplexné projekty [14].

Výhody Unity:

- Intuitívne a priateľské užívateľské rozhranie Unity zjednodušuje prácu vývojárov. Bez ohľadu na ich znalosti.
- Disponuje vlastnou knižnicou s veľkým repozitárom predpripravených assetov, pluginov a nástrojov. Toto urýchľuje vývoj ponukou hotových assetov.
- Má veľkú a aktívnu komunitu vývojárov, umelcov a nadšencov v ekosystéme. Komunita poskytuje cenné zdroje, návody a podporu, čím vytvára spolupracujúce ekosystému.
- Jadro engine je naprogramované v C++ zabezpečuje tým efektivitu pre robustnejšie projekty, avšak samotné hry sú programované v C# pre lepší workflow

Nevýhody Unity:

- Pre používanie pokročilejších funkcií a služieb si zaplatiť cenovo náročné predplatné ak je vývojár samostatná osoba.
- Nedostatočná podpora vizuálneho skriptovania voči Unreal Engine
- Pochopenie pokročilejších funkcií je dosť náročné
- Komplexná optimalizácia pre robustné a vysoko nákladové hry [11].

1.2 Vývojové prostredie

IDE je softvér pre tvorbu aplikácií, ktorý sa skladá z bežných vývojárskych nástrojov spojených do jedného grafického rozhrania. V IDE bežne nájdeme editoru kódu, kompilátor a debugger. IDE umožňuje vývojárom zefektívniť programovanie kódu, ďalšie funkcie IDE pomáhajú organizovať pracovný postup vývojárov a riešiť problémy, pričom väčšina moderných IDE obsahuje nástroje na identifikáciu chýb v reálnom čase a zvýrazňovanie syntaxe, meniť kód počas runtime a ďalšie. V dnešnej dobe je programovanie bez IDE priam nepredstaviteľné a časovo náročné, ale stále možné používaním textových editorov [15].

Unity podporuje hŕstku textových editorov a všetky tri najpopulárnejšie IDE na programovanie v C# čo sú voľne dostupné bezplatné Visual Studio a Visual Studio Code od Microsoft a proprietárny editor Rider od spoločnosti JetBrains, pričom Visual Studio je predvolené a najpodporovanejšie IDE na tvorbu C# skriptov v prostredí Unity, rozbor týchto programov si rozoberieme v nasledujúcich podkapitolách [16].

1.2.1 Visual Studio

Je integrované vývojové prostredie (IDE) pre vývojárov v .NET a C++ na platformu Windows. Obsahuje kompletnú sadu nástrojov a funkcií na zrýchlenie a vylepšenie všetkých fáz vývoja softvéru [17]. MS poskytuje 3 edície Visual Studio :

- Community – voľne dostupná, ponúka všetky podstatné nástroje na kompletný vývoj.
- Professional – Platená edícia, potrebná pre ziskové spoločnosti.
- Enterprise – Platená edícia, ponúka navyše voči ostatným edíciám testovacie nástroje, expresnú diagnostiku a dodatočné nástroje pre prácu s Xamarin frameworkom [18].

1.2.2 Jetbrains Rider

Je cross-platformové platené IDE pre vývoj .NET a hier vyvíjané spoločnosťou Jetbrains. Ponúka pokročilé nástroje na vývoj webových aplikácií pomocou ASP.NET. Podporu pre herný vývoj integráciou s Unity, cloudový vývoj [19].

1.2.3 Visual Studio Code

Je samotný multiplatformový editor zdrojového kód, primárne stavaný na vývoj webu s intergrovanou podporou pre Javascript, Nodejs a TypeScript [17]. VS Code narozdiel od svojej konkurencie obsahuje bohatú sadu rozšírení udržiavaných komunitou. Pomocou týchto rozšírení je možné VS Code premeniť na takmer plne integrované vývojové prostredie pre ľubovoľný jazyk [20].

1.3 3D Modelovacie softvéry

3D modely, ktoré reprezentujú fyzický objekt alebo teleso pomocou skupiny bodov v priestore, spájané rôznymi geometrickými entitami, ako sú čiary, trojuholníky, zakrivené povrchy a pod.. 3D modely, ktoré sú skupinou dátových bodov, môžu byť vytvorené ručne, algoritmicky alebo pomocou skenovacieho softvéru, ich povrchy môžu byť ďalej zadefinované mapovaním textúr.

3D Modelovanie popisuje použitie softvérových nástrojov, ako sú CAD (Computer-Aided Design), k vytvoreniu 3D digitálnej reprezentácie objektu. 3D modelovanie sa využíva v automobilovom dizajne, výrobe priemyselných zariadení, architektúre, strojárstve a vývoje hier [21].

1.3.1 Blender

Je bezplatný multiplatformový open-source 3D modelovací nástroj, vhodný pre jednotlivcov. Podporuje celú 3D pipeline, čo zahŕňa modelovanie, manipuláciu, animácie, simulovanie, vykresľovanie, kompozíciu a sledovanie pohybu, vhodné pre herný dizajn a do videí. Pre pokročilých používateľov ponúka Blender API pre Python skriptovanie na prispôsobenie aplikácie a písanie špecializovaných nástrojov, ktoré sa neskôr zahŕňajú do budúcich verzií Blenderu [22].

Výhody Blender:

- Voľne dostupný open-source pre záujemcov 3D modelovania.
- Možnosť tvorby 2D animácií pomocou Grease Pencil.
- Obsahuje video editačné funkcie, čo konkurencia neponúka.
- Poskytuje Eevee, čo je vykresľovanie v reálnom čase pre svetelné efekty a scény.

Nevýhody Blender:

- Neintuitívne používateľské rozhranie pre začiatočníkov.
- Slabšie vykresľovacie schopnosti ako má konkurencia.
- Chýba mu špecializácia, exceluje v modelovaní a textúrovaní, ale zaostáva v požiadavkách 3D umelcov na pohybové grafiky [23].

1.3.2 Cinema 4D

Cinema 4D je profesionálny softvér na modelovanie, animáciu, simuláciu a vykresľovanie 3D. Je to rýchly, výkonný, stabilný a flexibilný balík nástrojov pre efektívny a prístupný dizajn, VFX, AR/MR/VR, vývoj hier a všetkých typov vizualizácie [24]. Cinema 4D je hlavne používaný v kreatívnom priemysle pre 3D vizualizáciu a produktovú vizualizáciu, taktiež je často využívaných vo filmoch a televíznych seriáloch [25].

Výhody Cinema 4D:

- Bez pochyby najľahší na naučenie a ovládnutie pre začiatočníkov.
- Poskytuje vynikajúce možnosti simulácie častíc, ako sú mraky, oheň, voda, dym a prášok.
- Ideálny pre kombináciu pohybových grafík a vizuálnych efektov

Nevýhody Cinema 4D:

- Neexistuje bezplatná verzia, (Študentská licencia stojí 80€ ročne).
- Vyžaduje vysoko výkonný počítač, ak chce používateľ vykresliť scénu za jednu hodinu a menej.
- Pre náročnejšie projekty, je potrebné zakúpiť aj rozšírenia tretích strán [23].

1.3.3 Autodesk 3ds Max

Je platený počítačový program na tvorbu 3D modelov a animovania digitálnych obrázkov. Patrí medzi najpopulárnejšie programy v priemysle počítačovej grafiky, obľúbený medzi vývojármi hier, TV štúdiami a architektmi. Stojí za ním spoločnosť Autodesk známa svojím softvérom ako je AutoCad a Maya. 3ds Max je používaný hlavne na modelovanie postáv a na vykresľovanie fotorealistických obrázkov budov a iných objektov [26].

Výhody 3ds max:

- Ponúka širokú škálu funkcií a možností, zahŕňajúc programovanie a prispôsobovanie.
- Skriptovací jazyk maxscript je relatívne ľahký na naučenie.

- Má vynikajúci editačný nástroj edit poly modifier a systém spline na trhu.

Nevýhody 3ds max:

- Cenovo náročný pre zakúpenie, existuje však študentská licencia
- Naučiť sa tento výkonný 3D softvér vyžaduje čas.
- Môžu nastať kompatibilné problémy s iným softvérom Autodesk, ktoré si vyžadujú veľa doplnkov na funkčnosť.
- 3ds Max nie je multiplatformový, funguje iba na systéme Windows [27].

1.4 Grafický softvér

Je typ počítačového programu pre tvorbu a úpravu obrázkov. Na dnešnom trhu je veľký rozsah dostupných programov pre tento účel. Jedná sa o jednoduché programy až po komplexné programy, ktoré umožňujú tvoriť dokonca aj 3D modely a animácie. Grafický softvér vieme rozdeliť do štyroch základných kategórií [28]:

- Grafické programy pre vektorovú grafiku – obrázky sa tvoria z čiar a tvarov, nestrácajú na kvalite pri zväčšení a zmenšení, používajú sa hlavne pri tvorbe GUI.
- Grafické programy pre rastrovú grafiku – obrázok je zložený z pixelov, strata kvality pri zmenšení a zväčšení, používa sa pre tvorbu textúr.
- Grafické programy pre 3D grafiku – tvorba 3D modelov, používa sa prevažne v hernom dizajne.
- Grafické programy pre animácie – tvorba pohyblivých obrázkov, používa sa prevažne pre filmy, reklamy a hry [28].

1.4.1 GIMP

Celým názvom GNU Image Manipulation Program je open-source cross-platformový editor obrázkov. Ktorý je ponúka veľkú sadu nástrojov pre grafických dizajnérov, fotografov, vedcov. K nim je ešte možné vďaka open-source distribúcií doinštalovať dodatočné balíčky tretích strán na zvýšenie produktivity [29].

Výhody GIMP:

- Open-source bezplatná alternatíva pre Adobe Photoshop.
- Jednoduchý UI, vhodné pre začiatočníkov a pre malé projekty.
- Možnosť vylepšenia základnej verzie pomocou balíčkou.

Nevýhody GIMP:

- Menší počet nástrojov a funkcionalít ako Photoshop
- Nepodporuje prácu s RAW formátom obrázkov
- Malá integrácia s iným softvérom [32].

1.4.2 Adobe Photoshop

Je platená [30] softvérová aplikácia na úpravu obrázkov, retušovanie fotografií, digitálny dizajn pre Windows a MacOS. Ponúka možnosť používateľom upravovať a vylepšovať obrázky a ilustrácie vďaka jeho bohatou sadou nástrojov. Je to najpoužívanejší softvér na úpravu obrázkov dokonca aj videí cez vrstvy. Adobe ponúka niekoľko verzií PS, ktoré sa líšia ich cenou a funkcionalitou [31].

Najnovšie verzie PS obsahujú nástroje s umelou inteligenciou pre generovanie obrázkov, úpravu nedostatkových detailov generatívne rozšírenie obrázku a ďalšie. Týmto kombinuje Photoshop silu umelej inteligencie a presnosť jeho nástrojov a funkcionalít pre väčšiu kontrolu nad tvorbou grafického obsahu [30].

Výhody PS:

- Integrácia generatívnej umelej inteligencie do novších verzií.
- Veľká komunita tvoriaca návody pre nové verzie PS.
- Nespočetný počet funkcionalít a nástroj na editáciu fotiek po grafický dizajn.

Nevýhody PS:

- Pre plnú verziu neexistuje doživotná licencia, iba možnosť kúpy cez mesačné predplatné.
- Študentské licencie sú iba zľavnené, neexistuje možnosť používania zadarmo .
- Voči konkurencií dosť cenovo náročný grafický nástroj [32].

1.5 Generatívna umelá inteligencia

Generatívna AI ponúka používateľom rýchlo generovať obsah na základne rôznych vstupov. Typy vstupov môžu byť rôzne, napríklad: text, obrázky, zvuky, animácie a iné.

Tieto modely umelej inteligencie využívajú neurónové siete na identifikovanie vzorov a štruktúr existujúcich údajov z databázy s cieľom vygenerovať nový originálny obsah. Najväčší prínos pre generatívne modely priniesla schopnosť využitia rôznych prístupov

učenia vrátane kontrolovaného a nekontrolovaného učenia na trénovanie. Týmto sa zefektívnilo trénovanie pre veľké množstvá dát a vytvorenie základných modelov pre ďalšie špecializovanie na dané témy. Medzi základné modely patria napríklad GPT-3 a Stable Diffusion, ktoré sa následne použijú ako základ pre tvorbu špecializovaných aplikácií ako je ChatGPT, ktorý generuje na základe vstupu, textový výstup. Stable Diffusion narodiel od toho používa textový vstup pre generáciu obrázkov [33].

1.5.1 NVIDIA Canvas

Je grafický nástroj, ktorý pomocou umelej inteligencie mení základné ťahy štetcom na realistické textúry krajín. Intuitívny dizajn aplikácie umožňuje rýchlo nakresliť realistické koncepty a následne ich doladiť v Photoshope vďaka možnosti exportovania obrázkov do formátu podporovaných PS. Základná verzia podporuje niekoľko textúr ako piesok, zem, rôzne typy vôd, skalnaté a kamenisté povrchy, oblohu a trávnaté porasty, k týmto textúram je ešte možné zvoliť niekoľko variácií každej textúry. Umožňuje aj import vlastných textúr. Nevýhodou je že aplikáciu je možné nainštalovať iba na platformu Windows a grafickou kartou série RTX [34].

1.5.2 Midjourney

Predstavuje typ pokročilej generatívnej AI, ktorá dokáže na základe textového vstupu vytvoriť takmer dokonale realistické obrázky. Narozdiel od konkurencie funguje Midjourney cez bota na platforme Discord, pričom kvalita generovaných obrázkov sa líši na základe textového vstupu, ktorý treba detailne špecifikovať [35].

Výhody Midjouney AI:

- Ak si používateľ zakúpi predplatné, má k dispozícií takmer nekonečnú knižnicu vygenerovaných obrázkov spolu s ich textovým vstupom.
- V minulosti jediná dobrá alternatíva pre rýchle a lacné generovanie kvalitných obrázkov a textúr

Nevýhody Midjourney AI:

- Nie je open-source.
- Na využitie treba zakúpiť predplatné.
- Používateľ potrebuje mať Discord účet a musí byť pripojený aspoň na hlavný Midjourney server aby mal prístup k botovi [35].

1.5.3 Fooocus

Je voľne dostupný AI generátor obrázkov, ktorý prijíma textový vstup a následne ho mení na obrázkový výstup. Určený pre marketérov a rýchlu tvorbu obsahu. Vychádza zo základného modelu Stable Diffusion XL najnovším modelom od StabilityAI. Popri textovom vstupe si používateľ vie aplikovať rôzne umelecké filtre [36].

Výhody Foocus AI:

- Je voľne dostupný a open-source.
- Možnosť nainštalovať si ho na vlastný počítač a používať bez internetu.
- Možnosť doinštalovania rôznych balíčkov tém, pre rôzne generátory.

Nevýhody Foocus AI:

- Vyžaduje veľa pamäte.
- Generácia obrázku závisí od výkonu grafickej karty [36].

Po analýze dostupných technológií a softvéru sme si pre vývoj aplikácie zvolili, ktorý softvér bude pre nás ponúkať najvhodnejšie funkcionality a nástroje. Zvolili sme si herný engine Unity, pre jeho jednoduchosť a programovací jazyk C#. VS Code na programovanie C# skriptov do Unity, vďaka jeho ľahkosti ako IDE. Jednoduché sférické objekty budú vytvorené pomocou Blender a pre editáciu textúr sme si zvolili Photoshop, ktorý poskytuje veľkú knižnicu štetcov na texturovanie.

2 Konkurenčné aplikácie

Existuje viacero simulácií vesmíru, zameraných na rôzne aspekty jeho funkcionality a zložitosti. V tejto kapitole si preto podrobne rozoberieme štyri najpopulárnejšie simulácie, analyzujeme ich výhody, nevýhody a nedostatky. Cieľom tohto rozboru je poskytnúť podklady pre návrh a koncept našej vlastnej simulácie vesmíru. Keďže téma vesmíru a jeho simulácií je komplexná a náročná, je dôležité mať prehľad o existujúcich riešeniach a ich obmedzeniach pred vlastným návrhom. Tento kritický pohľad na súčasné simulácie nám umožní identifikovať nedostatky a možnosti zlepšenia.

2.1 Universe Sandbox

Je multiplatformová komplexná simulácia vesmíru, dostupná dokonca aj na AR/VR Primárnym cieľom aplikácie je simulácia gravitácie, teploty a klímy. Hoci sa v súčasnosti obmedzuje na newtonovskú fyziku, prebiehajú snahy o integráciu efektov relativity, tak aby neboli obmedzené výpočtovými nárokmi. Hlavným cieľom aplikácie je vytvoriť časovo stabilnú simuláciu n-body problému bez použitia kvantových počítačov [37]. Pre tých, ktorí chcú vytvárať simulácie slnečnej sústavy, predstavuje Universe Sandbox sofistikované riešenie, ktoré umožňuje interaktívne skúmanie a experimentovanie s vesmírnymi javmi [38].

Výhody:

- Rôzne scenáre, najvyšší počet zo všetkých dostupných simulácií.
- Databáza vesmírnych telies obsahuje viac ako päťdesiat tisíc objektov.
- Možnosť doinštalovania rôznych balíčkov cez Steam workshop vytvorených používateľmi a komunitou.
- Expertná a detailná simulácia atmosféry Zeme a Marsu.
- Možnosť tvorby detailnej vlastnej slnečnej sústavy.
- Náhodne generované vesmírne objekty.

Nevýhody:

- Platený softvér, neexistuje demo ani voľne dostupná verzia.
- Náročný a robustný softvér na výpočtovú silu.
- Pohyb kamery je dosť vyhľadený a je možné, že používateľ aj miernym pohybom pôjde až príliš ďaleko od svojho cieľu.

- Simulácia času razantne neovplyvňuje vlastnosti objektov.

2.2 Space Engine

Patrí medzi najpopulárnejšiu simuláciu vesmíru. Primárne využitie Space Engine je vo dokumentárnych filmoch a videí. Jedná sa takisto o typ simulácie sandbox. Pričom rozdiel medzi konkurenciou je možnosť v prvej osobe preskúmať celý viditeľný vesmír, iné galaxie, hviezdne sústavy a podobné. Planéty a telesá sú procedurálne generované. Space Engine sa primárne sústreďuje na aspekt skúmania okolitého vesmíru ako jeho fyzikálnu simuláciu. Preto jediná simulácia gravitácie je možná iba medzi planétami a ich hviezdou [39].

Výhody:

- Veľká databáza vesmírnych telies.
- Vysoko kvalitné textúry a procedurálne generované telesá.
- Mierka vesmíru je 1:1.
- Možnosť simulácie pristátia vesmírnou loďou na planétu.
- Kamera z prvej osoby.

Nevýhody:

- Platený softvér, neexistuje voľne dostupná verzia ani demo.
- Pomerne náročný na grafiku.
- Neexistujúca simulácia času s gravitáciou.

2.3 SpaceSim

Je pomerne nová simulácia vesmíru, jedná sa tiež o typ simulácie sandbox. Tento typ simulácie umožňuje používateľovi vkladať dostupné vesmírne telesá a upravovať ich fyzikálne vlastnosti. Taktiež ponúka rôzne preddefinované scenáre. Softvér vznikol v roku 2018, čo ho robí najmladším spomedzi porovnávaných softvérov od českého vývojára Pavla Ševečka. Jeho primárnou úlohou je simulovať N-body problém [37, 40].

Výhody:

- Open-source a voľne dostupný na stiahnutie.
- Rôzne scenária, ktoré konkurenčné programy zatiaľ neponúkajú.
- Možnosť prispôsobenia textúr priamo v aplikácií.
- Aplikácia zaberá minimálny priestor na disku

- Možnosť robenia videa z simulácie

Nevýhody:

- Počtom objektov a častíc klesá rýchlosť simulovania času.
- Nedoladený, môžu nastať chyby, pri ktorých treba aplikáciu reštartovať.
- Aj keď autor tvrdí, že je UI dizajn intuitívny a ľahký na použitie [40], je pre začiatočníka komplexnejší ako UI konkurencie.

2.4 NASA's Eyes

Jedná sa o jednoduchú simuláciu našej slnečnej sústavy, poskytnutej na hlavnej stránke Národného úradu pre letectvo a vesmír (NASA). Simulácia zahŕňa všetky planéty našej slnečnej sústavy a po priblížení na ľubovoľnú planétu aj jej mesiace. Pričom je možné simulovať aj čas. Dodatočne je možné zapnúť aj vybrané asteroidy, kométy a súhvezdia. V skratke sa dá povedať, že simulácia obsahuje všetky vesmírne telesá, ktoré NASA v minulosti a súčasnosti preskúmava [41].

Výhody:

- Dostupné na webe.
- Kvalitné textúry vesmírnych telies.
- Zobrazenie historických udalostí ako Voyager 1.

Nevýhody:

- Neintuitívny dizajn UI.
- Simulácia času je možná iba v malom rozmedzí od 1949 po 2049.
- Limitovaný pohyb kamery a polohy hráča.

2.5 Zhrnutie aplikácií

Z dôkladnej analýzy sme zistili, že aplikácie sa sústreďujú prevažne na simuláciu gravitácie pomocou N-body problému, pričom obetujú možnosť simulácie v čase, nakoľko ide o komplexný problém, ktorý je veľmi ťažko možné predpovedať pri veľkých časových rozdieloch [37]. Taktiež počas ich analýzy a testovania sme zistili, že pohyb kamery a samotného hráča nemá dobre vyriešená ani jedna aplikácia. Z týchto poznatkov vieme navrhnúť aplikáciu, ktorá by vyplnila tieto nedostatky a zaplnila dieru na trhu.

3 Návrh a Implementácia

Táto kapitola obsahuje návrh a implementáciu aplikácie, ktorá simuluje životný cyklus slnečnej sústavy, od jej formovania až po zánik. Pred každou implementáciou predchádza návrh. Kvalitný návrh je kľúčový pre hladký a bezproblémový vývoj softvéru. Zlý návrh môže viesť k komplikovanému a neprehľadnému stavu samotného vývoju.

3.6 Návrh

Prvé kroky návrhu pozostávajú z vhodne určených požiadavok samotného používateľa aplikácie. Tento krok je nesmierne potrebným aby sme zaručili kvalitný UX. Tento návrh následne odkonzultujeme a predstavíme vzorke používateľov a v prípade nedostatkov ich upravíme.

3.6.1 Špecifikácia požiadaviek

Používateľské požiadavky sa dajú rozdeliť do 3 skupín a to: funkcionálne, nefunkcionálne, doménové.

Funkcionálne požiadavky

- Používateľ musí byť schopný pozastaviť čas simulácie.
- Používateľ má možnosť voľby perspektívy kamery.
- Aplikácia musí ponúkať informácie používateľovi v adekvátnom časovom intervale.
- Aplikácia musí byť spustiteľná aj pre výkonovo slabšie počítače.
- Používateľ má mať možnosť pozorovať vlastnosti telies.
- Oneskorenie medzi vstupom používateľa a reakciou aplikácie by malo byť minimálne.

Nefunkcionálne požiadavky

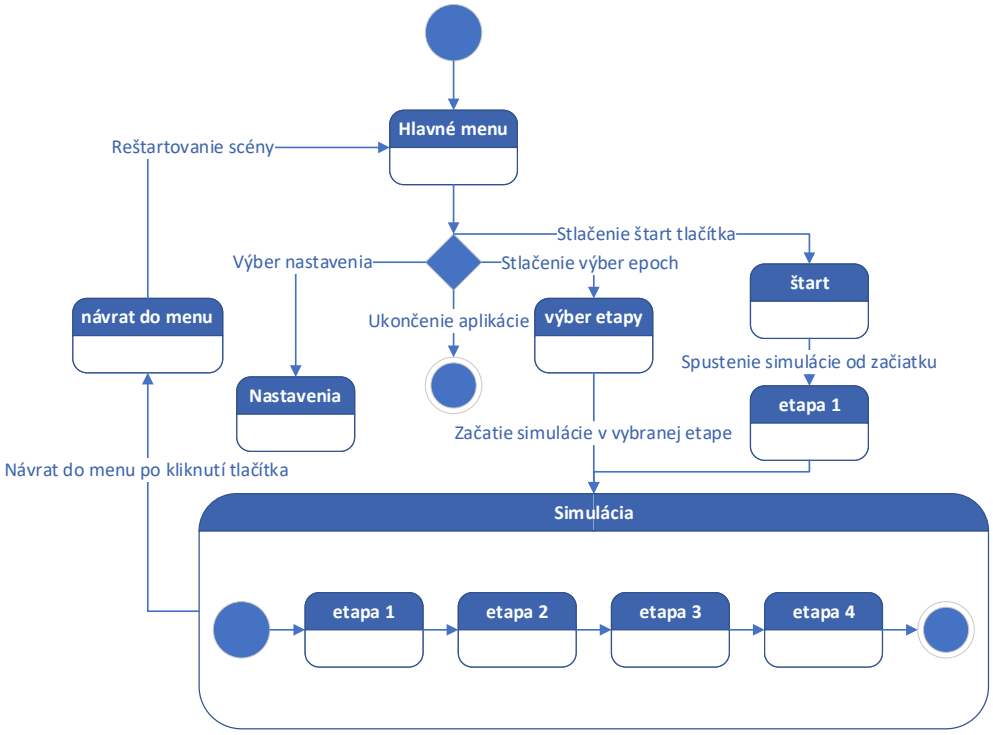
- Aplikácia má moderné a prehľadné používateľské rozhranie.
- Aplikácia funguje správne v časoch veľkej záťaže.
- UI by malo byť intuitívne a ľahko použiteľné pre nových používateľov.

Doménové požiadavky

- Aplikácia musí byť prispôsobená na rôzne veľkosti obrazovky.

3.6.2 Konceptuálny návrh

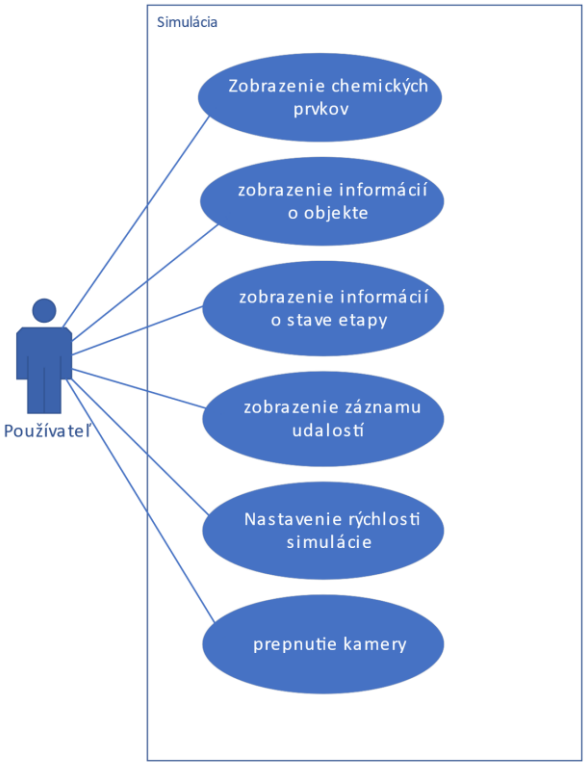
Po stanovení správnych požiadaviek, spravíme koncept aplikácie. Tento vieme navrhnúť pomocou rôznych UML diagramov [42]. Pomocou stavového diagramu navrhneme celkový chod aplikácie od jej spustenia až po vypnutie. Stanovíme si rôzne stavy, v ktorých sa používateľ môže nachádzať.



Obr. 1: Stavový diagram návrhu aplikácie

Prvý stav je hlavné menu. Používateľ by mal vedieť sa dostať do stavu nastavení, spustiť simuláciu alebo sa dostať výberu spustenia. V nastaveniach má používateľ možnosť úpravu rozličných preferencií, ako napríklad, nastavenie zvuku, hudby, rozlíšenia, levelu detailu. Z hlavného menu sa používateľ vie dostať do výberu etáp. V tomto stave má na výber 4 etapy, začiatok, skorá slnečná sústava, súčasnosť, budúcnosť. Zároveň z každého stavu, ktorý je dostupný z hlavného menu, sa dá dostať späť do hlavného menu. Najdôležitejší stav je stav simulácie. Tento stav sa delí na 4 podstavy. Každý podstav je etapa vo vývoji slnečnej sústavy. Každý stav etapy nadväzuje pritom na jeho ďalší, pričom sa v čase nedá vrátiť do predošlého stavu. Zo simulácie sa taktiež dá dostať do hlavného menu.

Ako ďalšie si spravíme diagram prípadov použitia pre používateľa. S týmto návrhom si určíme čo by mal používateľ v simulácií byť schopný spraviť. Pri tomto návrhu opäť komunikujeme so vzorkou používateľov a upravujeme diagram podľa ich nápadov a požiadaviek.



Obr. 2: Diagram použitia simulácie

Výsledný návrh je vidieť na obrázku. Používateľ má počas simulácie možností: zobraziť záznam udalostí, informácie o objekte a možnosť prepnutie kamery.

Konceptuálny návrh sme s týmto krokom dokončili. Z tohto konceptu si vieme vydedukovať čo používateľ očakáva od aplikácií simulujúcej vznik a zánik.

3.6.3 Očakávania od aplikácie

Od výsledného stavu aplikácie očakávame, že bude spustiteľná a stabilná. Dôraz sa má hlavne klásť na UX a informovanosť používateľa. Aplikácia by mala disponovať moderným grafickým rozhraním, pričom by nemalo byť nadmerne komplikované a neprehľadné, predo všetkým musí byť responzívne. Vstupy by mali byť pre používateľa prirodzené a nesmie byť veľmi oneskorené tzv. špongiové.

3.6.4 Rozkúskovanie vývoju

Vývoj si rozdelíme na niekoľko kúskov/etáp. Toto rozdelenie je veľmi podstatné nakoľko nám to vie výrazne sprehľadniť vývoj a vyvarovať sa zauzlenia pri neskoršom vývoji.

Etapy :

1. Počiatočné nastavenie projektu
2. Rozdelenie časovej osi
3. Rozdelenie funkcionality na časti
4. Návrh GUI
5. Funkcionalita simulácie
6. Doladenie výslednej simulácie

V etapách 2 až 6 míľnikové kroky vo vývoji aplikácií neustále testujeme a konzultujeme s používateľmi garantujeme tým, že výsledná aplikácia je presne podľa predstáv používateľa.

3.6.5 Nastavenie projektu

V prvej etape si správne nastavíme prostredie Unity a integráciu VS Code. V Unity si pomocou Shell skriptu vytvoríme základné priečinky na rozdelenie. Kód (ďalej len skripty) si nastavíme, aby sa ukladali do priečinku _Scripts, pričom podtržník pred S sme doplnili čisto z pohodlia, aby sa nám tento priečinok v prehľadávači ukazoval na vrchu. Animation slúži na ukladanie animácií, Prefabs na uloženie prefabrikátov, Resources obsahuje skriptovacie objekty, tieto slúžia ako dátové kontajnery, Sprites slúži na ukladanie textúr a súbor Sound na zvuk. Tieto priečinky sú základné pre každý projekt, neskôr vo vývoji sme si vytvoríme ďalšie súbory, ktoré sú špecifické pre našu aplikáciu.

3.6.6 Rozdelenie časovej osi

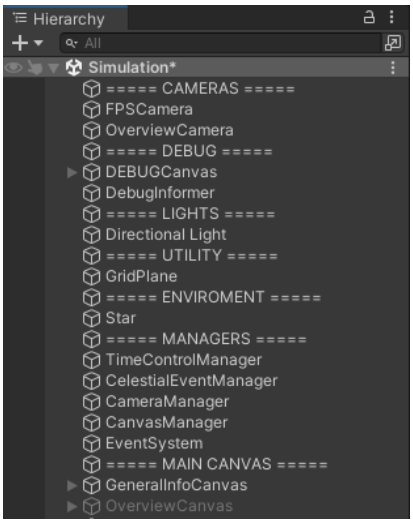
Po nastavení projektu nahliadneme do hlavného cieľa aplikácie zobraziť celú časovú líniu od začiatku po koniec našej slnečnej sústavy. Z dostupnej literatúry zistíme, že slnečná sústava je stará cca. 4,6 miliárd rokov [43], a taktiež že definitívny koniec nášho slnka nastane za približne 1 biliardu rokov (10^{15} pre lepšiu predstavu) [44]. Z týchto dostupných informácií sme sa rozhodneme, že prvý možný štart bude začiatok hneď pri tvorbe

a posledný počas transformácie červeného obra na bieleho trpaslíka, keďže v tomto štádiu sa už moc zaujímavého nedialo, urýchlime simuláciu času tak, aby schladenie bieleho trpaslíka na čierneho [44] netrvalo pridlho. Ako tretí možný štart zvolíme súčasnosť. Posledný štvrtý štart vyberieme opäť podľa odbornej literatúry, obdobie migrácie je primárne zaujímavé zmenami obežných dráh našich planét a migráciou na ich dnešné polohy [43]. Výsledná dejová línia vyzerá s našimi rozhodnutiami nasledovne:

Obr. 3: Časová os simulácie (nie v mierke)

3.6.7 Rozdelenie funkcionality

Rozdelením funkcionality na menšie bloky zaručíme nezávislosť skriptov medzi sebou. Takto zabezpečíme, že napr. ladiace skripty sú nezávislé od ostatných skriptov a nemajú žiadne priame referencie na iné skripty. Spravíme si niekoľko blokov, napríklad skripty zodpovedné za GUI majú priame referencie iba medzi inými skriptami na GUI, to platí aj pre kód ovplyvňujúci čas, fyziku, pohyb a pod. V kapitoly „Implementácia“ a jej podkapitol sa budeme viacej zaoberať týmto rozdelením a ako využívať jeho benefity v náš prospech. Podobné rozdelenie spravíme aj v hierarchickom prieskumníku v Unity. To dosiahneme vytvorením prázdnym objektov v scéne a ich vhodným pomenovaním ako je vidno na obrázku.

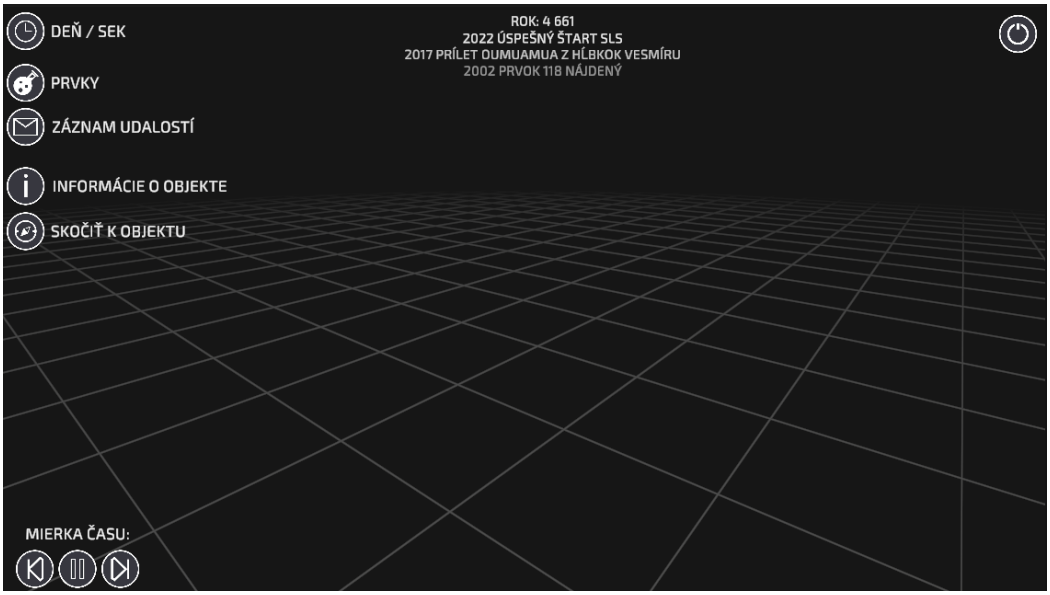


Obr. 4: Blokové rozdelenie v scéne

Časom ako sa aplikácia vyvíja pribúdajú objekty do scény, ktoré rýchlo napĺňajú priestor v prieskumníku. Spravením tzv. blokových objektov si nastavíme ostatné objekty relevantné k názvu blokového objektu ako „deti“ tohto objektu, tým sa nám uvoľní priestor a prehľad objektov v scéne sa zlepší.

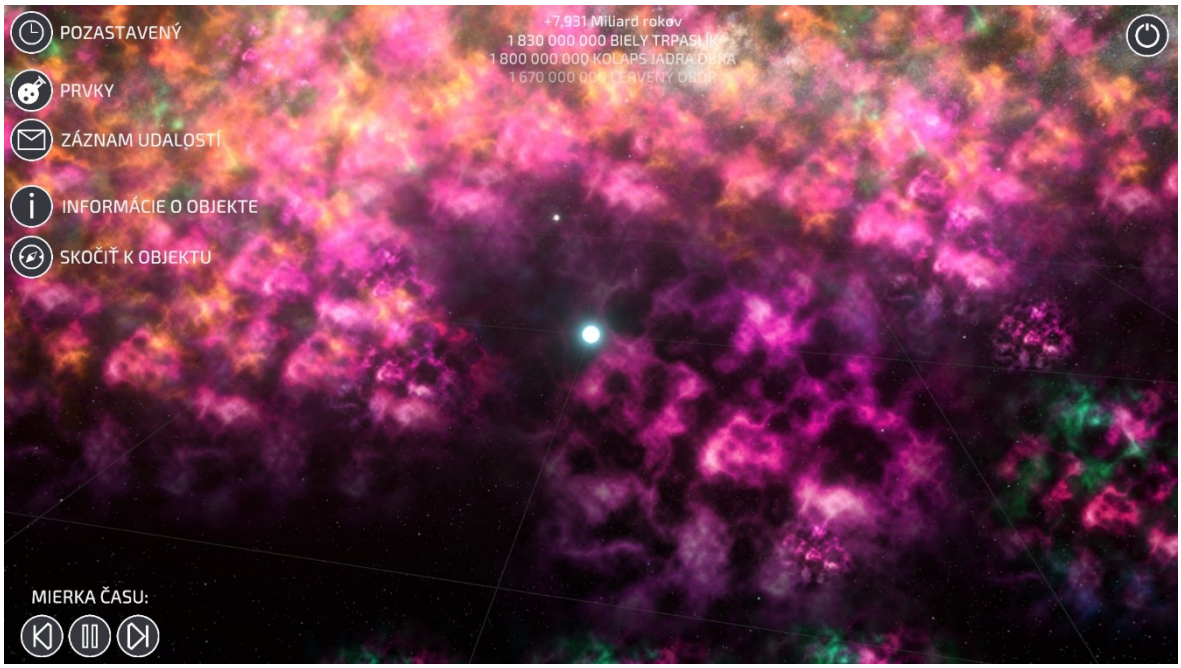
3.7 Návrh GUI

Pri dizajnovaní GUI si zase zavoláme hŕstku nami zvolených používateľov, predstavíme im počiatočný dizajn a rozloženie grafických elementov, tlačidiel a textov. Následne skúšame rôzne pozície a kombinácie rozloženia spolu s ich návrhmi. Takto sa vyvarujeme, redizajnu v neskorších fázach vývoju. Tento krok je kľúčový aj z hľadiska našej predošlej logiky pri rozdeľovaní funkcionality. Vytvorením správneho a pre používateľa akceptovateľného GUI sa tak vyvarujeme neskoršiemu návratu k tomuto vývojárskemu bloku. Dôležité je dbať na správne umiestnenie prvkov podľa ich úlohy a dôležitosti, aby sa nestalo, že prvok, ktorý má podstatnú úlohu v našej aplikácií skončí na mieste kde ho používateľský zrak nezachytí. Jeden z takýchto prvkov je práve časomiera, tento prvok sa skladá z dvoch častí, ikonky a textu, kde text sa aktualizuje podľa aktuálneho urýchlenia času napr. min / sek, pričom časť pred lomkou znamená rýchlosť simulácie a časť za lomkou čas v realite. Po komunikácií s používateľom sa rozhodneme tento grafický prvkov umiestniť do ľavého horného rohu obrazovky. Túto praktiku tak aplikujeme na všetky prvky, ktoré chceme pridať do našej aplikácie.



Obr. 5: Počiatočný GUI dizajn

Z obrázku je vidieť počiatočné schválené rozloženie GUI podľa používateľa. Ďalšie kroky sa zaoberajú návrhom prvkov po ich zakliknutí.



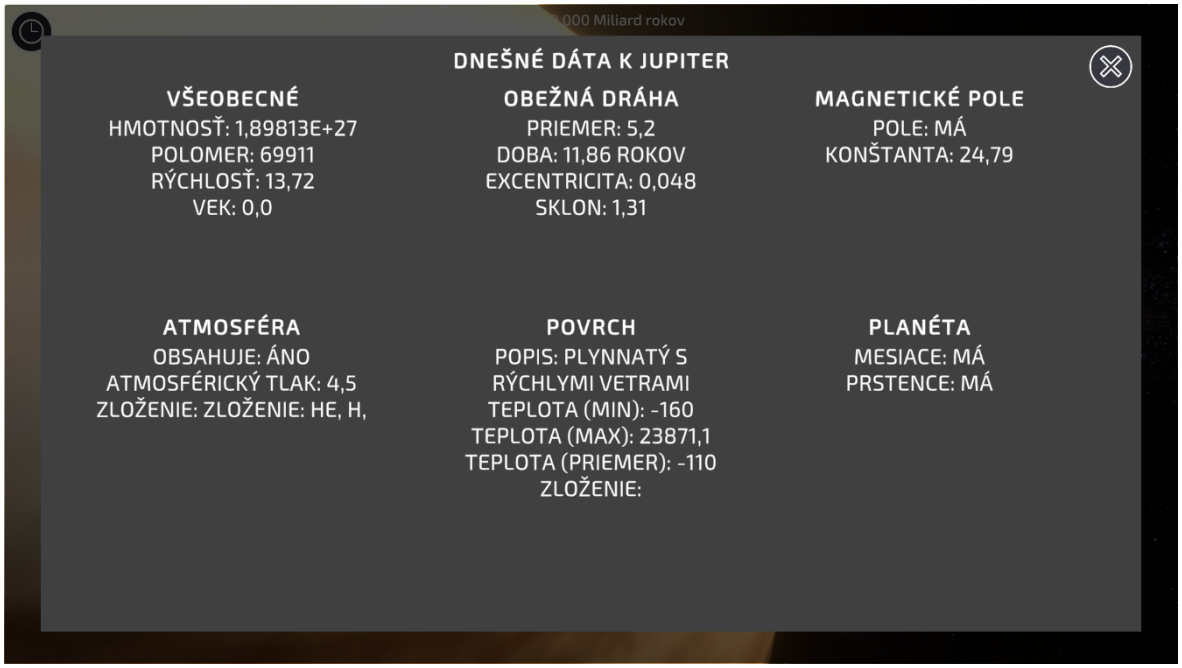
Obr. 6: GUI v konečnej podobe

3.7.1 Záznam udalostí

Záznam navrhujeme s úmyslom uchovávania všetkých udalostí, ktoré od začiatku simulácie nastali. Tieto udalosti sú rôzneho typu a bližšie sa k nim venujeme v nasledujúcich kapitolách. Udalosť sa najprv používateľovi zobrazí v hlavnom GUI hneď pod aktuálnym rokom (viď obrázok), s postupom času ako nastávajú ďalšie udalosti bude táto udalosť irelevantná a ukladá sa do záznamu udalostí, kde si používateľ vie pomocou kurzoru prejsť všetky udalosti v chronologickom poradí.

3.7.2 Informácie o objekte

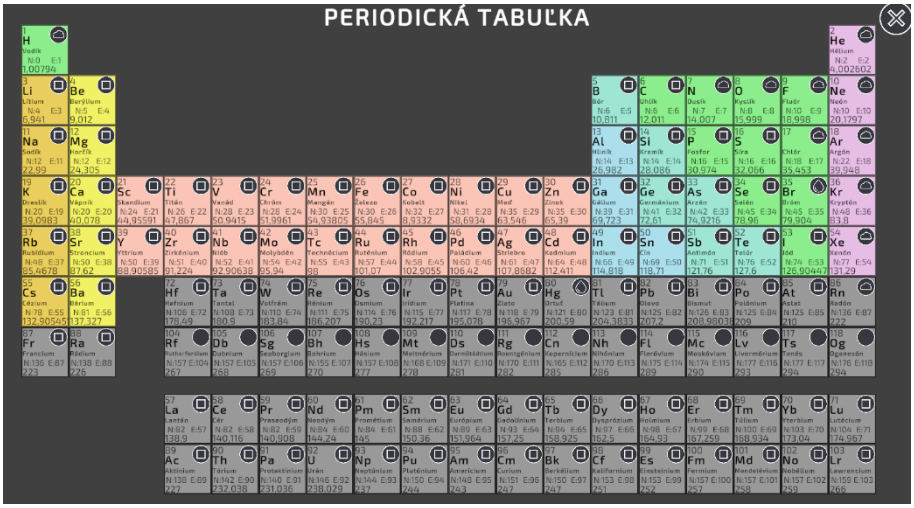
Patrí medzi najpodstatnejšie GUI, keďže používateľovi zobrazí informácie o príslušnom objekte. Toto rozhranie musíme navrhnúť tak, že používateľ dostáva iba relevantné dáta k prislúchajúcemu objektu a nenastane to, že sa zobrazia informácie, ktoré s objektom nesúvisia. Rozhranie sa skladá napokon z viacerých textových prvkov, ktoré je nutné správne umiestniť podľa relevantnosti voči simulácií.



Obr. 7: Informácie o objekte

3.7.3 Chemické prvky

Pri prvej konzultácií s používateľom bolo navrhnuté implementovať aj interaktívnu periodickú tabuľku prvkov. Prvotný návrh tabuľky vytvoríme s myšlienkou zvýrazňovať chemické prvky podľa ich vytvorenia v slnečnej sústave.



Obr. 8: Periodická tabuľka

3.7.4 Časť pre ladenie

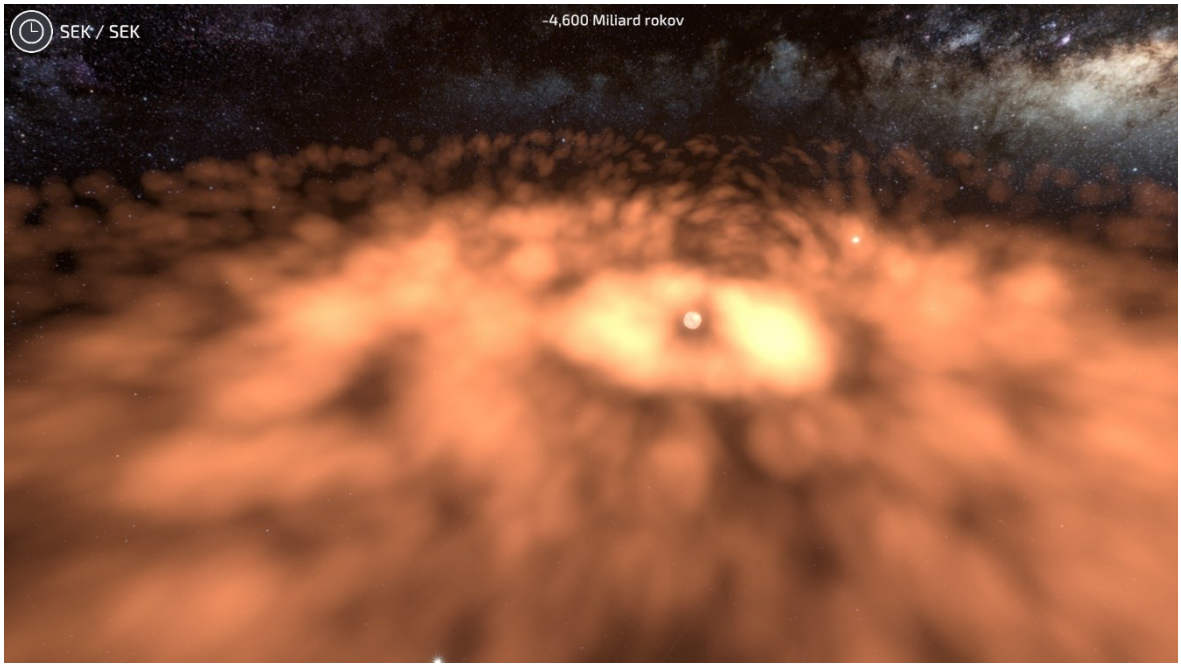
Pri testovaní nám príde vhod aj spätná väzba od programu, koľko snímok za sekundu dosahuje aplikácia pri záťaži, spotreba pamäte, počet vykresľovaných objektov, vypisovanie

chýb atď.. Jedná sa pritom o jednoduchý návrh, pričom zoberieme najmenej používanú časť plochy a dosadíme tam zopár textových elementov, ktoré následne napojíme na skript.

V tomto okamihu sme úspešne navrhli základný koncept aplikácie. Stanovili sme si základné ciele a požiadavky, ktoré má aplikácia spĺňať. Efektívne sme si rozdelili a rozkúskovali vývoj na viacero prepojených blokov, ktoré nám časovo uľahčia implementačnú časť projektu, ktorej sa budeme podrobnejšie venovať v nasledujúcej kapitole.

3.8 Implementácia

V tejto kapitole sa venujeme vývoju simulačnej aplikácie, opíšeme si praktiky a vymoženosti herného engine Unity, ktorý použijeme na vývoj a priblížime si s akými komplikáciami si musíme dať rady.



Obr. 9: Prvá etapa: Proto Slnko

3.8.1 Hlavné menu

Hlavné menu tvorí podstatnú časť každej aplikácie, preto nesmie chýbať ani tu. Z návrhu vieme, že bude pozostávať z viacerých sekcií. Implementácia hlavného menu je celkom priamočiara. Vytvoríme si novú prázdnu scénu v Unity. Do nej ako prvé pridáme „Canvas“. „Canvas“ v Unity je kontajner pre GUI elementy, ktorý slúži na ich vykreslenie [45]. V ňom pridáme šesť textových elementov: meno aplikácii, štart, etapy, nastavenia, vypnúť a meno

autora. Pre texty štart, etapy, nastavenia, vypnúť pridáme „Button“ komponent, ktorý pridá textu funkcionalitu tlačidla. Posledný krok spočíva v naprogramovaní funkcionality hlavného menu, ktoré je pomerne jednoduché vďaka funkciám, ktoré nám poskytuje Unity. Pre vytvorenie sekcie etapy a nastavenia si zoberieme „Canvas“ hlavného menu a duplikujeme ho, premenujeme text, prehodíme logiku tlačidiel, aby sme sa mohli prepínať medzi všetkými sekciami. V sekcií nastavenia pridáme dva posuvníky na zvuk a hudbu, zaškrtávacie políčko pre zobrazenie na celú obrazovku a rozbaľovací zoznam pre rozlíšenie. Tlačidlo vypnúť premenujeme na „späť“ a prehodíme funkcionalitu, aby po kliknutí sa vrátime na začiatočnú ponuku. Výhody, ktoré nám Unity poskytuje si ukážeme na útržkoch skript. Napríklad také vypnutie Unity aplikácie vieme naprogramovať takto:

```
public void Exit()
{
    StartCoroutine(WaitForAnimationDisable(null));
    Application.Quit();
    Debug.Log("Quit");
}
```

Výpis 1: Ukážka funkcie na vypnutie aplikácie

Tvorenie kódu pre nastavenie aplikácie je taktiež priamočiare a jednoduché vďaka poskytnutým funkciám od Unity. Jediná časť, pri ktorej si musíme pomôcť, je získanie všetkých dostupných rozlíšení, ktoré máme na výber.

```
private void Start()
{
    resolutions = Screen.resolutions;
    resolutionDropdown.ClearOptions();
    List<string> options = new();
    int resIndex = 0;
    for (int i = 0; i < resolutions.Length; i++)
    {
        string option = resolutions[i].width + " * " + resolutions[i].height
                        + " " + resolutions[i].refreshRate + "hz";
        options.Add(option);
        if (resolutions[i].width == Screen.currentResolution.width &&
            resolutions[i].height == Screen.currentResolution.height)
                resIndex = i;
    }
    resolutionDropdown.AddOptions(options);
    resolutionDropdown.value = resIndex;
```

```
    resolutionDropdown.RefreshShownValue();
}
```

Výpis 2: Získanie a pridanie rozlíšení do rozbaľovacieho zoznamu [46]

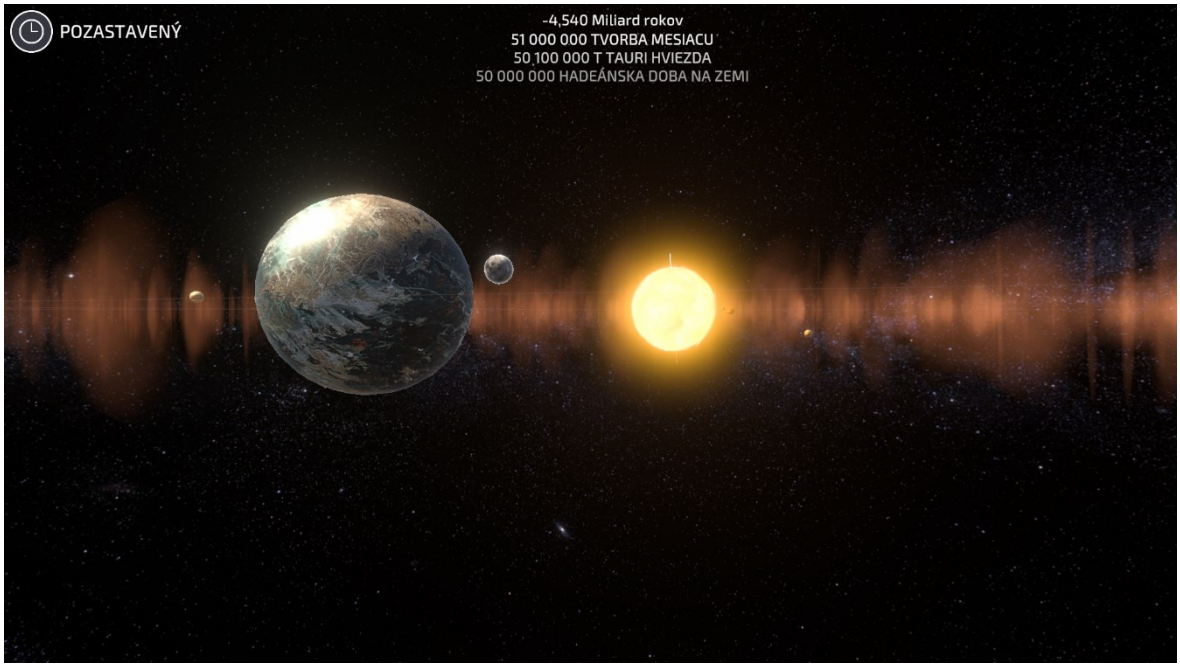
Pre sekciu voľba etapy taktiež pozmeníme texty a pridáme jednoduchý skript na načítanie príslušnej scény, ktorú zvolíme cez Unity editor.

Obr. 10: Hlavné menu

3.8.2 Kamerový systém a pohyb

Z analýzy konkurenčných aplikácií vieme, že najdôležitejším krokom je navrhnúť ideálny kamerový systém, taký ktorý bude používateľovi prirodzený. V konceptuálnom návrhu sme si stanovili, že používateľ bude mať možnosť prepínať kameru. Najvhodnejší spôsob bude kombinovať perspektívu kamery. Prvou možnosťou bude kamera z prvej osoby. Používateľ sa bude pohybovať pomocou kláves „WASD“, a otáčať myšou. K tomu mu pridáme ešte možnosť zrýchlenia pohybu počas držania klávesy „Shift“ (možnosť šprintovania). Musíme dbať aj nato, že mnohé GUI prvky, ktoré sme navrhli v tomto režime nebudú dostupné, nakoľko je pohyb myši už okupovaný kamerou. K riešeniu tohto problému sa dostaneme v neskoršej dedikovanej kapitole. Základný pohyb a perspektívu kamery máme spracovanú. Druhá perspektíva by bola kamera „zhora“. V tomto stave by kamera bola umiestnená vo vyvýšenej polohe nad objektom, na ktorý bude ukazovať. Pohyb v tejto perspektíve nemusíme komplexne riešiť, nakoľko jednoduchým riešením bude pohyb skokmi kamerou zameraných objektov. Toto vieme docieliť kliknutím myši tak, že sa nám kamera presunie

bližšie ku kliknutému objektu a bude naň ukazovať. V tejto perspektíve vieme uplatniť aj naše navrhnuté GUI, nakoľko vieme kliknutie myši využívať na stlačenie tlačidiel, ktoré nám zobrazia navrhnuté sekcie GUI. Prepínanie kamier vyriešime jednoduchým stlačením ľubovoľne zvoleného tlačidla na klávesnici, bude vhodné túto klávesovú skratku v neskorej fáze vývoja aj niekde umiestniť, aby o nej používateľ vedel.



Obr. 11: Pohľad z prvej osoby

```
public void MovePlayer()
{
    float targetSpeed = Input.GetKey(KeyCode.LeftShift)
                        ? PlayerSprintSpeed : PlayerSpeed;
    CurrentSpeed = Mathf.MoveTowards(CurrentSpeed, targetSpeed,
                                     Acceleration * Time.deltaTime);

    float horizontalInput = Input.GetAxis("Horizontal");
    float verticalInput = Input.GetAxis("Vertical");

    Vector3 movement = CurrentSpeed * Time.deltaTime *
                       new Vector3(horizontalInput, 0f, verticalInput);
    transform.Translate(movement);
}
```

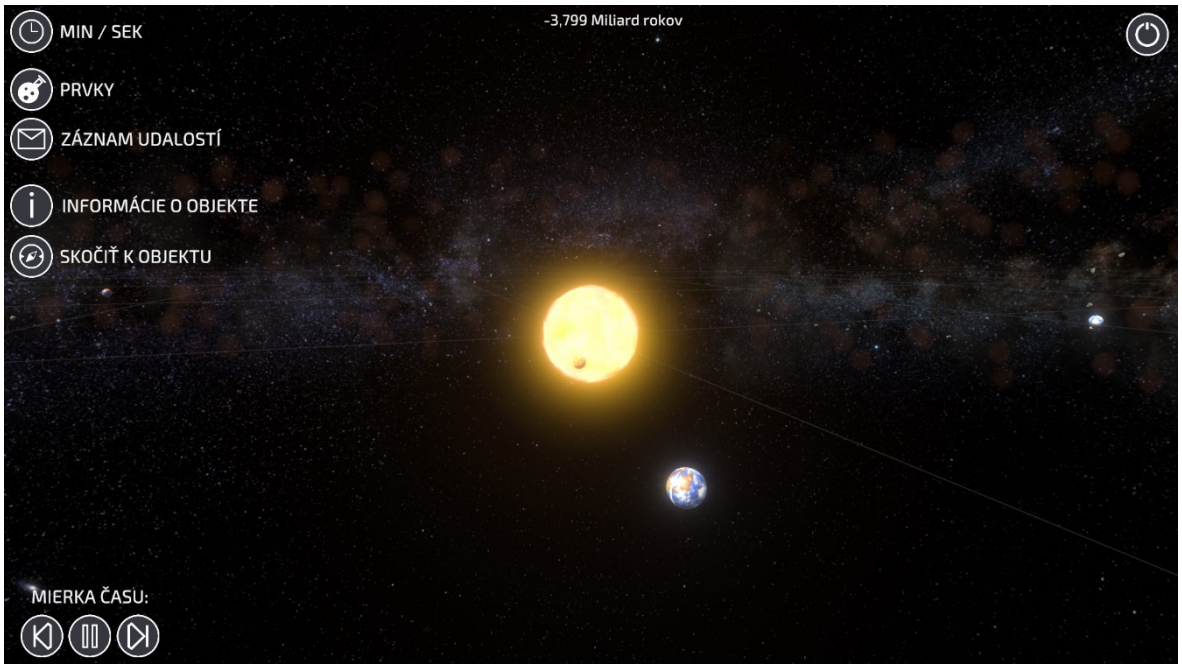
Výpis 3: Aktualizácia polohy hráča z prvej osoby

Hore uvedená funkcia berie vstup používateľa z klávesnice, pričom funkcie `Input.GetAxis()` menia klávesový vstup na hodnotu vektoru z hodnotami od <-1,1>. Tieto hodnoty následne vynásobíme rýchlosťou a časovou konštantou. Týmto dostaneme vektor, ktorý odovzdáme funkcií `transform.Translate(movement)`, ktorým nám premení tento vektor na pohyb objektu.

```
public void UpdateCameraAngles()
{
    float axisX = Input.GetAxisRaw("Mouse X");
    float axisY = Input.GetAxisRaw("Mouse Y");
    if (Input.GetMouseButton(0))
    {
        CamController.OrbitalAngle += axisX * DragSpeed / 100;
        CamController.ElevationAngle -= axisY * DragSpeed / 100;
    }
}
```

Výpis 4: Výpočet uhlu alternatívnej kamery

Uvedenú funkciu využíva druhý kamerový skript, kde kamera je nad objektom, na ktorý ukazuje. Funkcia spravuje vstup polohy kurzora myši, počas držania ľavého tlačidla, potom sa hodnota os aplikuje na uhly kamery.



Obr. 12: Pohľad kamery "zvrchu"

Funkcionality perspektív kamier rozdelíme do dvoch samostatných skriptov, následne vytvoríme jeden manažér, kde uvedieme referencie na tieto skripty, v manažéri

doprogramujeme logiku volania funkcií, spracujeme podmienky, kedy ktoré funkcie skriptov využívať. Najdôležitejším dizajnom tohto manažéra, ale aj ostatných je, že existuje v scéne iba jedna inštancia tzv. „singleton“. Tento návrhový vzor pri programovaní ďalších skriptov využijeme, aby sme nevytvárali referencie na tento objekt v každom skripte, ale mohli nastavovať jeho vlastnosti napriamo. V Unity scéne vytvoríme ďalšiu kameru nastavíme jej tag „SecondaryCamera“ a pridáme jej skript vrchnej kamery, na hlavnú kameru hodíme skript na pohyb z prvej osoby. Vytvoríme si ešte prázdny objekt, ktorý slúži ako manažér, jemu pridáme skript kamerového manažéra. Týmto sme vytvorili jednoduché ovládanie a pohyb pre používateľa.

3.8.3 Scriptable objekty

Dobrým a kvalitným spôsobom ako v Unity pracovať s veľkým množstvom dát, je pomocou tzv. „Scriptable Object“ [47] tried. V našom prípade ich použijeme prevažne na vytvorenie dátových kontajnerov. Výhodou je, že „Scriptable Object“ sa vytvára ako „asset“, teda je nezávislý od scény a jeho dáta sú v každej scéne rovnaké [47]. Táto vlastnosť je zároveň aj nevýhodou, keď chceme dáta meniť alebo ich modifikovať. Povedzme, že Slnku priradíme takýto dátový kontajner pre jeho fyzikálne vlastnosti a chceme ich časom meniť. Zmeny zostanú uložené aj po ukončení behu aplikácie, teda napríklad po návrate do hlavného menu. Tento nedostatok vieme vyriešiť tým, že vytvoríme inštanciu tejto triedy, pomocou ktorej vieme modifikovať dáta, ktoré nechceme aby sa zapísali do kontajneru.

```
public GenericCOData CreateGenericCODataInstance(GenericCOData data)
{
    GenericCOData newData = ScriptableObject.CreateInstance<GenericCOData>();
    newData.Mass = data.Mass;
    newData.Radius = data.Radius;
    newData.Velocity = data.Velocity;
    newData.Age = data.Age;
    newData.ObjectName  = data.ObjectName;
    return newData;
}
```

Výpis 5: Vytvorenie inštancie pre všeobecné dáta telesa

Hore uvedená metóda vytvára novú inštanciu pre všeobecné vlastnosti vesmírneho telesa, inštanciu vytvárame preto, že v priebehu simulácie sa tieto dáta časom menia, napr. „Age“ časom narastá a nebude konštantný. Týmto štýlom vytvárame väčšinu dátových premenných tried ako „Scriptable Object“, aby sme sprehľadnili kód. Referenciu na takýto dátový kontajner vieme v triede vytvoriť nasledovne:

```
public class CelestialObject : MonoBehaviour

{
    [SerializeField] private GenericCOData celestialObjectData;

}
```

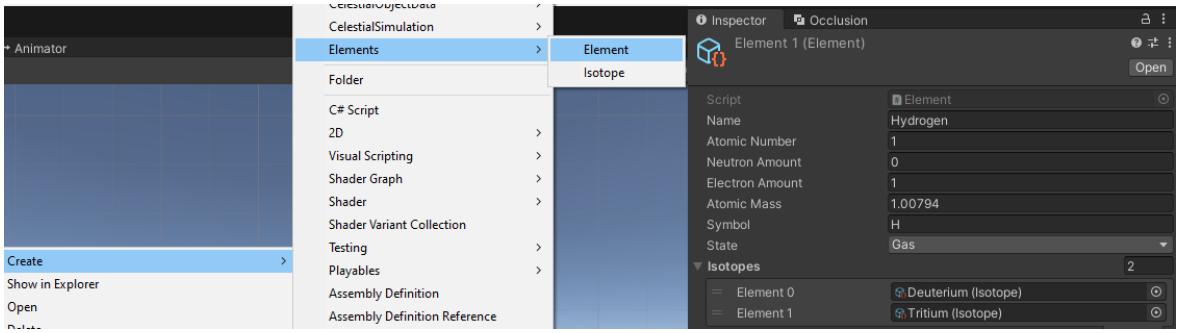
Výpis 6: Vytvorenie referencie na dátový kontajner

Dátové kontajnery máme naprogramované a využívame ich na sprehľadnenie kódu. Teraz si ich vytvoríme v prostredí Unity, doplníme ich hodnoty a napojíme na skripty, ktoré ich vyžadujú. Do skriptov, ktorých objekty sa vytvoria počas simulácie, vieme doplniť metódu, ktorá nám prehľadá zadefinovaný priečinok a získa tak referenciu na dáta, ktoré vyžaduje. Predtým ako pôjdeme vytvoriť „Scriptable Object“ ako „asset“ si musíme do skriptu doplniť ešte dva podstatné riadky kódu.

```
[System.Serializable]
[CreateAssetMenu(fileName = "New Element", menuName = "Elements/Element")]
public class Element : ScriptableObject
{
    [field: SerializeField] public string Name { get; private set; }
    [field: SerializeField] public int AtomicNumber { get; private set; }
    ....
}
```

Výpis 7: Štruktúra kódu Chemického prvku

Nad definíciou triedy Element doplníme anotáciu `[System.Serializable]`, ktorá nám skonvertuje triedu tak aby Unity vedel rozpoznať jej premenné a `[CreateAssetMenu]`, s ktorou definujeme pod akou ponukou chceme mať tlačidlo na vytvorenie nového „assetu“ a čo bude jeho predvolené meno pri vytvorení.



Obr. 13: Vytvorenie Element „Scriptable Object“

3.8.4 Grafické používateľské rozhranie

Implementácia nášho GUI je pomerne jednoduchá, avšak časovo náročnejšia. Začneme vytvorením viacerých objektov typu „Canvas“. Vhodné je vždy si rozložiť, aby jeden „Canvas“ obsahoval prvky iba pre jednu funkcionalitu. Takto napr. vytvoríme „Canvas“ pre menu pozastavenia. Doň budeme vkladať iba GUI prvky súvisiace s týmto menu, teda pozadie a zopár tlačidiel. Ďalej si rozdelíme celý návrh grafického rozhrania do blokov. Zvolený postup nám uľahčí prehľadnosť počas implementácie a zrýchli vykresľovanie počas behu aplikácie.

Obr. 14: Rozdelenie GUI

Na obrázku je vidieť výsledné rozdelenie celého GUI do viacerých „Canvasov“. Vytvoríme manažér pre GUI a naprogramujeme prvú funkcionalitu prepínania medzi nimi. Z manažéra spravíme „singleton“ a vytvoríme referencie na skripty. Každý „Canvas“ má svoj vlastný skript, do ktorého vieme pristúpiť cez referenciu pomocou manažéra. Najdôležitejšia časť kódu týchto elementov je ich správne prepínanie stavu.

```
public void ToggleUI()
{
    celestialObjectInfoCanvas.gameObject.SetActive
                    (!celestialObjectInfoCanvas.gameObject.activeInHierarchy);
```

```
    if (celestialObjectInfoCanvas.gameObject.activeInHierarchy)
    {
        PlayerController.Instance.InMenu = true;
        PlayerController.Instance.InSubMenuOpen = true;
    }
    else
    {
        PlayerController.Instance.InMenu = false;
        PlayerController.Instance.InSubMenuOpen = false;
    }
}
```

Výpis 8: Prepínanie viditeľnosti GUI

Hore uvedená metóda implementuje prepínanie viditeľnosti GUI pre používateľa v oboch režimoch kamery. Taktiež nastavíme pomocou jedinej inštancie hráčskeho manažéra, aby sme zamedzili pohyb hráča ak je dané menu zapnuté.

Väčšina ponúk, ktorých vytvárame je statická, teda nemenia sa počas priebehu simulácie, tieto ponuky vytvárame cez Unity editor, pridávame texty a pridáme tlačidlo na zatvorenie ponuky. Niektoré ponuky sa dynamicky upravujú počas simulácie sú to záznam udalostí a informácie o vesmírnych objektoch, v menšom množstve aj periodická tabuľka prvkov. Aby sme zabránili priamym referenciám, ponuky upravujeme pomocou „Unity Events“ alebo „C# Events“. Tieto „Eventy“ sú jedinečné vlastnosťou, že ak objekt metódy, ktorá je volaná cez „Event“ neexistuje, tak nenastane výnimka. Druhou výbornou vlastnosťou je zamedzenie priamych referencií na objekty, čím zjednodušíme prehľadnosť skriptov [48].

```
public void SetExistenceOfElement(string elementName)
{
    if (elementName == elementData.Name)
    {
        existsInSolarSystem = true;
        SetColors(bgColor);
    }
}
```

Výpis 9: Metóda aktualizáciu výskytu prvku

Horeuvedená metóda je napojená na prvok GUI v periodickej tabuľke prvkov. Ak sa prvok ešte nevyskytuje v slnečnej sústave, jeho ikonka je čiernobiela. Prvý výskyt prvku zavolá

túto funkciu, čo jej nastaví farebné pozadie. Túto metódu voláme cez „Event“, definovaný v manažéri udalostí. Volanie tejto funkcie vyzerá nasledovne:

```
[SerializeField] private UnityEvent<string> onElementCreation;
private void Start()
{
    List<UIElement> elements = FindObjectsOfType<UIElement>(true).ToList();
    foreach (UIElement element in elements)
    {
        onElementCreation?.AddListener(element.SetExistenceOfElement);
    }
    UpdateEventDisplay();
    UpdateFullEventLog();
}
```

Výpis 10: Pridanie „Event Listener“ na metódu

Manažér si pri inicializácií nájde v scéne všetky objekty typu „UIElement“ pridá ich do dátovej štruktúry List a pomocou „foreach“ cyklu pridá do „Unity Eventu“ poslucháča na metódu vyfarbenia pozadia elementu.

Obdobným spôsobom implementujeme aj „Eventy“ pre zobrazenie informácie o objekte, tu použijeme „Event“, ktorý sa zavolá ak používateľ sa nachádza v určenej blízkosti objektu. Ak sa používateľ nachádza v druhej kamerovej perspektívy vieme, tento „Event“ priradiť na kliknutie myši.

3.8.5 Časomiera

Ďalším neoddeliteľným prvkom, ktorý treba vytvoriť je urýchlenie času. Na tento účel vieme použiť zabudovanú časomieru od Unity. Po krátkej implementácií však zistíme, že toto urýchlenie nebude dostačujúce nakoľko maximálna hodnota je iba 100-násobok. Preto si navrhneme vlastnú časomieru. Ako prvé je potrebné si určiť v akých krokoch budeme môcť upravovať rýchlosť času. Po tomto určení budeme čas reprezentovať ako v realite. Vytvoríme si ďalší manažér a vytvoríme mu niekoľko premenných, metód na modifikáciu rýchlosti kroku a napokon otestujeme našu implementáciu.

```
private void CalculateTime()
{
    ElapsedTime += Time.smoothDeltaTime * StellarTimeScale;
    // Calculate years and remaining seconds
    Long years = ElapsedTime / 31536000;
```

```
    YearCount = (long)Math.Floor(years);
    double remainingSeconds = ElapsedTime - (double)(YearCount * 31536000);
    // Update other time units
    dayCount = (long)(remainingSeconds / 86400) % 365;
    hourCount = (long)(remainingSeconds / 3600) % 24;
    minuteCount = (long)(remainingSeconds / 60) % 60;
    secondCount = (long)remainingSeconds % 60;
}
```

Výpis 11: Výpočet času

Vyššie uvedený kód vykoná prepočet uplynutého času od začatia simulácie na sekundy, minúty, hodiny, dni a roky. `Time.smoothDeltaTime` je časová jednotka poskytnutá od Unity medzi posledným a aktuálnym vykresleným snímkam [49]. Vynásobením touto konštantou dostaneme vyhladené plynutie času. Funkcionalitu zmeny aktuálneho kroku prepojíme s GUI tlačidlami. Následne túto funkcionalitu musíme dostatočne otestovať, keďže ide o srdce našej simulácie. Avšak tu zistíme, že pri vysokých číslach nám nastane chyba. Chyba sa prejaví v prepočte uplynulého času na sekundy, minúty atď.. Po ladení zistíme, že dátový typ `double` v tomto prípade stráca svoju presnosť pri kalkulácii desatinných miest. Tento nedostatok vieme upraviť modifikáciou premenných, využívajúcich dátové typy `double`. Pri ďalšom testovaní si uvedomíme, že ani dátový typ `long` nebude dostatočný. Napravenie tohto nedostatku dosiahneme vymenením dátových typov za `BigInteger` a `decimal`. Tieto poskytuje knižnica `System.Numerics` a `System`. Nedostatkom týchto dátových typov je, že Unity Editor nepodporuje zobrazenie ich hodnôt v editore. Otestujeme modifikovanú verziu našej časomiery. Po teste úspešne prekročíme hranicu 10^{15} rokov, spomalíme čas a otestujeme či sa správne vypočítavajú sekundy a ďalšie jednotky. Týmto sme úspešne navrhli vlastnú časomieru. Ovplyvniť rýchlosť ostatných objektov vieme pomocou globálnej premennej `StellarTimeScale`, ktorú sme si zadefinovali v manažérovi.

```
public static MainTimeController Instance { get; private set; }

public BigInteger StellarTimeScale { get; set; } = 1;

public BigInteger YearCount { get; private set; } = 0;

public decimal ElapsedTime { get; private set; } = 0;

private void CalculateTime()
{
    ElapsedTime += (decimal)Time.smoothDeltaTime * (decimal)StellarTimeScale;
    decimal years = ElapsedTime / 31536000;
```

```
    YearCount = (long)Math.Floor(years);
    decimal remainingSeconds = ElapsedTime - (decimal)(YearCount * 31536000);
    dayCount = (long)(remainingSeconds / 86400) % 365;
    hourCount = (long)(remainingSeconds / 3600) % 24;
    minuteCount = (long)(remainingSeconds / 60) % 60;
    secondCount = (long)remainingSeconds % 60;
}
```

Výpis 12: Finálna implementácia časomiery

Obr. 15: Správny výpočet časových jednotiek

3.8.6 Manažér udalostí

Posledným robustným systémom pre informovanosť používateľa je vytvorenie manažéra na zobrazenie udalostí. Tieto udalosti budú rôzneho typu, primárnym cieľom bude informovať čo sa práve deje. Keďže naše rozpätie času je obsiahle, budeme pre jednoduchosť zobrazovať udalosti vždy na začiatku roku, v ktorom sa nastali. Najprv si vytvoríme novú triedu ako štruktúru pre udalosť, bude obsahovať kľúčové slovo, popis, rok, a typ. Typ udalosti si zadeklarujeme ako enum a zadefinujeme si typy udalostí.

```
[System.Serializable]
public enum CelestialEventType { SocietyEvent, ElementCreationEvent, PlanetEvent,
AsteroidEvent, StarEvent, OuterSpaceEvent, MoonEvent };
[System.Serializable]
public class CelestialEvent
{
    [field: SerializeField] public long Year { get; set; }
    [field: SerializeField] public CelestialEventType EventType { get; set; }
    [field: SerializeField] public string Keyword { get; set; }
```

```
    [field: SerializeField] public string Description { get; set; }
}
```

Výpis 13: Štruktúra triedy pre udalosti

Ako ďalšie si vytvoríme „Scriptable Object“, ktorý bude obsahovať zoznam objektov typu `CelestialEvent`. Vytvoríme ešte manažéra udalostí, štruktúra je podobná ako priuž existujúcich manažéroch. Tento manažér obsahuje viacero metód, pričom väčšina z nich slúži na zapisovanie udalostí do záznamu, iné filtrujú prichádzajúce udalosti a volajú pomocou „Eventov“ metódy objektov v scéne. Manažér je vytvorený tak, že je ľahko doplniteľný, ak chceme detailnejšie zápisy o udalostiach.

3.8.7 Rozdelenie vesmírnych objektov

Simulácia slnečnej sústavy bez planét, slnka a asteroidov nedáva zmysel. Ďalším krokom je základná implementácia generického vesmírneho telesa, od ktorého sa vieme odvíjať pri tvorbe ostatných špecifických telies ako sú planéty, mesiace, asteroidy a slnko. Pri tvorbe generického objektu si musíme však uvedomiť jeho veľkú podstatu, ktorou je zjednodušiť tvorbu ostatných telies. V programovaní sa tento návrhový vzor nazýva dedičnosť a polymorfizmus. Unity ponúka ešte alternatívu pomocou, ktorej vieme zadefinovať, že skript na jeho bezchybnú funkčnosť potrebuje, aby mal objekt komponent špecifikovaného typu. Oba prístupy majú svoje výhody, my si zvolíme prístup, ktorý ponúka Unity. Táto voľba bola primárne zvolená pre prehľadnosť kódu a atribútov tried, pri nastavovaní a ladení cez Unity editor.

Prvým krokom je teda naprogramovať základnú štruktúru generického vesmírneho objektu. Zadeklarujeme si aby skript potreboval komponent „SphereCollider“ a tento komponent využijeme na funkcionalitu zobrazenia informácií o objekte, keď bude používateľ v dostatočnej blízkosti, ktorú si vieme zadefinovať. Aby sme mali vek objektu synchronizovaný s celkovým časom, vytvoríme si obdobný skript na časomieru, s rozdielom, že čas sa inicializuje pri tvorbe objektu. Pridáme preto ako povinný komponent „LifeSpanController“. Takto nám Unity Editor zabráni odstránenie týchto dvoch komponentov z objektu nakoľko náš skript „CelestialObject“ je závislý od týchto dvoch komponentov. Náš generický skript bude taktiež obsahovať metódy na pohyb po zadefinovanej osi okolo bodu v scéne, pohyb okolo vlastnej osi a logiku pre kolízie.

Dátatové premenné, ktoré bude objekt obsahovať vytvoríme cez „Scriptable Objecty“ a zadeklarujeme na kontajnery referencie, pričom tu vytvoríme ešte kópie dátových kontajnerov, tieto poslúžia ako dočasné úložisko, nakoľko tieto údaje sa budú počas simulácie meniť.

Posledný krok pri vytvorení základnej implementácie vesmírnych telies je vytvoriť špecifické skripty pre planéty a pod.. Každému skriptu nastavíme aby bol potreboval komponent typu „CelestialObject“, čo je náš skript pre generický vesmírny objekt. Takto vytvoríme skripty pre asteroidy, mesiace a Slnko.

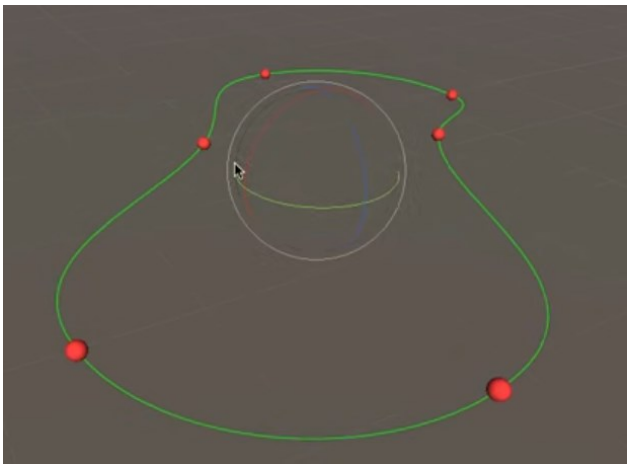
```
[RequireComponent(typeof(CelestialObject))]
public class Planet : MonoBehaviour
{
    [Header("Moons & Rings Information")]
    [SerializeField] private PlanetData planetData;
}
```

Výpis 14: Kostra triedy pre planétu

3.8.8 Pohyb telies

Na dokončenie jednoduchej simulácie je potrebné vytvoriť pohyb telies. Z odbornej literatúry vieme, že telesá v slnečnej sústave majú primárne pohyb okolo osi nejakého väčšieho objektu, táto os môže nadobúdať kruhový a oválny tvar. Tento pohyb je tvorený vzájomným gravitačným vplyvom dvoch telies. Naša implementácia však kvôli komplexnosti tohto problému bude iná. Objekty sa budú pohybovať po tzv. „cestách“, ktoré si zadefinujeme. Algoritmus na tento problém už vytvárať nemusíme. Unity ponúka riešenie cez „Splines“. V verzii 2021.3, s ktorou pracujeme táto funkcionalita ešte nie je podporovaná. Preto sa musíme spoliehať na takúto funkcionalitu od tretích strán.

Vytvorenie spomínanej cesty spočíva v zadefinovaní niekoľkých bodov, medzi ktorými následne algoritmus vytvorí cestu už za nás. Ako ďalšie vytvoríme skript na úpravu dráhy telesa.



Obr. 16: Vytvorenie cesty cez „Splines“

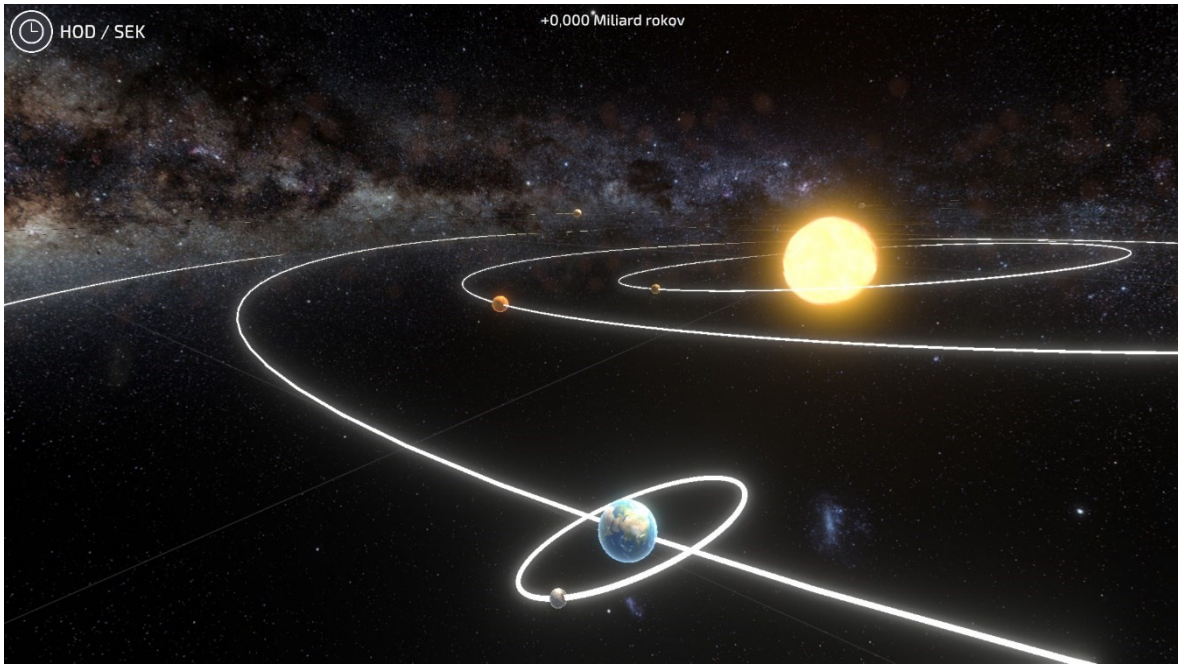
Takto vieme presnejšie modifikovať trajektóriu telies v čase. Strácame týmto síce realistickú autentickosť gravitačného vplyvu medzi telesami, ale zabránime tomu, že sa nám telesá pri rýchlom plynutí času stanú nekontrolovateľné pri meniacich sa hodnotách fyzikálnych vlastností, ktoré takýto pohyb a trajektóriu ovplyvňujú. Tento spôsob pohybu sme taktiež zvolili kvôli časovej komplexnosti problému. Štruktúra skriptov však umožňuje pomerne bezproblémové implementovanie pohybu podľa newtonského zákona, muselo by sa premodelovať menenie trajektórií v čase.

Pre nedostatok času, sa nakoniec rozhodneme pre odlišnú alternatívu, pomocou vlastného skriptu. Logika obežnej dráhy spočíva v troch súradniciach x,y,z, ktorými vieme udávať os, po ktorej sa nám bude objekt pohybovať.

```
float angle = currentTime * orbitSpeed;
float x = Mathf.Cos(angle) * xRadius;
float y = Mathf.Sin(angle) * yRadius;
float z = Mathf.Sin(angle) * zRadius;
offset = new(x, y, z);
Quaternion tiltRotation = Quaternion.Euler(tiltAngle, 0, 0);
offset = tiltRotation * offset;
```

Výpis 15: Výpočet pozície na osy

Horeuvedený kúsok kódu, nám pomocou goniometrických funkcií sínus a kosínus vypočíta presnú polohu objektu na jeho dráhe, pripočíta k nemu sklon dráhy. Takto vieme pohodlne vytvoriť dráhy pre planéty a mesiace.



Obr. 17: Zobrazenie ôs planét

3.9 Finalizácia aplikácie

Základnú štruktúru aplikácie máme vytvorenú, následne upravujeme skripty a objekty, aby bol náš softvér spoľahlivý a ponúkal dobrý UX. Pridáme efekty, „post processing“, detailnejšie objekty a upravujeme kód pre lepšiu čitateľnosť. Samozrejme nezabúdame na pravidelné testovanie s vybranými používateľmi.

4 Testovanie

Aplikáciu testujeme po každom dokončení funkčného bloku kódu. Testovanie si vieme rozdeliť na viacero častí a variant. Ako prvú vždy testujeme požadovanú funkčnosť posledného aplikovaného bloku kódu, napr. po implementácií základného GUI sa sústreďujeme na testovanie jeho funkcionality a efektívneho rozloženia. Ako vývojári vieme dobre otestovať iba jeho funkčnosť, nakoľko pri implementovaní a ladení si postupom zvykneme na rozloženie GUI elementov v našom prípade. Nezískame tým autentický zážitok a pocit, či je dizajn optimálny pre bežného používateľa. Riešením je výber používateľov, ktorým veríme, že náš produkt nezverejnia širšej verejnosti. Rozošleme im najaktuálnejší „build“ aplikácie a necháme ich testovať. Následne sa s nimi spojíme cez diskusné fórum, alebo skupinový hovor a vypočujeme si ich pocity a nápady na zlepšenie.

4.10 User acceptance testing

Najaktuálnejšiu verziu projektu rozošleme našim zvoleným testerom a nedáme im žiadne inštrukcie. Necháme ich aby si aplikáciu preskúmali a osvojili, takto je vysoká šanca na to aby sme zachytili chyby, ktoré sa dostali do verzie nedopatrením, nakoľko používateľ skúša po prvý krát aplikáciu a nevie, čo sú správne vstupy. Takýmto všeobecným testovaním vieme testovať aj funkcionalitu aj dizajn najaktuálnejšieho implementovaného bloku.

4.11 Cielené testovanie

Po všeobecnom testovaní sú testujú používatelia oboznámení s aktuálnym softvérom a vedia ako približne funguje. Cielené testovanie bude u nás spočívať v tom, že tester dostane za úlohu, za každú cenu vykonať nepovolenú akciu, vyvolať okrajový prípad, v najlepšom zhodiť celú aplikáciu jeho vstupmi. Toto testovanie vieme vykonať aj my vývojári nakoľko náš softvér poznáme najlepšie, ale môže sa stať, že neodhalíme všetky chyby, práve kvôli dobrým znalostiam o našej aplikácií. Preto tieto testovania kombinujeme s vybranými používateľmi a tým docielime vysoké pokrytie.

Tieto varianty testov skúšame najprv my, vývojári a ak sme spokojní s výsledkom nášho testovania, rozdáme „build“ aplikácie vzorke používateľov a dodáme im príslušné inštrukcie k testovaniu. Tento krok opakujeme, kým nám to časový priestor medzi blokmi umožní.

Záver

Táto bakalárska práca úspešne splnila svoj cieľ, ktorým bol návrh a implementácia aplikácie simulujúcej slnečnej sústavy od jej vzniku až po jej samostatný zánik. Aplikácia tak ponúka jedinečný interaktívny zážitok preskúmať našu slnečnú sústavu v čase. Dodatočne bol optimalizovaný aplikačný kód, aby sa umožnilo jednoduché pridávanie a vylepšovanie objektov, informácií a pod..

Prvým našim cieľom bolo spraviť prieskum dostupných technológií na tvorbu aplikácií, tvorenie 3D objektov a textúr. Hlbokou analýzou a porovnaním najpopulárnejších programov pre vývoj a dizajn sme vybrali najvhodnejšie moderné technológie na vytvorenie našej aplikácie. Tieto technológie boli herný engine Unity pre tvorbu aplikácie, VS Code ako IDE pre písanie kódu, a Fooocus, Blender a Photshop ako softvér pre grafický dizajn.

Ďalej sme vykonali dôkladnú analýzu existujúcich aplikácií zaoberajúcich sa vesmírnou tematikou a astronómiou. Zamerali sme sa na ich hlavnú funkcionalitu a preverili sme nedostatky týchto softvérov a zakomponovali do požiadaviek našej aplikácie. Taktiež sme určili identitu našej aplikácie.

Posledným krokom našej práce bol návrh a implementácia, následne testovanie a dolaďovanie nedostatkov. Po odbornej analýze konkurenčného softvéru sme vytvorili návrh nášho produktu, ten sme odkonzultovali s malou vzorkou používateľov, pričom ich pripomienky a nápady na zlepšenie softvéru sme zakomponovali do výsledného návrhu. Po dokončení návrhu sme rozdelili vývoj na viacero funkčných blokov, pre hladší a menej komplikovaný vývoj. Počas implementácie bolo vykonaných viacero testovacích behov s používateľmi a ich spätná väzba počas vývoja tak zaručila urýchlenie niektorých častí vývoja.

Výsledná aplikácia tak spĺňa základnú požiadavku, cestovanie v čase a lepšie zobrazenie vývoja slnečnej sústavy. Potenciálne využitie výslednej verzie aplikácii je primárne motivácia mladých ľudí k štúdiu vesmíru, astronómie a kozmického inžinierstva, ktoré sa vyučuje aj na FEI STU, ale aj lepšie objasniť životný cyklus vesmírneho komplexu ako je slnečná sústava. Aplikáciu je možné vylepšiť v rámci grafiky. Pridaním ďalších informácií, ktoré nie sú obsiahnuté v aktuálnej verzií. Simulácia by sa taktiež mohla rozšíriť o editor vlastnej slnečnej sústavy, nahradiť fyziku poskytovanú Unity vlastnou fyzikou cez N-body problém.

Zoznam použitej literatúry

1. **Petrova, M.** *Why there is a new global race to the moon*, [online]. 20.01.2024, [cit. 09.05.2024]. Dostupné z: https://www.cnbc.com/2024/01/20/why-there-is-a-new-global-race-to-the-moon-.html
2. **SAYKILI, A.** *Higher education in the digital age: The impact of digital connective technologies*. Journal of Educational Technology and Online Learning, 2019, 2.1: 1-15. [cit. 10.05.2024]. Dostupné z: https://eric.ed.gov/?id=ED591364
3. **Huntington B. Goulding J. Pitchford N. J.** *Pedagogical features of interactive apps for effective learning of foundational skills*, British Journal of Educational Technology, 2023. [cit. 10.05.2024] ISSN 14678535. Dostupné z: https://bera-journals.onlinelibrary.wiley.com/doi/full/10.1111/bjet.13317
4. **ABRAMOV, D.** *Interactive Analysis Tools for Visualizing the Universe*. University of California, Santa Cruz, 2023. [cit. 10.05.2024]. Dostupné z: https://www.proquest.com/openview/68f7fe43529067a68fe04903f9714252/1?pq-origsite=gscholar&cbl=18750&diss=y
5. **Bhatti K. Abela P. Hoppe J.** *How do you learn and master a new game engine or game framework quickly and efficiently?,* [online]. 2023, [cit 12.05.2024]. Dostupné z: https://www.linkedin.com/advice/0/how-do-you-learn-master-new-game-engine-framework-quickly
6. **arm,** *Game Engines,* [online]. 2021, [cit. 10.05.2024]. Dostupné z: https://www.arm.com/glossary/gaming-engines
7. **Unreal Engine**, *Unreal engine features,* [online]. 2018, [cit. 10.05.2024]. Dostupné z: https://www.unrealengine.com/en-US/features
8. **AL L., Hussain A. J**. *The Path of unity or the Path of unreal? A Comparative Study on Suitability for Game Development*. Journal of Student Research, 2019, [cit. 11.05.2024]. Dostupné z: https://www.jsr.org/index.php/path/article/view/976/823
9. **LINIETSKY, J.** *Godot 2.0: Talking with the Creator,* [online]. 2016, [cit. 11.05.2024]. Dostupné z: https://imetatech.io/blog/unity-unreal-godot-comparison
10. **HOLFELD, J.** *On the relevance of the Godot Engine in the indie game development industry*, [online]. arXiv preprint arXiv:2401.01909, 2023, [cit. 11.05.2024]. Dostupné z: https://arxiv.org/abs/2401.01909

11. **Sabiq**, *Comparison of Unity vs Unreal Engine Vs Godot,* [online]. 2023, [cit. 11.05.2024]. Dostupné z: https://imetatech.io/blog/unity-unreal-godot-comparison
12. **Studio Cherno,** *Hazel Engine,* [online]. 2022, [cit. 13.05.2024]. Dostupné z: https://hazelengine.com/
13. **Chernikov Y.** *I ACCIDENTALLY Created Hazel's Greatest Feature,* [online]. 2024, [cit. 13.05.2024]. Dostupné z: https://www.youtube.com/watch?v=yMRp9DVZYnI
14. **Unity**, *Unity Engine* [online]. 2024, [cit. 10.05.2024]. Dostupné z: https://unity.com/
15. **Red Hat**, *What is an IDE?,* [online]. 2019, [cit. 12.05.2024]. Dostupné z: https://www.redhat.com/en/topics/middleware/what-is-ide
16. **Unity Documentation,** *Integrated development environment (IDE) support,* [online]. 2022, [cit. 12.05.2024]. Dostupné z: https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html
17. **Microsoft**, *Visual Studio,* [online]. 2022, [cit. 12.05.2024]. Dostupné z: https://visualstudio.microsoft.com/cs/
18. **Microsoft,** *Compare Visual Studio Editions 2022,* [online]. 2022, [cit. 13.05.2024]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/compare/
19. **JetBrains**, *Rider,* [online]. 2017, [cit. 12.05.2024]. Dostupné z: https://www.jetbrains.com/rider/
20. **Palatkar S.** *VS Code Extensions For Near IDE Experience,* [online]. 2018, [cit. 13.05.2024]. Dostupné z: https://faun.pub/vs-code-extensions-for-complete-ide-experience-bca5bb2f0f90
21. **Siemens,** *3D Modeling,* [online]. 2021, [cit. 13.05.2024]. Dostupné z: https://www.sw.siemens.com/en-US/technology/3d-modeling/
22. **Blender,** *About,* [online]. 2019, [cit. 13.05.2024]. Dostupné z: https://www.blender.org/about/
23. **RebusFarm,** *Cinema 4D vs. Blender: Choosing the Right 3D Software,* [online]. 2024, [cit. 13.05.2024]. Dostupné z: https://rebusfarm.net/blog/blender-vs-cinema-4d-choosing-the-right-3d-software
24. **Maxon,** *Cinema 4D,* [online]. 2024, [cit. 13.05.2024]. Dostupné z: https://www.maxon.net/en/cinema-4d
25. **Render Farm,** *What is Cinema 4D used for?,* [online]. 2022, [cit. 13.05.2024]. Dostupné z: https://garagefarm.net/blog/what-is-cinema-4d-used-for

26. **Petty, J.** *What is 3ds Max & What is it Used For?,* [online]. 2016, [cit. 13.05.2024]. Dostupné z: https://conceptartempire.com/what-is-3ds-max/
27. **iMashup,** *3ds Max: Pros, Cons, Quirks, and Links,* [online]. 2018, [cit. 13.05.2024]. Dostupné z: https://medium.com/imeshup/3ds-max-pros-cons-quirks-and-links-a2a48832dbbe
28. **GeeksForGeeks,** *What is Graphics Software?,* [online]. 2022, [cit. 13.05.2024]. Dostupné z: https://www.geeksforgeeks.org/what-is-graphics-software/
29. **GIMP,** *GIMP - GNU Image Manipulation Program,* [online]. 1998, [cit. 13.05.2024]. Dostupné z: https://www.gimp.org/
30. **Adobe,** *Photoshop,* [online]. 2024, [cit. 14.05.2024]. Dostupné z: https://www.adobe.com/sk/products/photoshop.html
31. **American Graphics Institute,** *What is Photoshop,* [online]. 2022, [cit. 14.05.2024]. Dostupné z: https://www.agitraining.com/adobe/photoshop/classes/what-is-photoshop
32. **Shtanakova A.** *GIMP vs Photoshop: Which Photo Editor is Better?,* [online]. 2023, [cit. 14.05.2024]. Dostupné z: https://skylum.com/cs/blog/gimp-vs-photoshop
33. **Nvidia,** *What is Generative AI?,* [online]. 2024, [cit. 14.05.2024]. Dostupné z: https://www.nvidia.com/en-us/glossary/generative-ai/
34. **Nvidia,** *Nvidia Canvas,* [online]. 2023, [cit. 14.05.2024]. Dostupné z: https://www.nvidia.com/en-eu/studio/canvas/
35. **Wankhede C.** *What is Midjourney AI and how does it work?,* [online]. 2024, [cit. 14.05.2024]. Dostupné z: https://www.androidauthority.com/what-is-midjourney-3324590/
36. **Fooocus,** *Foocus: The Most Advanced AI Image Generator 2024,* [online]. 2024, [cit. 14.05.2024]. Dostupné z: https://fooocusai.com/
37. **Jar,** *Working Through the N-Body Problem in Universe Sandbox²,* [online]. 2016, [cit. 14.05.2024]. Dostupné z: https://universesandbox.com/blog/2016/02/n-body-problem/
38. **Giant Army,** *Universe Sandbox,* [online]. 2018, [cit. 14.05.2024]. Dostupné z: https://universesandbox.com/
39. **Cosmographic Software LLC,** *Space Engine,* [online]. 2019, [cit. 14.05.2024] Dostupné z: https://spaceengine.org/

40. **Ševeček P.** *SpaceSim,* [online]. 2018, [cit. 15.05.2024]. Dostupné z: https://pavelsevecek.github.io/
41. **NASA,** *Solar System Exploration,* [online]. 2024, [cit. 15.05.2024]. Dostupné z: https://science.nasa.gov/solar-system/
42. **UÍM**, *Softvérové inžinierstvo*, [online]. 2023, [cit. 15.05.2024]. Dostupné z: https://uim.fei.stuba.sk/predmet/b-swi/
43. **SEA,** *The Birth of the Solar System,* [online]. 2021. [cit. 16.05.2024]. Dostupné z: https://www.youtube.com/watch?v=d-z0eQOEzkE&list=PLCn-jE0y6UU7bCcMjaoy-ir6-brWHXJEY
44. **SEA,** *The Final Age of the Solar System,* [online]. 2021. [cit. 16.05.2024]. Dostupné z: https://www.youtube.com/watch?v=oWaDUiBHDY8&list=PLCn-jE0y6UU7bCcMjaoy-ir6-brWHXJEY
45. **Unity Documentation,** *Canvas*, [online]. 2021, [cit. 17.05.2001]. Dostupné z: https://docs.unity3d.com/2021.3/Documentation/Manual/class-Canvas.html
46. **Thirslund, A.** *SETTINGS MENU in Unity,* [online]. 2017, [cit. 17.05.2024]. Dostupné z: https://www.youtube.com/watch?v=YOaYQrN1oYQ
47. **Unity Documentation,** *Scriptable Object*, [online]. 2021, [cit. 17.05.2024]. Dostupné z: https://docs.unity3d.com/2021.3/Documentation/Manual/class-ScriptableObject.html
48. **Unity Documentation,** *Unity Events,* [online]. 2021, [cit. 17.05.2024]. Dostupné z: https://docs.unity3d.com/2021.3/Documentation/Manual/UnityEvents.html
49. **Unity Documentation,** *Time,* [online]. 2021, [cit. 17.05.2024]. Dostupné z: https://docs.unity3d.com/2021.3/Documentation/ScriptReference/Time.html

Prílohy

A Štruktúra prílohy aplikácie

Štruktúra prílohy aplikácie obsahuje zdrojové kódy súborov k interaktívnej aplikácií Vznik a zánik slnečnej sústavy v súbore projekt/BP. Bakalársku prácu vo formáte DOCX a PDF a priložený súbor build/BP so spustiteľným .exe súborom na spustenie aplikácie.

/projekt/BP.zip – komprimovaný priečinok so zdrojovými kódmi
/build/BP-final – priečinok so spustiteľným .exe
/BP_Janik.docx – bakalárska práca v DOCX formáte
/BP_Janik.pdf – bakalárska práca v PDF formáte
Alternatívny link na zdrojové kódy – https://github.com/IMPArchyS/BP

B Technická dokumentácia

Zobrazenie informácií o objekte

V prvej osobe využívame na získanie informácií o objekte „Collider“. Na kameru pridáme komponenty „Sphere Collider“ a „RigidBody“. Pre komponent „RigidBody“ zapneme vlastnosť „Is Kinematic“, tá slúži aby sa fyzika na daný objekt aplikovala iba cez skripty. Tieto dva komponenty nám umožnia detekovať pohyb objektu v rámci iného objektu, pričom iný objekt musí taktiež mať komponent typu „Collider“ na špecifickom type však nezáleží. Ak chceme, aby pri vstupe našej kamery do vzdialenosti planéty skript reagoval, musí mať „Collider“ planéty nastavenú vlastnosť „Is Trigger“. Teraz vieme využiť nasledovné metódy:

```
[SerializeField] private UnityEvent<CelestialObject> onPlayerEnter;
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("MainCamera") && currentCelestial == null)
    {
        onPlayerEnter?.Invoke(this);
        currentCelestial = this;
    }
}
```

Výpis 16: Vstup Kamery do dosahu objektu

```
[SerializeField] private UnityEvent onPlayerExit;
private void OnTriggerExit(Collider other)
{
    if (other.gameObject.CompareTag("MainCamera") && currentCelestial == this)
    {
        onPlayerExit?.Invoke();
        currentCelestial = null;
    }
}
```

Výpis 17: Výstup kamery z dosahu objektu

`onPlayerEnter`, `onPlayerExit` sú „Unity Eventy“, ktoré sú zavolané spúšťajú metódy v skripte „CelestialObjectInfo“. Tieto aktualizujú dáta na textových elementoch pri zobrazení menu o objekte. Dôležité je poznamenať, že `onPlayerEnter` event obsahuje

parameter typu „CelestialObject“, a ten obsahuje všetky informácie, ktoré chceme používateľovi zobraziť. Telá volaných metód pomocou „eventov“ vyzerajú takto:

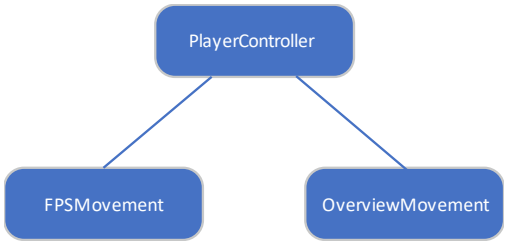
```
public void OnEnterRange(CelestialObject celestialObject)
{
    Debug.Log("Hello from CSO-Info");
    currentObject = celestialObject;
}

public void OnExitRange()
{
    Debug.Log("Bye from CSO-Info");
    currentObject = null;
}
```

Výpis 18: Telá volaných metód na aktualizáciu dát

„CelastialObjectInfo“, trieda volaných metód, taktiež obsahuje metódy na priradenie dátových hodnôt „currentObject“ k príslušným textovým prvkom.

Kamerový kontrolér

Podrobným popisom implementácie kamery a pohybu používateľa sa zaoberá kapitola 3.8.2. V scéne sú 2 kamery, kameru pre prvú osobu budeme ďalej nazývať iba „fps“, a druhú kameru „ovm“. Na obe kamery sme pridali korešpondujúce skripty na ich funkcionalitu. Objekt „CameraManager“ obsahuje komponent „PlayerController“ čo je náš kamerový kontrolér. Tu vieme nastaviť rôzne parametre ako: rýchlosť pohybu, zrýchlenie, citlivosť a referencie na skripty „fps“ a „ovm“.



Obr. 18: Štruktúra Kamerového kontroléra

Ak chceme pridať funkcionalitu do niektorého z módov kamery editujeme ich skripty. Následne pridáme volanie ich metód do kontroléra. Ten dodatočne obsahuje zopár stavových premenných na správne nastavenie daného módu.

Prepojenie počítania času s GUI

Používateľovi sa zobrazuje aktuálny čas v rokoch. Ak chceme pridať ďalší krok pre modifikáciu rýchlosti času editujeme premenné „timeScales“, „timeUnits“ v „MainTimeController“ skripte a pridáme preklad do metódy v „TranslateToSlovak“ skripte.

```
private readonly BigInteger[] timeScales = {
        1, 60, 3600, 86400, 31536000,
        315360000, 3153600000, 31536000000, 315360000000,
        3153600000000, 31536000000000, 315360000000000, 3153600000000000,
        BigInteger.Parse("31536000000000000000"),
        BigInteger.Parse("315360000000000000000")
    };
private readonly string[] timeUnits = {
        "sec", "min", "hr", "day", "yr",
        "10 yrs", "100 yrs", "1000 yrs", "10k yrs",
        "100k yrs", "1mil yrs", "100mil yrs", "1bil yrs",
        "1mld yrs", "100mld yrs"
    };
```

Výpis 19: Premenné pre časové kroky

Pre väčšie čísla však už musíme používať metódy dátového typu „BigInteger“, nakoľko nám už vysoké číselné konštanty samotné C# nepodporuje a nevie ich skompilovať.

V GUI je prepočet času zobrazovaný tak, že do 4,6 miliárd rokov sa nám čas zobrazuje v záporných hodnotách, pričom rok „0“ predstavuje približne aktuálny rok 2024. Od tohto okamihu sa v GUI roky pripočítavajú v kladných číslach. V kóde nerobíme žiadne prepočty a využívame vymoženosti C# funkcie `ToString()`, ktorej vieme nastaviť požadované formátovanie ako parameter. Časomiera je podrobne opísaná kapitole 3.8.5.

```
if (YearCount >= 4600000000)
{
    timeText.text = "+" + ((decimal)YearCount / 1000000000m).ToString("N3")
                    + " Miliard rokov";
}
else
{
    timeText.text = (((decimal)YearCount - 4600000000) / 1000000000m)
                    .ToString("N3") + " Miliard rokov";
}
```

Výpis 20: Formátovanie výpisu času

C Používateľská príručka

Hlavné menu

Po spustení aplikácii máte na výber si nastaviť vlastnosti aplikácie, ako je rozlíšenie, hlasitosti hudby, zvuku a grafických detailov. Ak chcete spustiť simuláciu od úplného začiatku stlačíte na text štart. Simulácia sa načíta a ocitnete sa priamo v nej. Ak chcete si zvoliť inú etapu, napríklad súčasnosť, kliknutím na text „Etapy“ sa dostanete do ďalšieho menu s možnosťami a zvolíte možnosť, ktorú chcete.

Zvolenie etapy

Po zvolení etapy cez etapové menu, budete načítaný do simulácie, ak sa chcete vrátiť, ale späť v čase, budete sa musieť vrátiť do hlavného menu a zvoliť skoršiu etapu.

Ovládanie

V priestore sa pohybujete pomocou klávesnice a myši. Po štarte simulácií ste v prvej osobe. Pohľad otáčate pohybom myši a pohybujete sa pomocou kláves „WASD“. Ak chcete zmeniť kameru stlačíte tlačidlo „P“. V tejto perspektíve sa neviete pohybovať, ale viete skákať po objektoch slnečnej sústavy pomocou kliknutí ľavého tlačidla na myške, obzerať sa viete držaním ľavého tlačidla myši a vzájomným pohybom myši. V tejto perspektíve máte možnosti zobrazenia rôznych detailov pomocou tlačidiel. Tieto detaily sa dajú zhliadnuť aj v kamere prvej osoby pomocou klávesových skratiek, (všetky klávesové skratky sú napísane nižšie v príručke).

Klávesové skratky

WASD – pohyb v prvej osobe
P – prepnutie perspektívy kamery
T – zobrazenie ponuky používateľa v prvej osobe
O – vypnutie/zapnutie obežných dráh telies
M – debugovacie menu
ESC – zobrazenie ponuky návratu do hlavného menu

D Zdroje k použitej hudbe a zvukom

V tejto prílohe sú uvedené zdroje k použitým zvukovým efektom a hudbe v aplikácií. Všetky zdroje majú licenciu na voľné nekomerčné použitie.

Hudba

Hlavné menu: https://www.free-stock-music.com/glitch-looking-at-the-night-sky.html
Simulácia: https://tunetank.com/track/3802-stasis/

Zvuk

Tlačidlo: https://www.zapsplat.com/page/2/?s=button+click&post_type=music&sound-effect-category-id

E Zdroje k použitým objektom

V tejto prílohe sú uvedené zdroje k použitým materiálom, objektom, efektom v aplikácií. Všetky zdroje majú licenciu na voľné nekomerčné použitie.

Simulácia

Path Creator: https://assetstore.unity.com/packages/tools/utilities/b-zier-path-creator-136082

Asteroidy: https://assetstore.unity.com/packages/3d/environments/asteroids-pack-84988

Planéty: https://assetstore.unity.com/packages/3d/environments/planets-of-the-solar-system-3d-90219

Obloha: https://assetstore.unity.com/packages/2d/textures-materials/dynamic-space-background-lite-104606

Dodatočné planéty a asteroidy: https://assetstore.unity.com/packages/3d/environments/sci-fi/vast-outer-space-38913

Variácie planét: https://assetstore.unity.com/packages/3d/environments/sci-fi/animated-planets-206071

GUI

Ikonky: https://assetstore.unity.com/packages/2d/gui/icons/clean-vector-icons-132084