Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# BAKALÁŘSKÁ PRÁCE

Aplikace pro astronomy

2020

Štěpán Šumpík

Vedoucí práce: RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika, prezenční forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | Štěpán Šumpík |
| Název práce: | Aplikace pro astronomy |
| Typ práce: | bakalářská práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2020 |
| Studijní obor: | Aplikovaná informatika, prezenční forma |
| Vedoucí práce: | RNDr. Arnošt Večerka |
| Počet stran: | 45 |
| Přílohy: | 1 CD/DVD |
| Jazyk práce: | český |

**Bibliograhic info**

| | |
|---|---|
| Author: | Štěpán Šumpík |
| Title: | Application for astronomers |
| Thesis type: | bachelor thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2020 |
| Study field: | Applied Computer Science, full-time form |
| Supervisor: | RNDr. Arnošt Večerka |
| Page count: | 45 |
| Supplements: | 1 CD/DVD |
| Thesis language: | Czech |

## Anotace

*Aplikace určená pro astronomická pozorování polohy planet Sluneční soustavy. Skládá se ze dvou částí. První část slouží k pozorování planet, Slunce a Měsíce ze zemského povrchu. Druhá část slouží k pozorování vzdálenosti planet a Slunce z vesmíru.*

## Synopsis

*Application designed for astronomical observations of the planets' position in the Solar System. It consists of two parts. The first part can be used to observe planets, Sun and Moon from the Earth's surface. The second part is used to observe the distance of planets and Sun from space.*

**Klíčová slova:** Sluneční soustava; astronomie; vesmír; pozorování oblohy

**Keywords:** Solar System; astronomy; universe; observational astronomy

Děkuji vedoucímu práce, RNDr. Arnoštu Večerkovi, za podnětné připomínky při vytváření práce a za utváření vize výsledné aplikace.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce podpis autora

# Obsah

# Seznam obrázků



# Seznam zdrojových kódů

# 1 Úvod

Bakalářská práce Aplikace pro astronomy se skládá ze dvou modelů Sluneční soustavy.

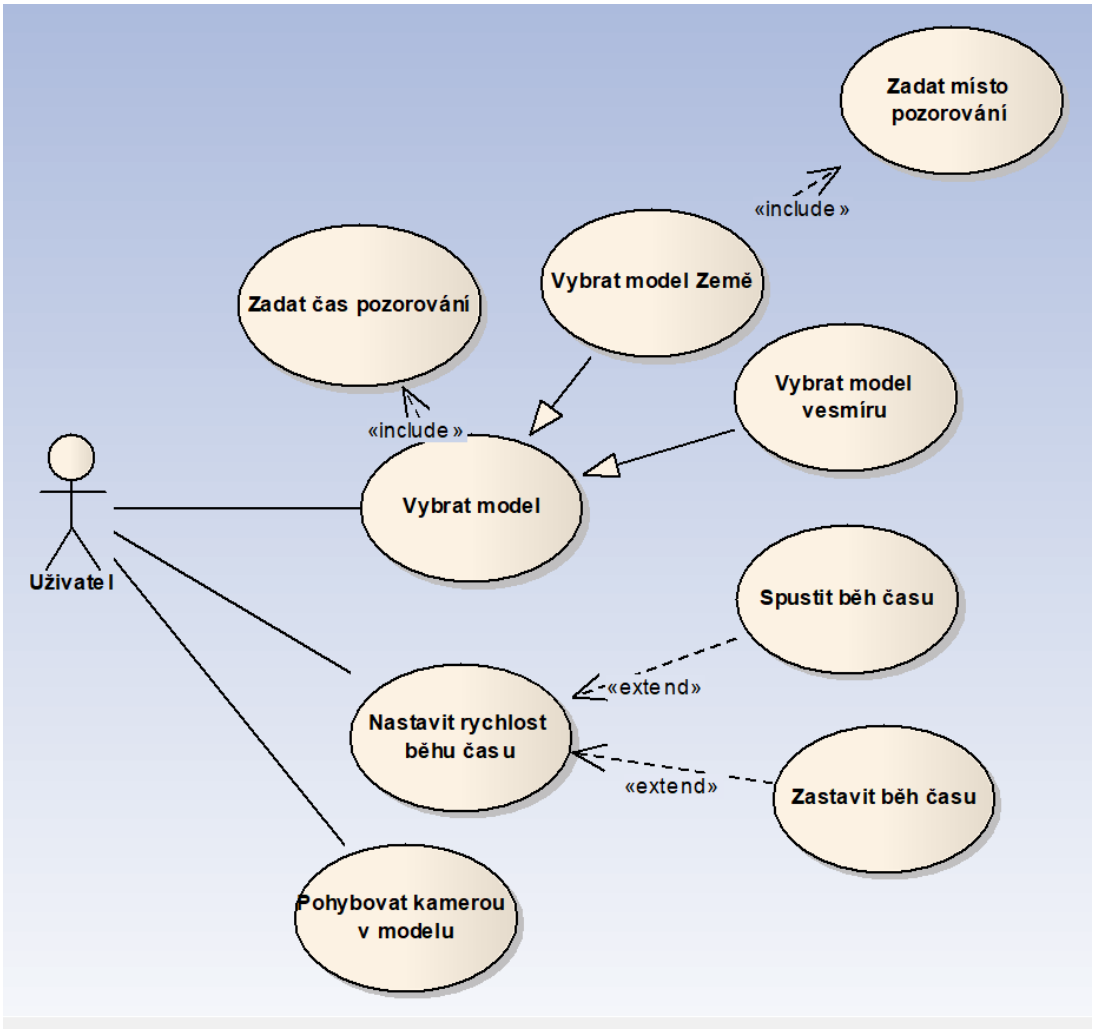
První model zobrazuje polohu planet Sluneční soustavy a Měsíce na simulované zemské obloze vzhledem k zeměpisným souřadnicím a času. Uživatel aplikace má v tomto modelu možnost zvolit zeměpisné souřadnice a čas. Poloha planet se podle nových údajů aktualizuje. Dále tento model simuluje běh času. Lze zvolit rychlost, s jakou se planety posunují po obloze.

Druhý model zobrazuje planety a jejich oběžné dráhy z vesmírného pohledu. Podobně jako v prvním modelu má uživatel možnost zvolit si konkrétní datum pozorování a rychlost běhu času. Poměry vzdáleností mezi oběžnými dráhami a jejich délkami jsou zachovány. Poměry velikostí planet však z praktických důvodů nejsou zachovány ani v jednom modelu.

# 2 Analýza požadavků

## 2.1 Požadavky ze zadání práce

Na základě zadání bakalářské práce vyplynulo množství požadavků. Nejdůležitějším je možnost uživatele pozorovat planety ze zemského povrchu a z vesmíru. Kvůli tomu jsem se rozhodl rozdělit výslednou aplikaci na dvě části — modely. První model zobrazuje planety Sluneční soustavy ze zemského povrchu. Druhý model má uživateli umožnit pozorovat vzdálenosti mezi planetami ve vesmíru. U obou modelů má být možnost výběru času pozorování.



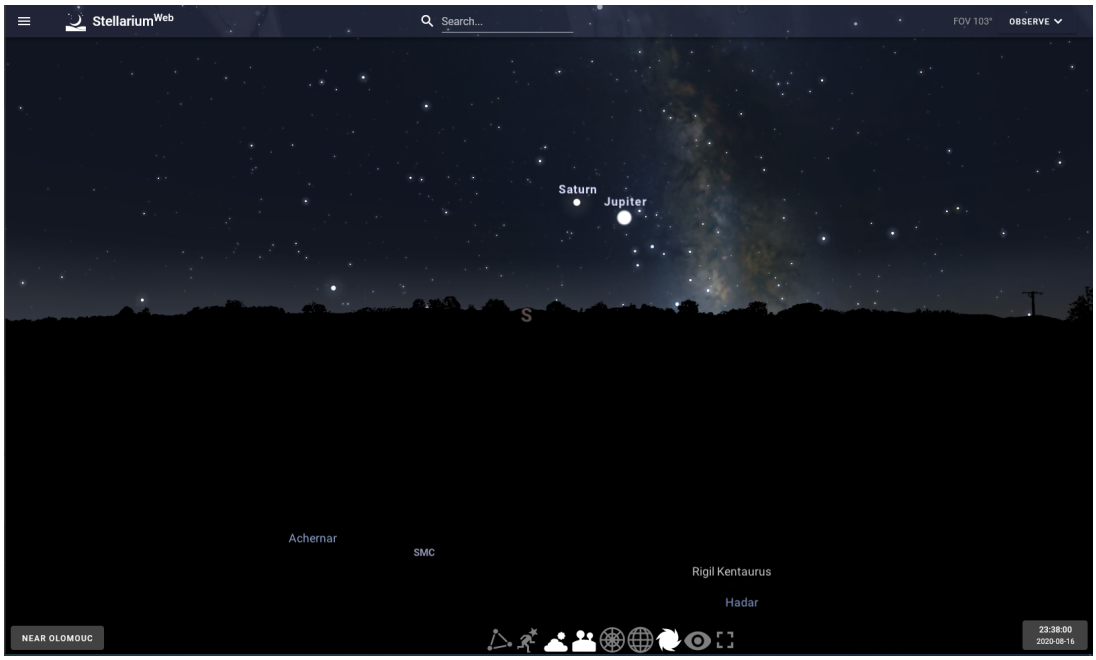

Obrázek 1: Use Case diagram počátečních požadavků

U požadavků na aplikaci jsem se inspiroval u již existujících aplikací pro astronomická pozorování.

## 2.2 Existující aplikace

### 2.2.1 Stellarium

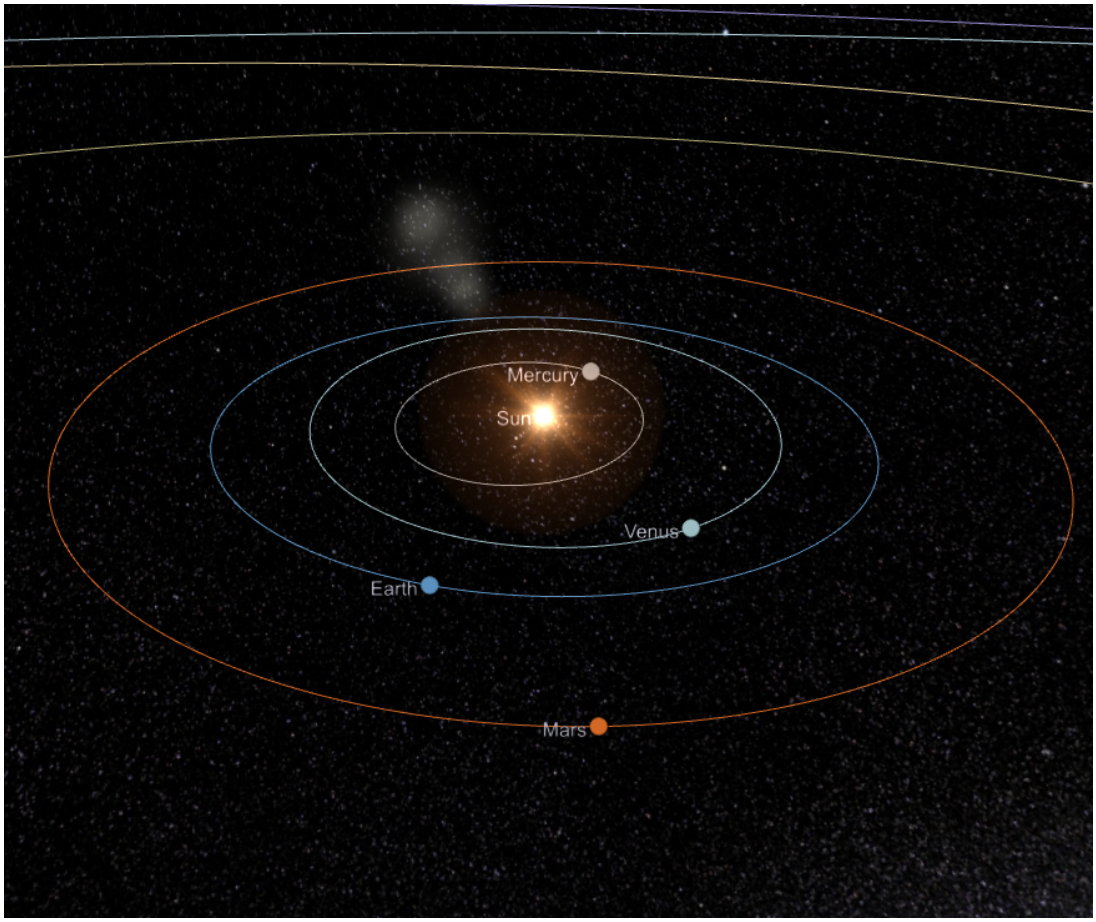
Stellarium [1] je aplikace pro pozorování souhvězdí, planet a jiných vesmírných objektů ze zemského povrchu. Aplikace umožňuje nastavovat polohu, čas pozorování, zorné pole kamery a jiné. Kromě planet může astronom pozorovat třeba souhvězdí nebo zapnout *grid* na lepší určení polohy pozorovaného objektu. Při zadávání času se ale uživatel nevyhne velkému množství klikání. Není zde také možnost nastavit rychlost, s jakou čas ubíhá. Ovládání aplikace mi přijde až moc citlivé na pohyb myši. Svými možnostmi a příjemnou grafickou stránkou pro mě však tato aplikace dalece předbíhá konkurenci.




Obrázek 2: Obrázek z aplikace Stellarium

### 2.2.2 The Sky LIVE

The Sky LIVE [2] je webová aplikace pro pozorování planet z vesmíru. Aplikace umožňuje nastavovat čas pozorování, rychlost ubíhání času a vzdálenost kamery ve vesmíru. Uživatelské rozhraní v aplikaci mi přišlo mírně matoucí a nastavení ubíhání času je trochu omezené. Graficky je pěkná a velmi se mi líbí zobrazení oběžných drah u planet.




Obrázek 3: Obrázek z aplikace Skylive

### 2.2.3 Solar System Scope

Solar System Scope [3] je další webovou aplikací na pozorování planet, obsahuje jak model na pozorování planet z povrchu Země, tak vesmírný model. Ani jeden však dle mého názoru neprezentuje správně. Ve vesmírném modelu jsou planety příliš velké a přítomnost rotace planet způsobuje chaos. Textura povrchu Země je zase rušivá. Program však používá textury získáné vesmírným pozorováním, a proto jsou jednotlivé planety graficky velmi pěkné. Navíc program obsahuje velké množství informací o jednotlivých planetách.

### 2.2.4 Mobilní aplikace

Velkou výhodou mobilních astronomických aplikací je gyroskop, GPS a přenosnost. Tyto aplikace fungují na principu rozšířené reality, kdy po namíření kamery mobilu na část oblohy se na displeji zobrazí přítomné vesmírné objekty. Nevýhodou těchto aplikací většinou bývá složité ovládání a malá obrazovka na pozorování.


Obrázek 4: Obrázek z aplikace Solar System Scope

## 2.3 Vize aplikace

Po krátkém zvážení jsem se rozhodl neudělat mobilní aplikaci, ale desktopovou. Díky zanalyzování funkcí, uživatelských rozhraní a grafické stránky několika aplikací jsem došel k těmto požadavkům na aplikaci:

- obsažení světových stran pro lepší orientaci (Země)
- možnost nastavit si zorné pole kamery (Země)
- dobře viditelné oběžné dráhy u planet (vesmír)
- možnost oddálit a přiblížit kameru (vesmír)
- jména u planet
- dobrou grafickou stránku
- jednoduché zadávání času a rychlosti
- intuitivní ovládání

# 3 Vývoj aplikace

## 3.1 Vývojová prostředí

Vývoj aplikace od základu není příliš efektivní, proto se k vývoji často používají softwarová rozhraní. Tato rozhraní mohou vývoj velmi usnadnit, zrychlit a jednodušit. Na výběr je velké množství možností vývoje. Při výběru rozhraní je vhodné zvážit několik kritérií.
Například:

- typ aplikace
- účel aplikace
- uživatele, kteří budou aplikaci používat
- složitost vývoje na vybrané platformě
- vizuální design
- zkušenosti vývojáře

## 3.2 Herní engine

Herní engine je komplexní, víceúčelový nástroj na tvorbu her, aplikací a multimediálního obsahu. Nabízí prostředí na efektivní vývoj, někdy bez nutnosti znalosti skriptování. Herní enginy zahrnují mnoho různých oblastí herního vývoje jako renderování, fyziku, zvuk, animaci, umělou inteligenci a možnosti vytvořit uživatelské rozhraní.[4]

Na vývoji velkých projektů se podílí mnoho lidí různých profesí. Někteří z nich, například umělečtí designéři, nemají s programováním žádné či skoro žádné zkušenosti. Dobrý herní engine je navržen tak, aby byl uživatelsky přívětivý pro všechny typy profesí. Zároveň by toto kritérium nemělo omezovat funkčnost a možnosti enginu. Herní enginy taktéž umožňují výslednou aplikaci exportovat na různé platformy, včetně mobilních zařízení, jen za pomocí minimálních změn v originální verzi.

Herní enginy, ač sdílí velké množství funkcí, mohou se velmi lišit. Třeba přístupem k vývoji nebo cílením na různé potřeby vývojářů. Další zase nepožadují znalost programování nebo jsou založeny na principu webových technologií. Některé enginy si lze upravit či rozšířit uživatelem. Vybrat si není tedy jednoduché.[5]

## 3.3 Typické funkce herního enginu

- renderování 2D/3D grafiky
- práce se vstupem od uživatele: klávesnice, myš, dotyková zařízení
- herní smyčka, přepočítávání herních událostí pro každý snímek
- fyzikální engine pro detekci kolizí
- zvuk
- animace
- přiřazování paměti
- paralelizace procesů [6]

## 3.4 Další funkce herních enginů

- skriptování
- umělá inteligence
- sdílení informací po síti
- streamování
- podpora lokalizace pro vytvořené programy
- multi-platformní vývoj [6]

## 3.5 Unreal Engine 4

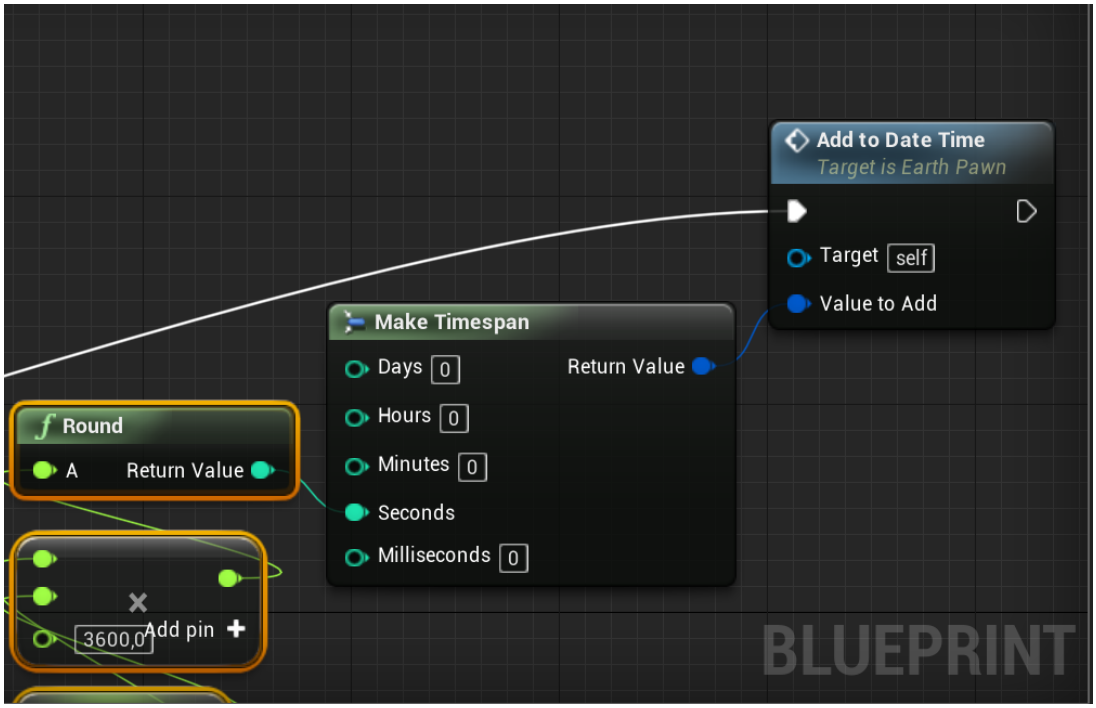
Unreal Engine 4 je engine vyvíjený firmou Epic Games. Programovací jazyk používaný v enginu je C++. Ve čtvrté verzi tohoto enginu přibyla možnost implementovat C++ kód přímo do enginu, čehož využívám i v této bakalářské práci.[7] Unreal Engine je pro nekomerční použití zdarma.

## 3.6 Unreal Engine 4 Blueprints

Kromě C++ Unreal Engine nabízí možnost používat tzv. Blueprints Visual Scripting. Tento skriptovací systém umožňuje skriptování pomocí uzlového rozhraní pro vytvoření herních prvků v editoru enginu (Unreal Editor). Používá se zejména pro definování tříd objektů v enginu. Tento systém je flexibilní a výkonný, umožňuje využít celou řadu konceptů a nástrojů používaných v programování. Je vhodný i pro méně zkušené vývojáře.[8]

Každý z uzlů Blueprintu má specifickou funkci, vstupy, výstupy. Spojením uzlů se tvoří logika celého programu. Blueprinty mohou obsahovat funkce, makra,

události. Mají také proměnné, datové typy, pole, enums, hash mapy. Většina komponentů v Unreal Enginu jsou Blueprinty. Designéři vytvářejí hru pomocí Blueprintů. Umělci navrhou materiál, též Blueprint a stejně tak umělá inteligence používá Blueprinty. [4]




Obrázek 5: Blueprints funkce


Obrázek 6: Blueprints materiál

## 3.7 Aplikace pro astronomy

Když jsem vybíral téma bakalářské práce, tak kvůli mé fascinaci vesmírem byla Aplikace pro astronomy jasná volba. Výběr platformy, na které aplikaci vyvíjet, tak jednoduchý nebyl. Když jsem však viděl grafické možnosti Unreal Enginu, měl jsem jasno, ač jsem s ním neměl předchozí zkušenosti. Oproti konkurenčním enginům je UE4 vhodný pro pokročilejší uživatele, protože podporuje visual scripting a má komplexnější grafické prostředí. Není jednoduchý na naučení a vyžaduje silnější hardware, než většina ostatních enginů. Nejvíce se hodí pro desktopové hry a aplikace, protože nabízí více funkcí a lepší grafické renderování než konkurence. [5]

# 4 Části aplikace

## 4.1 Unreal Engine Levels

Všechno, co lze v aplikaci vidět a s čím lze interagovat se nachazí ve struktuře zvané Level. Level se skládá z modelů, oblastí, osvětlení, Blueprintů a dalších částí, které utvářejí výslednou aplikaci. [9]

## 4.2 Unreal Engine AActors

Actor je základní třída objektů, které mohou být umístěny do levelu. Actory mohou obsahovat ActorComponents. Ty určují všechny jeho vlastnosti, jak se bude Actor pohybovat, renderovat, z jakých modelů se bude skládat a další. Actor také obsahuje proměnné a funkce, které je možno volat při běhu aplikace. [10]

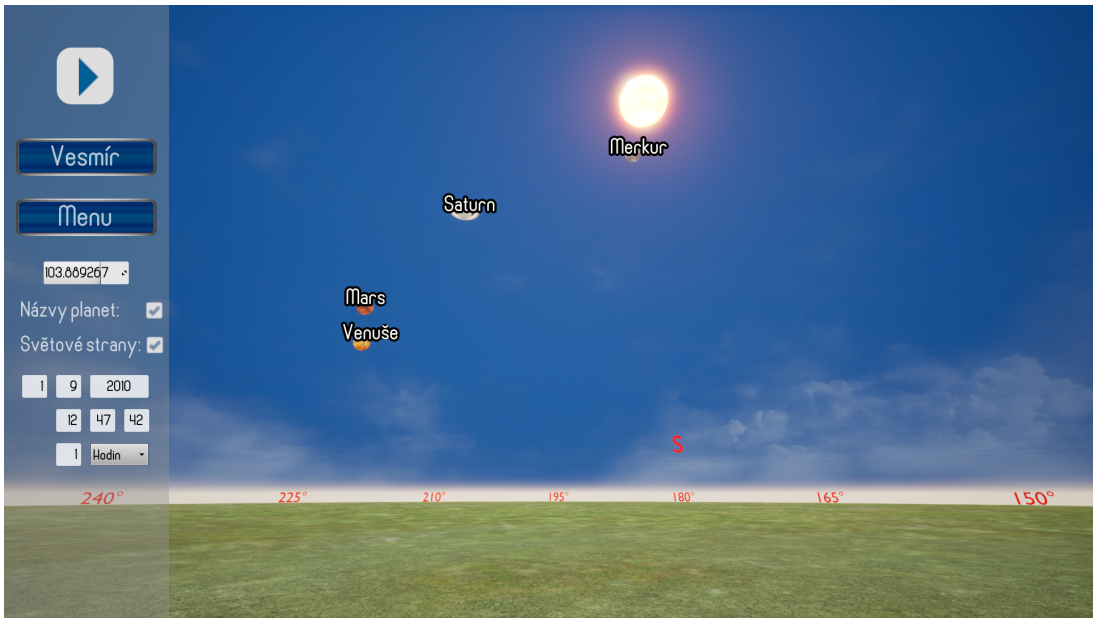
## 4.3 MainMenu Level

MainMenuLevel je level v Aplikaci pro astronomy, sloužící pro nastavení polohy, časové zóny a jako rozcestník do levelů EarthLevel a SpaceLevel, které zobrazují modely Sluneční soustavy. Tento level kromě uživateského rozhraní obsahuje Actor představujícího Zemi a Actor představujícího Slunce a další Volumes upravující vzhled levelu.


Obrázek 7: MainMenuLevel

## 4.4 Earth Level

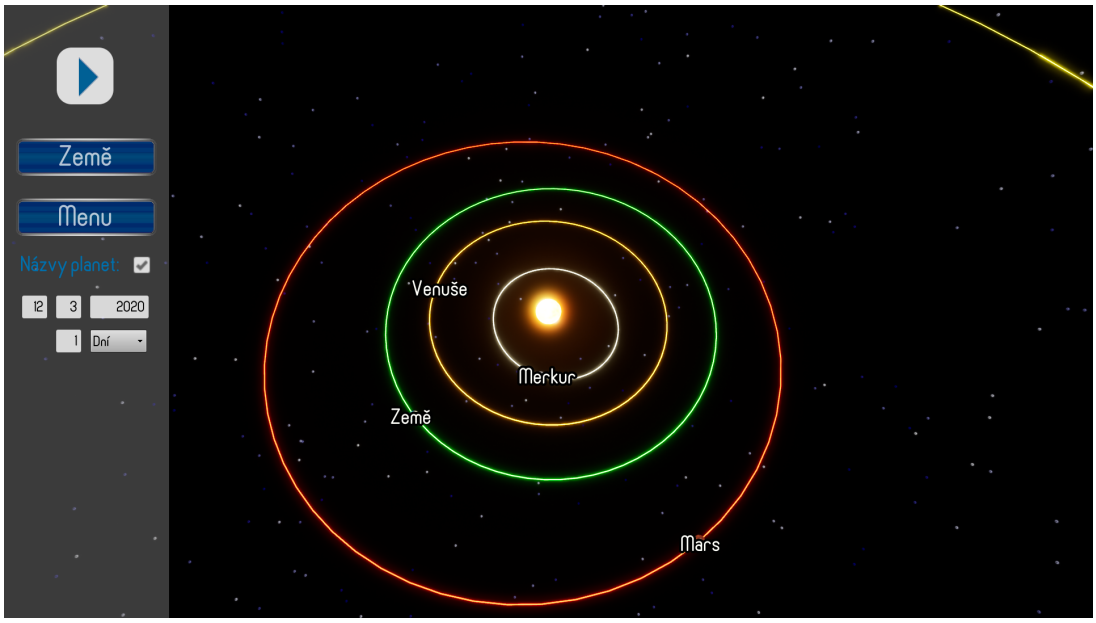
EarthLevel je level pro Sluneční soustavu z pohledu člověka stojícího na povrchu Země. Uživatelské rozhraní v tomto levelu slouží k nastavení data, rychlosti běhu času a nastavení zorného pole kamery. Uživatel má možnost pozorovat polohy planet, Slunce a Měsíce. Taktéž umožňuje přístup do MainMenuLevel a SpaceLevel. Level obsahuje Actory jako jednotlivé planety, Slunce a Měsíc.




Obrázek 8: EarthLevel

## 4.5 Space Level

SpaceLevel je level znázorňující Sluneční soustavu z pohledu z vesmíru. Uživatelské rozhraní v tomto levelu slouží k nastavení data a rychlosti běhu času. Taktéž umožňuje přístup do MainMenuLevel a EarthLevel. Obsahuje Actory jako jednotlivé planety a Slunce.

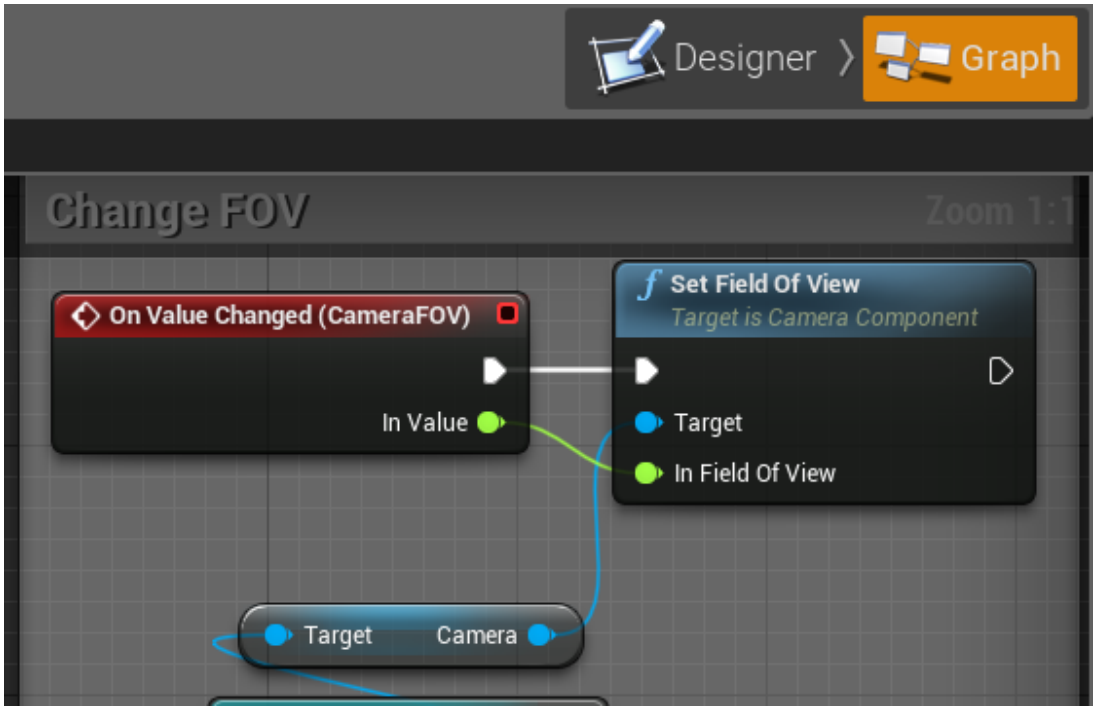



Obrázek 9: SpaceLevel

# 5 Uživatelská rozhraní

## 5.1 Unreal Engine UMG

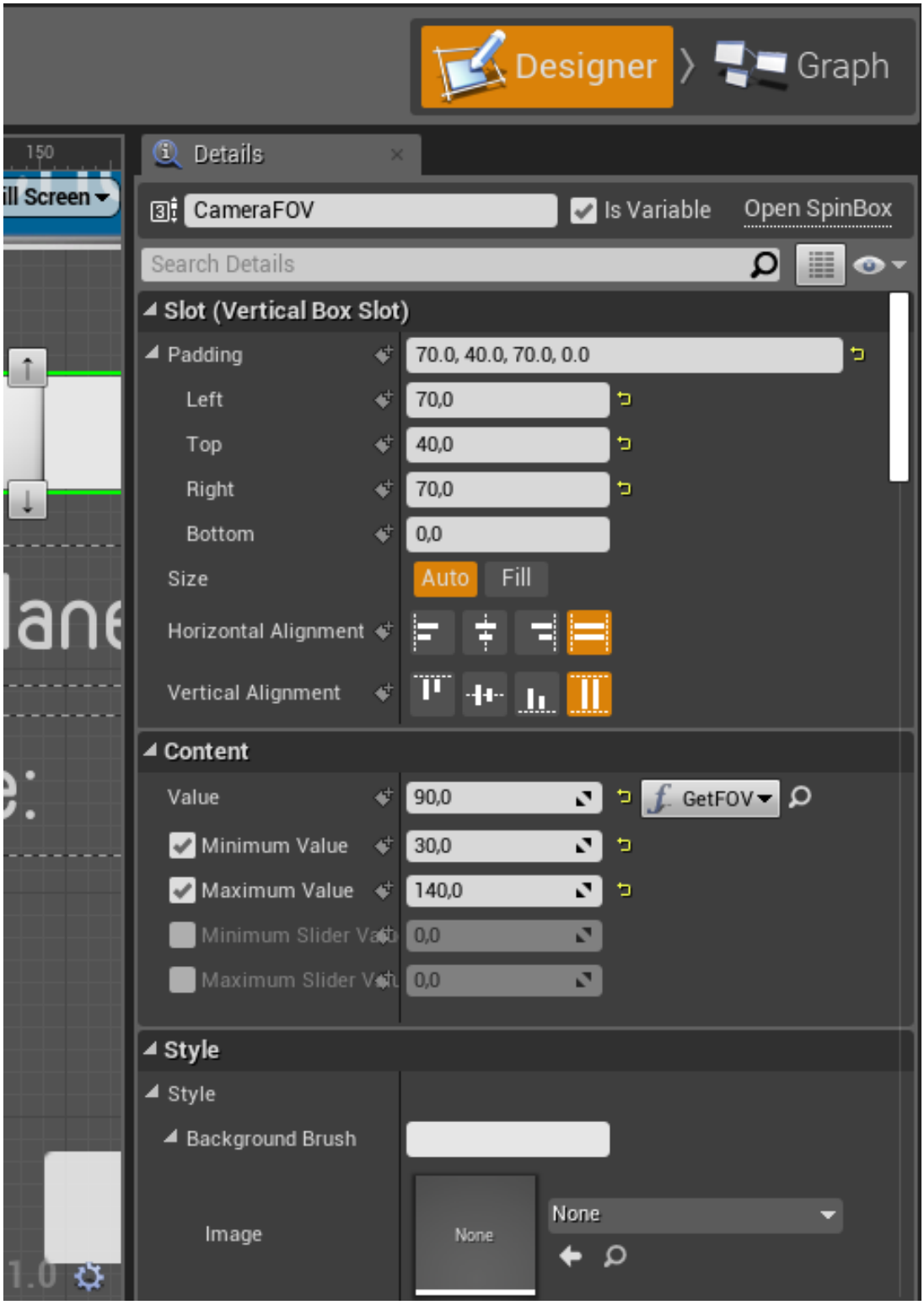
Unreal Motion Graphics UI Designer (UMG) je vizuální nástroj na vytváření uživatelského rozhraní. Zahrnuje všechny prvky jako HUD (head-up-display) menu a další grafická rozhraní, která se mají uživateli prezentovat. Základ UMG, Widgety, jsou Blueprinty sloužící k vytváření rozhraní. Nabývají mnoha podob známých i z jiných softwarových rozhraní jako tlačítka, zaškrávací pole a další. Jako všechny Blueprints i je lze upravit, popřípadě si vytvořit vlastní — a to jak v enginu, tak v C++. Konstrukce Widgetu se skládá ze dvou panelů. Designer panel slouží k nastavení vzhledu Widgetu a jeho rozvržení. Graph panel dodává Widgetům jejich funkcionalitu. [11]




Obrázek 10: Graph panel

## 5.2 Unreal Engine Pawn

Třída Pawn je základní třídou sloužící ke kontrole Actoru hráčem nebo umělou inteligencí. Pawn je fyzickou reprezentací uživatele nebo AI entity v levelu. Pawn určuje, jak uživatel nebo umělá inteligence vypadá v levelu a jak s ním interaguje. Tyto interakce zahrnují například kolize s jinými Actory nebo Volumes. Pawn reprezentuje vlastnosti jako polohu, rotaci a velikost hráče nebo entity v aplikaci. [12]




Obrázek 11: Designer panel

## 5.3 Menu Button

Pro potřeby této aplikace jsem vytvořil vlastní menu tlačítko, které používám ve všech levelech aplikace. Po stisknutí se přehraje zvuk. Textura tlačítka je mnou vytvořená v grafickém programu GIMP. [13]

## 5.4 Planet Name Widget

Planet Name Widget je widget obsahující text. Text Widgetu je nastaven tak, aby měl vzhledem ke kameře Pawnu vždy stejnou rotaci a velikost. Planet Name Widget je použit jako komponent v Actoru Planet, ze kterého dědí všechny jednotlivé planety. Každá planeta si nastavuje hodnotu textu sama.

# 6 Textury a materiály

Textury a materiály jsou důležité části každé aplikace. Spolu s osvětlením určují, jak vypadají jednotlivé objekty v levelu. Textura je popisem vlastností povrch je podstatná pro vnímání struktury, barvy a kvality objektu. Textura je vzorek, který může být buď pravidelný, nebo nepravidelný. Prvek textury se nazývá texel. Textura je spjata s materiálem, který by měl povrch tělesa jednoznačně popisovat. Z historických důvodů, zejména však pro zjednodušení mnoha operací, se materiál a textura oddělují a aplikují ve dvou krocích. [14]

## 6.1 Textury

Textury jsou obrázky používáné v materiálech. Jsou namapovány na povrch, na kterém je aplikován materiál. Textury jsou aplikovány buď přímo — například barvou nebo se vytvoří maska z pixelů textury (texelů) a je použita v materiálu. Textury mohou být použity i na uživatelské rozhraní. Obvykle jsou vytvořeny v grafických programech jako GIMP nebo Photoshop, a potom importovány do programu, ve kterém jsou použity, v tomto případě Unreal Enginu.

Jeden materiál může využívat několik textur, které jsou vzorkovány a použity pro více účelů. Například jeden materiál může využívat barevnou texturu, emisivní texturu, normálovou mapu a jiné. Unreal Engine umožňuje vytvoření dynamických textur. Textura objektu se v tomto případě mění po nějaké události v levelu. [15]

## 6.2 Materiály

Materiál je aplikován na 3D model objektu, který určuje, jak bude daný objekt vypadat. To zahrnuje interakci materiálu se světlem a povrchem, které jsou vypočítány matematickými funkcemi a výrazy. V Unreal Enginu lze nastavit mnoho z těchto vlastností přímo v enginu. Materiál mimo jiné definuje povrch, barvu, lesklost, průhlednost, tření a reakci na fyzikální engine. [16]

# 7 Knihovna Ephemeris

Pro výpočet polohy planet v aplikaci používám knihovnu Ephemeris, napsanou v C++, původně vytvořenou pro Arduino Mega. [17] Knihovna vychází z knihy Jeana Meea Astronomical Algorithms. (Jean Meeus) [18]

## 7.1 Zadání data a souřadnic

Pro výpočet polohy je nejprve třeba zadat zeměpisné souřadnice, tzn. zeměpisnou šířku a délku a čas, ve kterém se mají planety zobrazit.
Příklad:

```
1 // Zadání zeměpisné šířky a délky
2 Ephemeris::setLocationOnEarth(49,35,32, 17,15,48);
3 // Zadání času
4 int day=4,month=9,year=2020,hour=8,minute=0,second=0;
5 // Výpočet souřadnic
6 SolarSystemObject planet = Ephemeris::solarSystemObjectAtDateAndTime
    (Mars, day, month, year, hour, minute, second);
```

Zdrojový kód 1: C++

SolarSystemObject je struktura, která obsahuje množství proměnných jako jsou rovníkové souřadnice, obzorníkové souřadnice, vzdálenost od Země a další.

## 7.2 Průběh výpočtu

Nejprve se datum převede do juliánského kalendáře, potom se poloha planet vypočítá pro jednotlivé planety. Algoritmy pro výpočet polohy využívají zejména informace o oběžné dráze planety — délku vzestupného uzlu, sklon dráhy, šířku pericentra, ekcentricitu atd. [17]

# 8 Lokace

Pro svůj program jsem potřeboval tabulku lokací měst ve světě. Pro tyto informace využívám github projekt. [19] Tabulku ve formátu *csv* jsem propojil s Unreal Enginem. Vytvořil jsem novou datovou strukturu a následně tabulku nahrál ve formátu vytvořené struktury. Časové zóny nejsou součástí tabulky a musí se zadávat v aplikaci manuálně. Je to z důvodu jejich velké složitosti (například Čína používá jen jedno časové pásmo nebo dělení času na letní a zimní, které akceptují jen některé státy).

# 9 MainMenu Level

## 9.1 Spuštění levelu.

Při spuštění levelu se nejprve povolí použití kurzoru myši, následně se vytvoří MainMenuUI Widget. Načte se uložená pozice, která obsahuje poslední zadanou polohu a časovou zónu a určí spuštění hudby.

## 9.2 Rozhraní levelu

Rozhraní se skládá ze dvou částí: MainMenuUI a SettingsUI. MainMenuUI obsahuje: název aplikace, tlačítka do EarthLevelu a SpaceLevelu, tlačítko *Nastavení* (SettingsUI), tlačítko na ukončení aplikace a znak Univerzity Palackého s textem.

Po kliknutí na tlačítko *Země* nebo *Vesmír* se uloží nastavení z proměnných Pawnu MainMenuPawn (Latitude, Longitude, Timezone) do SaveGame slotu a otevře se EarthLevel respektive SpaceLevel. Po kliknutí na tlačítko *Nastavení* se odstraní MainMenuUI z levelu a vytvoří se SettingsUI. SettingsUI se skládá z nadpisu, tlačítka *Zpět*, polí pro uživatelský vstup pro zadání zeměpisné šířky a délky a vstupní pole pro zadání časové zóny. Při vytvoření SettingsUI se z MainMenuPawn nahrají naposledy použitá lokace a časová zóna a zapíší se do vstupních polí. Tlačítko *Zpět* odstraní SettingsUI z levelu a vytvoří MainMenuUI Widget a přidá jej do levelu. Hodnoty uložené v polích pro uživatelský vstup se zapíší do proměnných MainMenuPawn, Latitude a Longitude a Timezone. Uživatel má možnost měnit časovou zónu, zeměpisnou šířku a délku. Uživatelský vstup je ošetřen pro invalidní hodnoty.

## 9.3 Actory v levelu

Level obsahuje dva Actory třídy SpaceOrb, EarthPlanetNight a Sun. EarthPlanetNight je Actor dědící z Actor Planet a ta dědící z Actor SpaceOrb. Statickým modelem obou je koule.

Materiál EarthPlanetNight se skládá z amosféry, což je modrá barva na okrajích 3D modelu, která ma parametr jas a Fresnel, který určuje sytost barvy směrem od středu 3D modelu. Další částí je textura mraků, které se pomocí Panner otáčí kolem modelu. Tyto textury se sčítají s texturou noční Země. Materiál obsahuje také texturu Specular map, což je mapa ovlivňující lesklost výsledného materiálu (voda je lesklejší než zemní plocha). Dalším nastavitelným parametrem je hrubost, též ovlivňující lesk materiálu. Další texturou je normálová mapa, která zprostředkovává iluzi rozdílu výšek v materiálu a nakonec záře atmosféry, která má podobné parametry jako atmosféra a zajišťuje, že planeta svítí ve vesmíru.

Materiál SunMenu dědí z Materiálu Sun. Materiál Sun se skládá z textury Slunce, parametrů Specular a Roughness, určující lesklost a odraživost materiálu a textury Slunce několikrát vynásobené, definující záři Slunce. SunMenu má sílu záření vyšší, než je v materiálu Sun, jinak jde o identický materiál.

Textury jednolitvých planet jsou založeny na pozorováních NASA, konkrétněji sond Messenger, Viking a Cassini a Hubbleova teleskopu. [3]

## 9.4 Sky Sphere

SkySphere je Blueprint v Unreal Enginu znázorňující oblohu levelu. V levelech MainMenuLevel a SpaceLevel používám SpaceSphere. Vytvořil jsem Cubemap texturu, což je textura skládající se ze šesti textur dohromady tvořící krychli. Na to jsem použil program Spacescape, [20] což je freeware program určený na vytváření kubických map vesmíru. Vytvořenou texturu jsem importoval do Unreal Enginu a použil v materiálu MSpace. Tento materiál jsem použil na upravenou verzi SkySphere.

## 9.5 Ostatní prvky

Level obsahuje AmbientSound, který se dá v nastavení vypnout, hudba je použita se souhlasem autora. [21] odlesky na kameře (lensflare), a Volume ovlivňující jas Slunce.

# 10 Earth Level

Level zobrazující planety Sluneční soustavy ze zemského povrchu.

## 10.1 Spuštění levelu.

Při spuštění levelu se povolí použití kurzoru myši v levelu, vytvoří se EarthUI Widget a přidá se do levelu. Nastaví se možnost uživatele interagovat jak s rozhraním, tak levelem.

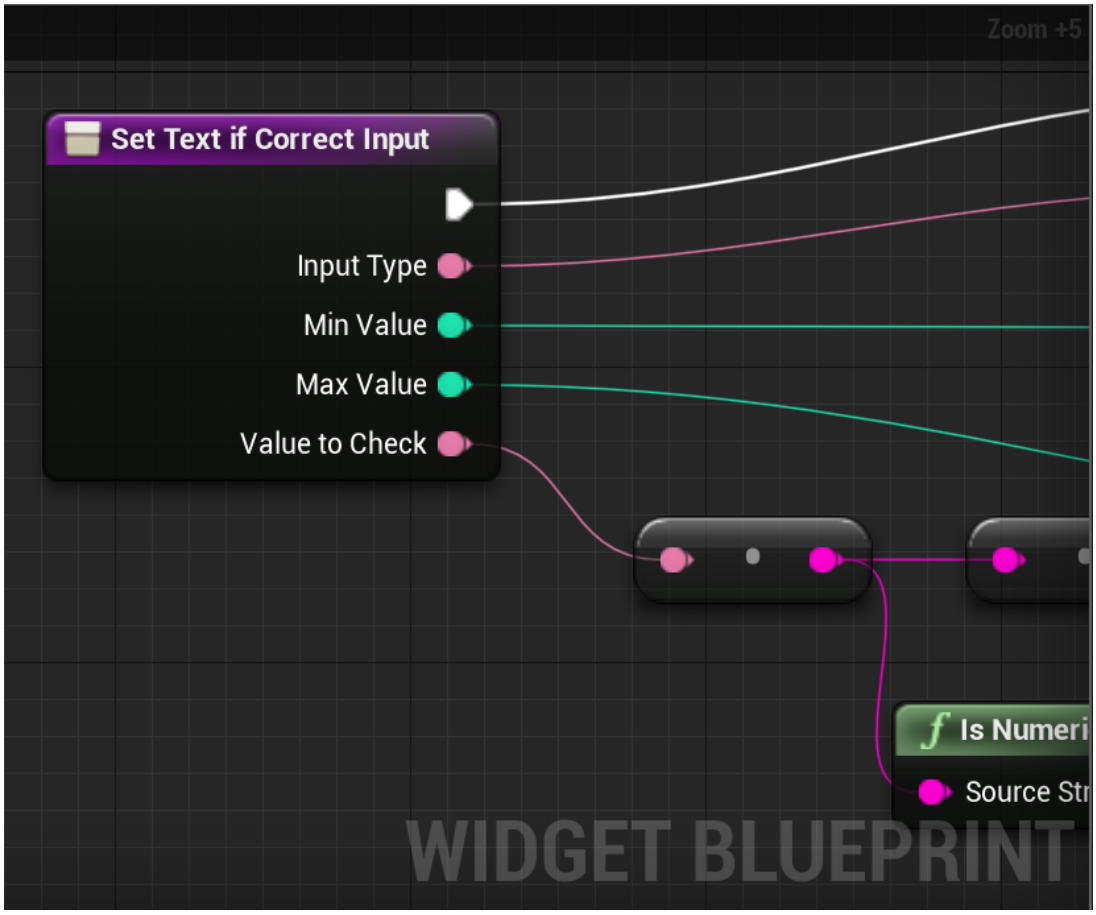
## 10.2 Rozhraní levelu

EarthUI se skládá z několika částí:

- Checkbox RunningState — udává, zda v levelu ubíhá čas a poloha planet se aktualizuje. Grafiku RunningState jsem vytvořil v programu GIMP.
- tlačítko *Vesmír* — otevře SpaceLevel
- FOV slider — pro kameru
- Checkbox PlanetNames — přepíná zobrazení jmen u planet
- Checkbox DegreesVisible — přepíná zobrazení světových stran
- pole pro uživatelský vstup času
- pole pro uživatelský vstup rychlosti

Planety se pohybují, když je RunningState nastaven na *true*. Při změně PlanetNames se pro všechny Actory třídy Planet nastaví visibilita jména na novou hodnotu, stejně tak při změně RunningState se změní bool hodnota proměnné Running, kterou vlastní EarthPawn na novou hodnotu určenou IsChecked. Při změně slideru FOV se nová hodnota předá kameře EarthPawn, která se podle ní aktualizuje.

Pole pro uživatelské vstupy pro každý snímek používají funkce, pomocí kterých získají časovou hodnotu od proměnné Datetime EarthPawnu. Hodnoty jsou ve funkcích dále upravovány pomocí proměnné Timezone, zadané v nastavení. Hodnoty z funkcí se pak v polích pro vstupy zobrazí. Při změně hodnoty ve vstupu uživatelem se běh času zastaví. Po změně hodnoty se upraví poloha planet. Vstupy jsou ošetřeny proti invalidním hodnotám, včetně přestupných roků. Nastavení času v aplikaci je limitováno dolní a horní hranicí. Pokud uživatel změní hodnoty vstupu při běhu času v aplikaci, změní se hodnota proměnné Running, kterou vlastní EarthPawn na *false* a CheckedState Checkboxu RunningState se změní na *unchecked*.




Obrázek 12: Část funkce na kontrolu vstupů

## 10.3 Spuštění EarthPawn

Před událostí BeginPlay, která se spustí po otevření levelu se zkonstruuje EarthPawn, kterého může uživatel ovládat. Načtou se data z SaveGame slotu a přiřadí se hodnoty proměnných Latitude, Longitude EarthPawnu. Timezone hodnota se přiřadí až při otevření levelu. Následně se spustí funkce SpawnPlanets a Planet Moves, která posune planety, Slunce a Měsíc na jejich místa na obloze.

## 10.4 Funkce SpawnPlanets

Po spuštění funkce se nastaví čas, proměnná Datetime. Na to se použíje funkce v enginu UtcNow, která vrátí čas běžící na zařízení. Potom se vytvoří Blueprint třídy Astrolabe. Třída Astrolabe je napsaná v C++ propojující knihovnu Ephemeris s Unreal Enginem. Funkce pokračuje vytvořením Actorů planet podle jejich jmen a jejich umístění do středu levelu (na souřadnice $x,y,z$ (0,0,0)).

## 10.5 Funkce PlanetMoves

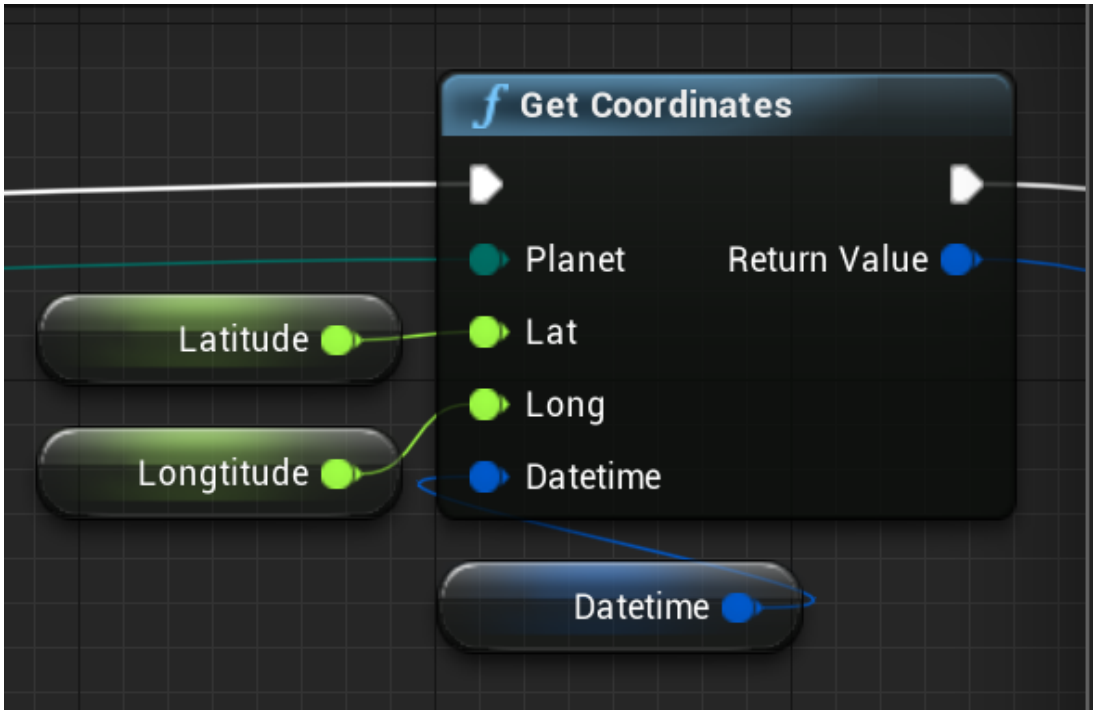
Funkce PlanetMoves slouží pro aktualizaci polohy po změně času, proměnné Datetime uložené v EarthPawnu. Zavolá se funkce GetCoordinates s proměnnými Latitude, Longitude, které uživatel při spuštění levelu načetl z uložené pozice. Další proměnnou která se předá funkci je jméno planety, jejíž poloha se zrovna aktualizuje.

### 10.5.1 Funkce GetCoordinates

Funkce GetCoordinates je napsaná v C++ a propojená s Unreal Enginem skrze Astrolabe Blueprint.

```
1  /* Funkce GetCoordinates používaná v Unreal Enginu
2  EPlanetEnum je je enum třída přiřazující jménu planety číslo
3  FDateTime je datový tip Unreal Enginu*/
4
5  FVector2D UAstrolabe::GetCoordinates(EPlanetEnum Planet, float Lat,
       float Long, FDateTime Datetime)
6  {
7    //Zadání souřadnic
8    Ephemeris::setLocationOnEarth(Lat, Long);
9    int day = Datetime.GetDay(), month = Datetime.GetMonth(), year =
         Datetime.GetYear(), hour = Datetime.GetHour(), minute =
         Datetime.GetMinute(), second = Datetime.GetSecond();
10
11   //Převod jména planety na její index a vytvoření proměnné index s
         aktuálními souřadnicemi zvolené planety
12   SolarSystemObjectIndex p = static_cast<SolarSystemObjectIndex>((
         uint8)Planet);
13   SolarSystemObject planet = Ephemeris::
         solarSystemObjectAtDateAndTime(p, day, month, year, hour,
         minute, second);
14
15   //Horizontální souřadnice vráceny zpět do Unrealu jako 2D vektor
16   return FVector2D(planet.horiCoordinates.azi, planet.
         horiCoordinates.alt);
17 }
```

Zdrojový kód 2: C++




Obrázek 13: Funkce GetCoordinates jako Blueprint v UE

### 10.5.2 Funkce SphereToCartesian

Souřadnice získané funkcí GetCoordinates jsou vráceny ve sférické soustavě souřadnic v prostoru. Pro zobrazení Actorů používá Unreal Engine trojrozměrnou kartézskou soustavu souřadnic v eukleidovském prostoru, souřadnice musíme převést. Funkce SphereToCartesian má dva parametry SphereInput a RadiusInput. SphereInput je 2D Vektor vrácený funkcí GetCoordinates. RadiusInput je mnou určená konstanta určující vzdálenost planet od středu souřadnicového systému. [22] Po převodu souřadnic se pozice Actoru vybrané planety změní na nové souřadnice. Na převod jsou použity tyto funkce:

$$x = rsin\theta cos\phi$$
$$y = rsin\theta sin\phi$$
$$z = rcos\theta$$

## 10.6 Událost EventTick

EventTick je událost, která se spustí při každém snímku aplikace. Pro EarthPawn nejdříve zkontroloje, zda má běžet čas, tzn. bool proměnná je nastavená na *true* a nedosáhlo se horního limitu pro čas. Potom se rychlost zadaná uživatelem, SpeedInput, přidá k proměnné Datetime pomocí funkce SetSpeed. Funkce využívá delta sekundy, které popisují, kolik času uplynulo od předchozího snímku. Nakonec se zavolá funkce PlanetMoves popsaná výše.

EventTick má jako událost i Blueprint EarthLevel. Zde má na starosti měnit parametr ZenithColor podle hodnoty proměnné Datetime, kterou vlastní EarthPawn. Funkce podle hodin a minut z EarthPawnu mění od 06.00-08.00 a 18.00-20.00 barvu oblohy tím, že mění hodnotu *value* barevného modelu Přičítá k hodnotě *value* a od 12.01-24.00 od *value* hodnotu odečítá. HSV model se potom převede do RGB modelu a výslednou barvu přiřadí proměnné ZenithColor získané z reference SkySphere obsažené v levelu. Dalším parametrem, ke kterému se tímto způsobem přičítá respektive odečítá hodnota je StarBrightness.

## 10.7 EarthPawn pohyb

EarthPawn se skládá pouze ze statické kamery, se kterou se lze rozhlížet v levelu. EarthPawnu jsou přiřazeny události MouseY a MouseX zajišťující ovladatelnost ve směru os $x$ a $y$.

## 10.8 Planet Actory

Jak bylo zmíněno, ve funkci SpawnPlanets se vytvoří Actory příslušných planet. Tyto Actory dědí z materiálu MPlanet. U každého materiálu se nastaví parametry EmissiveStrength, Roughness, MainTexture a EmissiveTexture. Výjimkou je Země, která obsahuje navíc ještě několik textur, normálovou texturu, specular texturu, texturu mraků a atmosféru. Každý Actor planety si změní proměnnou HeredityName na jméno, které bude viditelné pro uživatele.

## 10.9 Ostatní komponenty

V levelu se nachází 3D model zemského povrchu, který má tvar oválu. Materiál Grass, který je na tento model aplikován je upravený materiál, který je obsažen v Unreal Enginu jako jeden ze základních materiálů. Model planet (detailnější koule) a světové strany jsou mnou vytvořené 3D modely v programu Blender [23] a importovány do Unreal Enginu. Level obsahuje Volumes na úpravu odrazů a osvětlení.

# 11 Space Level

Level zobrazující polohu planet Sluneční soustavy z vesmíru.

## 11.1 Spuštění levelu.

Při spuštění levelu se povolí použití kurzoru myši v levelu, vytvoří se SpaceUI Widget a přidá se do levelu. Nastaví se možnost uživatele interagovat jak s rozhraním, tak levelem.

## 11.2 Rozhraní levelu

SpaceUI se skládá z několika částí:

- Checkbox RunningState — udává, zda v levelu ubíhá čas a poloha planet se aktualizuje
- tlačítko *Země* — otevře SpaceLevel
- Checkbox PlanetNames — přepíná zobrazení jmen u planet
- pole pro uživatelský vstup času
- pole pro uživatelský vstup rychlosti

Pole pro uživatelské vstupy pro každý snímek používají funkce, pomocí kterých získají časovou hodnotu od proměnné Datetime SpacePawnu. Hodnoty z funkcí se pak v polích pro vstupy zobrazí. Při změně hodnoty ve vstupu uživatelem se běh času zastaví. Po změně hodnoty se upraví poloha planet. Vstupy jsou ošetřeny proti invalidním hodnotám, včetně přestupných roků. Nastavení času v aplikaci je limitováno dolní a horní hranicí. Pokud uživatel změní hodnoty vstupu při běhu času v aplikaci, změní se hodnota proměnné Running, kterou vlastní SpacePawn na *false* a CheckedState Checkboxu RunningState se změní na *unchecked*.

## 11.3 Spuštění SpacePawn

Před událostí BeginPlay, která se spustí po otevření levelu se zkonstruuje SpacePawn ovládaný uživatelem a zavolají se funkce SpawnPlanet, SpawnOrbits a PlanetMoves.

## 11.4 Funkce SpawnPlanets

Nastaví čas, proměnná Datetime. Na to se použíje funkce v enginu UtcNow, která vrátí čas běžící v zařízení. Vytvoří se Blueprint třídy Astrolabe. Potom se podle jmen planety vytvoří jejich Actory a umístí do středu levelu (souřadnice *x,y,z* (0,0,0)). Rozdíl je v tom, že v SpaceLevelu nezůstavá velikost planet po celou

dobu běhu levelu stejná. Velikost planety totiž záleží na vzdálenosti kamery od středu levelu.

## 11.5 Funkce SpawnOrbits

Každá z planet ve vesmíru má viditelnou oběžnou dráhu, po které obíhá kolem Slunce. Funkce SpawnOrbits vezme všechny Actory třídy PlanetOrbit a pro každého spustí funkci SpawnPlanetOrbit, což je funkce Actoru PlanetOrbit.

### 11.5.1 Funkce SpawnPlanetOrbit

Každá oběžná dráha planety dědí z třídy PlanetOrbit a nastavuje si hodnotu proměnných NumberOfDays a NumberOfJumps, PlanetToOrbit a Orbit PlanetMaterial. Oběžná dráha planety se vytvoří za pomocí funkce PlanetMoves a komponentů Spline křivky. Následně se spustí *for* cyklus od indexu 0 do hodnoty proměnné NumberOfJumps. V tomto cyklu se přičítá hodnota proměnné NumberOfDays k proměnné Datetime SpacePawnu a zavolá se funkce PlanetMoves, která posune planetu na novou polohu. Pomocí proměnné PlanetToOrbit se identifikuje planeta, pro kterou počítáme oběžnou dráhu a jejímu spline komponentu přidáme SplinePoint. Přidaný SplinePoint bude mít souřadnice právě posunuté planety. Na konci cyklu bude Spline oběžné dráhy planety mít NumberOfJumps bodů, jejichž vzdálenost od sebe je určena funkcí PlanetMoves a proměnnou NumberOfDays. Po výpočtu těchto bodů se zavolá funkce SpawnOrbitMesh. Na konci funkce se načte původní hodnota proměnné Datetime uložená do pomocné proměnné na začátku funkce.

### 11.5.2 Funkce SpawnOrbitMesh

V cyklu se projdou všechny body Spline komponentu a mezi každé dva se vloží 3D model válce (StaticMesh). Každé této části je přiřazen materiál PlanetMaterial a nastaven směr umístění modelu. Každému SplinePointu je změněno měřítko (scale).

## 11.6 Funkce PlanetMoves

Funkce PlanetMoves slouží pro aktualizaci polohy po změně času, proměnné Datetime v SpacePawn. Spustí se funkce GetHelioCoordinates s proměnnou Datetime a planetou, pro kterou se momentálně poloha počítá.

### 11.6.1 Funkce GetHelioCoordinates

Funkce GetHelioCoordinates je funkce napsaná v C++ propojená s Unreal Enginem skrze Astrolabe. Zaregistroval jsem, že rovníkové souřadnice, se kterými pracuje základní funkce knihovny Ephemeris, solarSystemObjectAtDateAndTime, jsou vypočítány pomocí heliocentrických souřadnic. Z informace o poloze planet

ve vesmíru se vypočítá poloha planet na obloze. Funkce nebyla přístupná, proto jsem změnil modifikátor jejího přístupu.

```
1  /* Funkce GetHelioCoordinates používaná v Unreal Enginu
2  EPlanetEnum je je enum třída přiřazující jménu planety číslo
3  FDateTime je datový tip Unreal Enginu*/
4
5  FVector UAstrolabe::GetHelioCoordinates(EPlanetEnum Planet,
       FDateTime Datetime)
6  {
7
8    int day = Datetime.GetDay(), month = Datetime.GetMonth(), year =
         Datetime.GetYear(), hours = Datetime.GetHour(), minutes =
         Datetime.GetMinute(), seconds = Datetime.GetSecond();
9
10   //Převod na juliánský den, s kterým funkce pracuje
11   JulianDay jd = Calendar::julianDayForDateAndTime(day, month, year,
         hours, minutes, seconds);
12   FLOAT T = T_WITH_JD(jd.day, jd.time);
13
14   //Převod jména planety na její index
15   SolarSystemObjectIndex p = static_cast<SolarSystemObjectIndex>((
         uint8)Planet);
16   HeliocentricCoordinates helio = Ephemeris::
         heliocentricCoordinatesForPlanetAndT(p, T);
17
18   //Heliocentrické souřadnice vráceny zpět do Unrealu jako 2D vektor
19   return FVector(helio.lat, helio.lon, helio.radius);
20 }
```

Zdrojový kód 3: C++

### 11.6.2 Funkce HelioToCartesian

Heliocentrické ekliptické souřadnice (obdélníkové souřadnice) vráceny funkcí GetHelioCoordinates, aby byly použitelné v Unreal Enginu musí být převedeny do kartézského systému souřadnic. [24]

$$x = rcos\alpha cos\delta$$

$$y = rcos\alpha sin\delta$$

$$z = rcos\delta$$

## 11.7 Událost EventTick

Událost EventTick je u SpacePawn identická jako u EarthPawn, spustí se při každém snímku aplikace. Nejdříve zkontroluje, zda má běžet čas, bool proměnná

Running je nastavená na *true*. A zkontroluje, zda se nepřesáhla maximální hodnota času. Následně se vezme rychlost zadaná uživatelem v SpeedInput a přičte se k proměnné Datetime. Zavolá se zavolá funkce PlanetMoves popsaná výše, která aktualizuje polohu planet.

## 11.8 SpacePawn pohyb

Actor SpacePawn se skládá z kamery a komponentu SpringArm. [25] SpringArm komponent slouží k rozšíření, zúžení zorného pole a vzdálenosti kamery od Actoru. SpacePawn také používá události MouseY a MouseX, které zajišťují ovladatelnost ve směru os $x$ a $y$. To znamená uživatel aplikace se může rozhlížet. Díky SpringArm komponentu se ale může rozhlížet ve všech úhlech. Je v tom omezen délkou SpringArm.

### 11.8.1 Zoom

Délku SpringArm lze ovlivnit pomocí kolečka myši, kterému jsou přiřazeny události MouseWheelUp a MouseWheelDown pro obě možnosti pohybu. U těchto událostí se nejprve zkontroluje, jestli kamera nepřekročila minimální nebo maximální vzdálenost, nemůže se více přiblížit, respektive vzdálit od centra levelu (souřadnic $x,y,z$ (0,0,0)). Pokud je tato podmínka splněna, zavolá se funkce Zoom. Ta vynásobí velikost SpringArm s hodnotou funkci předané. V případě MouseWheelUp je to záporná hodnota, délka SpringArm se zmenší. V případě MouseWheelDown je to kladná hodnota, délka SpringArm se zvětší. Aby zůstaly planety a oběžné dráhy vizuálně stejně velké, zvětšují se spolu se SpringArm.

U planet se spustí *for* cyklus na všechny Actory třídy SpaceOrb, planety a Slunce, pro každou se spočítá nová hodnota Scale. Délka SpringArm se vydělí mimimální vzdáleností kamery, kam až se může uživatel přiblížit a vynásobí s proměnnou StartingPlanetScale označující začáteční měřítko planet. U oběžných drah se zavolá *for* cyklus pro všechny Actory třídy PlanetOrbit a pro každou oběžnou dráhu se zavolá *for* cyklus na její části. Proměnná StartingOrbitScale se stejně jako u planet vynásobí s délkou SpringArm a vydělí mimimální vzdáleností kamery. U každé části se nastaví měřítko počátečních bodů a měřítko koncových bodů. Bohužel nelze nastavit měřítko všech částí Spline naráz.

## 11.9 Ostatní prvky

Pro PlanetActory platí to stejné jako v EarthLevel. I v tomto levelu jsou Volumes ovlivňující interakci planet se světlem.

# 12 Testování

Díky možnosti spouštění aplikace přímo v editoru se chyby vzniklé při vývoji velmi snadno odhalí. Při vývoji lze aplikaci spouštět přímo a chyby tak rychle odhalit. Testování funkcí, materiálů, Blueprintů apod. probíhalo tedy dynamicky.

# 13 Programátorská příručka

Složka SolarSystem v přiloženém DVD obsahuje projekt, ve kterém byla tato aplikace vyvíjena. Ten obsahuje všechny Blueprinty, Actory, Levely, Widgety a funkce popsané v této práci. Po instalaci Unreal Enginu lze projekt spustit pomocí souboru: SolarSystem.uproject.

# 14 Uživatelská příručka

Aplikace se spouští pomocí souboru SolarSystem.exe ve složce SolarSystemApplication. Po spuštění aplikace je uživatel v menu. Odtud může otevřít level *Země*, level *Vesmír*, *Nastavení*, nebo aplikaci ukončit. Uživatel může uživatel zadávat vstupy jako rychlost, datum a zapínat běh času. Ovládání je popsáno v souboru README.TXT.

# Závěr

Dnešní možnosti vývoje aplikací jsou obrovské. I když jsem ze začátku neměl žádnou zkušenost s vývojem aplikací v herních enginech, díky studiu na katedře informatiky, online dokumentaci a tutoriálům[26] jsem neměl problém tyto technologie rychle pochopit a využít nabyté vědomosti k vytvoření aplikace popsané v této práci. Výsledkem je graficky dobře vypadající aplikace pro astronomy a nadšence do vesmíru, ve které si mohou nastavit datum a sledovat polohu planet Sluneční soustavy jak na zemském povrchu, tak z pohledu vesmíru.

# Conclusions

Today's application development capabilities are enormous. Despite having no previous experience with application development in game engines, I quickly managed to understand these technologies, and apply the acquired knowledge in creating the application described in this thesis. I was helped by my studies of Informatics at this institute, as well as by the online documentation and tutorials. [26] The result is a graphically good-looking application for astronomers and space enthusiasts, in which they can set the date and observe position of planets in the Solar System's on the Earth's surface or from the outer space.

# A Obsah přiloženého CD/DVD

Obsah DVD:

**bin/**
: Pokud není na počítači nainstalovný DirectX verze June 2010 nebo novější, je třeba jej nainstalovat ze souboru: directxJun2010redist.exe obsaženém ve složce. Aplikace se spouští SolarSystem.exe.

**doc/**
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

**src/**
: Ve složce Source je knihovna Ephemeris společně s Astrolabe popsané v této práci. Ve složce Content lze nalézt použité fonty, textury a zvuky. Vytvořené třídy a funkce jsou součástí projektu SolarSystem.uproject. Návod na spuštění projektu je v souboru readme.txt.

**readme.txt**
: Instrukce pro instalaci a spuštění Unreal Enginu 4.22.3 ve kterém byla aplikace vyvíjena. Popisuje také ovládání v levelech aplikace.

# Literatura


[1] TEAM, Stellarium. *Stellarium* [online]. [cit. 2020-8-16]. Dostupný z: ⟨https://stellarium-web.org/⟩.

[2] SKYLIVE. *The SkyLIVE* [online]. [cit. 2020-8-16]. Dostupný z: ⟨https://theskylive.com/3dsolarsystem⟩.

[3] TEAM, Solar System Scope. *Solar Textures* [online]. [cit. 2020-8-8]. Dostupný z: ⟨https://www.solarsystemscope.com/⟩.

[4] ŠMÍD, Antonín. Comparison of Unity and Unreal Engine. [online]. 2017, [cit. 2020-8-4]. Dostupný z: ⟨https://dcgi.fel.cvut.cz/theses/2017/smidanto⟩.

[5] CHRISTOPOULOU, Eleftheria; XINOGALOS, Stelios. Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *International Journal of Serious Games*. 2017, vol. 4, s. 21–36. Dostupný také z: ⟨http://dx.doi.org/10.17083/ijsg.v4i4.194⟩.

[6] EPIC GAMES, INC. *Blueprints Visual Scripting* [online]. [Cit. 2020-8-4]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html⟩.

[7] EPIC GAMES, INC. *Unreal Engine 4 Features* [online]. [Cit. 2020-8-4]. Dostupný z: ⟨https://www.unrealengine.com/en-US/features⟩.

[8] ANDRADE, A. Game engines: a survey. *EAI Endorsed Transactions on Game-Based Learning*. 2015, roč. 2, s. 150615. Dostupný také z: ⟨http://dx.doi.org/10.4108/eai.5-11-2015.150615⟩.

[9] EPIC GAMES, INC. *Unreal Engine 4 Levels* [online]. [Cit. 2020-8-4]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Engine/Levels/index.html⟩.

[10] EPIC GAMES, INC. *Unreal Engine 4 AActors* [online]. [Cit. 2020-8-4]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/API/Runtime/Engine/GameFramework/AActor/index.html⟩.

[11] EPIC GAMES, INC. *Unreal Engine 4 UMG UI Designer* [online]. [Cit. 2020-8-4]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Engine/UMG/index.html⟩.

[12] EPIC GAMES, INC. *Unreal Engine 4 Pawn* [online]. [Cit. 2020-8-6]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Gameplay/Framework/Pawn/index.html⟩.

[13] *GNU Image Manipulation Program*. [online]. [cit. 2020-8-16]. Dostupný z: ⟨https://www.blender.org/⟩.

[14] MENDELOVA UNIVERZITA, Brno. *Textury* [online]. [Cit. 2020-8-6]. Dostupný z: ⟨https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=23680⟩.

[15] EPIC GAMES, INC. *Unreal Engine 4 Textures* [online]. [Cit. 2020-8-6]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Engine/Content/Types/Textures/index.html⟩.

[16] EPIC GAMES, INC. *Unreal Engine 4 Materials* [online]. [Cit. 2020-8-6]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/index.html⟩.

[17] MARCHAND, Sebastian. *Ephemeris* [online]. [cit. 2020-8-6]. Dostupný z: ⟨https://github.com/MarScaper/ephemeris⟩.

[18] MEEUS, Jean H. *Astronomical Algorithms*. 1991. ISBN 0943396352.

[19] BAHAR, Habib Ullah. *Ephemeris* [online]. [Cit. 2020-8-8]. Dostupný z: ⟨https://github.com/bahar/WorldCityLocations⟩.

[20] PETERSON, Alex. *Spacescape* [online]. [cit. 2020-8-16]. Dostupný z: ⟨http://alexcpeterson.com/spacescape/⟩.

[21] SKYY, Electric. *Relaxation music- deep space* [online]. [cit. 2020-8-16]. Dostupný z: ⟨https://www.youtube.com/watch?v=VVGb7JU4TSU⟩.

[22] MGR. ING. ARCH. PETR KURFÜRST, Ph.D. *Kulové (sférické) souřadnice* [online]. [Cit. 2020-8-9]. Dostupný z: ⟨https://www.shorturl.at/aeoz0⟩.

[23] FOUNDATION, The Blender. *blender* [online]. [cit. 2020-8-16]. Dostupný z: ⟨https://www.blender.org/⟩.

[24] SPACE ASTRONOMY, Belgian Institute for. *Coordinate systems and transformations* [online]. [Cit. 2020-8-11]. Dostupný z: ⟨https://www.spenvis.oma.be/help/background/coortran/coortran.html⟩.

[25] EPIC GAMES, INC. *Unreal Engine 4 SpringArmComponent* [online]. [Cit. 2020-8-11]. Dostupný z: ⟨https://docs.unrealengine.com/en-US/Gameplay/HowTo/UsingCameras/SpringArmComponents/index.html⟩.

[26] ULIBARRI, Stephen. *Unreal Engine 4 Tutorials* [online]. [cit. 2020-8-12]. Dostupný z: ⟨https://www.udemy.com/user/stephen-ulibarri-3/⟩.