Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj aplikace pro simulaci hospodářského cyklu: vytvoření herní logiky a implementace vlivů hospodářských politik

Vypracoval: Miroslav Kolesnik

Vedoucí práce: Mgr. Radim Remeš, Ph.D.

České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miroslav KOLESNIK**
Osobní číslo:	**E20044**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vývoj aplikace pro simulaci hospodářského cyklu: vytvoření herní logiky a implementace vlivů hospodářských politik**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je naprogramovat herní aplikaci, která bude fungovat jako simulátor hospodářského cyklu. Pomocí nástrojů hospodářské politiky bude možné ovlivňovat makroekonomické indikátory. Autor naprogramuje funkční logiku aplikace, tj. pouze back-endovou část, bez uživatelského prostředí. Vývoj bude probíhat v herním engingu Unity a v programovacím jazyku C#.

Metodický postup:

1. Studium odborné literatury.
2. Popis použitých technologií pro vývoj aplikace.
3. Návrh, popis vývoje a implementace aplikace.
4. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.

Rozsah pracovní zprávy:	**40-50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Baron, D. (2021). *Game Development Patterns with Unity 2021.* Birmingham, Spojené království: Packt Publishing.
2. Brusca, V. (2021). *Advanced Unity Game Development: Build Professional Games with Unity, C#, and Visual Studio.* New York Apress.
3. Hardman, C. (2020). *Game Programming with Unity and C#: A Complete Beginner´s Guide.* New York, USA: Apress.
4. *Hands-On Unity 2021 Game Development.* (2021). Birmingham, Spojené království: Packt Publishing.

Vedoucí bakalářské práce: **Mgr. Radim Remeš, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **22. března 2023**
Termín odevzdání bakalářské práce: **12. dubna 2024**

L.S.

doc. RNDr. Zuzana Dvořáková Líšková, Ph.D.
děkanka

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 23. března 2023

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum: 14.4.2023 Podpis studenta: Kolesnik

Poděkování

Děkuji panu Mgr. Radimovi Remešovi, Ph.D. za trpělivost, cennou zpětnou vazbu a konstruktivní kritiku poskytnutou při psaní této bakalářské práce. Dále děkuji celému týmu – Jakubovi Tupému, Pavlovi Kocianovi a Matějovi Přibylovi za spolupráci na projektu.

Obsah

1. Úvod

Cílem této práce je naprogramovat backendovou část 2D videohry ve které uživatel pomocí nástrojů fiskální a monetární politiky ovlivňuje průběh hospodářského cyklu.

Toto téma jsem si vybral za prvé, protože programování a vývoj videoher jsou nejen mým koníčkem ale i profesionálním zaměřením a za druhé, jelikož kombinuje obě části mého oboru, tj. ekonomika a informatika.

Výsledná hra by měla zajímavým a poutavým způsobem učit její uživatele makroekonomickou teorii fiskálních a monetárních nástrojů, které vlády zemí používají pro ovlivňování makroekonomických ukazatelů hospodářského cyklu jako jsou například HDP, inflace, nezaměstnanost aj.

Tato práce je součástí skupinové práce a zabývá se pouze částí vývoje backendu videohry, nezabývá se tedy vývojem uživatelského rozhraní a nerozebírá makroekonomickou teorii která se zabývá fiskální a monetární politikou – tyto oblasti jsou součástí bakalářských prací mých spolupracovníků.

V teoretické části seznámím čtenáře se základními pojmy v problematice vývoje softwaru a videoher, rozeberu některé druhy a žánry videoher, vysvětlím pojem IDE a mezi jakými si lze vybírat při vývoji v Unity, popíšu hlavní pojmy a prostředí v Unity, a na konec vysvětlím zásadní fáze při vývoji videoher.

Na začátku praktické části stručně popíšu, jak probíhala plánovací a před produkční fáze vývoje videohry. Dále ukážu proces aktualizace pracovního prostředí pomocí správy verzí Plastic SCM pro spolupráci v týmu. Potom vysvětlím, jaká jsou pravidla hry a proces jejího průběhu a na konec ukážu, jakým způsobem jsem makroekonomickou logiku implementoval do kódu a jak pak tento kód byl použit při konstrukci videohry.

2. Základní pojmy

2.1 Videohra

„Elektronická hra, také nazývaná počítačová hra nebo videohra je jakákoli interaktivní hra provázená počítačovými obvody. Zařízení nebo "platformy", na kterých se hrají elektronické hry zahrnují sdílené a osobní počítače pro obecné účely, arkádové konzole, video konzole připojené k domácím televizorům, ruční herní zařízení a mobilní zařízení, jako jsou mobilní telefony a serverové sítě. Termín videohra může být použit k reprezentaci všech těchto formátů, nebo může odkazovat pouze na hry hrané na zařízeních s video displeji: televizi a arkádových konzolích“ (Lowood, 2021).

2.2 Herní engine

Herní Engine je software, který je rozšiřitelný a dá se použít jako základ pro tvorbu videoher bez značných modifikací softwaru. Herní enginy jsou často vytvořené pouze pro specifickou hru či pro specifický žánr hry (např. střílečka, závodní hry nebo multiplayer hry) společností, která danou hru vyvíjí. Existují však i obecné víceúčelové druhy herních enginů aplikovatelných pro různé druhy videoher, tato výhoda však často znamená, že je víceúčelový engine méně optimalizovaný pro konkrétní žánr videohry než engine, který je vytvořený specificky pro daný žánr videohry (Gregory, 2015).

Hlavní výhodou použití herního enginu při tvorbě videohry je, že v sobě již obsahuje potřebné nástroje pro tvorbu videoher jako jsou grafika, kolize a fyzika, animace, rendering, vizuální efekty, audio, nástroje pro debugging, input systém, networking a mnoho dalších.

2.3 Unity

Unity je velmi používané, výkonné multiplatformní vývojové prostředí a runtime engine pro hry podporující širokou škálu platforem. Pomocí Unity mohou vývojáři distribuovat své hry na velkém množství mobilních konzolových a počítačových operačních systémů. Dokonce podporuje Webplayer pro nasazení na všechny hlavní webové prohlížeče (Gregory, 2015).

Poskytuje přehledný editor, ve kterém lze manipulovat s objekty, které tvoří danou scénu a možnost přímo testovat herní funkčnost v editoru bez buildování aplikace, což výrazně šetří čas. Dále poskytuje celou řadu nástrojů a knihoven pro vývoj videoher (viz. 2.1). Starší verze Unity podporovali skriptovací jazyky UnityScript, C# a Boo. Nové verze Unity však podporují pouze C#.

Jako herní engine pro vývoj tohoto projektu náš tým zvolil Unity za prvé, protože je to bezplatný herní engine s intuitivním uživatelským rozhraním a velkým množstvím dokumentace a návodů dostupných na internetu jak v textové, tak ve video formě, a za druhé, protože jako skriptovací jazyk používá C#, se kterým jsem dobře seznámený.

2.4 Programovací jazyk

Programovací jazyk je jakýmkoli z různých jazyků sloužících k vyjádření souboru instrukcí pro počítač. Tyto instrukce mohou být spuštěny přímo, pokud jsou ve formě strojového jazyka, nebo po jednoduchém substitučním procesu, pokud jsou vyjádřeny v jazyce symbolických adres (Assembly), nebo po překladu „vyššího“ programovacího jazyka (Hemmendinger, 2022).

Dle Hemmendingera (2022) Existuje mnoho programovacích jazyků pro různé účely:

- Strojový kód nebo Assembly pro přímou komunikaci s počítačem
- Algoritmické programovací jazyky pro vyjádření matematických nebo symbolických výpočtů
- Databázové programovací jazyky jako např. SQL
- Vzdělávací jazyky, které mají zjednodušenou syntaxi a jsou zaměřené na snazší výuku programování pro začátečníky nebo studenty škol
- Objektově orientované jazyky, což jsou jazyky, které fungují na principu „všechno je objekt“ což zjednodušuje programování a organizaci kódu
- Deklarativní jazyky, kde programátor spíše specifikuje cíl nebo čeho by mělo být dosaženo, než jak by mělo být cíle dosaženo
- Programovací jazyky pro formátování dokumentů
- Značkovací jazyky, používané na obohacení textu na webových stránkách
- Skriptovací jazyky pro web

2.5 Správa verzí

Správa verzí (známá také jako správa zdrojového kódu) používá nástroje ke sledování změn nebo úprav provedených ve zdrojovém kódu. Umožňuje rychlou a efektivní spolupráci mezi vývojáři a zachovává integritu kódu. Díky tomu umožňuje týmům vývojářů pracovat bez obav v konfliktu v kódu (Unity Technologies, 2022-b).

V prostředí softwarového inženýrství téměř vždy na vývoji softwaru spolupracuje tým programátorů. Každý programátor pracuje na vlastním zařízení, často nejen na pracovním ale i

na osobním, což znamená, že je potřeba mít nástroj, který týmu umožní zdrojový kód jednoduše stahovat, modifikovat, vytvářet nová odvětví a mít zálohy starých verzí a možnost se k těmto zálohám jednoduše vrátit pro případ, že by se projekt nějakým způsobem pokazil nebo by bylo potřeba vrátit zpět velkou část změn. Těmto nástrojům se říká nástroje pro správu zdrojového kódu (Source Code Management tools – SCM). Bez nástroje pro správu zdrojového kódu jsou tyto činnosti velmi obtížné a zabrali by výrazně více času. Programátoři by totiž museli manuálně posílat každou změnu v kódu což by kromě časové náročnosti téměř jistě vedlo k výrazným problémům ve funkci softwaru kvůli lidskému faktoru (Unity Technologies, 2022-b).

Systémy pro správu verzí se dělí na dva hlavní typy – centralizované a distribuované. Hlavní rozdíl je, že v centralizovaných systémech se vývojář napojí přímo na server a provádí změny na něm, zatímco v distribuovaným systému vývojáři mají na svém zařízení lokálně vlastní repozitář a změny sdílí pomocí vzdáleného repozitáře buď na serveru nebo přímo přes síť (Otte, 2009).

Podle Otte (2009) mezi základní pojmy správy verzí patří:

- **Repozitář (Repository)** – Ať už počítač nebo server, na kterém jsou uloženy všechny sledované soubory včetně jejich historie.
- **Revize / Verze (Revision / Version)** – Každá fáze v historii souboru je identifikovatelná revizí nebo verzí (typicky číslo.) Revize obsahuje soubor a jeho metadata. Uložená metadata se mohou lišit dle specifického používaného programu. Nejnovější revize se často nazývá hlavou.
- **Vyzvednutí (Checkout)** – Uživatel může provést vyzvednutí buď na specifickém souboru/souborech, nebo na celém repozitáři. Pomocí checkoutu lze aktualizovat repozitář.
- **Zapsání změn (Commit)** – Potvrzení změn v repozitáři. Typicky obsahuje commit komentář popisující provedené změny.
- **Aktualizace (Update)** – Aktualizování změn provedených jinými členy v repozitáři
- **Sloučení (Merge)** – Spojení dvou repozitářů do jednoho
- **Konflikt (Conflict)** – Konflikt mezi obsahem stejných souborů ve dvou repozitářích, při řešení konfliktů lze typicky zvolit ponechání buď zdroje (zdroj ze kterého se stahuje aktualizace) nebo destinace (místo do kterého se aktualizace kopíruje)
- **Větev (Branch)** – Odvětví z hlavního repozitáře

3. Druhy a žánry videoher

Od začátků herní industrie vzniklo mnoho druhů a žánrů videoher. Žánry se často kříží a kombinují s dalšími žánry a vytvářejí další pod-žánry. Všechny videohry se dají rozdělit do dvou hlavních kategorií – dvourozměrné hry (2D) a třírozměrné hry (3D).

3.1 Sandboxové videohry

V Sandboxové hře často hráč hraje nelineárně, má otevřené prostředí a má sám se rozhoduje to tom co bude zrovna dělat. Hráč nemá pevně stanovené cíle, nebo směr, kterým by měl postupovat. Cíle ve hře lze splnit jakýmkoli způsobem a kdy se hráči zachce. Sandboxové hry poskytují hráči více pohlcující zážitek a podporuje experimentaci. Mezi známé Sandboxové tituly patří například hry Minecraft, Grand Theft Auto, nebo The Sims (Pavlovic, 2020).

3.2 Strategie v reálném čase

Strategie v reálném čase, označované zkratkou RTS (Real time strategy) jsou jedním z nejstarších žánrů videoher. V typické RTS hře hráči a umělý hráči ovládají frakce a soupeří mezi sebou v „reálném čase“. Tyto hry většinou obsahují ovládání herních zdrojů a mapy, na kterou shlíží shora dolů. Mezi známé RTS tituly patří například Warcraft, Age of Empires, nebo Command & Conquer (Pavlovic, 2020).

3.3 Střílečky

Střílečky jsou také jeden z žánrů, které existují prakticky od počátku herní industrie. Dělí se primárně na tzv. FPS a TPS druhy. V FPS střílečkách vidí hráč svět z pohledu první osoby čili tak jak vidí svět každý člověk, zatímco v TPS hrách vidí hráč svoji postavu ze shora. Některé videohry umožňují obě tyto zobrazení, takže je lze klasifikovat jako FPS i TPS střílečky. Premisa těchto videoher je poměrně samovysvětlující – hráč má zbraně a střílí po nepříteli, ať už jinému reálnému hráči nebo umělému nepříteli. Tento žánr se často kombinuje s jinými žánry videoher jako například ve hře Grand Theft Auto 5. Mezi tradiční FPS/TPS hry patří například hry HALO, Gears of War nebo DOOM (Pavlovic, 2020).

3.4 Online bojové arény pro více hráčů

Tento žánr se označuje zkratkou MOBA (Multiplayer online battle arena) a je to žánr který je z části podobný strategiím. Týmy hráčů proti sobě soupeří o kontrolu mapy a nahlíží na ni z horního pohledu. Kombinuje ovládání zdrojů a mapy s kompeticí mezi individuálními hráči. Hlavní rozdíl mezi MOBO a RTS žánry je, že v MOBO hráč typicky ovládá jen jednu postavu,

zatímco v RTS hrách ovládá skupiny postav. V MOBA hrách má také prioritu multiplayer, spolupráce a kompetice mezi hráči (Pavlovic, 2020).

3.5 RPG hry

Zkratka RPG je akronymem „Role Playing Game“, čili hra, ve které hráč hraje jakousi fiktivní roli. Základní premise RPG her je vytvoření a ovládání své vlastní postavy, která má úroveň a s postupem se času stává čím dál lepší a silnější tím, že se úroveň zvedá bojováním, plněním úkolů, prozkoumáváním mapy a další. RPG žánr se postupem času vyvinul do mnoha dalších pod-žánrů jako například ARPG, CRPG, MMORPG a další. Každý pod-žánr má vlastní charakteristiky a hratelnost. Mezi známé RPG hry patří například Skyrim, The Witcher 3, nebo Fallout 4 (Pavlovic, 2020).

3.6 Sportovní hry

Žánr sportovních her je velmi rozmanitý a zahrnuje všechny druhy sportů jako například fotbal, basketbal, hokej ale i závody aut a mnoho dalších. Sportovní hry jsou jedním z největších a nejvýdělečnějších druhů videoher v herní industrii nejen kvůli obrovskému počtu sportovních fanoušků po celém světě ale i protože jsou často sponzorovány nebo dokonce vytvářeny obrovskými sportovními společnostmi jako je NBA, NFL nebo FIFA (Pavlovic, 2020).

3.7 Párty hry a hlavolamy

Hlavolamy a párty hry se často navzájem protínají. Oba kladou důraz na herní mechaniky a můžeme od nich očekávat hru založenou na tématu nebo na tradiční stolní videohře či kartové hře, přičemž párty hry jsou většinou určené pro více hráčů a kladou větší důraz na hratelnost, zatímco v hlavolamech je hlavním cílem řešení problémů. Oba žánry mohou bát dost různorodé a velmi často se mixují s jinými herními žánry. Mezi známé párty videohry patří například Jackbox Party Pack a do hlavolamů lze zařadit například The Talos Principle nebo Portal 2 (Pavlovic, 2020).

3.8 Akční dobrodružství

Akční dobrodružství se hluboce zaměřují na příběhové zápletky a bojové mechaniky prostřednictvím zapojení příběhu a těsné herní mechaniky. Tato kategorie, stejně jako mnoho dalších herních žánrů je velmi různorodá a může zahrnovat celou řadu videoher. Mezi první videohry tohoto žánru patří například hra Legend of Zelda, která dodnes slouží jako inspirace

pro jiné hry tohoto žánru. Do této kategorie můžeme zařadit například hry jako Star Wars Jedi: Fallen Order, Sekiro: Shadows Die Twice, nebo Assassin's Creed franšízu (Pavlovic, 2020).

3.9 Hry o přežití

Jak vyplývá z jejich názvu, v těchto videohrách je hlavním cílem hráče přežívat ve vytvořeném prostředí pomocí spravování a využití dostupných zdrojů. Témata těchto videoher se většinou inspirují přežitím v divočině, na opuštěném ostrově nebo v postapokalyptickém prostředí. Většinou zahrnují herní mechaniky pro výrobu předmětů, které jim pomáhají přežívat, sběru a využití předmětů, které se nachází v okolí a stavby různých přístřešků nebo i budov. Mezi známé tituly tohoto žánru patří například Don't Starve, The Forst nebo nově například Valheim (Pavlovic, 2020).

3.10 Platformer hry

Platformer hry jsou jedním z nejstarších žánrů videoher a na rozdíl od ostatních žánrů jsou v základu poměrně stejnorodé ale také jsou jedním z nejpopulárnějších žánrů na světě. Jsou to 2D hry, ve kterých se hráč různě pohybuje, skáče, leze, střílí a prozkoumává své okolí po cestě k cíli. Vznikli na základě klasických videoher jako jsou Donkey Kong, Super Mario Bros nebo Sonic the Hedgehog. Mezi novější populární tituly tohoto žánru patří například Cuphead (Pavlovic, 2020).

3.11 Vzdělávací videohry

3.11.1 Studie

Výzkumy, které byli prováděny v ranní fázi videoher se zaměřovali hlavně na jejich negativní dopady na hráče jako například agresivita nebo závislost, což velmi negativně ovlivnilo tehdejší veřejné vnímání všech druhů videoher. Mnoho novějších studií ale tvrdí, že tyto negativní efekty v podstatě skoro neexistují a že naopak mohou videohry mít pozitivní účinky na jejich hráče. Některé studie dokonce tvrdí, že děti, které hrají videohry mají zvýšený intelekt a motoriku oproti jejich vrstevníkům, kteří je nehrají (González-González & Blanco-Izquierdo, 2012).

Lze s jistotou říci, že videohry dokážou efektivně upoutat pozornost dětí a teenagerů. Je také známým faktem, že se děti učí lépe, pokud je to baví a že při tom dokážou udržet pozornost déle než při tradičních výukových metodách. Z tohoto důvodu mají videohry silný potenciál jako takzvaná edu-tainment média.

3.11.2 Edu-tainment

Obecně Edu-tainment znamená výuka zábavou. V počítačovém prostředí je klasifikován jako druh videoher odměňujících učení, nebo může také zahrnovat například různá videa, animace, příběhy nebo jiný vizuální vzdělávací materiál přístupný přes obrazovku počítače. Účelem edu-tainmentu v počítačovém prostředí je přilákat studenty a udržet jejich pozornost pomocí různých animací a poutavých barev a vizuálu nebo pomocí jakéhosi interaktivního prvku (Aksakal, 2015).

3.11.3 Vzdělávací vlastnosti videoher

Videohry mohou demotivované studenty inspirovat se začít učit, mohou jim umožnit prozkoumat svět a naučit je vytvářet si životní cíle a odolávat překážky. Mohou také do vyučovacího procesu přinést interaktivní prvky, což dále zlepšuje efektivitu výuky. Dokážou studenty seznámit s počítačovou technologií, čímž mohou položit základ budoucím informatikům. A protože videohry nenesou žádné reálné následky, mohou studentům umožnit experimentovat bez strachu z negativních důsledků jako například při simulaci nebezpečných chemických reakcí nebo ovládání auta či letadla (Griffiths, 2002).

4. Integrované vývojové prostředí

Integrované vývojové prostředí, také označované zkratkou IDE, která je akronymem slov „Integrated Development Environment“ je software pomocí kterého vývojáři programují aplikace. Kombinuje všechny základní nástroje potřebné pro programování do jednoho uživatelského rozhraní (RedHat, 2019).

Podle článku od RedHat (2019) mezi nástroje, které IDE poskytují patří například:

- **Editor kódu** – textový editor který programátorům pomáhá při psaní kódu pomocí zvýraznění syntaxe, vizuálních nápověd, automatického dokončování kódu a kontrolování chyb při psaní.
- **Automatizace lokálního buildu** – nástroje automatizují jednoduché opakovatelné úkoly tím, že je provádí automaticky při buildování aplikace. Tyto úkoly jsou například kompilace kódu do binárního, zabalení binárního kódu a provádění různých testů.
- **Debugger** – software pro testování softwaru který dokáže přímo najít a zobrazit chybu v originálním kódu

Hlavním důvodem použití integrovaných vývojových prostředí je, že silně urychlují práci programátorů tím, že automatizuje nastavení utilit a urychluje rychlost psaní kódu pomocí automatického doplňování a zvýrazňování chyb při psaní. Nové verze dokonce poskytují umělou inteligenci, která je schopná sama psát části kódu. I když je možné vyvíjet aplikace bez IDE a manuálně integrovat utility přes textový editor, v dnešní době nenajdete skoro žádný vývojový tým, který tak pracuje. Takový tým by nejen musel mít vývojáře, který vůbec ví jak, se používají jednotlivé nástroje, ale také by to znamenalo, že stráví výrazně déle na tvorbě aplikace. Přesto existují případy, kdy vývojáři pracují bez IDE, například když potřebují mít kontrolu nad přizpůsobením prostředí ve kterém pracují nebo pokud potřebují získat hlubší pochopení o tom, jak funguje jejich kód (RedHat, 2019).

4.1 IDE pro Unity

Unity dává programátorům na výběr mezi třemi různými vývojovými prostředí, a to Visual Studio, Visual Studio Code a JetBrains Rider. Volba jednoho z těchto prostředí závisí na osobní preferenci programátora.

4.1.1 Visual Studio

„Integrované vývojové prostředí (IDE) Visual Studio je kreativní spouštěcí panel, který můžete použít k úpravám, ladění a sestavování kódu a k publikování aplikace. Nad rámec standardního editoru a ladicího programu, které poskytuje většina integrovaných vývojových prostředí (IDE), obsahuje Visual Studio kompilátory, nástroje pro doplňování kódu, grafické návrháře a mnoho dalších funkcí, které zlepšují proces vývoje softwaru.“ (Microsoft, n.d.)

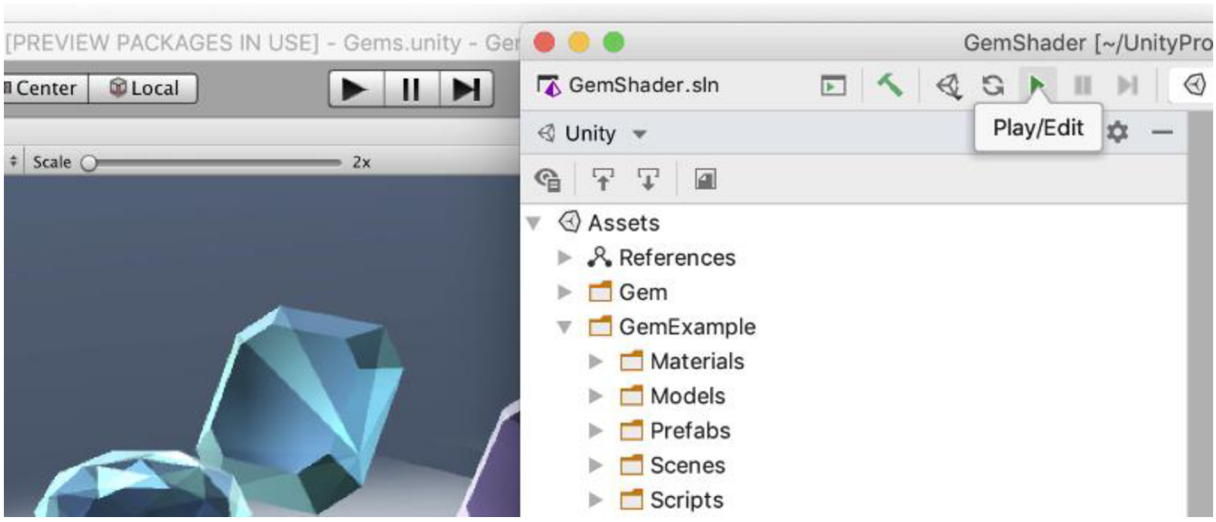
4.1.2 Visual Studio Code

„Visual Studio Code je lehký, ale výkonný editor zdrojového kódu, který běží na počítači a je k dispozici pro Windows, MacOS a Linux. Obsahuje integrovanou podporu pro JavaScript, TypeScript a Node.js a má bohatý ekosystém rozšíření pro další jazyky a moduly runtime (například C++, C#, Java, Python, PHP, Go, .NET).“ (Microsoft, n.d.)

4.1.3 JetBrains Rider

JetBrains Rider je rychlé a výkonné multiplatformní .NET vývojové prostředí dostupné na operačních systémech Mac, Windows a Linux. Poskytuje také podporu přímo pro vývoj v Unity, a dokonce se při otevření tohoto IDE automaticky nastaví jako vývojové prostředí Unity. Velkou výhodou použití JetBrains Rider pro vývoj v unity je integrovaná komunikace mezi IDE a Unity, která umožňuje zapínání play módu, posouvání se po jednom snímku, a pozastavování hry, což můžeme vidět na obrázku 1. Poskytuje také nápovědy při psaní kódu specifické pro Unity a mnoho dalších funkcí přímo v grafickém rozhraní IDE (JetBrains, n.d.).

Obrázek 1: Nástroje spuštění, pozastavení a posunu snímků v uživatelském rozhraní JetBrains Rider



Zdroj: JetBrains, n.d.

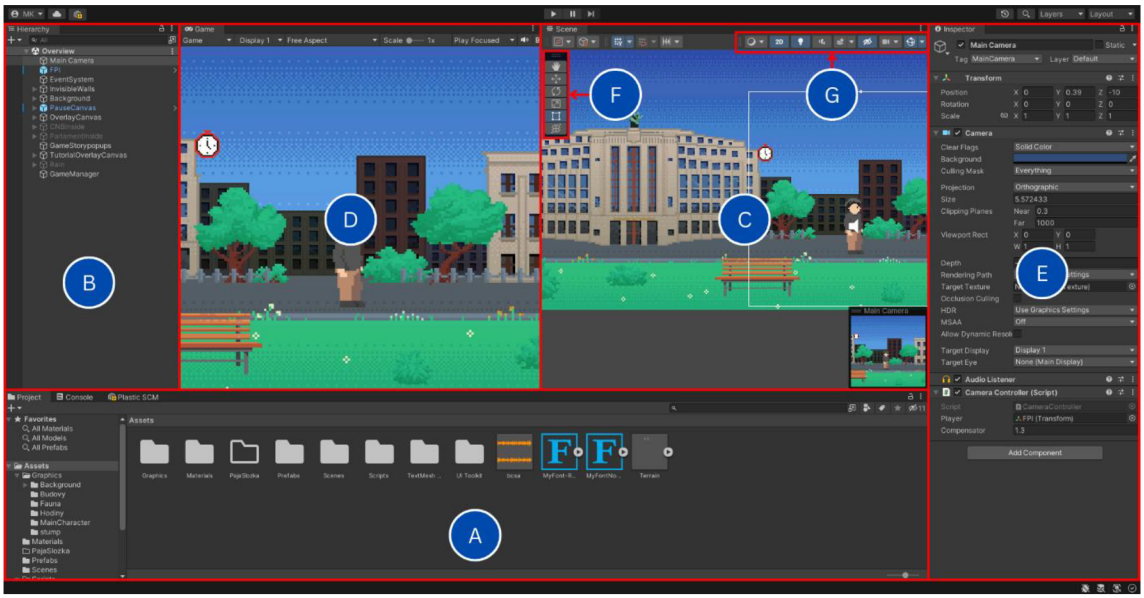
5. Vývojové prostředí Unity

Unity editor je hlavní prostředí, ve kterém vývojáři, ať už programátoři, grafici nebo zvukový designeři vytvářejí hry nebo aplikace. Poskytuje všechny potřebné nástroje pro herní vývoj, jako například procházení assetů, vytváření scén a objektů ve scénách, navigace po scénách, runtime testování a debuggování aplikace a mnoho dalších. Unity editor je také přizpůsobitelný a podporuje rozšiřování vývojářských nástrojů buď vytvářením vlastních nástrojů pomocí C# skriptů, nebo importováním pluginů od třetích stran. Unity poskytuje editor jak pro 2D, tak pro 3D herní prostředí. Tyto dva editory jsou identické, mají pouze jiná nastavení. V praktické části této práce bude použita 2D šablona.

5.1 Okno projektu

Okno projektu je ve výchozím nastavení umístěné ve spodní části na obrazovce v editoru. (Označeno jako A na obr. 2) Slouží jako prohlížeč souborů, pro vytvořené soubory v adresáři projektu. Dvě hlavní složky, které může uživatel v projektovém okně prohlížet jsou složky „Assets“ a „Packages“, které se nachází v kořenové složce projektu. Složka „Assets“ obsahuje všechny soubory, které vytvářejí vývojáři, například scény, grafika, zvukové efekty, skripty, animace a další. Složka „Packages“ uchovává všechny stažené balíky. Balíky obsahují soubory funkcí, o které rozšiřují projekt, do kterého jsou vložené. V případě, že uživatel neví, kde přesně se nachází soubor, který hledá, může použít funkci vyhledávání umístěnou na horní části okna (Unity Technologies, 2023-a).

Obrázek 2: Rozložení oken v Unity editoru



Zdroj: autor, 2023

Při práci na projektech v Unity je velmi důležité dodržovat zásady organizace souborů v projektu. Pokud by vývojáři jednoduše ukládali všechny vytvořené soubory do složky „Assets“, i relativně malý projekt by se rychle stal velmi nepřehledným. Základem je vytvořit hlavní složky pro jednotlivé typy souborů, typicky například scény, skripty, audio, grafika, animace, efekty, textury, materiály, prefaby nebo nastavení. Tyto hlavní složky pak lze rozdělovat na specifické podsložky, pokud je pro uživatele přehlednější část assetů uložit na jedno společné místo reprezentující například jednotlivou funkci nebo část aplikace. Dále je důležité volit správná a deskriptivní jména souborů, která zachycují jejich funkci nebo význam pro snazší identifikaci a vyhledávání (Unity Technologies, 2022-a).

5.2 Okno hierarchie

Okno hierarchie obsahuje hierarchickou strukturu objektů v aktuálně otevřené scéně. (Označeno jako B na obr. 2) Každý objekt ve scéně v sobě může obsahovat jeden nebo více podřízených objektů. Podřízený objekt je, stejně jako jeho nadřazený objekt, také další objekt ve scéně, pouze je umístěn pod jeho nadřazeným objektem, to znamená, že stejně jako jeho nadřazený objekt v sobě může také obsahovat podřízené objekty. Tato struktura nadřazených a podřízených objektů tvoří hierarchii scény (Unity Technologies, 2023-g).

5.3 Zobrazení scény a herní zobrazení

Herní zobrazení je okno, ve kterém běží herní testování aplikace po jejím spuštění. (Označeno jako D na obr. 2) Ve výchozím nastavení jsou okna zobrazení scény a herního zobrazení na stejném místě a lze mezi nimi přepínat pomocí záložek umístěných na jejich horní části (Unity Technologies, 2023-e).

Pomocí okna scény, umístěného uprostřed obrazovky (Označeno jako C na obr. 2) se může uživatel volně pohybovat po scéně, aniž by aplikaci zapínal. Může vybírat a interagovat s objekty ve scéně, pohybovat a otáčet je, nebo měnit jejich velikost pomocí panelu nástrojů v levé horní části okna (Unity Technologies, 2023-f).

5.4 Inspektor

Okno inspektoru se ve výchozím nastavení editoru nachází na pravé straně obrazovky. (Označeno jako E na obr. 2) Inspektor slouží jako místo, kde lze zobrazovat informace a upravovat parametry objektů ve scéně, nebo assetů v adresáři. Při kliknutí na asset v prohlížeči se v okně inspektoru zobrazí informace a nastavení daného assetu. Při kliknutí na objekt ve

scéně se zobrazí jednotlivé komponenty daného objektu, které lze libovolně upravovat, odstraňovat nebo přidávat (Unity Technologies, 2023-d).

5.5 Asset

Assety jsou reprezentací jakéhokoli souboru, který se dá použít ve vyvíjené hře nebo aplikaci. Asset může vzniknout buď mimo Unity, jako například 3D model, audio soubor nebo obrázek, nebo může vzniknout v Unity, jako například Animation Controller, Audio Mixer nebo Render Texture (Unity Technologies, 2017).

5.6 Scéna

Scény jsou místo, ve kterém uživatel pracuje s obsahem v Unity. Jsou to assety, které obsahují celou nebo části hry nebo aplikace. Lze například vytvořit jednoduchou hru v jedné scéně, zatímco více komplexní hry mohou být rozdělené na více scén (Unity Technologies, 2023-c).

5.7 Komponent

Komponenty jsou jakékoli funkce, které lze přiřadit jednotlivým objektům ve scéně. V podstatě tvoří veškerou funkcionalitu dané scény. Ať už pohyb hráče, fyziku nebo gravitaci, efekty nebo animace, všechny funkce ve scéně musí být připnuté na objektech ve scéně (Unity Technologies, 2023-h).

5.8 Skriptovatelný objekt

Skriptovatelný objekt je datový kontejner, který se používá pro ukládání velkého množství dat nezávisle na instancích třídy. Jeden z hlavních případů jejich použití je snížení využití paměti projektu tím, že se vyhnete kopírování hodnot. Je to užitečné, pokud má projekt prefab, který ukládá neměnná data v připnutých „Monobehaviour“ skriptech. Pokaždé, když se vytvoří instance takovéhoto prefabu, získá svou vlastní kopii těchto dat. Místo toho můžeme k uložení dat použít skriptovatelný objekt a poté k nim přistupovat pomocí reference ze všech prefabů, to znamená, že v paměti existuje pouze jedna kopie dat. Stejně jako „Monobehaviour“ se skriptovací objekty („ScriptableObject“) odvozují ze základního objektu Unity, ale na rozdíl od „Monobehaviour“ se skriptovatelné objekty nedají připojit k herním objektům. Místo toho se ukládají jako assety v projektu (Unity Technologies, 2023-i).

5.9 Prefab

Slovo prefab je odvozené z anglického slova „Prefabricated“, čili něco, co je předpřipravené. Prefab systém v Unity umožňuje vytvářet, konfigurovat a ukládat herní objekty se všemi jeho komponentami, hodnotami, vlastnostmi, a podřízenými objekty jako opakovatelně použitelný asset. Prefab funguje jako šablona, ze které lze vytvářet její instance ve scéně. Jakékoli úpravy, které se provedou na prefabu se automaticky projeví na instancích daného prefabu, což umožňuje provádět rozsáhlé změny v celém projektu, aniž by se museli provádět stejné úpravy u každé kopie prefabu. Nicméně, to neznamená, že všechny instance prefabu musí být vyloženě totožné. Pokud je třeba, aby se jednotlivá instance lišila od ostatních, může se nastavení jednotlivé instance přepsat, nebo lze také vytvářet variace prefabu (Unity Technologies, 2023-j).

5.10 Základní navigace a úprava scény

Ve 2D projektech lze pomocí překryté vrstvy (označené na obr. 2 jako G) přepínat mezi 3D a 2D zobrazením kliknutím na ikonku s názvem „2D“. Přepínat mezi nástroji pro navigaci a úpravu scény lze pomocí panelu nástrojů (označených na obr, 2 jako F). 3D a 2D zobrazení mají některé rozdíly především v možnosti otáčení pohledu a dostupnosti některých navigačních funkcí (Unity Technologies, 2023-l).

5.10.1 Pohyb po scéně

Za pohyb po scéně odpovídá nástroj zobrazení označený na panelu nástrojů jako ikonka ruky. Nástroj zobrazení se dá přepnout buď zmáčknutím na jeho ikonku nebo zmáčknutím klávesové zkratky Q (Unity Technologies, 2023-l).

Ve 2D zobrazení lze kameru scény pohybovat pouze po ose X a Y. Nejjednodušší způsob pohybu po 2D scéně je pomocí pravého nebo středního tlačítka na myši (ve 2D zobrazení zastupují stejnou funkci). Po zmáčknutí tlačítka se kamera pohybuje spolu s pohybem myši. Pokud je v panelu nástrojů zvolen nástroj zobrazení, lze se stejným způsobem pohybovat i levým tlačítkem myši. Pokud uživatel preferuje pohyb pomocí klávesnice, může použít tlačítka šipek (Unity Technologies, 2023-l).

Ve 3D zobrazení pomocí pravého tlačítka po dobu jeho zmáčknutí lze otáčet kameru a pohybovat se po scéně pomocí kláves W, A, S a D. Pokud uživatel stiskne klávesu Shift, je pohyb po scéně zrychlený. Pomocí stisknutí středního tlačítka myši (nebo levého, pokud je

aktivován nástroj zobrazení) se lze pohybovat pouze po X a Y momentálního úhlu kamery (Unity Technologies, 2023-l).

5.10.2 Pohyb a úprava objektů

S objekty se dá po scéně pohybovat pomocí nástroje pohybu označeného ikonkou čtyř šipek jdoucích od sebe (klávesová zkratka W). Uživatel může buď kliknutím na objekt v hierarchii, nebo ve scéně označit objekt a pohybovat s ním po potřebné ose pomocí šipek, které se po kliknutí objeví. Pokud chce s objektem pohybovat volně, a ne po ose, může kliknout na čtverec (ve 3D zobrazení kvádr), který se nachází v centru osových šipek (Unity Technologies, 2023-l).

Otáčet lze objekty zvolením nástroje otáčení označeného šipek v cyklu (klávesová zkratka E). Po zvolení nástroje otáčení se na objektu zobrazí barevně oddělené kruhy reprezentující osu, po které objekt otáčí. Kliknutím a posunutím těchto kruhů lze s objektem otáčet po dané ose (Unity Technologies, 2023-l).

Velikost objektu se dá změnit pomocí nástroje zvětšení označeného čtvercem, ve kterém se nachází menší čtverec s šipkou vycházející z jeho rohu (klávesová zkratka R). Pomocí zobrazených osových šipek se čtvercovým hrotem lze stejným způsobem jako u pohybu nebo otáčení upravovat velikost po dané ose nebo se kliknutím na šedý čtverec uprostřed dá velikost upravovat na všech osách najednou (Unity Technologies, 2023-l).

Pozice, rotace nebo velikost objektů lze také upravovat přímo v Transform komponentu objektu úpravou jejich X, Y a Z hodnot.

6. Fáze vývoje videoher

6.1 Plánovací fáze

Podle Devina Pickella (2019) Mezi základní otázky, na které musíme odpovědět, než začne vývoj videohry jsou:

- Jaký druh videohry budeme vyvíjet?
- Bude hra 3D nebo 2D?
- Jaké budou hlavní funkce videohry
- Jaké postavy se v ní budou vyskytovat?
- Kdy a kde se hra nachází?
- Kdo je cílová skupina videohry?
- Na jaké platformě nebo operačním systému bude hra dostupná?

Odpovědi na tyto základní otázky jsou jednou z nejtěžších částí vývoje videohry, protože slouží jako opora pro celý proces vývoje videohry. Nejen stanoví standardy pro vývojáře, ale také naznačí vydavatelům přehled o tom, co můžou od produktu očekávat (Pickell, 2019).

Dalším důležitým krokem v plánovací fázi je ověření vytvořeného konceptu, pomocí kterého lze zjistit, zda organizace, která hru vyvíjí je schopná dosáhnout finálního produktu (Pickell, 2019).

Podle Devina Pickella lze koncept ověřit pomocí následujících otázek:

- Jaká je odhadovaná částka vývoje dané videohry?
- Má organizace technologické dovednosti na splnění vize?
- Použijeme existující engine, nebo vytvoříme vlastní?
- Jak velký tým se musí na vývoji podílet?
- Budeme najímat externí hlasové herce nebo spisovatele?
- Jaký je odhadovaný čas pro vytvoření hry?
- Jak budeme videohry zpeněžovat?

6.2 Před-produkční fáze

V před-produkční fázi se stanoví, jakým způsobem budou uskutečněny nápady, které byly vymyšleny v plánovací fázi. Je toho dosaženo kolaborací mezi spisovateli, umělci, návrháři, vývojáři a ostatními důležitými odděleními. Musí se navzájem dohodnout, na rozsahu projektu, vytvořit grafický styl, navrhnout prototypy postav, herní prostředí, uživatelské rozhraní, stanovit, jak se bude hra ovládat a mnoho dalších, aby mohli stanovit, jak na sebe budou tyto části navazovat a jak budou spolupracovat (Pickell, 2019).

Pokud hru vyvíjí nezávislý vývojář, často pracuje na projektu skoro sám a je za všechno odpovědný, což znamená, že sice má více práce, ale zároveň nemusí komunikovat s jinými vývojáři a může se rozhodovat sám což je na druhou stranu výhodou. V před-produkční fázi začínají nápady nabývat tělesné formy. Málokdy se ale stane, že videohra projde fázemi plánování a preprodukce bez žádných obětí. Rozhodnutí, která vzniknou v před-produkční fázi jsou základem, ze kterého bude vývoj v dalších fázích pokračovat (Bramble, 2023).

6.3 Produkční fáze

Tato fáze je časově a obtížnostně nejnáročnější a část produkce. Během této fázi probíhá vytváření všech částí, ze kterých se skládá videohra. Vytváří se všechny možné 3D modely, nahrávají se dialogy nebo zvuky herního prostředí, navrhuje se struktura jednotlivých úrovní nebo herních prostředí a píše se kód, který hru řídí. Po celém produkčním procesu vedení kontrolují kvalitu a dodržování časového rozvrhu. Tato fáze může trvat měsíce nebo roky (Pickell, 2019).

6.4 Testovací fáze

Testování a kontrola kvality každé herní funkce a mechaniky je velmi důležité. Hra, která ještě nebyla řádně otestována není připravená ani na Alpha vydání. Testeři musí hru kompletně projít a vyhledat veškeré chyby, zneužitelné funkce, problémy s grafikou a vykreslováním, rozbité ovládání, chyby v dialogu a další neošetřené problémy ve hře. Hru lze testovat ze dvou stejně důležitých pohledů, a to za prvé z hlediska funkčnosti a za druhé z hlediska její kvality a poskytnutí uživateli zážitku ze hry (tzv. fun factor) (Pickell, 2019).

Fun factor je to, co dělá videohru zábavnou a poutavou. Pro dobrý zážitek ze hry je třeba zajistit, například že je příběh poutavý, herní mechaniky jsou příjemné a zábavné, nebo že řešení problémů je pro hráče odměňující. Dobrý zážitek ze hry je pro prodej stejně důležitý jako technická stabilita videohry (Bramble, 2023).

6.5 Předstartovní fáze

V Předstartovní fázi se organizace připravuje na vydání první verze videohry. Zabývá se hlavně marketingem a propagací videohry. Organizace vytváří různé trailery či upoutávky a vydává je buď na různých sociálních médiích, nebo na herních konferencích jako například E3 nebo PAX. Menší nezávislá studia často nemají peněžní zdroje na masivní propagaci, zde však často hraje roli crowdfunding nebo je v dnešní době je také populární a často velmi úspěšné propagovat videohru pomocí známých osobností na platformách YouTube nebo Twitch (Pickell, 2019).

Předstartovní fáze je jedna z nejnapínavějších ale zároveň psychicky náročných fází vývoje. Organizace je připravena představit veřejnosti svůj produkt nad kterým dlouho pracuje, ale nemůže vědět, zda jej publikum pojme pozitivně, negativně, nebo neprojeví zájem vůbec (Bramble, 2023).

6.6 Fáze vydání hry

Poslední měsíce před vydáním jsou většinou stráveny opravou a odstraňováním velkých zásob chyb, ať už starých nebo nově objevených v testovací fázi. Chyby jsou většinou seřazeny do hierarchie na základě jejich důležitosti, přičemž nejvyšší důležitost mají chyby způsobující pád hry. Dále paralelně s odstraňováním chyb se vývojáři snaží co nejvíce zdokonalit funkce a vizuál hry před jejím vydáním (Pickell, 2019).

6.7 Po-produkční fáze

Po uvedení videohry na trh začíná po-produkční fáze vývoje. I když organizace strávila roky na vývoji první verze, je pořád velké množství práce, která ji čeká. I pro největší herní společnosti je prakticky nemožné vydat videohru bez chyb a je nutné pokračovat v jejich identifikaci a odstranění. Často k tomu také pomáhá to, že byla videohra vypuštěna do širokého prostředí hráčů, který mohou chyby nahlašovat ať už vyplněním speciálních formulářů nebo jejich zveřejněním na různých herních fórech. Další součástí po-produkční fáze je průběžné vydávání aktualizací. Tyto aktualizace mohou obsahovat buď výše zmíněné opravené chyby, nebo mohou obsahovat kompletně nový obsah či DLC. DLC (Downloadable Content – Stažitelný obsah) může obsahovat celou řadu nového obsahu, ať už nové úrovně, příběhy, rozšíření, herní režimy a další. V dnešní době je běžné průběžně vydávat čerstvý obsah, protože to zvyšuje počet hráčů a přitažlivost hry (Pickell, 2019).

7. Praktická část

7.1 Plánovací proces

Proces začal základní vizí – videohra, ve které pomocí makroekonomických nástrojů bude hráč ovládat průběh hospodářského cyklu. Pro vývoj videohry vznikl čtyřčlenný tým, který byl rozdělený na dvě zaměření. První dva členové se zaměřili na makroekonomickou teorii a logiku, dle které se hra řídí, a druhý dva na proces vývoje videohry. Mezi vývojáři byl vývoj dále rozdělen do dvou částí – frontendová část, tj. tvorba uživatelského rozhraní, grafiky nebo animací a backendová část, do které spadá implementace makroekonomické teorie, vytvoření herní logiky a propojení s uživatelským rozhraním. V této práci se zabývám právě backendovou části.

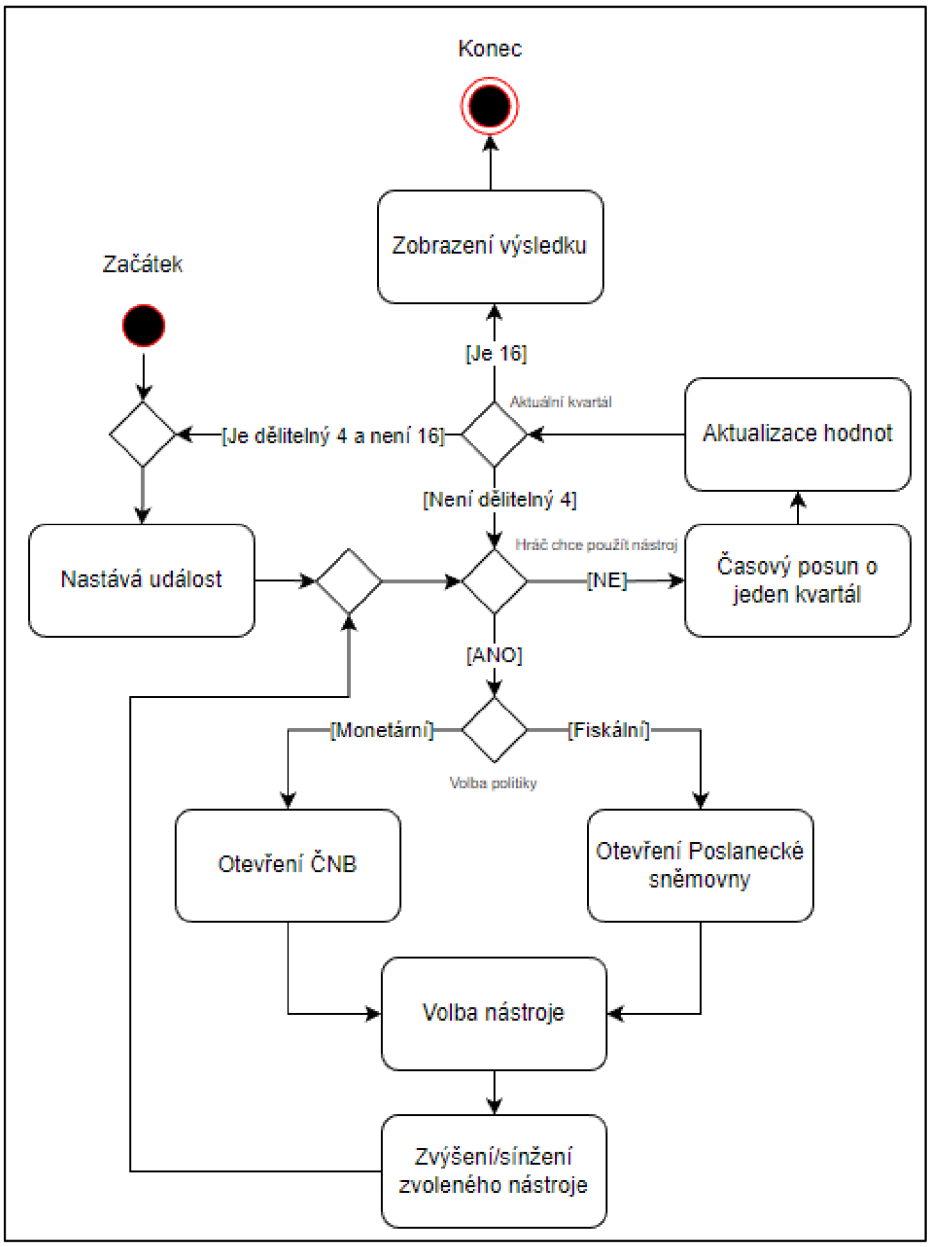
Po diskusi vize a stanovení rolí v týmu byl vytvořen základní koncept hry. Stanovili se pravidla hry, byl navrhnut konkrétní průběh a hlavní herní mechaniky, zvolena herní dimenze, grafika a styl videohry.

Na základě konceptu bylo frontendovým programátorem vytvořeno herní prostředí, hlavní postava a obecně kostra videohry, do které bylo nutno přidat funkcionality a mechaniky hry. Dále tým teoretiků vytvořil makroekonomickou logiku, dle které se videohra řídí. Tato logika mi byla předána a na jejím základě v kombinaci s frontendovou kostrou začala fáze vytváření funkčnosti a implementace této logiky do videohry.

7.2 Průběh a pravidla hry

Videohra začíná tím, že se uživateli oznámí událost, která nějakým způsobem negativně ovlivňuje hospodářský cyklus. Hráč může, ale nemusí, pomocí pohybu po prostředí vejít do jedné z vládních budov, které po jejich otevření poskytují uživateli rozhraní s nástroji, pomocí kterých mohou neutralizovat nevhodný směr ekonomie a udržet makroekonomické ukazatele v jejich přirozených hodnotách. Tyto budovy jsou Česká národní banka a Poslanecká sněmovna. Po otevření budovy mohou buď zobrazit graf aktuálního průběhu ekonomie, nebo použit nástroj pro její regulaci. Každý nástroj lze do určité míry buď snížit, nebo zvýšit, což bude mít dopad na další ekonomické období. Hráč se může události věnovat po dobu osmi kvartálů, než začne další významná ekonomická událost. Pro posunutí do dalšího kvartálu uživatel musí kliknout na obrázek hodinek v levém horním rohu obrazovky. Takto bude koloběh probíhat 8 let neboli 32 kvartálů, po kterých se hra ukončí a uživateli bude oznámen výsledek ohledně kvality jeho výkonu. Průběh hry je grafický znázorněn na obrázku 3.

Obrázek 3: Diagram aktivit průběhu videohry



Zdroj: autor, 2023

7.3 Ekonomická logika

7.3.1 Měřené hodnoty

Ve hře se se měří tři základní makroekonomické ukazatele – HDP, inflace a nezaměstnanost. Tyto tři hodnoty může hráč průběžně zobrazovat pomocí grafu v budově České národní banky nebo Poslanecké sněmovny. Jejich hodnoty jsou reprezentovány jako desetinná čísla, jejichž hodnoty znamenají nárust nebo pokles předchozí hodnoty. Pomocí historie změn ukazatelů se v uživatelském rozhraní vykresluje graf. Nezaměstnanost a inflace na se grafu zobrazuje v procentech, zatímco HDP je převedeno a zobrazeno v jednotkách miliard.

Tyto makroekonomické ukazatele se každý ekonomický kvartál mění. Dle jejich průměrných hodnot z veřejných dat za posledních 30 let byly určeny jejich následující ideální hodnoty:

- Meziroční HDP: +1.9%
- Nezaměstnanost: 5%
- Meziroční inflace: +2.8%

7.3.2 Události

Události jsou makroekonomické změny v simulovaném světě videohry, které určitým způsobem ovlivňují průběhy makroekonomických ukazatelů. To, jakým způsobem daná událost ovlivňuje ukazatele je určené pomocí hodnot, které vypracovali kolegové z teoretické části projektu. V rámci této práce není třeba znát logiku za těmito hodnotami, pouze jak se používají v logice. Každá událost má přiřazené desetinné číslo pro její efekt na každý ukazatel.

V následujících tabulkách 1 a 2 jsou sepsané všechny vytvořené události, které se budou ve hře vyskytovat a jakým způsobem ovlivňují ukazatele za jeden kvartál.

Tabulka 1: První část ekonomických událostí

Název	Ozbrojený konflikt	Energetická událost kladná	Energetická událost záporná	Pandemie
HDP	-0.5%	+1.25%	-1.25%	-1.75%
Inflace	+2.5%	-0.25%	+1%	-2.25%
Nezaměstnanost	-0.25%	-0.1%	-0.1%	+0.5%

Zdroj: autor, 2023

Tabulka 2: Druhá část ekonomických událostí

Název	Přírodní událost kladná	Přírodní událost záporná	Uzavření hranic
HDP	+0.375%	-0.375%	-1.25%
Inflace	-0.125%	+0.75%	-0.625%
Nezaměstnanost	-0.25%	+0.175%	+0.875%

Zdroj: autor, 2023

7.3.3 Nástroje

Nástroje jsou prostředky, pomocí kterých lze ovlivňovat hospodářský cyklus ve hře. Používají se jako obrana proti událostem a v podstatě je jejich princip založený na podobné logice jako události. Každý nástroj má ale ve hře svůj aktuální stupeň, který lze nastavit buď na snížený, zvýšený nebo neutrální. Tento stupeň určuje, jakým způsobem daný nástroj ovlivňuje ukazatele.

Následující tabulky 3, 4, 5 a 6 reprezentují vytvořené nástroje. Index značí stupeň, který může hráč zvolit přičemž -1 znamená snížení, 0 neutrální a 1 zvýšení hodnoty nástroje.

Tabulka 3: Hodnoty efektu daně z přijmu

Daň z příjmu			
Index	-1	0	1
HDP	1.5%	0.5%	-1.5%
Nezaměstnanost	-0.4%	-0.05%	0.1%
Inflace	0.3%	0.05%	-0.15%

Zdroj: autor, 2023

Tabulka 4: Hodnoty efektu státních výdajů

Státní výdaje			
Index	-1	0	1
HDP	-0.5%	-0.25%	0.25%
Nezaměstnanost	0.25%	0.05%	-0.15%
Inflace	-0.25%	-0.1%	0.25%

Zdroj: autor, 2023

Tabulka 5: Hodnoty efektu úrokové míry

Úroková míra			
Index	-1	0	1
HDP	0.1%	0.05%	-0.05%
Nezaměstnanost	-0.1%	0%	0.1%
Inflace	1.5%	0.1%	-0.75%

Zdroj: autor, 2023

Tabulka 6: Hodnoty efektu podmínek úvěru

Změna podmínek úvěrů			
Index	-1	0	1
HDP	0.05%	-0.25%	-0.4%
Nezaměstnanost	-0.15%	-0.05%	0.05%
Inflace	0.5%	0.1%	-0.75%

Zdroj: autor, 2023

7.3.4 Hospodářský cyklus

Hospodářský cyklus je ve hře nastaven tak, že každé dva roky (8 kvartálů) působí nová událost. Pro každý nový kvartál je nutné provést výpočet aktuálních hodnot ukazatelů. Ve výpočtu se kombinuje ideální hodnota ukazatele, vliv aktuální události, vliv všech nástrojů dle jejich aktuálně zvolených stupňů, a nakonec jsou výsledky v mikroměřítku poupraveny pomocí náhodného prvku. Následující vzorce mi byli předány od teoretického týmu a jsou v práci použité pro výpočet hodnot ukazatelů pro nový kvartál:

$$X_N = (T_N[I_T] + G_N[I_G] + R_N[I_R] + L_N[I_L] + E_N) * R_N$$

$$R_N = rnd(min, max) * 0.01$$

$$min = (d * 10) + 70$$

$$max = (d * 10) + 100$$

$$d = | N[n] - i |$$

X = Další hodnota ukazatele

N = Ukazatel

T_N = Vliv daně z přijmu na daný ukazatel

G_N = Vliv státních výdajů na daný ukazatel

R_N = Vliv úrokové míry na daný ukazatel

L_N = Vliv podmínek úvěru na daný ukazatel

E_N = Vliv události na na daný ukazatel

R_N = Náhodný prvek pro daný ukazatel

Ix = Index daného nástroje

rnd = funkce náhodné hodnoty

min = minimální hranice

max = maximální hranice

d = odchylka

n = poslední index ukazatele

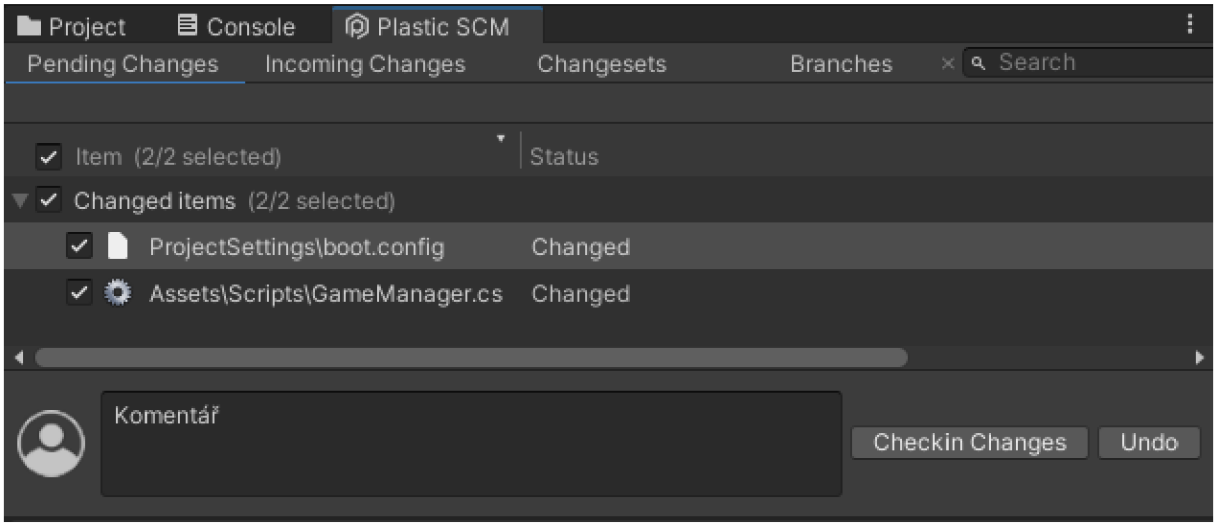
i = ideální hodnota ukazatele

7.4 Správa verzí

Na tvorbě projektu se podílím já a můj kolega, který se zabývá uživatelským rozhraním. Z tohoto důvodu bylo nutné nastavit a implementovat systém pro správu verzí. Zvolili jsme Plastic SCM, protože je do určité hranice úložiště a uživatelů zdarma, je přímo podporovaný Unity a jednoduše se používá.

Provedené změny byli na cloudu průběžně aktualizovány pomocí okna Plastic SCM přímo v Unity prostředí, kde se vybrali změny, které bylo nutno nahrát, napsal komentář a stisknulo tlačítko „Checkin chnages“, jak lze vidět na obr. 4. Případně bylo možné stejného výsledku dosáhnout přes desktopovou aplikaci Plastic SCM kliknutím pravého tlačítka myši na repozitář a výběrem „Checkin“ z vyskakovací nabídky, což otevřelo okno, ve kterém se vybrali a vyplnili stejné údaje jako v Unity verzi a opět se stisknutím tlačítka „Checkin“ změny nahráli na cloud.

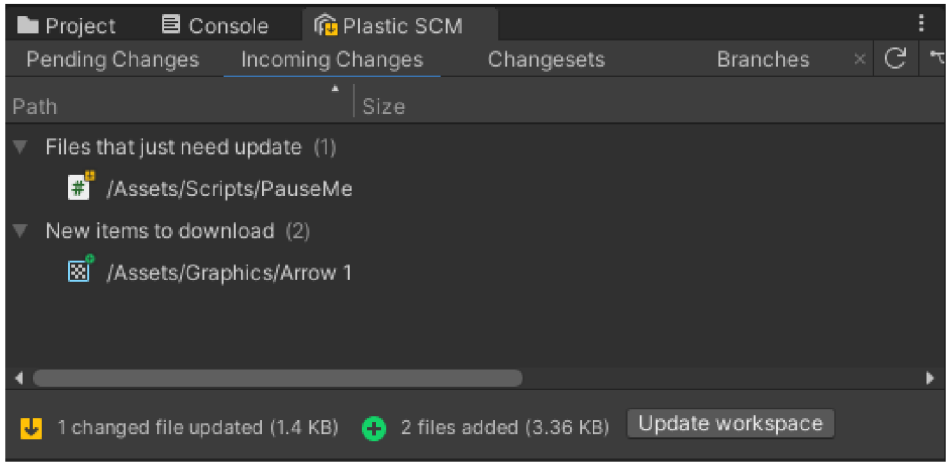
Obrázek 4: Nahrávání změn do cloudu Plastic SCM.



Zdroj: autor, 2023

Aktualizace pracovního prostoru probíhala přes záložku „Incoming Changes“ v okně Plastic SCM. Stisknutím tlačítka „Update Workspace“, které lze vidět na obrázku 5 probíhala aktualizace. Pokud vznikli konflikty mezi lokálním a serverovým repositářem, museli se nejdříve vyřešit.

Obrázek 5: Aktualizace pracovního prostoru



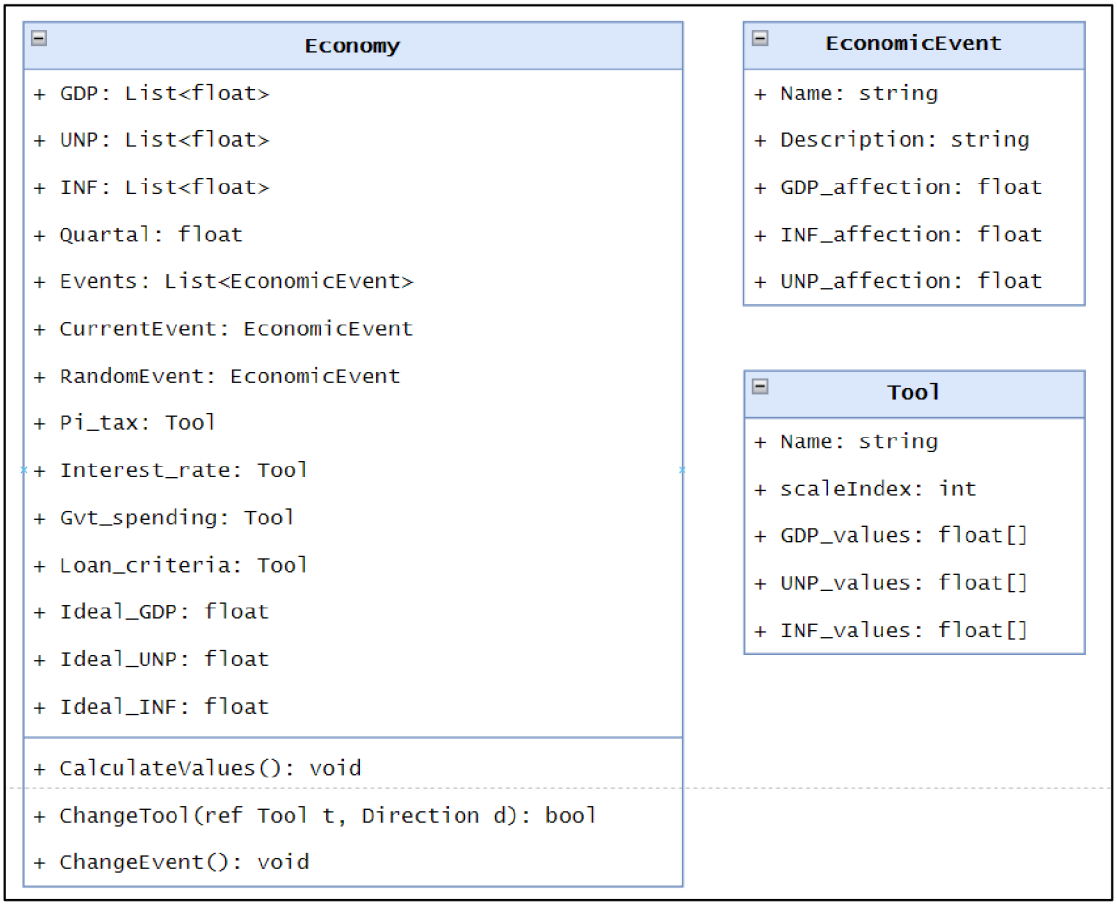
Zdroj: autor, 2023

7.5 Převedení logiky do kódu

7.5.1 Třída Economy

Třída „Economy“ je hlavní třída, která drží a zpracovává hodnoty ekonomie. Její grafická reprezentace je znázorněna pomocí diagramu tříd na obrázku 6. Hodnoty ukazatelů HDP, inflace a nezaměstnanosti se ukládají v rámci datového typu List jako hodnoty s desetinným číslem. Tyto listy slouží jako historie ukazatelů, podle které se v uživatelském rozhraní vykresluje graf. „Quartal“ je celé číslo reprezentující aktuální kvartál hospodářského cyklu. Všechny události, které jsou ve hře dostupné jsou uloženy v listu nazvaném „Events“. Systém vybírá náhodné události do hry pomocí proměnné „RandomEvent“ která vrací událost z listu „Events“ na náhodném indexu od nuly do hodnoty posledního indexu listu. Individuální nástroje, které jsou hráči dostupné jsou uložené ve třídě Ekonomie a nazvané „Pi_tax“, „Interest_rate“, „Gvt_spending“ a „Loan_criteria“. Ideální hodnoty každého ukazatele slouží zaprvé jako prvek ve vzorcích pro výpočet náhodného prvku a také slouží jako startovní hodnoty ekonomie.

Obrázek 6: Diagram tříd popisující datovou strukturu logiky



Zdroj: autor, 2023

Metoda GetNewValue()

Protože se výpočet všech ukazatelů počítá stejným vzorcem, pouze s jinými proměnnými tak se dá kód zkrátit pomocí jedné metody vracející výsledek pro zadaný ukazatel. Za toto odpovídá metoda „GetNewValue()“. Tato metoda má celkem 7 parametrů, které by se nevešli do obrázku diagramu tříd, jsou však popsané v zdrojovém kódu 1:

Zdrojový kód 1: Metoda GetNewValue()

```
/// <summary> Vzorec pro spočítání nové hodnoty ukazatele
public float GetNewValue(List<float> IND, float IdealValue,
                         float PITax_Inf, float GVSpend_Inf,
                         float IntRate_Inf, float LoanCrit_Inf,
                         float affection)
{
    // Výpočet náhodného prvku
    float ind_deviation = Math.Abs(IND[IND.Count - 1] - IdealValue);
    float ind_higher = ind_deviation * 10 + 70;
    float ind_lower = ind_deviation * 10 + 100;
    float rnd_element = (float)(new System.Random().NextDouble() *
                        (ind_higher - ind_lower) + ind_lower) * 0.01f;

    // Efekty působí pouze v prvním a druhém kvartálu události
    // Pokud kvartál není první nebo druhý efekt je nula
    if (!(Quartal % 8 == 0) && !(Quartal % 8 == 1))
    {
        affection = 0;
    }
    // Suma zvolených hodnot nástrojů a efektu události na ukazatel
    float Sum = (PITax_Inf + GVSpend_Inf + IntRate_Inf + LoanCrit_Inf + affection);
    // Přidání náhodného prvku do finálního výpočtu a návrat hodnoty
    return Sum * rnd_element;
}
```

Zdroj: autor, 2023

Jako parametry se do metody podává list hodnot konkrétního indikátoru, jeho ideální hodnota, aktuální efekty všech nástrojů a ve finále efekt aktuální události.

Pro výpočet náhodného prvku je nejdříve nutné určit odchylku, a její hranice. Odchylka je vypočítaná jako absolutní hodnota rozdílu mezi poslední a ideální hodnotou ukazatele. Horní a dolní hranice se určí vynásobením odchylky deseti a přičtením konstant, které byly vybrané teoretickým týmem.

Následně se spočítá náhodný prvek vytvořením náhodné hodnoty mezi předem určenými hranicemi a převedením této hodnoty na procenta.

V pravidlech hry je stanoveno, že událost působí po první dva kvartály od jejího začátku. To se prověřuje zbytkem po dělení aktuálního kvartálu osmi. Pokud je zbytek nula nebo jedna,

nacházíme se v prvním nebo druhém kvartále události a ponechává se efekt události působit. Pokud tomu však tak není, efekt se nuluje.

Na konec je spočítána suma efektů specifikovaných nástrojů a událostí, která je vynásobena spočítaným náhodným prvkem a vracena jako desetinné číslo.

Metoda CalculateValues()

Metoda „CalculateValues()“ odpovídá za výpočet všech hodnot ukazatelů pro nový kvartál na základě zvolených nástrojů a působící události a za aktualizaci listů ukazatelů. Dále je zde nastaven limit minimální nezaměstnanosti (nezaměstnanost nemůže být menší než nula a v praktice neexistuje ani nulová nezaměstnanost, znamenalo by to, že je stát stoprocentně zaměstnaný). Jak lze vidět v podmínce ve zdrojovém kódu 2, pokud se má nezaměstnanost snížit na hodnotu menší než jedno procento, je uměle nastavena na jedno procento plus náhodná hodnota mezi 0 a 0,3.

Zdrojový kód 2: Metoda CalculateValues()

```
public void CalculateValues()
{
    float finalGDP = GetNewValue(GDP, Ideal_GDP,
                                 Pi_tax.GDP_values[Pi_tax.scaleIndex],
                                 Gvt_spending.GDP_values[Gvt_spending.scaleIndex],
                                 Interest_rate.GDP_values[Interest_rate.scaleIndex],
                                 Loan_criteria.GDP_values[Loan_criteria.scaleIndex],
                                 CurrentEvent.GDP_affection);
    float finalINF = GetNewValue(INF, Ideal_INF,
                                 Pi_tax.INF_values[Pi_tax.scaleIndex],
                                 Gvt_spending.INF_values[Gvt_spending.scaleIndex],
                                 Interest_rate.INF_values[Interest_rate.scaleIndex],
                                 Loan_criteria.INF_values[Loan_criteria.scaleIndex],
                                 CurrentEvent.INF_affection);
    float finalUNP = GetNewValue(UNP, Ideal_UNP,
                                 Pi_tax.UNP_values[Pi_tax.scaleIndex],
                                 Gvt_spending.UNP_values[Gvt_spending.scaleIndex],
                                 Interest_rate.UNP_values[Interest_rate.scaleIndex],
                                 Loan_criteria.UNP_values[Loan_criteria.scaleIndex],
                                 CurrentEvent.UNP_affection);

    if (UNP[UNP.Count - 1] - finalUNP < 1f) // Hranice minimální nezaměsntanosti
    {
        // Pokud je budoucí nezaměstnanost měnší než jedna,
        // nastaví se na náhodnou hodnotu mezi 0 a +0.3%
        finalUNP = (float)new System.Random().NextDouble() * 0.3f;
    }

    GDP.Add(UNP[UNP.Count - 1] + finalGDP);
    GDP_total.Add(GDP_total[GDP_total.Count - 1] * (1 + (finalGDP / 100)));
    UNP.Add(UNP[UNP.Count - 1] + finalUNP);
    INF.Add(INF[INF.Count - 1] + finalINF);
}
```

Zdroj: autor, 2023

Metoda ChangeToolIndex()

Metoda „ChangeToolIndex()“ slouží pouze jako prostředek pro změnu indexu daného nástroje pro jiné třídy. Nástroj, jehož index chceme změnit je definovaný parametrem "t" datového typu "Tool" s klíčovým slovem ref pro přímý odkaz na nástroj. Novou hodnotu indexu do metody posíláme přes parametr "val" datového typu int.

Zdrojový kód 3: Metoda ChangeToolIndex()

```
public void ChangeToolIndex(ref Tool t, int val)
{
    t.scaleIndex = val;
}
```

Zdroj: autor, 2023

Metoda ChangeEvent()

Poslední metoda „ChangeEvent()“ podobně jako předchozí je pomocná metoda pro jiné třídy a pouze nastavuje aktuální událost na novou událost náhodně vybranou z listu událostí.

7.5.2 Třída EconomicEvent

Třída „EconomicEvent()“ je datový kontejner pro konkrétní události které mohou v herním světě vzniknout. Každá událost je pojmenována přes pole „Name“ a její popis je uložen v poli „Description“. Efekty, které mají události na ekonomii jsou reprezentovány desetinnými čísly pod názvy „GDP_values“, „UNP_values“ a „INF_values“.

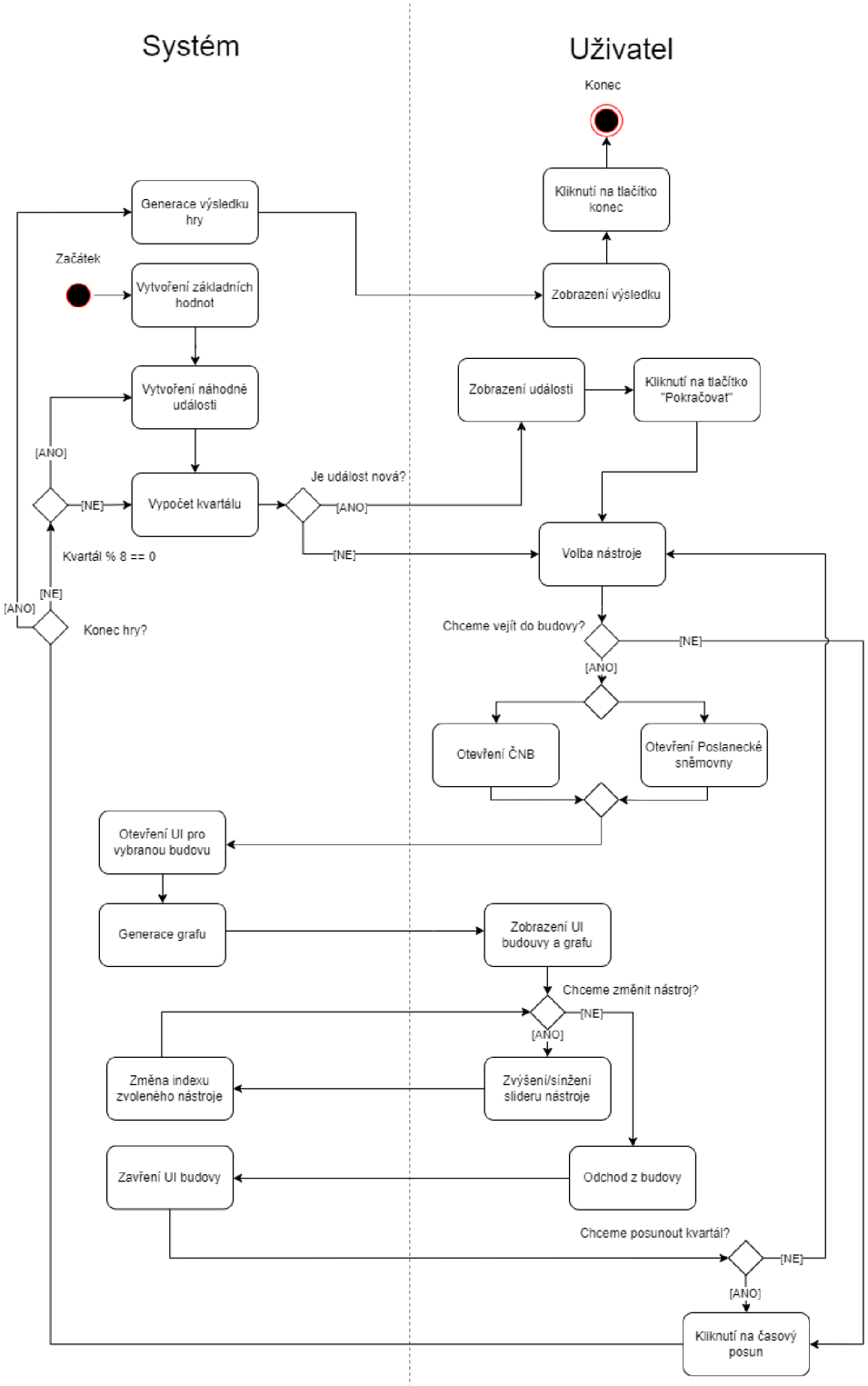
7.5.3 Třída Tool

Třída „Tool“ pomáhá zpřehlednit, jak jsou momentálně jednotlivé nástroje nastaveny a jaké hodnoty působí na ukazatele na škále nástroje. Nástroj je pojmenovaný ve vlastnosti „Name“. Efekty na ukazatele jsou vždy uložené a přednastavené v adekvátním poli ve formě desetinného čísla. „scaleIndex“ určuje, na jakém indexovacím čísle se momentálně nástroj nachází. Tato pole jsou v systému nastavená tak, že snížené hodnoty nástroje jsou uložené na nulovém indexu, neutrální na prvním a zvýšené na druhém.

7.6 Propojení backendu s frontendem

To, jakým způsobem komunikuje systém s uživatelem v průběhu hry je znázorněno na obr. 7.

Obrázek 7: Diagram aktivit komunikace mezi uživatelem a systémem



Zdroj: autor, 2023

7.6.1 Ukládání herních dat

Pro ukládání hodnot konkrétních před-vytvořených událostí a nástrojů bylo nejdříve třídám „Tool“ a „EconomicEvent“ přidaná dědičnost třídy „ScriptableObject“. Instanci třídy, která dědí „ScriptableObject“ lze vytvářet přímo v Unity prostředí, pokud má třída ještě přidaný atribut [CreateAssetMenu] se specifikovaným názvem. Nastavení atributu [CreateAssetMenu] je znázorněné ve zdrojovém kódu 4.

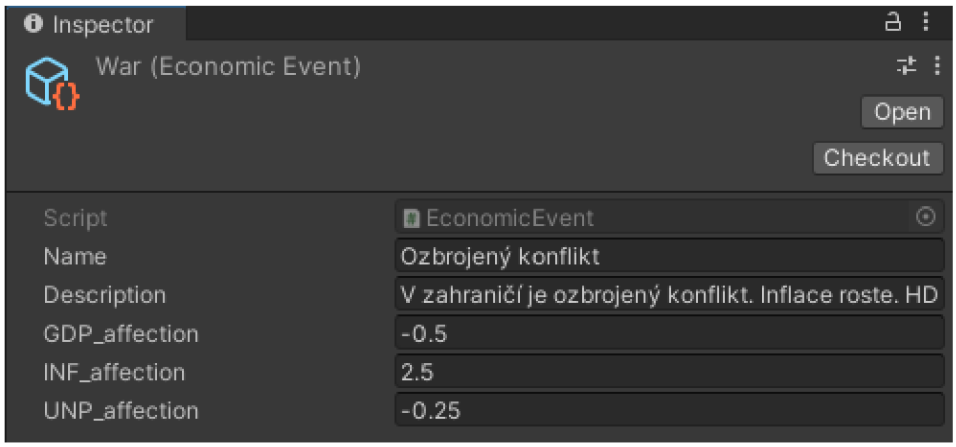
Zdrojový kód 4: Dědění třídy ScriptableObject a nastavení atributu CreateAssetMenu pro třídu Tool

```
[CreateAssetMenu(fileName = "New Tool", menuName = "Macroeconomic Tool")]
public class Tool : ScriptableObject
{
    public string Name;

    public int scaleIndex = 1;

    public float[] GDP_values;

    public float[] UNP_values;

    public float[] INF_values;
}
```

Zdroj: autor, 2023

Instanci takto nastavené třídy lze vytvářet jednoduše kliknutím pravého tlačítka myši v okně projektu a kliknutím na Vytvořit -> "Název třídy" ve vyskakovací nabídce. To vytvoří soubor, reprezentující instanci a v inspektoru se nastaví parametry třídy, jak lze vidět na obrázku 8. Vytváření instancí tímto způsobem znamená, že nově přidané nástroje nebo události lze jednoduše vytvořit v Unity prostředí místo kódu, což usnadnilo vývoj aplikace.

Obrázek 8: Nastavení parametrů pro vytvořenou instanci události



Zdroj: autor, 2023

7.6.2 Třída GameManager

Třída „GameManager“ je odpovědná za veškerou komunikaci mezi systémem a uživatelem. Řídí zobrazování a aktualizaci dat uživatelského rozhraní. Zaznamenává uživatelské vstupy, pomocí kterých upravuje hodnoty nástrojů, a posouvá kvartály. Po dobu vývoje také poskytovala informace pro ladění a testování přes konzoli v Unity prostředí.

Pro napojení s herními objekty a uživatelským rozhraním dědí „GameManager“ třída „MonoBehaviour“ třídu, aby mohla být připojena jako komponent ve scéně do kterého se vkládají instance uživatelského rozhraní nebo některé základní hodnoty.

7.6.3 Vlastnosti třídy GameManager

Vlastnosti, které třída GameManager ovládá, byly nejdříve definovány pomocí veřejných polí s odpovídajícím datovým typem, jak lze vidět ve zdrojovém kódu 5. Následně byli jejich instance a hodnoty ve scéně přiřazeny v uživatelském rozhraní komponenty.

Zdrojový kód 5: Definované vlastnosti třídy GameManager

```
public GameObject newsPrefab;
public PauseMenu pauseMenu;
public Tool Pi_tax;
public Tool Interest_rate;
public Tool Gvt_spending;
public Tool Loan_criteria;
public Slider Pi_tax_slider;
public Slider Interest_rate_slider;
public Slider Gvt_spending_slider;
public Slider Loan_criteria_slider;
public float Starting_GDP;

[DoNotSerialize] public Economy economy = new Economy();
public List<EconomicEvent> Events = new List<EconomicEvent>();
```

Zdroj: autor, 2023

- „newsPrefab“ odkazuje na prefab novin, jehož instance je vytvořena při každé nové události
- „pauseMenu“ se používá pro zastavení průběhu hry
- Všechny vlastnosti typu „Tool“ odkazují na jednotlivé předvytvořené nástroje
- Všechny vlastnosti typu „Slider“ odkazují na posuvníky, které ovládají nástroje
- „Starting_GDP“ je počáteční hodnota HDP a nastavuje se v UI komponenty

- „economy“ drží aktuální data o ekonomii ve hře
- Do listu „Events“ se v UI komponenty vkládá libovolný počet událostí ze kterých bude systém náhodně vybírat každých 8 kvartálů.

7.6.4 Metoda Start() třídy GameManager

Metoda „Start()“, ve třídách, které dědí „MonoBehaviour“ slouží podobně jako konstruktor třídy a je volána při startu hry, veškerá nastavení, která musí být provedena na začátku hry jsou provedena v této třídě.

Jak lze vidět ze zdrojového kódu 6, v metodě „Start()“ se nejdříve přenesou instance nástrojů nastavené v komponentu „GameManager“ do instance třídy economy, nastaví její počáteční HDP a zavolá se metoda pro výběr náhodné události. Ostatní metody, které lze vidět ve zdrojovém kódu 6 rozebírám v následujících kapitolách.

Zdrojový kód 6: Metoda Start() třídy GameManager

```
void Start()
{
    // Zádladní hodnoty ekonomie
    economy.GDP_total.Add(Starting_GDP);
    economy.Pi_tax = Pi_tax;
    economy.Interest_rate = Interest_rate;
    economy.Gvt_spending = Gvt_spending;
    economy.Loan_criteria = Loan_criteria;
    economy.Events = Events;
    economy.ChangeEvent();
    // Metody, které je notno provést při startu hry
    ResetIndexes();
    ShowEventNews();
    NextQuartal();
    AddSliderListeners();
}
```

Zdroj: autor, 2023

7.6.5 Nová událost

Informace o začátku nové události jsou uživateli poskytnuté prostřednictvím novin, kde nadpis hlavního článku reprezentuje název události a text článku poskytuje uživateli bližší informace o události. Od frontendového programátoru mi byl připraven prefab těchto novin, ve kterém jsou textová pole, která stačí v kódu vyplnit hodnotami ze třídy události. Za zobrazení tohoto prefabu uživateli odpovídá metoda „ShowEventNews()“.

K ovládání textu prefabu byl vytvořen jednoduchý „MonoBehaviour“ skript který lze vidět ve zdrojovém kódu 7. Ten v sobě pouze drží instance textových polí reprezentujících název a popis události a tlačítka „Pokračovat“ ke kterému se přiřazuje funkce, která noviny zavře a zruší pauzu hry. Tento skript je pak jako komponent připnutý na prvním podřazeným objektu prefabu.

Zdrojový kód 7: Třída News

```
public class News : MonoBehaviour
{
    public TextMeshProUGUI title;
    public TextMeshProUGUI description;
    public Button continueButton;
}
```

Zdroj: autor, 2023

V metodě ShowEventNews(), znázorněné ve zdrojovém kódu 8, se nejprve zastaví průběh hry přes instanci třídy „PauseMenu“. Dále se vytvoří instance prefabu novin ve scéně, což způsobí jejich zobrazení na obrazovce. Dále se přes třídu „News“ nastaví název a popis události. Na konec se tlačítku přiřadí funkce „ContinueGame()“. Když se tlačítko zmáčkne, zruší se pauza hry a instance prefabu novin se zničí metodou „Destroy()“.

Zdrojový kód 8: Metoda ShowEventNews()

```
// Zobrazení novin uživateli
public void ShowEventNews()
{
    // Zastavení hry
    pauseMenu.PauseWithoutMenu();
    // Instance prefabu do scény
    GameObject news = Instantiate(newsPrefab);
    // Získání komponenty News
    News newsScript = news.GetComponentInChildren<News>();
    // Nastavení textu a funkce tlačítka "Pokračovat"
    newsScript.title.text = economy.CurrentEvent.Name;
    newsScript.description.text = economy.CurrentEvent.Description;
    newsScript.continueButton.onClick.AddListener(() => ContinueGame(news, pauseMenu));

    // Funkce tlačítka pokračovat
    void ContinueGame(GameObject n, PauseMenu p)
    {
        // Při zmáčknutí tlačítka se zruší zastavení hry a zničí instance prefabu
        p.ResumeWithouMenu();
        Destroy(n);
    }
}
```

Zdroj: autor, 2023

7.6.6 Změna nástrojů

Uživatel mění úroveň nástrojů přes posuvníky. Tyto posuvníky jsou nastavené tak, že mohou držet hodnoty celých čísel od 0 do 2. Tato čísla jsou zvolená, protože reprezentují index, který se používá pro volbu hodnot z polí desetinných čísel ve třídě „Tool“. Komponenta, která odpovídá za tyto posuvníky se jmenuje Slider. Tato komponenta poskytuje možnost kontrolovat změnu v hodnotách posuvníku pomocí události, „onValueChanged“ která se zavolá při změně. Každému posuvníku se tedy musí přidat „Listener“ (v překladu „posluchač“), který se vykoná, když se změna provede. Za toto odpovídá třída „AddSliderListeners()“, která se volá v metodě „Start()“, aby byli tyto funkce přiřazeny hned na začátku hry. Každému individuálnímu posuvníku je přiřazena metoda „SetToolScaleIndex()“ s odkazem na nástroj za který posuvník odpovídá a hodnotou posuvníku. Jak lze vidět na obrázku 9, i když by to nemělo být přes posuvník možné, kontroluje se, zda se přes parametr nepodává hodnota menší než nula a větší než dva, protože kdyby se tak stalo způsobilo by to výjimku indexu mimo rozsah ve třídě „Tool“, protože maximální index pro pole hodnot nástroje je 2.

Zdrojový kód 9: Metody AddSliderListeners() a SetToolScaleIndex()

```
public void AddSliderListeners()
{
    Pi_tax_slider.onValueChanged.AddListener(delegate {
        SetToolScaleIndex(ref economy.Pi_tax, (int)Pi_tax_slider.value);
    });
    Interest_rate_slider.onValueChanged.AddListener(delegate {
        SetToolScaleIndex(ref economy.Interest_rate, (int)Interest_rate_slider.value);
    });
    Gvt_spending_slider.onValueChanged.AddListener(delegate {
        SetToolScaleIndex(ref economy.Gvt_spending, (int)Gvt_spending_slider.value);
    });
    Loan_criteria_slider.onValueChanged.AddListener(delegate {
        SetToolScaleIndex(ref economy.Loan_criteria, (int)Loan_criteria_slider.value);
    });
}

void SetToolScaleIndex(ref Tool t, int val)
{
    if (val < 0 || val > 2)
    {
        Debug.LogWarning("Invalid tool index value passed!");
        return;
    }
    economy.ChangeToolIndex(ref t, val);
}
```

Zdroj: autor, 2023

7.6.7 Posun kvartálu

Kvartál se posouvá pokaždé, když uživatel klikne na tlačítko hodinek. Odpovídá za to metoda „NextQuartal()“. Průběh této metody je znázorněn ve zdrojovém kódu 10. Jako první se při zavolání metody zvýší kvartál o jedna. Následně probíhá kontrola, zda je kvartál dělitelný osmi a zároveň se nerovná nule. Pokud se tato podmínka splní, znamená to, že zbytek po dělení aktuálního kvartálu je nula, což znamená že je čas vytvořit novou náhodnou událost. Nulový kvartál je vyloučen, protože se první náhodná událost vybírá v metodě „Start()“.

Aby nová náhodná událost nebyla stejná jako přechozí, je použit „while“ cyklus, ve kterém se vybírá náhodná událost, dokud se nebude lišit od předchozí. Protože v případě, kdyby byla v seznamu pouze jedna událost by while cyklus nikdy neskončil je tento případ ošetřen podmínkou, která zkontroluje, zda je počet událostí roven 1 a pokud ano, zavolá metodu „ChangeEvent()“ pouze jednou.

Pokud se jedná o osmý, šestnáctý a dvacátý čtvrtý kvartál, zobrazí se uživateli noviny obsahující informace o nové události. Pokud se jedná od třicátý druhý, znamená to, že hra končí a uživateli se zobrazí noviny ve kterých je sepsán výsledek jeho výkonu.

Zdrojový kód 10: Metoda NextQuartal()

```
public void NextQuartal()
{
    economy.Quartal++;
    if (economy.Quartal % 8 == 0 && economy.Quartal != 0)
    {
        string previousEventName = economy.CurrentEvent.Name;

        if (Events.Count == 1)
            economy.ChangeEvent();
        else
        {
            do
            {
                economy.ChangeEvent();
            } while (economy.CurrentEvent.Name == previousEventName);
        }
        if (economy.Quartal != 32)
            ShowEventNews();
        else
            ShowEndingNews();
    }
    economy.CalculateValues();
    PrintDebugInfo();
}
```

Zdroj: autor, 2023

7.6.8 Reprezentace stavu ekonomie v prostředí

Uživateli je průběžně naznačováno, zda je ekonomie v pořádku, nebo ne pomocí objektů ve scéně. Například pokud je ekonomie ve špatném stavu, může uživatel v prostředí uvidět bezdomovce ležícího na lavičce. Tyto „pozitivní“ a „negativní objekty“ se průběžně zobrazují a schovávají. Tento úkol má na starosti metoda „UpdateEnvironment()“, jejíž strukturu lze vidět ve zdrojovém kódu 11.

Tato metoda nejprve získá hodnocení ekonomiky pomocí metody GetEvaluation(), s návratovým typem „Evaluation“. Dále deaktivuje všechny pozitivní, negativní a neutrální objekty ve scéně a na základě návratové hodnoty uložené v proměnné „ev“ aktivuje adekvátní objekty reprezentující stav ekonomiky. Aktivace a deaktivace probíhá pomocí metody „SetActive()“, která jako parametr přijímá „boolean“, přičemž „true“ znamená aktivování a „false“ znamená deaktivování.

Zdrojový kód 11: Metoda UpdateEnvironment()

```
public void UpdateEnvironment()
{
    Evaluation ev = GetEvaluation();
    ExcellentEnvironment.SetActive(false);
    GoodEnvironment.SetActive(false);
    BadEnvironment.SetActive(false);

    switch (ev)
    {
        case Evaluation.Excellent:
            ExcellentEnvironment.SetActive(true);
            break;
        case Evaluation.Good:
            GoodEnvironment.SetActive(true);
            break;
        case Evaluation.Bad:
            BadEnvironment.SetActive(true);
            break;
    }
}
```

Zdroj: autor, 2023

Stav ekonomie byl rozdělen na čtyři základní stavy, které reprezentuje výčtový typ „Evaluation“, který může nabýt hodnot „Excellent“, „Good“ a „Bad“. Za získání této hodnoty v systému odpovídá metoda „GetEvaluation()“.

Zdrojový kód 12: Metoda GetEvaluation()

```
private Evaluation GetEvaluation()
{
    float gdpScore, unpScore, infScore = 0;

    // Nezaměstnanost
    float idealUnpMin = 3;
    float idealUnpMax = 5;
    unpScore = GetScore(idealUnpMin, idealUnpMax, economy.UNP);
    // inflace
    float idealInfMin = 1;
    float idealInfMax = 4;
    infScore = GetScore(idealInfMin, idealInfMax, economy.INF);
    // HDP
    int gdpBound = 2;
    gdpScore = GetGDPScore(gdpBound, economy.GDP);

    float avg = new float[] { gdpScore, unpScore, gdpScore }.Average();

    if (avg > 0.2) return Evaluation.Excellent;
    else if (avg < -0.2) return Evaluation.Bad;
    else return Evaluation.Good;

    float GetScore(float min, float max, List<float> indicator)...

    float GetGDPScore(float lowerBound, List<float> indicator)...
}
```

Zdroj: autor, 2023

Jak lze vidět ve zdrojovém kódu 11, výkon hráče se počítá jako průměr mezi skórem všech ukazatelů. Skóre může nabýt hodnoty od -1 do 1, s tím že pokud je hodnota menší než -0.2, vrací se výsledek „Bad“ a pokud je větší než 0.2, vrací se výsledek „Excellent“. Pokud je hodnota v rozmezí od -0.2 do 0.2, vrací se výsledek „Good“. Inflace a nezaměstnanost mají stejný algoritmus výpočtu skóre a odpovídá za něj metoda „GetScore()“, kterou lze vidět ve zdrojovém kódu 13.

V kódu se do volané metody „GetScore()“ v parametrech nastaví minimální a maximální hranice pro výpočet nezaměstnanosti nebo inflace a předá List s hodnotami daného ukazatele. Algoritmus funguje tak, že pro každý případ v historii ukazatele kontroluje, pokud ukazatel nevyšel pryč z rozmezí ideálních hodnot. Pokud se tomu tak stane, sníží se proměnná „count“

o 1, jinak se o 1 zvýší. Ve finále se proměnná count vydělí počtem prvků v listu indikátorů, což ji promění na číslo od -1 do 1.

Zdrojový kód 13: Metoda GetScore()

```
float GetScore(float min, float max, List<float> indicator)
{
    int count = 0;
    foreach (float f in indicator)
    {
        if (f < min || f > max)
        {
            count--;
        }
        else
        {
            count++;
        }
    }
    return count / indicator.Count;
}
```

Zdroj: autor, 2023

Metoda „GetGDPScore()“ se od metody „GetScore()“ liší, protože kritéria, která stanovují, zda jsou hodnoty vhodné nebo nevhodné se liší. Výpočet této metody je znázorněn ve zdrojovém kódu 14.

Rozdíl v hodnocení skóre HDP oproti inflaci a nezaměstnanosti spočívá v tom, že nezaměstnanost a inflace se musí držet v přirozené hodnotě, zatímco HDP může volně růst. Minimální počet procent, o kterých by se mělo HDP za jeden kvartál zvýšit se stanovil na 0,5.

Metoda „GetGDPScore()“ prochází všechny hodnoty v listu indikátoru a porovnává jejich růst oproti předchozí hodnotě. Pokud se však jedná o první prvek v listu, znamená to, že nemá předchozí hodnotu. Navíc, pokud by se počítač pokusil získat prvek na indexu i -1, výsledek by byl -1 a způsobilo by to podmínku indexu mimo rozsah. Proto je tento případ ošetřen podmínkou kontrolující proměnnou „i“.

Systém, pomocí kterého se počítá samotné skóre je podobný jako u metody „GetScore()“, pouze obsahuje jinou podmínku.

Zdrojový kód 14: Metoda GetGDPScore()

```
float GetGDPScore(float lowerBound, List<float> indicator)
{
    float count = 0;

    for (int i = 0; i < indicator.Count; i++)
    {
        float dif = lowerBound; // Toto platí pouze pro první hodnotu

        if (i >= 1)
        {
            dif = indicator[i] - indicator[i - 1];
        }

        if (dif < lowerBound)
        {
            count--;
        }
        else
        {
            count++;
        }
    }
    return count / indicator.Count;
}
```

Zdroj: autor, 2023

7.6.9 Konec hry

Na konci hry se uživateli zobrazí stejné noviny jako při nové události ale název a text článku odráží výkon hráče v textové podobě. Pokud si hráč vedl výborně uvidí v novinách článek o výborném výkonu vlády, pokud si vedl dobře, uvidí článek s neutrálním postojem k vládě a pokud si vedl špatně, bude ve článku uvedeno, že se ve fiktivním státě, ve kterém se hra nachází vyslovila nedůvěra k vládě. Za zobrazení výsledku odpovídá metoda „ShowEndingNews()“ znázorněná ve zdrojovém kódu 15. Tato metoda funguje podobně jako metoda „ShowEventNews() s tím rozdílem, že zaprvé zjistí výkon hráče pomocí metody „GetEvaluation()“ a na jeho základě v novinách nastaví texty pro výsledek výkonu, a za druhé tlačítku „Pokračovat“ změní text na „Konec“ a přiřadí mu funkci pro načtení scény hlavního menu.

Zdrojový kód 15: Metoda ShowEndingNews()

```
// Display the news with the description of the event for a couple of seconds
public void ShowEndingNews()
{
    pauseMenu.PauseWithoutMenu();
    GameObject news = Instantiate(newsPrefab);
    News newsScript = news.GetComponentInChildren<News>();
    Evaluation ev = GetEvaluation();
    Ending ending = new Ending();

    switch (ev)
    {
        case Evaluation.Excellent:
            ending = excellentEnding;
            break;
        case Evaluation.Good:
            ending = goodEnding;
            break;
        case Evaluation.Bad:
            ending = badEnding;
            break;
    }

    newsScript.title.text = ending.title;
    newsScript.description.text = ending.text;
    newsScript.continueButton.onClick.AddListener(delegate { SceneManager.LoadScene(0); });
    newsScript.continueButton.gameObject.GetComponentInChildren<TextMeshProUGUI>().text = "Konec hry";
}
```

Zdroj: autor, 2023

8. Závěr

Na závěr lze říci, že práce přinesla pozitivní výsledky. Ekonomická logika využití nástrojů hospodářské politiky, světových událostí, průběhu hospodářského cyklu a stanovená pravidla byli úspěšně převedeny do C# kódu který byl pak implementován do hry. Propojení backendu na frontendu bylo také úspěchem a výsledkem je funkční videohra, která plní stanovená očekávání. V rámci týmové spolupráce na projektu autor přispěl znalostmi jazyka C# a enginu Unity, což výrazně zrychlilo vývoj videohry. Výsledná aplikace může sloužit jako základ, který se po řádném testování, ladění a přidání dodatečných herních funkcí a mechanik může využít k výuce teorie o fiskální a monetární politice. Celkově byla práce úspěšná a splnila stanovené požadavky.

Summary

This thesis is about programming the economic logic for a 2D videogame about using economic policies for management of the business cycle and implementing that logic in the videogame.

The theoretical part of this thesis introduces the reader to the concept of videogames, programming, and game development in Unity and in general.

The practical part of this thesis describes the process of converting the theory behind the effect of economic policies and world events on the business cycle into C# code and the implementation of this code into the videogame as well as programming the rules of the game.

Keywords

Unity, 2D videogame, programming, C#, game development, economic policies, business cycle

9. Seznam zdrojů

- Gregory, J. (2015). Game Engine Architecture (Druhé vydání). Boca Raton, USA: CRC Press.
- Hemmendinger, D. (2. prosince 2022). Computer programming language. Encyclopedia Britannica. [vid. 2023-03-20]. Dostupné z: https://www.britannica.com/technology/computer-programming-language
- Otte, S. (2009). Version Control Systems. Berlím, Německo.
- Pickell, D. (2019). The 7 Stages of Game Development. [vid. 2023-04-04]. Dostupné z: https://www.g2.com/articles/stages-of-game-development
- Bramble, R. (2023). The seven stages of game development. [vid. 2023-04-04]. Dostupné z: https://gamemaker.io/en/blog/stages-of-game-development
- Lowood, H. E. (2021). Electronic game. Encyclopedia Britannica. [vid. 2023-04-04]. Dostupné z: https://www.britannica.com/topic/electronic-game
- Pavlovic, D. (2020) Video Game Genres: Everything You Need to Know [vid. 2023-04-08]. Dostupné z: https://www.hp.com/us-en/shop/tech-takes/video-game-genres
- González-González, C., & Blanco-Izquierdo, F. (2012). Designing social videogames for educational uses, 58(1), 250-262.
- Griffiths, M. (2002). The educational benefits of videogames, 1-2.
- Aksakal, N. (2015). Theoretical View to The Approach of The Edutainment. Procedia - Social And Behavioral Sciences, 186, 1232-1239.
- RedHat. (2019). What is an IDE?. [vid. 2023-03-29]. Dostupné z: https://www.redhat.com/en/topics/middleware/what-is-ide
- Microsoft. (n.d.). Seznamte se s rodinou Visual Studio. [vid. 2023-03-29]. Dostupné z: https://visualstudio.microsoft.com/cs/#vs-section
- JetBrains (n.d.) Rider for Unity. [vid. 2023-03-29] Dostupné z: https://www.jetbrains.com/lp/dotnet-unity/
- Unity Technologies. (2023-a). Unity - Manual: The Project Window. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/ProjectView.html
- Unity Technologies. (2017). Unity – Manual: Asset Workflow. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/560/Documentation/Manual/AssetWorkflow.html

- Unity Technologies. (2023-c). Unity – Manual: Scenes. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/CreatingScenes.html
- Unity Technologies. (2023-d). Unity – Manual: The Inspector window. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/UsingTheInspector.html
- Unity Technologies. (2023-e). Unity – Manual: The Game view. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/GameView.html
- Unity Technologies. (2023-f). Unity – Manual: The Scene view. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/UsingTheSceneView.html
- Unity Technologies. (2023-g). Unity – Manual: The Hierarchy window. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/Hierarchy.html
- Unity Technologies. (2023-h). Unity – Manual: Introduction to components. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/Components.html
- Unity Technologies. (2023-i). Unity – Manual: ScriptableObject. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/class-ScriptableObject.html
- Unity Technologies. (2023-j). Unity – Manual: Prefabs. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/Prefabs.html
- Unity Technologies. (2022-a). Best practices for organizing your Uunity project. [vid. 2023-03-26]. Dostupné z: https://unity.com/how-to/organizing-your-project
- Unity Technologies. (2023-l). Unity – Manual: Scene view navigation. [vid. 2023-03-26]. Dostupné z: https://docs.unity3d.com/Manual/SceneViewNavigation.html
- Unity Technologies. (2022-b). What is version control?. [vid. 2023-03-21]. Dostupné z: https://unity.com/solutions/what-is-version-control

10. Seznam obrázků

11. Seznam ukázek zdrojového kódu

12. Seznam tabulek