

Využití umělé inteligence ve hrách

David Polašník

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	David Polaščík
Osobní číslo:	A22081
Studijní program:	B0613A140020 Softwarové inženýrství
Forma studia:	Prezenční
Téma práce:	Využití umělé inteligence ve hrách
Téma práce anglicky:	The Use of Artificial Intelligence in Games

Zásady pro vypracování

- Popište současný stav dané problematiky.
- Popište technologie, které budou v práci použité.
- Proveďte analýzu požadavků pro aplikaci.
- Navrhněte a realizujte aplikaci.
- Proveďte otestování aplikace.
- Zhodnotte provedené řešení a navrhněte případné vylepšení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. HARDMAN, Casey. *Game programming with Unity and C#: a complete beginner's guide*. New York: APress, 2020. ISBN 978-1-4842-5655-8.
2. MUSIOL, Martin. *Generative AI: Navigating the Course to the Artificial General Intelligence Future*. Hoboken, New Jersey: John Wiley & Sons, 2024. ISBN 978-1-394-20591-2.
3. BOND, Jeremy Gibson. *Introduction to game design, prototyping, and development: from concept to playable game with Unity and C#*. Second edition. Upper Saddle River, NJ: Addison-Wesley, 2018. ISBN 978-0-13-465986-2.
4. GOOGLE CLOUD, n.d. *Google Cloud Documentation: Find user guides, quickstarts, tutorials, use cases, code samples, and more*. Online. Dostupné z: <https://cloud.google.com/docs/>. [cit. 2024-09-11].
5. UNITY TECHNOLOGIES, n.d. *Unity Documentation: Comprehensive resources for getting started, tutorials, and reference guides*. Online. Dostupné z: <https://docs.unity.com/>. [cit. 2024-10-17].

Vedoucí bakalářské práce: **Ing. Dušan Fojtů, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a implementací aplikace virtuálního asistenta, která využívá pokročilé metody umělé inteligence. Cílem bylo vytvořit interaktivní systém schopný komunikovat s uživatelem prostřednictvím textu i hlasu. Aplikace byla vyvinuta v prostředí Unity a využívá generativní jazykové modely, technologii převodu textu na řeč (TTS) a rozpoznávání řeči (STT). Součástí systému je 3D model postavy se synchronizací rtů a emocí pomocí systému SALSA. Pro potřeby testování a interakce byla vytvořena stylizovaná 3D scéna. Projekt využívá kombinaci cloudových a lokálních řešení pro dosažení plynulé a kontextově vhodné komunikace. Zvláštní důraz byl kladen na modularitu architektury, škálovatelnost a přístupnost.

Klíčová slova: Unity, C#, ChatGPT, Gemini, Text-to-Speech, Speech Recognition, SALSA, virtuální asistent

ABSTRACT

This bachelor thesis focuses on the design and implementation of a virtual assistant application utilizing advanced artificial intelligence technologies. The goal was to develop an interactive system capable of communicating with the user through both text and voice. The application was developed using the Unity engine and integrates generative language models, text-to-speech synthesis (TTS), and speech recognition (STT). The system includes a 3D character model with lip sync and emotional expression handled by the SALSA framework. A stylized 3D environment was created to enhance immersion and engagement. The project combines both cloud-based and local solutions to ensure smooth and contextually relevant communication. Emphasis was placed on modular architecture, scalability, and accessibility.

Keywords: Unity, C#, ChatGPT, Gemini, Text-to-Speech, Speech Recognition, SALSA, virtual assistant

Rád bych vyjádřil upřímné poděkování všem, kteří mi poskytli podporu během zpracování této bakalářské práce i celého studia.

Především děkuji své rodině za neustálou podporu, trpělivost a porozumění, které mi byly oporou v náročných obdobích studia. Jejich důvěra a povzbuzení pro mě měly zásadní význam.

Mé upřímné poděkování patří také mé přítelkyni za její trpělivost, psychickou podporu a pochopení během dlouhých hodin práce, stejně jako za motivaci, která mi pomáhala překonat náročné chvíle.

Zvláštní poděkování si zaslouží vedoucí práce, Ing. Dušan Fojtů, Ph.D., za odborné vedení, cenné připomínky a vstřícný přístup v průběhu celé spolupráce. Jeho podněty výrazně přispěly ke zkvalitnění výsledné podoby této práce.

Všem výše jmenovaným patří mé srdečné díky.

OBSAH

ÚVOD.....	11
I TEORETICKÁ ČÁST	13
1 SOUČASNÝ STAV DANÉ PROBLEMATIKY.....	14
2 POUŽITÉ TECHNOLOGIE.....	17
2.1 HERNÍ ENGINE	17
2.2 PROGRAMOVACÍ JAZYK C#	18
2.3 GENERATIVNÍ MODEL Y	18
2.3.1 Gemini Flash API.....	19
2.3.2 ChatGPT-4o Mini.....	20
2.4 TECHNOLOGIE PŘEVODU TEXTU NA ŘEČ (TEXT-TO-SPEECH, TTS).....	20
2.4.1 ElevenLabs – Pokročilá syntéza hlasu na bázi AI	21
2.4.2 Piper TTS – Open-source řešení pro efektivní syntézu hlasu	21
2.5 ROZPOZNÁNÍ HLASU (SPEECH RECOGNITION)	21
2.5.1 Google Cloud Speech-to-Text.....	22
2.5.2 Apple Speech Framework	22
2.6 VYUŽITÍ BLENDERU PRO TVORBU 3D MODELŮ A ANIMACÍ.....	23
2.7 FIRESTORE DATABÁZE.....	24
2.7.1 Připojení k Firestore databázi.....	24
2.7.2 Ukládání dat do Firestore	25
2.7.3 Bezpečnostní pravidla a řízení přístupu	25
2.7.4 Výhody použití Firestore.....	25
2.8 GOOGLE COLAB PRO TRÉNOVÁNÍ VLASTNÍHO HLASOVÉHO MODELU.....	25
2.9 GOOGLE CLOUD TEXT-TO-SPEECH	27
2.9.1 Technologické pozadí a charakteristiky	27
2.9.2 Role v projektu a srovnání	28
2.9.3 Provozní aspekty	28
2.10 MICROSOFT AZURE TEXT-TO-SPEECH.....	29
2.10.1 Zařazení do projektu	30
2.10.2 Provozní charakteristiky.....	30
2.11 ROKOKO VISION.....	30
II PRAKTICKÁ ČÁST	32
3 ANALÝZA POŽADAVKŮ PRO APLIKACI.....	33
3.1 FUNKČNÍ POŽADAVKY	33
3.2 NEFUNKČNÍ POŽADAVKY	34
4 NÁVRH A REALIZACE APLIKACE.....	36
4.1 VOLBA VÝVOJOVÉHO PROSTŘEDÍ A NÁSTROJŮ	36

4.2	NÁVRH ARCHITEKTURY APLIKACE	37
4.3	TVORBA 3D MODELU A PROSTŘEDÍ	39
4.3.1	Prostředí:	39
4.3.2	Model virtuálního kamaráda:	40
4.3.3	Technické parametry:	41
4.4	REALIZACE: PŘÍPRAVA UNITY PROJEKTU A VÝVOJOVÉHO PROSTŘEDÍ.....	41
4.4.1	Krok 1: Otevření Unity Hubu a vytvoření nového projektu	41
4.4.2	Krok 2: Import assetů z Unity Asset Store a externích zdrojů	42
4.4.3	Krok 3: Vytvoření systémových GameObjectů ve scéně.....	43
4.4.4	Krok 4: Nastavení komponent pro rozpoznávání řeči (Speech Recognition)	44
4.4.5	Krok 5: Přidání a struktura uživatelského rozhraní (UI)	45
4.4.6	Krok 6: Nastavení lip sync systému pomocí SALSA	47
4.4.7	Krok 7: Import 3D modelu pokoje vytvořeného v Blenderu	48
4.4.8	Krok 9: Import 3D modelu postavy Kočky a nastavení Humanoid Rigu	48
4.4.9	Krok 10: Vytvoření Animator Controlleru a nastavení materiálů pro model kočky	50
4.5	PŘÍRAZENÍ SKRIPTŮ A PROPOJENÍ SYSTÉMOVÝCH KOMPONENT V INSPEKTORU	51
4.5.1	Krok 1: Managers	51
4.5.2	Krok 2: Speech	53
4.5.3	Krok 3: Propojení UI komponent se systémem v inspektoru.....	54
4.6	NASTAVENÍ SALSA	55
4.6.1	Krok 1 Nastavení komponenty SALSA	55
4.6.2	Krok 2: Nastavení Emoter	56
4.6.3	Krok 4: Nastavení očí a mrkání pomocí Eyes	56
4.6.4	Krok 5: Propojení s hlasovým výstupem	56
4.7	NASTAVENÍ ANIMATOR CONTROLLERU A IMPORT ANIMACÍ.....	56
4.7.1	Hlavní kroky:.....	57
5	TESTOVÁNÍ APLIKACE.....	58
5.1	MANUÁLNÍ TESTOVÁNÍ.....	58
5.2	UNIT TESTY	59
6	ZHODNOCENÍ PROVEDENÉHO ŘEŠENÍ A NÁVRH VYLEPŠENÍ.....	61
6.1	ZHODNOCENÍ PROVEDENÉHO ŘEŠENÍ.....	61
6.2	NÁVRH VYLEPŠENÍ	61
	ZÁVĚR.....	63
	SEZNAM POUŽITÉ LITERATURY	65
	SEZNAM OBRÁZKŮ	69
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	70
	SEZNAM PŘÍLOH.....	72

ÚVOD

Integrace umělé inteligence (AI) do herního vývoje představuje jeden z nejvýznamnějších a nejdynamičtější se rozvíjejících trendů v oblasti interaktivních aplikací. Potenciál AI transformovat herní zážitky – od chování nehratelných postav (NPC) přes generování obsahu až po personalizaci – je značný. Navzdory těmto možnostem však implementace AI v mnoha komerčních hrách a aplikacích často zaostává za technologickým pokrokem, který nabízí moderní jazykové modely a pokročilé metody strojového učení. AI bývá ve většině případů využívána pouze pro dílčí úkoly nebo jako doplněk ke stávajícím mechanikám, zatímco její využití jako jádra dynamických a adaptivních systémů zůstává spíše výjimečné. Hlubší integrace, v níž by AI autonomně řídila komplexní chování postav včetně animací, dynamicky přizpůsobovala herní svět nebo generovala emergentní narativní struktury, se doposud objevuje převážně v rámci experimentálních či úzce zaměřených projektů.

V tomto kontextu je analyzována pokročilá implementace AI při vývoji virtuálního kamaráda pro děti v prostředí Unity. Zaměření spočívá ve zkoumání procesů integrace více AI systémů s cílem dosáhnout přirozenější a plynulejší interakce oproti tradičním, převážně skriptovaným přístupům. Vyvíjená aplikace umožňuje komunikaci prostřednictvím textových zpráv i hlasového vstupu, čímž rozšiřuje možnosti interakce a zvyšuje míru ponoření do digitálního prostředí.

Klíčovou roli sehrává synergické propojení více AI komponent, zahrnující analýzu textových a hlasových vstupů, generování smysluplných odpovědí pomocí generativních modelů a jejich převod do přirozeně znějící řeči. Důraz je kladen na schopnost asistenta porozumět přirozenému jazyku, přizpůsobovat výstupy kontextu a reagovat autentickým, nepředvídatelným způsobem. Takový přístup nejen obohacuje herní zážitek, ale může zároveň podporovat rozvoj komunikačních dovedností dětí.

Technická část se soustředí na výzvy související s integrací generativních modelů, rozpoznáváním řeči a syntézou hlasu. Analyzovány jsou výkonnostní nároky, zajištění konzistence generovaného obsahu a návrh efektivní komunikace mezi jednotlivými moduly systému. Pozornost je věnována i problémům spojeným s běžícím provozem v reálném čase.

Přínosem realizovaného řešení je komplexní návrh a implementace interaktivního systému, který propojuje rozpoznávání hlasu, generativní AI a syntézu řeči do jednoho funkčního celku. Výsledný prototyp je schopen vést plynulou konverzaci v reálném čase, adaptovat se

na kontext a přizpůsobovat své reakce. Prezentované řešení může sloužit jako případová studie pro vývoj pokročilých AI aplikací v herním prostředí.

Cílem přehledu je poskytnout ucelený pohled na praktické využití moderní umělé inteligence v herním vývoji se zaměřením na technickou implementaci, dopad na uživatelský zážitek a potenciální směry budoucího vývoje v této dynamicky se rozvíjející oblasti.

I TEORETICKÁ ČÁST

1 SOUČASNÝ STAV DANÉ PROBLEMATIKY

Integrace umělé inteligence (AI) do herního průmyslu zažívá v posledních letech dynamický rozvoj, který zásadně mění způsob, jakým jsou hry navrhovány, vyvíjeny a prožívány hráči. Historicky se AI ve hrách soustředila především na funkční aspekty, jako je navigace postav ve světě (pathfinding), základní rozhodovací logika pomocí stavových automatů (Finite State Machines - FSMs) nebo simulace taktického chování protivníků. Tyto tradiční techniky sice umožnily vytvořit funkční a často zábavné herní světy, ale interakce s nehratelnými postavami (NPC) často trpěla předvídatelností, opakujícími se dialogy a omezenou schopností reagovat na dynamické akce hráče. NPC byly spíše reaktivními automaty řídicími se pevně danými skripty než autonomními entitami. [1]

Současný trend, poháněný exponenciálním růstem výkonu hardwaru a především průlomem v oblasti strojového učení a velkých jazykových modelů (Large Language Models - LLMs), směřuje k radikální proměně role AI ve hrách. Vývojáři stále více experimentují s implementací pokročilých AI modelů s cílem vytvořit NPC, které nejenže plní svou herní funkci, ale působí jako uvěřitelné, adaptabilní a skutečně interaktivní součásti herního světa. [2]

Klíčovou oblastí této transformace je konverzační AI a generativní modely. Místo spoléhání se na předem napsané dialogové stromy, které nabízí jen omezený počet voleb a reakcí, moderní přístupy využívají LLMs (jako jsou modely z rodiny GPT, Gemini, Llama a další) k dynamickému generování dialogů v reálném čase. Tyto modely jsou trénovány na obrovských datasetech textu a kódu, což jim umožňuje porozumět kontextu konverzace, udržovat určitou formu "paměti" napříč interakcemi a generovat relevantní, koherentní a často i kreativní odpovědi, které nejsou explicitně naprogramovány. [3] Tímto způsobem mohou NPC reagovat na neočekávané otázky hráčů, zapamatovat si předchozí interakce a přizpůsobit svůj tón a obsah sdělení aktuální situaci ve hře nebo dokonce vztahu s hráčem. Vývojáři mohou těmto modelům pomocí technik jako "prompt engineering" definovat specifickou osobnost, znalosti (např. herní lore) a cíle, aby NPC jednaly v souladu se svou rolí ve světě. [3]

Technologie jako NVIDIA ACE (Avatar Cloud Engine) představují komplexní platformy navržené specificky pro usnadnění tvorby těchto pokročilých digitálních postav. ACE integruje několik AI technologií do jednoho balíku: automatické rozpoznávání řeči (Automatic Speech Recognition - ASR) pro zpracování hlasového vstupu hráče, LLM pro

porozumění záměru a generování textové odpovědi, technologii převodu textu na řeč (Text-to-Speech - TTS) pro syntézu přirozeně znějícího hlasu a AI modely pro generování realistických animací obličeje (lip sync a výrazy), které odpovídají generované řeči. Tech demo, jako je ukázka interakce v kyberpunkovém stánku s ramenem, ilustrují potenciál těchto systémů vytvářet plynulé a uvěřitelné konverzace s NPC, které dalece přesahují možnosti tradičních metod. [5] Podobné ambice mají i další platformy a middleware, například Inworld AI nebo Convai, které nabízejí vývojářům nástroje pro integraci inteligentních NPC do jejich projektů. [4]

Mimo AAA herní vývoj vidíme principy pokročilé konverzační AI aplikované v nástrojích a aplikacích jako Replika AI. Ačkoliv se nejedná o hru v tradičním slova smyslu, Replika slouží jako příklad AI společníka navrženého pro budování osobního vztahu s uživatelem. Učí se z konverzací, přizpůsobuje svou osobnost a snaží se poskytovat empatickou a podpůrnou interakci. Zkušenosti z vývoje a používání aplikací jako Replika – jak pozitivní (schopnost AI navázat zdání vztahu), tak negativní (etické otázky, obavy o soukromí, "uncanny valley" efekt) – poskytují cenné lekce pro herní vývojáře usilující o vytvoření hlubších a emocionálně rezonujících interakcí s NPC.

Další oblastí, kde AI transformuje herní vývoj, je procedurální generování obsahu (Procedural Content Generation - PCG). Zatímco PCG není novinkou, moderní AI techniky, včetně generativních adversariálních sítí (GANs) a dalších modelů strojového učení, umožňují vytvářet komplexnější, koherentnější a estetičtější herní světy, textury, modely a dokonce i příběhové prvky s menším množstvím manuální práce. [2]

Navzdory obrovskému potenciálu čelí implementace pokročilé AI do her řadě výzev

Výpočetní náročnost: Provoz velkých jazykových modelů vyžaduje značné výpočetní zdroje, což je problematické pro real-time aplikace, jako jsou hry, zejména na koncových zařízeních hráčů (PC, konzole, mobily). Řešení se hledají v optimalizaci modelů (kvantizace, destilace), využití cloudových služeb pro inferenci nebo vývoji specializovaného hardwaru.

Konzistence a ovladatelnost: Zajistit, aby AI generované dialogy a chování byly konzistentní s herním světem, příběhem a osobností postavy, je náročné. Existuje riziko, že AI vygeneruje obsah, který je nevhodný, nelogický nebo "rozbíjí" herní fikci (tzv. halucinace modelů). Vývojáři musí implementovat robustní mechanismy pro filtrování, řízení a uzemnění AI odpovědí (např. pomocí Retrieval-Augmented Generation - RAG, které čerpá informace z definované znalostní báze). [3]

Integrace: Bezproblémové napojení AI systémů (zejména LLM) na stávající herní enginy, animační systémy, systémy questů a další herní logiku vyžaduje nové nástroje a postupy.

Etické otázky: Vznikají otázky týkající se soukromí hráčů (zejména při použití hlasového vstupu), možnosti manipulace hráčů prostřednictvím emocionálně sofistikovaných NPC, odpovědnosti za obsah generovaný AI a potenciálního dopadu na lidské vztahy a vnímání reality. [6]

Budoucnost AI ve hrách slibuje ještě hlubší integraci a sofistikovanější aplikace. Můžeme očekávat NPC s dlouhodobou pamětí, které si pamatují minulé interakce a vyvíjejí komplexní vztahy s hráčem, adaptivní herní světy reagující na hráčův styl hry v reálném čase, a příběhy, které se dynamicky větví a mění na základě skutečných hráčských voleb a interakcí s AI postavami. Konvergence různých AI technologií (počítačové vidění, zpracování přirozeného jazyka, generativní modely, posilové učení) pravděpodobně povede k vytvoření skutečně pohlcujících a nepředvídatelných herních zážitků, které posunou hranice interaktivní zábavy. [7]

2 POUŽITÉ TECHNOLOGIE

V této kapitole se podrobně zaměříme na použité technologie a nástroje, které byly klíčové při vývoji aplikace integrující umělou inteligenci pro interakci s uživatelem. Výběr těchto technologií byl pečlivě zvažován na základě požadavků na výkon, dostupnost a kompatibilitu s prostředím Unity, což je široce používaný herní engine známý svou flexibilitou a rozsáhlými možnostmi pro vývoj interaktivních aplikací.

2.1 Herní engine

Unity je výkonný herní engine vyvinutý společností Unity Technologies, který umožňuje vývojářům vytvářet interaktivní 2D a 3D aplikace a hry. Díky své flexibilitě a široké škále nástrojů se stal oblíbeným nástrojem pro vývoj na různých platformách, včetně mobilních zařízení, počítačů, konzolí a webu. [8]

Jednou z klíčových vlastností Unity je jeho podpora více platforem. Vývojáři mohou vytvářet aplikace a hry, které lze snadno exportovat a spouštět na různých zařízeních a operačních systémech, což výrazně zjednodušuje proces distribuce a rozšiřuje dosah aplikace. [8]

Unity také nabízí integrované vývojové prostředí (IDE) s uživatelsky přívětivým rozhraním, které zahrnuje nástroje pro tvorbu a úpravu scén, skriptování, animace a fyziku. Skriptování v Unity je primárně založeno na jazyce C#, což umožňuje vývojářům psát logiku aplikace a interakce s objekty ve hře. [9]

Pro potřeby integrace umělé inteligence (AI) nabízí Unity několik možností. Vývojáři mohou využít vestavěné nástroje a knihovny pro AI, jako jsou systémy pro navigaci a pathfinding, nebo integrovat externí AI služby a frameworky prostřednictvím API. Tato flexibilita umožňuje implementovat pokročilé AI funkce, jako je rozpoznávání hlasu, analýza textu nebo strojové učení, což je klíčové pro aplikace zaměřené na interakci s uživateli, například virtuálního kamaráda pro děti. [10]

Díky rozsáhlé dokumentaci a aktivní komunitě vývojářů je snadné najít podporu a zdroje pro řešení specifických problémů či rozšíření funkcionality aplikace. Unity se tak stává ideální volbou pro vývoj aplikací, které kombinují herní prvky s pokročilými AI technikami.

2.2 Programovací jazyk C#

C# je moderní, objektově orientovaný programovací jazyk vyvinutý společností Microsoft jako součást platformy .NET. Je navržen tak, aby byl jednoduchý, bezpečný a vhodný pro různé typy aplikací, včetně desktopových, webových a mobilních. Jeho syntaxe je podobná jazykům C++ a Java, což usnadňuje přechod pro vývojáře s předchozí zkušeností v těchto jazycích. [11]

prostředí Unity je C# primárním jazykem pro skriptování. Unity podporuje dvě .NET profily: .NET Standard 2.1 a .NET Framework 4.8. Výchozí nastavení je .NET Standard 2.1, které poskytuje širokou kompatibilitu napříč platformami. Pro projekty vyžadující specifické API dostupné pouze v .NET Framework 4.8 lze přepnout na tento profil v nastavení projektu.

[14]

Skriptování v C# v Unity umožňuje vývojářům definovat chování herních objektů, reagovat na uživatelské vstupy a implementovat herní logiku. Díky podpoře objektově orientovaného programování mohou vývojáři vytvářet modulární a znovupoužitelný kód, což usnadňuje údržbu a rozšiřování projektů. [12]

Pro začínající vývojáře nabízí Unity a Microsoft rozsáhlou dokumentaci a tutoriály, které pokrývají základy C# a jeho použití v Unity. Tato podpora zahrnuje příklady kódu, návody krok za krokem a osvědčené postupy, což usnadňuje naučit se efektivně využívat C# při vývoji her a interaktivních aplikací. [13]

Díky silné podpoře komunity a neustálému vývoji se C# stal jedním z nejpobulárnějších jazyků pro vývoj her v Unity, poskytující vývojářům výkonný nástroj pro tvorbu komplexních a interaktivních zážitků.

2.3 Generativní modely

Generativní modely jsou typem umělé inteligence, které se zaměřují na vytváření nových dat na základě vzorů naučených z trénovacího souboru. Na rozdíl od diskriminačních modelů, které pouze klasifikují vstupní data, generativní modely mají schopnost vytvářet nové vzorky, jež odpovídají rozdělení trénovacího souboru.[15]

Mezi nejvýznamnější generativní modely patří generativní adversariální sítě (GANs), variational autoencoders (VAEs) a transformerové modely.

- Generativní adversariální sítě (GANs) pracují na principu dvou sítí: generátoru a diskriminátoru, které mezi sebou soupeří. Generátor se snaží vytvářet realistická data, zatímco diskriminátor se je pokouší rozpoznat jako falešná nebo autentická. [15]
- Variational Autoencoders (VAEs) jsou modely, které se snaží zakódovat vstupní data do latentního prostoru a následně z něj generovat nová data. Jsou oblíbené zejména v úlohách generování obrazů a zvuků. [15]
- Transformer, jako je GPT (Generative Pre-trained Transformer), využívají mechaniku self-attention k porozumění kontextu textu a generování nového obsahu na základě poskytnutých vstupů. Tyto modely jsou široce využívány pro tvorbu textů, překládání jazyků a konverzační AI. [15]

V rámci této práce byla zvolena možnost využití dvou pokročilých generativních modelů umělé inteligence: ChatGPT-4o Mini a Gemini Flash API. Uživatel má možnost vybrat si jeden z těchto modelů v závislosti na požadavcích konkrétní aplikace. Oba modely využívají nejnovější technologie transformátorových architektur, které umožňují efektivní generování obsahu na základě zadaných vstupních parametrů.

2.3.1 Gemini Flash API

Gemini Flash API je model vyvinutý společností Google DeepMind, který je optimalizován pro rychlou tvorbu textového obsahu a analýzu velkých datových souborů. Díky své multimodální povaze je schopen zpracovávat nejen text, ale i další typy vstupních dat, včetně obrázků a zvuku. Využívá pokročilé mechanismy pozornosti (self-attention), což mu umožňuje generovat obsah s vysokou úrovní koherence a kontextového porozumění. Model je také hluboce integrován do ekosystému Google a je optimalizován pro aplikace, které vyžadují zpracování velkého objemu informací s důrazem na přesnost a rychlost odpovědí. [16]

2.3.2 ChatGPT-4o Mini

ChatGPT-4o Mini je lehčí verze modelu ChatGPT-4o vyvinutá společností OpenAI. Tento model je určen především pro rychlou a efektivní generaci textu, přičemž klade důraz na optimalizaci výpočetních nákladů při zachování vysoké úrovně přesnosti. Jeho hlavní využití spočívá v aplikacích, které vyžadují rychlé dialogové interakce, automatizované odpovědi a pokročilé schopnosti porozumění přirozenému jazyku. ChatGPT-4o Mini je navržen tak, aby poskytoval vyvážený poměr mezi výkonem a efektivitou, což z něj činí vhodnou volbu pro aplikace s omezenými výpočetními zdroji. [17]

2.4 Technologie převodu textu na řeč (Text-to-Speech, TTS)

V rámci této práce je využita technologie Text-to-Speech (TTS), která umožňuje převod psaného textu na mluvenou řeč. Moderní TTS systémy využívají pokročilé neuronové modely, které dokážou simulovat intonaci, důraz, emoce a přirozenou plynulost řeči. Tyto technologie nacházejí uplatnění v širokém spektru aplikací, včetně asistenčních systémů, automatizovaných hlasových odpovědí, přístupových technologií pro zrakově postižené a interaktivních virtuálních postav.

Typy syntézy řeči

Existují dvě hlavní metody syntézy řeči:

- Formantová syntéza – Tradiční metoda využívající matematické modely k syntéze jednotlivých zvukových jednotek (fonémů). Výstup má syntetický zvuk a nižší přirozenost, ale je efektivní z hlediska výpočetních nároků. [19]
- Korpusová syntéza – Modernější přístup, který využívá rozsáhlé databáze zaznamenaného hlasu. Výsledná řeč je přirozenější, ale metoda vyžaduje větší množství trénovacích dat a výpočetního výkonu. [20]

V rámci této práce jsou implementovány technologie – ElevenLabs i Piper TTS. Aplikace umožňuje přepínání mezi těmito dvěma systémy podle konkrétního scénáře:

- ElevenLabs je využíván pro vysoce kvalitní hlasové výstupy s možností přizpůsobení intonace a emocí.
- Piper TTS je volitelná alternativa pro offline syntézu řeči s nižšími výpočetními nároky.

V uživatelském rozhraní aplikace je dostupná možnost přepnutí mezi těmito dvěma systémy podle aktuálních požadavků uživatele.

2.4.1 ElevenLabs – Pokročilá syntéza hlasu na bázi AI

ElevenLabs je špičková platforma pro syntézu řeči založenou na hlubokém učení, která umožňuje vytvářet realistické hlasové výstupy s detailním řízením intonace, emocí a přirozeného projevu. Tento systém se vyznačuje:

- Hlasovou adaptací – možnost vytvořit vlastní hlas na základě krátkého vzorku.
- Podporou více jazyků a dialektů – včetně možnosti převodu mezi různými akcenty.
- Pokročilou emocí a intonací – AI je schopna simulovat různé emoce, jako je radost, smutek nebo vzrušení.

ElevenLabs se běžně používá v oblasti audioknih, mediální produkce, herního průmyslu a interaktivních aplikací, kde je důležitá vysoká kvalita hlasového výstupu. [21]

2.4.2 Piper TTS – Open-source řešení pro efektivní syntézu hlasu

Piper TTS je open-source systém pro TTS, který se zaměřuje na nízké latence a efektivitu běhu na méně výkonném hardwaru. Tento systém umožňuje:

- Trénování vlastních hlasových modelů – uživatel může vytvořit a upravit vlastní hlasový výstup.
- Optimalizaci pro edge computing – efektivní provoz na lokálních zařízeních bez nutnosti připojení k cloudu.
- Podporu více hlasových modelů – Piper umožňuje použití různých modelů pro syntézu řeči v různých jazycích.

Piper TTS je oblíbený zejména v open-source komunitě a projektech zaměřených na lokální hlasovou syntézu, kde je kladen důraz na soukromí a nezávislost na cloudových službách. [22]

2.5 Rozpoznání hlasu (Speech Recognition)

Rozpoznání hlasu (Speech Recognition) je technologie umožňující převod mluvené řeči na text. Tento proces využívá pokročilé algoritmy strojového učení a neuronové sítě, které analyzují zvukové signály a transformují je do digitální podoby. Moderní systémy rozpoznávání hlasu jsou schopny rozlišovat různé jazyky, akcenty a dokonce i emocionální

tón řeči, což umožňuje jejich široké uplatnění v digitálních asistentech, hlasovém ovládání zařízení, automatizovaných odpovědních systémech a přístupových technologiích pro uživatele se speciálními potřebami.

V rámci této práce jsou využity dva komplementární systémy pro rozpoznávání hlasu – Google Cloud Speech-to-Text API a Apple Speech Framework, přičemž každý z nich nabízí specifické výhody v závislosti na požadavcích aplikace.

2.5.1 Google Cloud Speech-to-Text

Google Cloud Speech-to-Text API je výkonná cloudová služba umožňující konverzi mluveného slova na text s využitím nejmodernějších neuronových sítí. Tento systém podporuje více než 125 jazyků a dialektů, včetně češtiny, a je optimalizován pro vysokou přesnost rozpoznání i v hlučném prostředí.

Mezi jeho hlavní vlastnosti patří:

- Rozpoznání v reálném čase – API umožňuje okamžitou transkripci mluveného slova, což je ideální pro interaktivní aplikace.
- Možnost trénování specifických modelů – umožňuje přizpůsobení modelu pro konkrétní použití, například odbornou terminologii či jména.
- Podpora více mluvčích a adaptace na hlučné prostředí – API je schopno analyzovat řeč v reálných podmínkách a rozlišovat mezi různými hlasy.
- Automatická interpunkce a formátování textu – generovaný text je již upravený s interpunkčními znaménky, což zvyšuje jeho čitelnost.

Google Cloud Speech-to-Text API je široce využíváno v firemních řešeních, chatbotových asistentech, hlasových ovládáních, mediálních aplikacích a v automatizovaných službách zákaznické podpory, kde je klíčová vysoká přesnost rozpoznání. [23]

2.5.2 Apple Speech Framework

Apple Speech Framework je nativní rozpoznávací technologie vyvinutá pro iOS, macOS, watchOS a tvOS, která umožňuje rychlou a bezpečnou konverzi hlasu na text. Na rozdíl od Google Cloud Speech API je primárně zaměřena na lokální rozpoznávání řeči, což zajišťuje nižší latenci a vyšší úroveň soukromí.

Hlavní vlastnosti Apple Speech Framework:

- Lokalizované rozpoznání – podpora více jazyků s optimalizací pro Apple zařízení.
- Offline režim – možnost rozpoznávání hlasu přímo na zařízení bez nutnosti připojení k internetu.
- Nízká latence a rychlá odezva – optimalizace pro aplikace běžící v reálném čase, jako je hlasové ovládání.
- Ochrana soukromí – rozpoznávání probíhá lokálně na zařízení, což minimalizuje přenos dat do cloudu.
- Integrace s ekosystémem Apple – možnost propojení se Siri, diktováním textu a dalšími nativními funkcemi systému iOS.

Apple Speech Framework je ideální pro mobilní aplikace, asistenční technologie a bezpečnostně orientované aplikace, kde je klíčová rychlost a ochrana soukromí uživatelů. [24]

2.6 Využití Blenderu pro tvorbu 3D modelů a animací

Blender je open-source software určený pro tvorbu 3D grafiky, animací a vizuálních efektů. Tento výkonný nástroj poskytuje široké možnosti v oblasti modelování, texturování, riggování, animace a renderování, čímž se stal jedním z nejrozšířenějších softwarů využívaných v herním průmyslu, filmové produkci a interaktivních aplikacích. Jeho popularita spočívá především v rozsáhlé komunitní podpoře, flexibilitě a neustálém vývoji, který umožňuje implementaci moderních animačních technik. [25]

V rámci této práce byl Blender využit k tvorbě 3D modelů postav, jejich riggování a implementaci blendshapes pro animaci lip sync. Proces modelování probíhal za použití polygonálního modelování, které umožňuje detailní úpravy topologie, UV mapování a texturování. Při optimalizaci modelů byl kladen důraz na nízkou polygonální náročnost, což je zásadní pro jejich efektivní využití v herním enginu. [26]

Pro animaci postav bylo nutné vytvořit rigging (kosterní strukturu), která umožňuje pohyb modelu v reálném čase. Blender poskytuje nástroje pro manipulaci s modelem a přizpůsobení pohybů, přičemž byl využit princip Inverse Kinematics (IK), který zajišťuje realistické pohyby končetin a celého těla. Tento systém umožňuje přirozenější animaci postavy s automatickým přizpůsobením polohy jednotlivých kloubů při pohybu. [27]

Další důležitou součástí bylo vytvoření blendshapes (shape keys), které slouží k synchronizaci pohybu rtů s hlasovým výstupem (lip sync). V rámci modelování byly definovány různé morfologické variace úst, odpovídající jednotlivým fonémům při vyslovování slov. Tyto tvarové změny byly následně implementovány do animačního systému, kde jsou řízeny podle zvukového vstupu, čímž se dosahuje realistické synchronizace mluvené řeči s pohybem rtů. [28]

2.7 Firestore databáze

Firestore je cloudová NoSQL databáze, která umožňuje ukládání dat ve formě dokumentů a kolekcí. I když je často využívána pro real-time aplikace, v rámci této implementace není využíván real-time přístup, ale Firestore slouží jako robustní databázové řešení pro ukládání a správu dat o uživatelských interakcích.

Firestore v aplikaci uchovává data ve formě kolekcí a dokumentů. Pro tuto aplikaci je klíčová kolekce "api_exchanges", kde jsou ukládány jednotlivé interakce mezi uživatelem a virtuálním asistentem. Tato kolekce obsahuje dokumenty, které zaznamenávají zprávy od uživatele, odpovědi AI, metadata o uživatelských interakcích (např. počet tokenů, časová razítka) a další relevantní údaje.

Dokumenty v kolekci api_exchanges jsou propojeny s konkrétními uživatelskými ID a obsahují podrobnosti o každé konverzaci, což umožňuje efektivní správu konverzačních historií a interakcí. [29]

2.7.1 Připojení k Firestore databázi

Pro připojení k Firestore databázi je nezbytné mít nastavené Firebase SDK v aplikaci. Proces připojení zahrnuje:

- Autentifikace uživatele: Uživatel musí být přihlášen, aby mohl komunikovat s Firestore. Používáte Firebase Auth pro přihlašování uživatelů a jejich autentifikaci.
- Inicializace Firebase: Po autentifikaci je třeba inicializovat Firestore, což umožňuje aplikaci připojit se k databázi a začít zapisovat nebo číst data. Firestore je připojen k aplikaci prostřednictvím SDK, které poskytuje potřebné metody pro práci s kolekcemi a dokumenty.
- Ukládání a čtení dat: Po inicializaci databáze mohou být data (např. zprávy uživatele, odpovědi modelu AI) uložena do kolekce api_exchanges. Tato data jsou

organizována do dokumentů, kde každý dokument reprezentuje jednu interakci a obsahuje všechny důležité informace jako text zprávy, metadat (např. tokeny) a časové razítko. [29]

2.7.2 Ukládání dat do Firestore

Aplikace využívá Firestore pro ukládání informací o interakcích mezi uživatelem a AI. Při každé nové interakci jsou následující informace uloženy do databáze:

- Všechny zprávy (od uživatele a od modelu AI).
- SystemInstruction: Pokyny pro AI, které definují její chování v rámci konverzace.
- Počet tokenů: Počet použitých tokenů během interakce.
- Časové razítko: Čas, kdy byla konverzace uložena.

Data jsou ukládána do kolekce `api_exchanges`, která je specifická pro každého uživatele. Každý dokument obsahuje detaily o interakci a je označen unikátním ID, které pomáhá propojit tuto informaci s konkrétní konverzací uživatele. [29]

2.7.3 Bezpečnostní pravidla a řízení přístupu

Firestore nabízí flexibilní způsob řízení přístupu k datům prostřednictvím bezpečnostních pravidel. Tato pravidla určují, kdo má přístup k čtení a zápisu do specifických kolekcí a dokumentů. Pro vaši aplikaci je nezbytné zajistit, aby pouze autentifikovaní uživatelé měli přístup k jejich osobním datům, jako jsou konverzace. [29]

2.7.4 Výhody použití Firestore

- Škálovatelnost: Firestore poskytuje automatické škálování, které je ideální pro aplikace s vysokým počtem uživatelů a interakcí.
- Flexibilita: Umožňuje strukturované ukládání dat do dokumentů a kolekcí, což je vhodné pro uložení komplexních datových struktur, jako jsou zprávy, metadat a časová razítka.
- Bezpečnostní pravidla: Firestore umožňuje detailní nastavení pravidel pro řízení přístupu, což zajišťuje bezpečné ukládání citlivých. [29]

2.8 Google Colab pro trénování vlastního hlasového modelu

Pro vytvoření unikátního hlasového projevu virtuálního asistenta v rámci této práce bylo nezbytné natrénovat vlastní hlasový model kompatibilní s open-source systémem Piper TTS

[22]. Vzhledem k výpočetní náročnosti tohoto procesu byl pro trénování zvolen nástroj Google Colaboratory (Google Colab). Jedná se o cloudovou platformu od společnosti Google, která poskytuje interaktivní prostředí ve formátu Jupyter notebooků, přístupné přes webový prohlížeč. Zásadní výhodou Colabu pro účely strojového učení je možnost bezplatného (s určitými limity) nebo placeného přístupu k výkonným hardwarovým akceleračtorům, konkrétně grafickým procesorovým jednotkám (GPU) a tensorovým procesorovým jednotkám (TPU). [30]

Proces trénování modelu pro syntézu řeči (TTS), jako je ten používaný v Piper TTS, je komplexní úloha hlubokého učení. Vyžaduje:

- Přípravu datového souboru: Sestavení kvalitního datasetu obsahujícího audio nahrávky mluveného slova (řádově hodiny záznamu) a jejich přesné textové transkripce. Kvalita a konzistence datového souboru (čistota nahrávek, správnost transkripce, jednotný styl mluvy) zásadně ovlivňuje kvalitu výsledného syntetizovaného hlasu.
- Předzpracování dat: Audio soubory a texty je nutné převést do formátu, kterému rozumí trénovací algoritmus. To obvykle zahrnuje segmentaci audia, normalizaci hlasitosti, převod textu na fonetickou reprezentaci a vytvoření tzv. mel-spektrogramů z audio signálů.
- Samotné trénování neuronové sítě: Vlastní trénování modelu, které spočívá v iterativním procházení datového souboru a upravování parametrů (vah) neuronové sítě tak, aby se minimalizovala chyba mezi predikovaným a skutečným výstupem (např. mezi generovaným a skutečným mel-spektrogramem). Tento krok je výpočetně nejnáročnější a vyžaduje velké množství maticových operací, které jsou efektivně paralelizovatelné na GPU.

Výhody využití Google Colab pro tento proces byly následující:

- Akcelerace pomocí GPU: Přístup k GPU (např. NVIDIA Tesla T4, K80, nebo novější v závislosti na dostupnosti) v prostředí Colab dramaticky zkrátí dobu potřebnou pro trénování modelu z potenciálních týdnů na běžném CPU na řádově dny nebo hodiny, v závislosti na velikosti datasetu a počtu trénovacích epoch. Bez této akcelerace by trénování vlastního modelu v rámci časových možností bakalářské práce bylo velmi obtížné.

- Předinstalované prostředí: Colab nabízí prostředí s předinstalovaným systémem Linux a mnoha klíčovými knihovnami pro Python a strojové učení (jako TensorFlow, PyTorch, NumPy, SciPy), což usnadnilo instalaci specifických závislostí potřebných pro trénovací skripty Piper TTS a minimalizovalo problémy s kompatibilitou.
- Integrace s Google Drive: Možnost snadno připojit osobní Google Drive umožnila ukládat trénovací data, konfigurační soubory a především výsledné natrénované modely a kontrolní body (checkpoints) tréninku přímo do cloudového úložiště.
- Interaktivita a reprodukovatelnost: Formát Jupyter notebooku umožnil rozdělit trénovací proces do logických kroků (načtení dat, konfigurace, spuštění tréninku, evaluace), přidávat popisky a vizualizace, což zlepšilo přehlednost a usnadnilo ladění a případnou reprodukci experimentů.

Výstupem trénovacího procesu v Google Colab byl soubor obsahující natrénované váhy neuronové sítě (obvykle ve formátu .onnx pro Piper TTS) a případně konfigurační soubor (.json). Tento soubor s modelem byl následně stažen a integrován do Unity aplikace, kde jej systém Piper TTS využívá pro generování syntetické řeči s natrénovaným vlastním hlasem. Použití Colabu tak představovalo klíčový krok pro dosažení personalizovaného hlasového výstupu virtuálního asistenta bez nutnosti investice do drahého lokálního hardwaru.

2.9 Google Cloud Text-to-Speech

Kromě specializované platformy ElevenLabs [21] a lokálního open-source řešení Piper TTS [22], byla pro rozšíření možností hlasového výstupu aplikace integrována a zkoumána také pokročilá technologie syntézy řeči nabízená službou Google Cloud Text-to-Speech (TTS). V rámci této služby byla věnována pozornost především jejím prémiovým, vysoce věrným (high-fidelity) hlasovým modelům. Tyto modely představují špičku v současných technologiích syntézy řeči a jsou výsledkem rozsáhlého výzkumu společnosti Google v oblasti hlubokého učení a generativních modelů pro audio. [31]

2.9.1 Technologické pozadí a charakteristiky

High-fidelity hlasy Google Cloud TTS typicky využívají pokročilé architektury neuronových sítí. Historicky významnou technologií v této oblasti byl WaveNet, hluboký generativní model, který přímo modeluje surovou vlnovou formu audio signálu. Tento přístup umožnil dosáhnout výrazně přirozenějšího a plynulejšího zvuku oproti starším metodám. Novější generace hlasů, často označované jako "Studio" hlasy, staví na těchto

základech a využívají ještě pokročilejší techniky (např. varianty modelů založených na architektuře Transformer nebo pokročilé vocodery) k dalšímu zlepšení kvality.

Klíčové charakteristiky, které odlišují tyto high-fidelity hlasy, zahrnují:

- Přírozená prozódie a intonace: Modely jsou schopny lépe zachytit a reprodukovat rytmus, melodii a důraz v řeči, čímž se výsledný projev blíží lidskému přednesu.
- Vysoká srozumitelnost a čistota zvuku: Generovaný zvuk má minimální artefakty, šum nebo zkreslení, což zajišťuje vynikající srozumitelnost i v náročnějších poslechových podmínkách.
- Plynulé přechody: Přechody mezi hláskami a slovy jsou hladké, bez nepřirozených skoků nebo "robotického" zabarvení.
- Schopnost vyjádřit nuance: Některé modely umožňují jemné řízení aspektů, jako je emoční náboj (i když často méně flexibilně než specializované platformy jako ElevenLabs) nebo styl přednesu (např. zpravodajský, konverzační).

2.9.2 Role v projektu a srovnání

Integrace Google Cloud high-fidelity hlasů do projektu sloužila primárně jako:

- Poskytnutí alternativy nejvyšší kvality: Nabídnout uživateli možnost volby mezi různými prémiovými hlasy (Google vs. ElevenLabs), které se mohou lišit v konkrétním zabarvení, dostupných jazycích/dialektech nebo specifických vlastnostech.
- Referenční bod (Benchmark): Stanovit referenční úroveň pro posouzení kvality ostatních implementovaných TTS řešení (zejména vlastního trénovaného modelu pro Piper TTS).
- Demonstrace škálovatelnosti a flexibility: Ukázat, že architektura aplikace umožňuje snadno integrovat různá externí API a cloudové služby pro TTS, což umožňuje budoucí rozšíření nebo změnu poskytovatele.

2.9.3 Provozní aspekty

Je důležité zdůraznit, že stejně jako ElevenLabs, i Google Cloud TTS, zejména při využití high-fidelity modelů, je cloudovou službou závislou na připojení k internetu. Syntéza řeči neprobíhá lokálně na zařízení uživatele, ale na serverech Google. Tento přístup umožňuje

využití výpočetně velmi náročných modelů bez zatížení koncového zařízení, avšak za cenu určité latence (doba potřebná pro odeslání textu a přijetí audio dat) a nutnosti síťové konektivity. Použití prémiových "Studio" nebo WaveNet hlasů je také spojeno s vyššími náklady v rámci cenového modelu Google Cloud Platform ve srovnání se standardními TTS hlasy.

Volba mezi Google Cloud high-fidelity hlasy a například ElevenLabs tak v praxi závisí na specifických požadavcích projektu: preferovaném hlasovém charakteru, potřebě jemného řízení emocí (kde může mít ElevenLabs výhodu), rozpočtu na provoz a akceptovatelné míře závislosti na cloudové infrastruktuře. V kontextu této práce poskytuje integrace Google Cloud TTS další cenný rozměr pro porovnání a demonstraci spektra dostupných TTS technologií.

2.10 Microsoft Azure Text-to-Speech

Jako další významná cloudová platforma nabízející pokročilé služby umělé inteligence byla do přehledu zahrnuta a pro účely srovnání a demonstrace flexibility integrována také služba Text-to-Speech (TTS) z portfolia Microsoft Azure Cognitive Services. Azure TTS představuje konkurenceschopné řešení k nabídkám Google Cloud [31] a specializovaným platformám jako ElevenLabs [21], poskytující širokou škálu možností pro syntézu řeči. [32]

Klíčovým prvkem Azure TTS jsou neuronové hlasy (Neural Voices). Podobně jako u konkurenčních řešení (např. WaveNet u Google), i zde se jedná o modely založené na hlubokých neuronových sítích, které jsou trénovány na rozsáhlých datasetech, aby dosáhly vysoké míry přirozenosti a plynulosti řeči. Tyto neuronové hlasy minimalizují robotický přízvuk a lépe napodobují lidskou intonaci a prozódii. [32]

Azure TTS nabízí řadu funkcí a vlastností relevantních pro interaktivní aplikace:

- Široký výběr hlasů a jazyků: Platforma poskytuje rozsáhlou knihovnu předtrénovaných neuronových hlasů v mnoha jazycích a dialektech, často s různými hlasovými charakteristikami (mužské, ženské, různého věku).
- Přizpůsobení výstupu: Umožňuje detailní kontrolu nad generovanou řečí pomocí standardu SSML (Speech Synthesis Markup Language). Vývojáři mohou upravovat rychlost řeči, výšku tónu, hlasitost, vkládat pauzy nebo specifikovat výslovnost konkrétních slov.

- Mluvní styly (Speaking Styles): Pro některé hlasy jsou k dispozici různé styly přednesu, jako je zpravodajský, konverzační, zákaznický servis nebo empatický, což umožňuje lépe přizpůsobit hlas kontextu aplikace.
- Vlastní neuronové hlasy (Custom Neural Voice): Azure nabízí také možnost trénování zcela vlastního hlasu na základě dodaných nahrávek, podobně jako ElevenLabs, což umožňuje vytvořit unikátní hlasovou identitu pro značku nebo postavu (tato pokročilá funkce však nebyla v rámci této práce přímo implementována, ale je relevantní pro kontext). [32]

2.10.1 Zařazení do projektu

Integrace Azure TTS sloužila především k doplnění spektra dostupných cloudových TTS řešení. Poskytla další datový bod pro srovnání kvality, přirozenosti a flexibility syntézy řeči od hlavních poskytovatelů cloudových služeb. Umožnila ověřit, že aplikační architektura je dostatečně flexibilní pro připojení různých TTS API s minimálními úpravami.

2.10.2 Provozní charakteristiky

Stejně jako ostatní zde zmíněné cloudové TTS služby (Google Cloud, ElevenLabs), i Azure TTS vyžaduje připojení k internetu pro komunikaci s API a generování řeči probíhá na serverech Microsoftu. Využití služby je zpoplatněno dle ceníku Azure, přičemž náklady se obvykle odvíjejí od počtu zpracovaných znaků textu a typu použitého hlasu (neuronové hlasy bývají dražší než starší standardní hlasy).

2.11 Rokoko Vision

Pro rychlejší a efektivnější tvorbu realistických animací pro 3D model virtuálního asistenta byla v rámci projektu využita technologie Rokoko Vision ve spojení se softwarem Rokoko Studio. Tento přístup umožňuje získávat data o pohybu (motion capture) přímo z videozáznamu pomocí algoritmů počítačového vidění a umělé inteligence. [33]

Tradiční metody animace, jako je klíčování (keyframing) nebo ruční práce s rigem v 3D softwaru [27], jsou časově náročné, zejména při potřebě velkého množství rozmanitých a přirozeně působících pohybů. Zároveň jsou komerční systémy pro motion capture založené na senzorech nebo optickém sledování často finančně nákladné a vyžadují specifické vybavení. Rokoko Vision nabízí cenově dostupnější alternativu založenou na zachycení

pohybu z běžného videozáznamu pořízeného standardní kamerou (např. webkamerou, kamerou telefonu). [33]

Rokoko Vision představuje moderní alternativu využívající algoritmy počítačového vidění a strojového učení k extrakci dat o pohybu přímo ze standardního 2D videozáznamu. Software analyzuje video pořízené běžnou kamerou (webkamera, telefon atd.) a dokáže rozpoznat a sledovat klíčové klouby a body na těle zaznamenané osoby. Na základě této analýzy následně rekonstruuje 3D pohyb této osoby. Tento proces je mnohem dostupnější, nevyžaduje žádný speciální hardware kromě kamery a dostatečně výkonného počítače.

Data zachycená v Rokoko Vision se následně zpracovávají a upravují v Rokoko Studiu. Rokoko Studio je centralizovaná platforma, která slouží pro záznam, správu a úpravu motion capture dat z různých zdrojů Rokoko (včetně Vision). Umožňuje data čistit od šumu, vyhlazovat, slučovat různé záznamy a především provádět retargeting. Retargeting je proces aplikace zachyceného pohybu z jedné kostry (v tomto případě rekonstruované lidské kostry z videa) na jinou kostru (rig 3D modelu kočky). Tento krok vyžaduje pečlivé namapování kostí a doladění měřítko, aby se pohyb správně přenesl na jinou anatomii postavy. [33]

II PRAKTICKÁ ČÁST

3 ANALÝZA POŽADAVKŮ PRO APLIKACI

Tato kapitola specifikuje funkční a nefunkční požadavky kladené na vyvíjenou aplikaci virtuálního asistenta.

3.1 Funkční požadavky

Funkční požadavky definují specifické funkce a chování, které musí aplikace poskytovat.

- Hlasový vstup uživatele: Aplikace musí umožnit uživateli zadávat vstupy prostřednictvím mluvené řeči. Musí být integrována podpora pro cloudovou službu Google Cloud Speech-to-Text API a nativní Apple Speech Framework, aby byla zajištěna funkčnost na platformách Android a iOS.
- Volba technologie rozpoznávání řeči: Aplikace musí poskytnout uživateli možnost výběru preferované technologie pro rozpoznávání řeči (Google Cloud nebo Apple Speech) v rámci nastavení aplikace.
- Generování odpovědi pomocí AI: Aplikace musí zpracovat textový vstup (získaný z rozpoznávání řeči nebo přímým zadáním) a generovat relevantní a kontextově odpovídající textové odpovědi pomocí externích generativních jazykových modelů. Součástí odpovědi modelu mohou být i specifické příkazy pro řízení chování a animací virtuálního asistenta. Musí být implementována podpora pro modely OpenAI ChatGPT-4o Mini a Google Gemini Flash API.
- Syntéza řeči (TTS): Aplikace musí převést textovou odpověď (generovanou AI) na syntetizovanou mluvenou řeč. Musí být integrována podpora pro cloudovou službu ElevenLabs, lokální systém Piper TTS (s možností využití vlastního trénovaného modelu), Google Cloud TTS a Microsoft Azure TTS.
- Volba technologie syntézy řeči: Aplikace musí umožnit uživateli vybrat preferovaný systém a hlas pro syntézu řeči v rámci nastavení.
- Perzistence interakcí: Aplikace musí ukládat historii konverzací mezi uživatelem a AI asistentem. Pro každou interakci musí být zaznamenány minimálně textové zprávy uživatele, textové odpovědi AI, identifikátor použitého jazykového modelu, počet tokenů (pokud je relevantní) a časové razítko. Data musí být ukládána do cloudové databáze Google Firestore v rámci kolekce specifické pro daného autentizovaného uživatele.

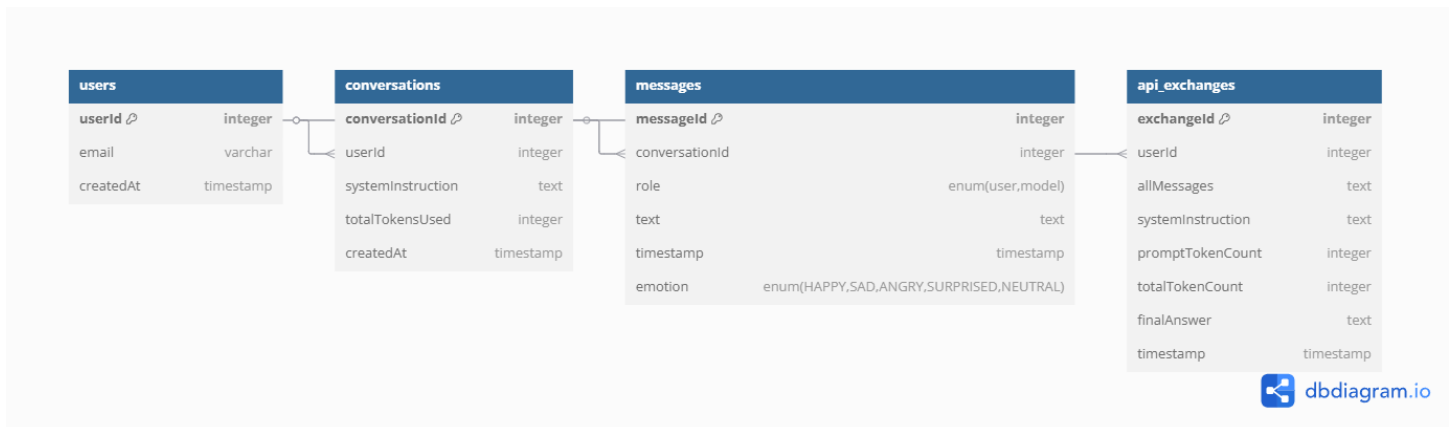
- Uživatelské rozhraní pro interakci: Aplikace musí poskytovat grafické uživatelské rozhraní (GUI) umožňující zobrazení konverzace, zadávání textových vstupů, aktivaci hlasového vstupu a přehrávání syntetizované řeči. GUI musí také zobrazovat 3D model virtuálního asistenta.
- Řízení animací 3D modelu: Aplikace musí interpretovat příkazy pro animace získané z odpovědi AI a spouštět odpovídající animace na 3D modelu virtuálního asistenta (např. gesta, změny postoje, specifické akce).
- Lip Sync (Synchronizace rtů): Aplikace musí analyzovat audio výstup ze syntézy řeči a v reálném čase synchronizovat pohyb rtů (např. pomocí blendshapes) 3D modelu virtuálního asistenta tak, aby odpovídal mluvenému slovu.

3.2 Nefunkční požadavky

Nefunkční požadavky definují kvalitativní atributy a omezení systému.

- Výkon a odezva: Aplikace musí být optimalizována pro plynulý běh na cílových mobilních platformách (Android, iOS). Latence mezi dokončením hlasového vstupu uživatele a začátkem přehrávání syntetizované odpovědi by měla být minimalizována (ideálně pod 3-5 sekund při stabilním síťovém připojení pro cloudové služby). Lokální operace (Piper TTS, Apple Speech) by měly mít nižší latenci. Přehrávání animací 3D modelu a lip sync nesmí negativně ovlivňovat plynulost aplikace ani způsobovat znatelné zpoždění v interakci.
- Bezpečnost a soukromí: Veškerá komunikace s externími API (Generativní modely, Cloudové TTS/STT, Firestore) musí probíhat šifrovaně (HTTPS). Přístup k datům v databázi Firestore musí být řízen bezpečnostními pravidly tak, aby uživatel měl přístup pouze ke svým vlastním datům. Citlivá data, jako jsou API klíče, nesmí být uložena v klientském kódu v čitelné podobě. Zpracování audio dat musí být v souladu s podmínkami jednotlivých služeb a principy ochrany osobních údajů.
- Kompatibilita: Aplikace musí být kompatibilní s operačním systémem Android verze 8.0 (Oreo) a vyšší a iOS verze 14.0 a vyšší. Aplikace by měla korektně fungovat na různých velikostech obrazovek mobilních telefonů a tabletů.
- Použitelnost: Uživatelské rozhraní musí být navrženo s ohledem na cílovou skupinu (děti), musí být intuitivní, snadno ovladatelné a vizuálně přívětivé. Počet kroků potřebných k zahájení a vedení konverzace musí být minimální.

- Škálovatelnost: Backendové služby (zejména Firestore a cloudové AI API) musí být schopny zvládnout rostoucí počet uživatelů a interakcí bez výrazného zhoršení výkonu nebo dostupnosti.
- Údržba a rozšiřitelnost: Kód aplikace by měl být strukturovaný, čitelný a modulární, aby umožňoval snadnou údržbu a budoucí rozšiřování o nové funkce nebo podporu dalších AI modelů a služeb.
- Vizuální kvalita: Animace 3D modelu a lip sync by měly působit plynule a co nejpřirozeněji v rámci zvoleného grafického stylu a technických možností.



Obrázek 1 Diagram Firestore databáze

4 NÁVRH A REALIZACE APLIKACE

Tato kapitola detailně popisuje jednotlivé fáze a kroky vedoucí k realizaci aplikace. Od výběru základních vývojových nástrojů a definování softwarové architektury, po detailní implementaci hlavních funkčních a technických celků aplikace. Cílem je představit, jakým způsobem byly požadavky převedeny do funkčního softwarového řešení.

4.1 Volba vývojového prostředí a nástrojů

V počáteční fázi projektu bylo klíčové zvolit vhodné vývojové prostředí a sadu nástrojů, které umožní efektivní a úspěšnou realizaci aplikace splňující specifikované požadavky. Zvolené nástroje musely podporovat multiplatformní vývoj pro mobilní zařízení, práci s 3D grafikou a animacemi a flexibilní integraci externích služeb umělé inteligence.

Na základě zhodnocení dostupných možností byly vybrány následující klíčové komponenty technologického stacku:

- Herní engine: Unity verze [6000.0.47f1] byl zvolen jako hlavní vývojová platforma díky své vynikající podpoře multiplatformního vývoje (zejména pro Android a iOS), robustním integrovaným nástrojům pro 3D a animace a otevřenosti pro integraci externích API a pluginů. Projekt byl nastaven jako 3D projekt cílený na mobilní zařízení.
- Programovací jazyk: Pro veškeré skriptování byl použit jazyk C# v prostředí Microsoft Visual Studio, využívající profil .NET Standard 2.1 pro širokou kompatibilitu.
- Tvorba 3D assetů a animací: Pro modelování, rigging, tvorbu blendshapes a animací 3D modelu virtuálního asistenta byl využit open-source software Blender. Pro rychlé získávání pohybových dat (motion capture) z videa byl doplnkově použit software Rokoko Vision ve spojení s Rokoko Studiem.
- Návrh uživatelského rozhraní (UI): Pro implementaci grafického rozhraní aplikace byl zvolen moderní systém Unity UI Toolkit, umožňující deklarativní návrh a stylování UI.
- Externí AI služby (Cloud API): Pro zajištění jádrových AI funkcionalit (generování textu, rozpoznávání a syntéza řeči) byly integrovány API od předních poskytovatelů cloudových služeb, konkrétně Google Gemini API, OpenAI ChatGPT API,

ElevenLabs API, Google Cloud Speech-to-Text API, Apple Speech Framework a Microsoft Azure Text-to-Speech API. Ačkoliv primární perzistence využívá lokální soubory, zvažována a popsána v teorii byla i databáze Google Firebase Firestore.

- Doplňkové knihovny a pluginy: Pro specifické úkoly, jako je automatický lip sync (SALSA LipSync Suite), snadnější integrace cloudových STT pluginů (FrostweepGames, KKSpeech) a práci s JSON daty (Newtonsoft.Json), byly do projektu přidány relevantní externí knihovny a pluginy. Pro lokální TTS pak knihovna LeastSquares.Overtone (pro Piper TTS).
- Správa verzí: Pro efektivní správu zdrojového kódu, sledování změn v průběhu vývoje, možnost návratu k předchozím verzím a zálohování byl použit systém správy verzí Unity Version Control. Tento systém, integrovaný v rámci platformy Unity, umožnil snadné nahrávání projektu a sledování historie změn přímo z vývojového prostředí.

Diagram Architektura vývojového prostředí a nástrojů viz. Příloha P I

4.2 Návrh Architektury Aplikace

Úspěšná realizace aplikace s komplexní funkcionalitou, jakou je virtuální asistent integrující různé AI služby, vyžaduje promyšlený návrh softwarové architektury. Architektura systému definuje strukturu aplikace, rozdělení do logických modulů a způsob jejich vzájemné komunikace. Dobře navržená architektura přispívá k přehlednosti kódu, usnadňuje vývoj, testování, údržbu a budoucí rozšiřování.

S ohledem na zvolené vývojové prostředí Unity a požadavky na modularitu, flexibilitu a výkon byla zvolena modulární, komponentově orientovaná architektura (Component-Based Architecture). Tento architektonický styl je pro Unity přirozený, jelikož engine sám je založen na principu připojování komponent (skriptů) k objektům ve scéně. Cílem bylo rozdělit funkcionalitu aplikace do menších, relativně nezávislých modulů (reprezentovaných Unity GameObjects a C# skripty), které mají jasně definované zodpovědnosti a komunikují spolu prostřednictvím řízených mechanismů (např. přímé volání metod přes reference, události, či statické instance pro manažery).

Hlavní subsystémy (moduly) a jejich zodpovědnosti v rámci navržené architektury jsou následující:

- **UI Vrstva (User Interface Layer):** Tento subsystém je zodpovědný za veškerou vizuální prezentaci aplikace a za příjem uživatelských vstupů prostřednictvím grafického rozhraní. Je implementován pomocí Unity UI Toolkit a spravován sadou C# skriptů (MainUIController.cs, ChatUIController.cs, SettingsUIController.cs, MemoryUI.cs, LevelDetailsUIController.cs, případně Popup.cs). Tyto kontrolery spravují konkrétní obrazovky nebo panely UI, reagují na interakce uživatele (kliknutí na tlačítka, zadávání textu) a signalizují tyto akce dalším subsystémům (např. manažeru interakce). Rovněž přijímají data k zobrazení z jiných částí systému.
- **Manažer Interakce (Interaction Manager):** Jádro řídicí logiky aplikace, implementované především ve skriptu ChatManager.cs. Plní roli orchestrátora celého konverzačního cyklu. Přijímá zpracované uživatelské vstupy (textové z ChatUIController, rozpoznanou řeč z STT modulů) a rozhoduje o dalším postupu. Deleguje požadavek na zpracování AI jádru, čeká na odpověď a koordinuje její prezentaci – odesílá text pro zobrazení do UI vrstvy a text pro syntézu řeči do modulu hlasových služeb. Řídí také stav interakce (např. blokuje uživatelský vstup, když AI zpracovává odpověď nebo mluví).
- **AI Jádro (AI Core):** Subsystém zodpovědný za komunikaci s externími cloudovými API pro generativní modely (Google Gemini, OpenAI ChatGPT), implementovaný ve skriptu ChatClientUnity.cs. Tento modul formátuje požadavky na AI modely (včetně historie konverzace pro kontext a systémových instrukcí), odesílá je, zpracovává přijaté odpovědi a parsuje z nich relevantní text nebo požadavky na volání funkcí. Komunikuje se správcem paměti pro získání kontextu konverzace.
- **Správce Paměti (Memory Manager):** Implementován ve skriptu MemoryManager.cs. Zajišťuje perzistenci dat a správu historie konverzace a uživatelského profilu aplikace. Ukládá a načítá data z lokálních souborů (JSON). Spravuje krátkodobou paměť potřebnou pro kontext při volání AI API a plnou historii pro zobrazení a případné sumarizace. Zodpovídá také za čištění textu od specifických příkazů před uložením pro AI.
- **Hlasové Služby (Voice Services):** Tento subsystém je rozdělen na moduly pro rozpoznávání řeči (Speech-to-Text - STT) a syntézu řeči (Text-to-Speech - TTS). Moduly STT (GoogleSpeech.cs pro Google Cloud STT a IOSSpeech.cs pro Apple Speech Framework) zachytávají audio vstup od uživatele a převádějí ho na text.

Modul TTS (KittyVoiceManager.cs) převádí textové odpovědi AI na mluvenou řeč pomocí vybraného TTS engine (ElevenLabs, Piper/Overtone, Google Cloud TTS, Azure TTS). Přepínání mezi STT systémy je řízeno skriptem SpeechModeManager.cs. Komunikace mezi těmito moduly a ChatManager probíhá primárně prostřednictvím událostí (events), signalizujících dokončení rozpoznávání nebo syntézy.

- **3D Model a Vizuální Interakce:** Tento subsystem zahrnuje samotný 3D model virtuálního asistenta s rigem a blendshapes, Unity Animator Controller pro řízení animací, a systém pro animaci obličeje a synchronizaci rtů (SALSA LipSync Suite). Skript KittyEmoter.cs funguje jako rozhraní, které přijímá příkazy pro animace a emoce (často extrahované z odpovědi AI) a spouští příslušné funkce na komponentách Animator a SALSA Emoter.
- **Perzistence Dat:** Ačkoliv teoretická část zmiňuje Google Firebase Firestore, implementace v rámci BP využívá lokální mechanismy pro ukládání dat mezi spuštěními aplikace. Tento subsystem zahrnuje ukládání historie a profilu (MemoryManager.cs do JSON souborů) a ukládání uživatelských nastavení a stavu levelovacího systému (SettingsUIController.cs, LevelSystem.cs pomocí UnityEngine.PlayerPrefs).
- **Doplňkové Herní Systémy:** Zahrnuje implementaci specifických herních mechanik, které obohacují interakci a přidávají gamifikační prvky. Patří sem levelovací systém (LevelSystem.cs), správa herní měny (CurrencyManager.cs), které demonstrují propojení AI s tradičními herními prvky.

Diagram Blokové schéma softwarové architektury aplikace viz. Příloha P II

4.3 Tvorba 3D Modelu a Prostředí

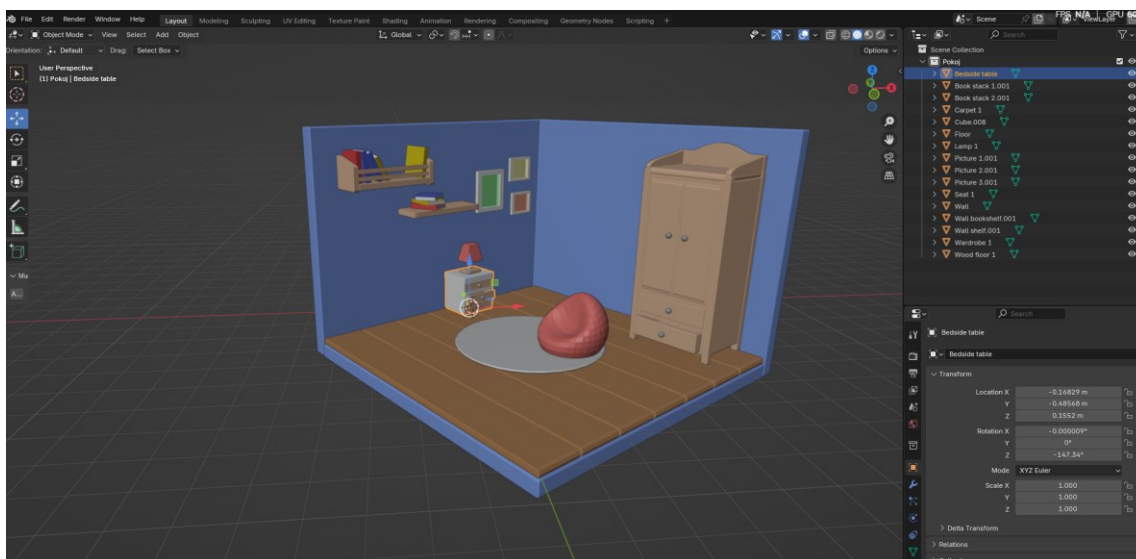
4.3.1 Prostředí:

Pro vytvoření herní scény bylo navrženo stylizované 3D prostředí odpovídající dětskému pokoji, ve kterém se pohybuje virtuální postava. Prostředí bylo vymodelováno v softwaru Blender, přičemž důraz byl kladen na jednoduchost, čitelnost a optimalizaci pro mobilní zařízení.

Scéna obsahuje základní konstrukční prvky jako stěny a podlahu, dále pak interiérové objekty – skříň, lampu, koberec, sedací vak, nástěnné police s knihami a několik obrázků na stěnách. Každý objekt byl vytvořen samostatně a uspořádán do přehledné hierarchie ve scéně.

Modelace probíhala pomocí polygonálního modelování s využitím základních modifikátorů a nástrojů pro transformaci a duplikaci objektů. Textury a barvy byly řešeny formou jednoduchých materiálů s jednotným barevným odstínem, čímž bylo dosaženo stylizovaného „low-poly“ vzhledu vhodného pro zacílení na mladší publikum.

Model scény byl exportován ve formátu .fbx a následně importován do složky Assets/Models v Unity. Po vložení do scény byly objekty vhodně rozmístěny a upraveny jejich transformace. Veškeré materiály byly zachovány nebo případně upraveny pro kompatibilitu s použitým URP profilem.



Obrázek 2 Návrh stylizovaného pokoje vytvořeného v Blenderu.

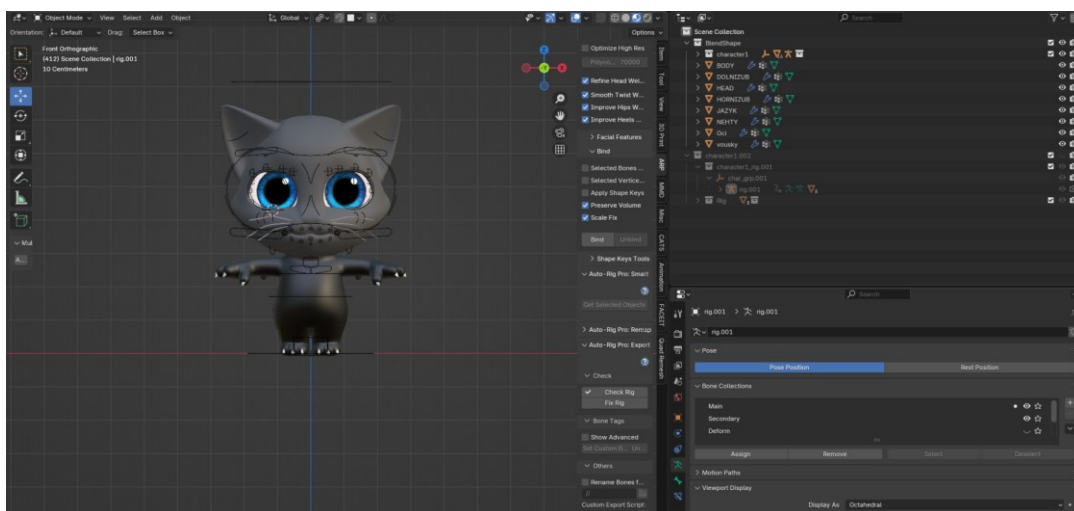
4.3.2 Model virtuálního kamaráda:

Model hlavní postavy – stylizované černé kočky – byl vytvořen v prostředí Blender s využitím technik digitálního modelování a riggování. Vizuální styl byl zvolen tak, aby působil přátelsky a byl atraktivní zejména pro dětské publikum.

Modelace probíhala primárně pomocí nástroje Sculpt Mode, který umožňuje organické tvarování postavy a dosažení měkkých zaoblených tvarů. Model je složen z několika částí, jako jsou tělo, hlava, oči, jazyk a vousky.

4.3.3 Technické parametry:

- **Blendshapes:** Model obsahuje celkem 52 shape keys kompatibilních se systémem Apple ARKit, což umožňuje pokročilou synchronizaci výrazu obličeje a pohybu rtů se zvukem (lip sync). Tyto shape keys zahrnují jednotlivé fonémy, výrazy a pohyby očí, tváří a čelistí.
- **Rigging:** Kosterní struktura byla vytvořena pomocí pluginu Auto-Rig Pro, který umožňuje efektivní vytváření a úpravu rigů pro animaci. Kosti byly vázány k mesh pomocí metody "Bind with Preserve Volume", což zajistilo stabilní deformace při animaci.
- **Texturey a materiály:** Pro oči byla použita ručně malovaná textura s vysokým leskem. Ostatní části těla využívají jednoduché stylizované materiály, které odpovídají barevné paletě celé scény.



Obrázek 3 Model kočky v Blenderu

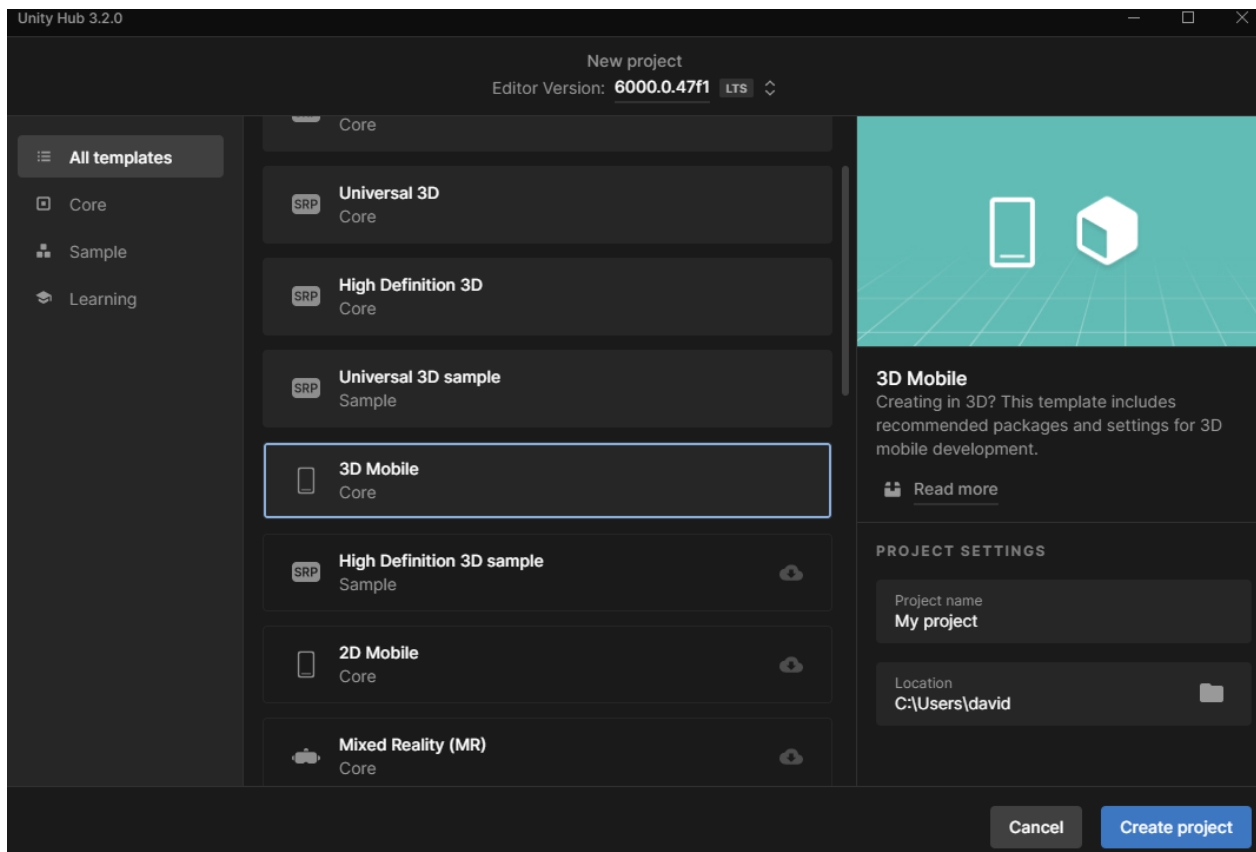
4.4 Realizace: Příprava Unity projektu a vývojového prostředí

Tato část popisuje kroky, které byly provedeny při zakládání projektu v Unity. Cílem je umožnit čtenáři vytvořit identické vývojové prostředí a reprodukovat vývojovou fázi aplikace. Pro maximální srozumitelnost je každý krok doplněn stručným komentářem a doporučenými nastaveními.

4.4.1 Krok 1: Otevření Unity Hubu a vytvoření nového projektu

1. Otevřete aplikaci Unity Hub.
2. Klikněte na tlačítko „New project“.

3. Zvolte šablonu 3D Mobile
4. Projekt pojmenujte například VirtualCompanionAI.
5. Vyberte vhodné umístění projektu a potvrďte vytvoření.



Obrázek 4 Výběr šablony 3D (Mobile)

4.4.2 Krok 2: Import assetů z Unity Asset Store a externích zdrojů

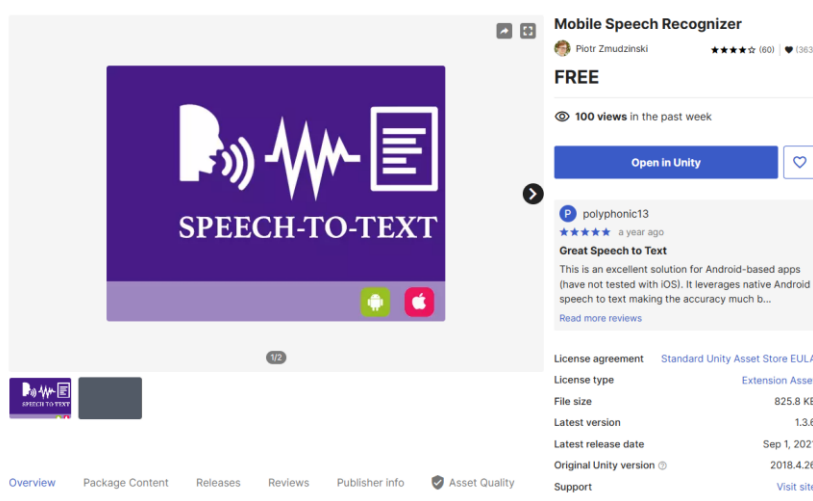
Po vytvoření základního projektu bylo nutné rozšířit funkcionalitu aplikace o pokročilé funkce pro zpracování hlasu a syntézu řeči. K tomu byly využity následující assety a datové sady:

1. Overtone TTS (LeastSquares.Overtone) – zakoupený asset z Unity Asset Store, sloužící jako rozhraní pro integraci systémů pro převod textu na řeč, včetně podpory Piper TTS. [34]
2. SALSA LipSync Suite – asset pro automatickou synchronizaci rtů 3D modelu s audio výstupem. Umožňuje jednoduché napojení na TTS systémy a animace obličeje. [35]

3. Speech Recognition using Google Cloud (od Frostweep Games) – plugin pro napojení na Google Cloud Speech-to-Text API, včetně podpory rozpoznávání v reálném čase. [36]
4. Mobile Speech Recognizer – assets: lokální rozpoznávání hlasu na zařízeních Android/iOS. [37]

Postup importu:

1. V Unity Editoru byla otevřena sekce Window > Asset Store nebo použit webový portál Asset Store.
2. Po přihlášení do uživatelského účtu byly assety nalezeny v sekci My Assets a pomocí tlačítka Import přidány do projektu.
3. Open-source assety byly staženy z GitHubu a manuálně přidány do složky Assets/Plugins.



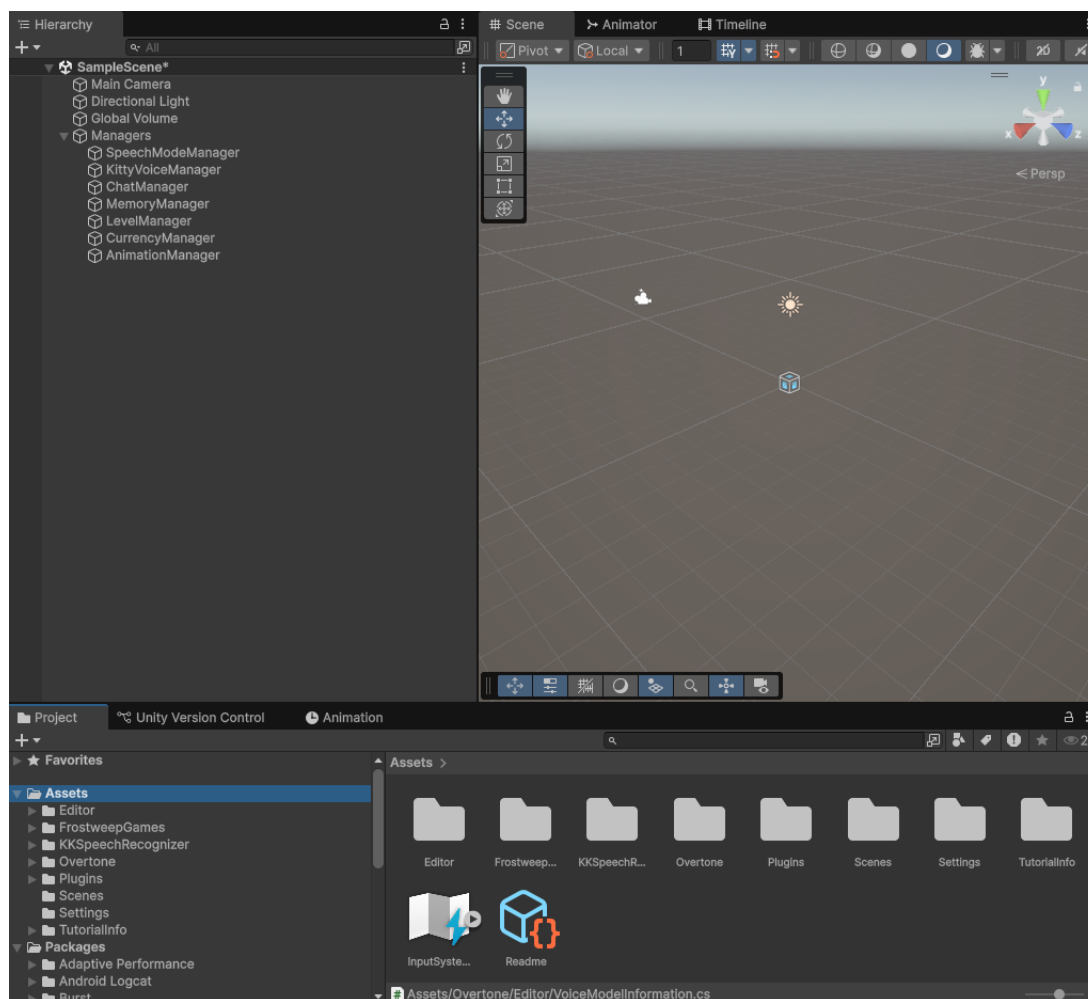
Obrázek 5 Import zakoupených assetů

4.4.3 Krok 3: Vytvoření systémových GameObjectů ve scéně

Po otevření výchozí scény SampleScene bylo vytvořeno systémové jádro scény, které spravuje aplikační logiku:

1. V Hierarchii byl vytvořen nový prázdný GameObject s názvem Managers.
2. Do tohoto objektu byly následně přidány následující podrízené GameObjecty, každý reprezentující specifickou komponentu systému: SpeechModeManager, KittyVoiceManager, ChatManager, MemoryManager, LevelManager, CurrencyManager, AnimationManager

3. Celá tato struktura je uspořádána tak, aby zůstala přehledná a snadno rozšiřitelná i při přidávání dalších manažerů nebo subsystémů.



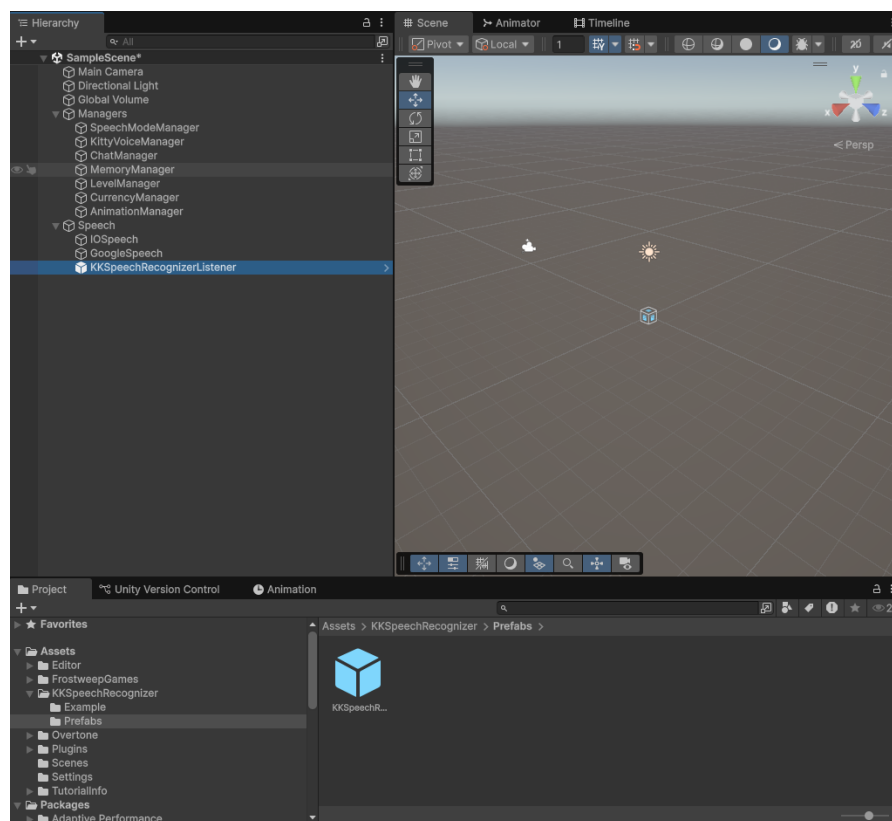
Obrázek 6 Hierarchie objektu Managers

4.4.4 Krok 4: Nastavení komponent pro rozpoznávání řeči (Speech Recognition)

Pro zajištění podpory hlasového vstupu byly do scény přidány samostatné komponenty pro zpracování řeči:

1. Byl vytvořen nový GameObject s názvem Speech.
2. Do něj byly přidány následující podrízené objekty, každý odpovídající specifickému systému rozpoznávání:
 - IOSpeech – modul pracující s Apple Speech Frameworkem pro iOS zařízení.
 - GoogleSpeech – modul propojený s Google Cloud Speech-to-Text API.

- KKSpeechRecognizerListener – prefab, který byl přetažen do scény z pluginu KKSpeech. Tento objekt zajišťuje propojení s nativními rozhraními iOS pro rozpoznávání řeči.
3. Tyto komponenty jsou řízeny centrálním skriptem SpeechModeManager, který volí aktivní vstupní metodu v závislosti na platformě a uživatelském nastavení.
 4. Celý subsystém Speech je navržen tak, aby byl snadno rozšiřitelný o další poskytovatele STT nebo simulátory vstupu.



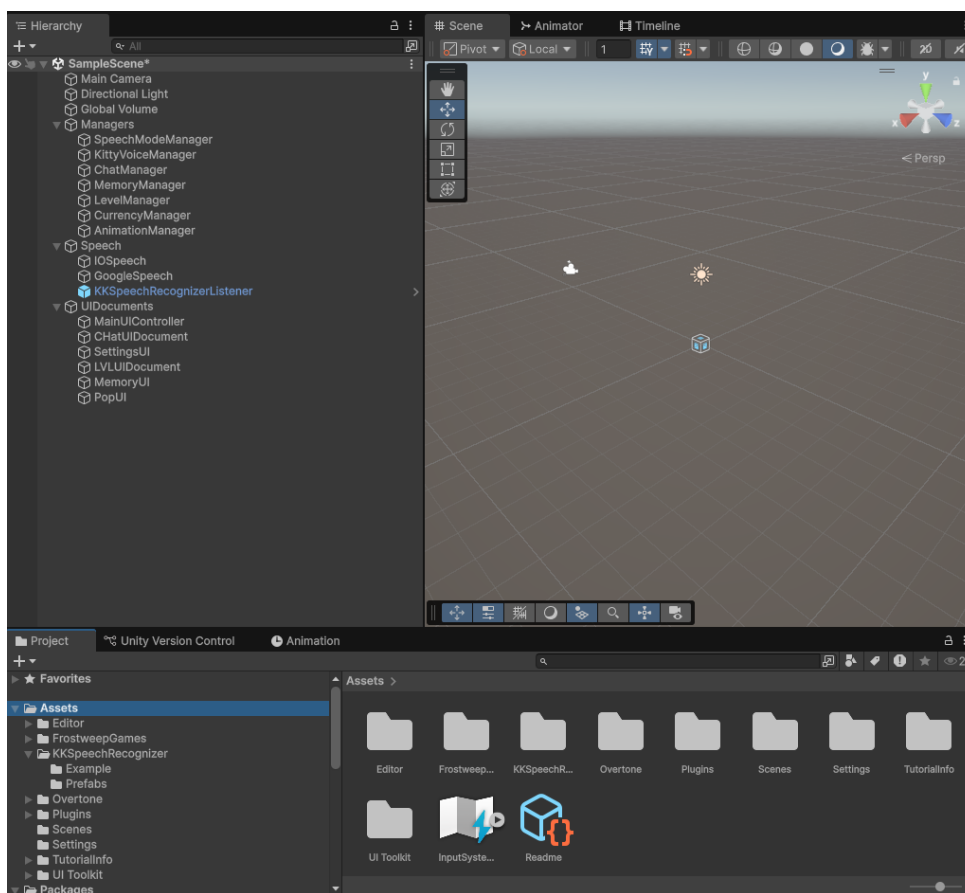
Obrázek 7 Hierarchie Speech

4.4.5 Krok 5: Přidání a struktura uživatelského rozhraní (UI)

Uživatelské rozhraní bylo vytvořeno pomocí systému UI Toolkit. Tento moderní přístup umožňuje oddělení vizuální vrstvy (pomocí .uxml a .uss souborů) od logiky a poskytuje vyšší flexibilitu oproti staršímu systému Canvas.

1. Ve scéně byl vytvořen nový GameObject s názvem UIDocuments.
2. Do něj byly přidány jednotlivé UI controllery jako samostatné GameObjecty:
 - MainUIController – centrální řídicí objekt pro přepínání panelů a obecné události v UI.

- ChatUIDocument – komponenta zodpovědná za zobrazování historie zpráv, chatovací okno a zadávání zpráv.
 - SettingsUI – panel s uživatelským nastavením, včetně výběru AI modelů a TTS/STT systémů.
 - LVLUIDocument – rozhraní pro zobrazování levelovacího systému.
 - MemoryUI – sekce zobrazující historii a přehled předchozích konverzací.
3. Každý z těchto objektů obsahuje komponentu UIDocument, napojenou na konkrétní .uxml soubor a doplněnou o příslušný C# skript (např. ChatUIController.cs), který řídí logiku UI prvků.
4. Panely jsou aktivovány/deaktivovány podle aktuálního stavu aplikace prostřednictvím MainUIController, který funguje jako koordinátor mezi UI a herní logikou (ChatManager, SpeechModeManager apod.).

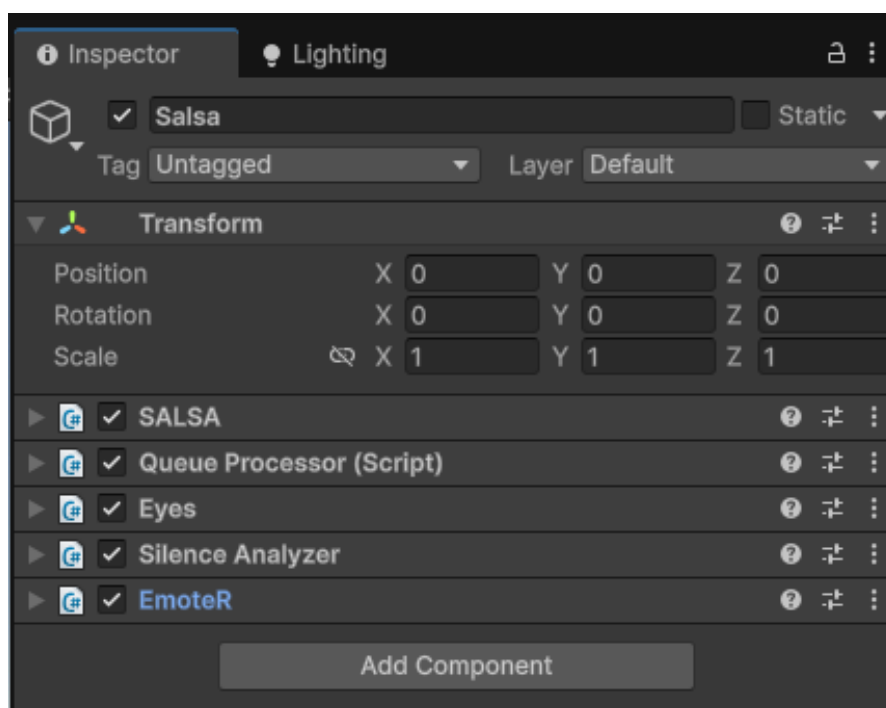


Obrázek 8 Hierarchie objektu UIDocuments

4.4.6 Krok 6: Nastavení lip sync systému pomocí SALSA

Aby mohla virtuální postava reagovat i vizuálně, byl do scény integrován systém pro synchronizaci pohybů úst a očí se zvukovým výstupem. K tomu slouží asset SALSA LipSync Suite

1. Ve scéně byl vytvořen nový GameObject s názvem Salsa.
2. Do tohoto GameObjectu byly přidány následující komponenty ze SALSA assetu:
 - SALSA – hlavní komponenta, která přijímá audio vstup a převádí jej na animace rtů.
 - Queue Processor – obsluhuje zpracování audio fronty a načítání dat.
 - Eyes – řídí směr pohledu a animace očí.
 - Silence Analyzer – rozpoznává pauzy a klidové stavy v řeči.
 - EmoteR – zajišťuje animace emocí obličeje (např. radost, zmatení, smutek).
3. Komponenty byly přidány manuálně v Unity Editoru prostřednictvím nabídky Add Component (viz Obrázek 9).
4. Tento objekt bude v dalších krocích propojen s výstupem z TTS systému (KittyVoiceManager), aby se animace rtů synchronizovaly s přehrávaným zvukem.

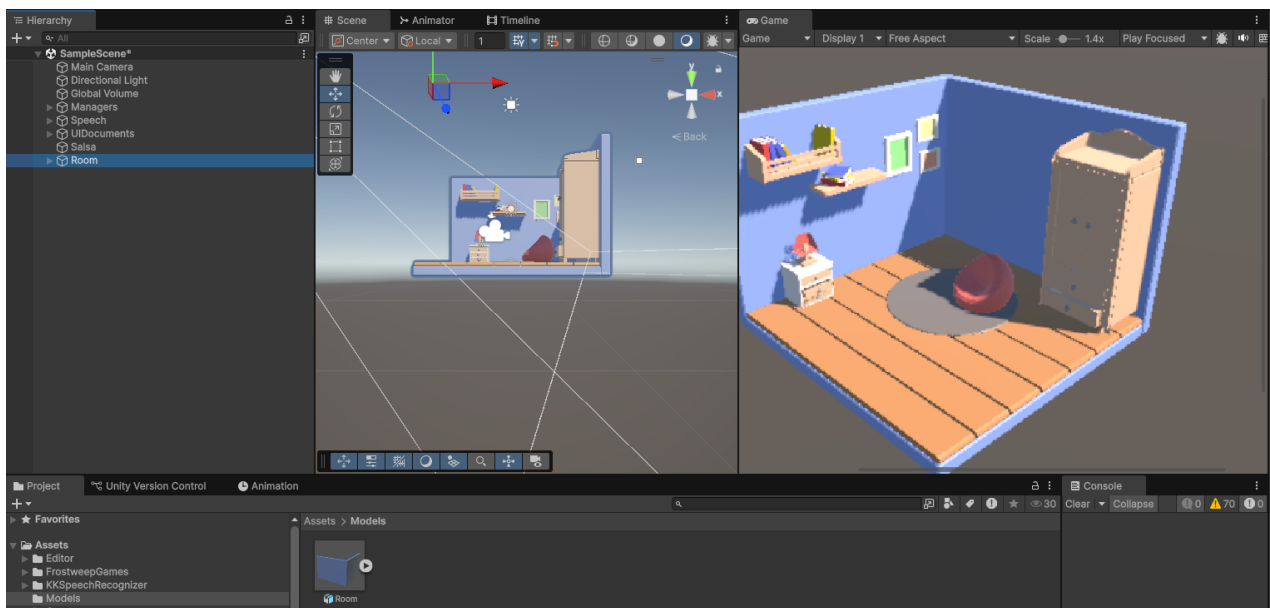


Obrázek 9 Komponenty přidané do GameObjectu Salsa

4.4.7 Krok 7: Import 3D modelu pokoje vytvořeného v Blenderu

Pro vytvoření prostředí, ve kterém se virtuální postava Kitty nachází, byl využit předem připravený 3D model dětského pokoje navržený v aplikaci Blender.

1. Finální model byl z Blenderu exportován ve formátu .fbx.
2. V Unity editoru byla ve složce Assets vytvořena nová podsložka s názvem Models.
3. Do této složky byl jednoduše přetažen .fbx soubor
4. Po importu byl model vložen do scény a správně umístěn:
 - Upravena pozice a měřítko podle potřeby.
 - Ověřena viditelnost a kolizní vlastnosti, pokud jsou potřeba.
 - V případě potřeby byly upraveny materiály a světla, aby scéna působila příjemně a hravě.



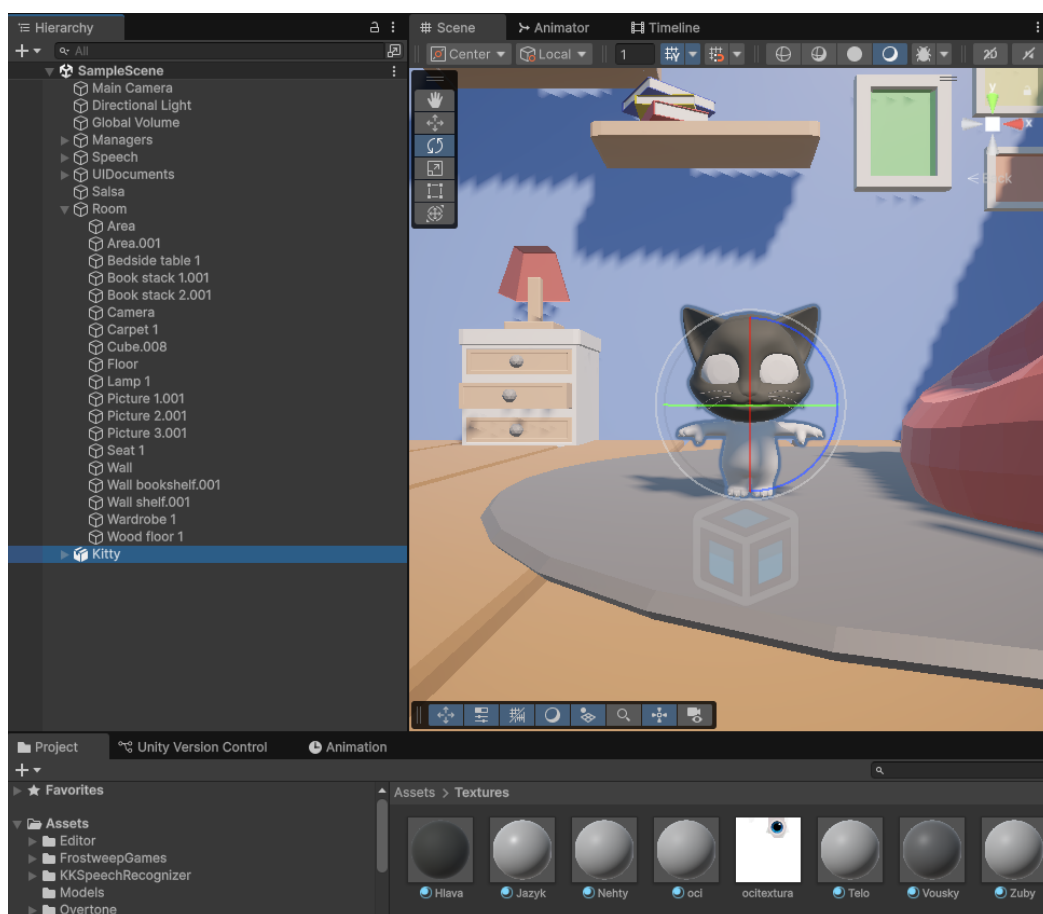
Obrázek 10 Import modelu pokoje

4.4.8 Krok 9: Import 3D modelu postavy Kočky a nastavení Humanoid Rigu

Hlavní interaktivní postavou aplikace je virtuální kočka, která reprezentuje AI kamarádka. Tento model byl vytvořen nebo upraven v externím 3D editoru a následně importován do Unity.

1. Do složky Assets/Models byl přidán exportovaný soubor kočky ve formátu .fbx.
2. Po výběru souboru v Unity byly v Inspectoru provedeny následující kroky:

- Rig záložka:
 - Typ riggingu byl změněn na Humanoid.
 - Kliknutím na Apply se potvrdilo nastavení, což umožňuje použití Humanoid animací.
 - Materials záložka:
 - Materiály byly nastaveny tak, aby se extrahovaly do nové složky:
 - Klik na tlačítko Extract Materials.
 - Vybrána cílová složka Assets/Textures, kam se materiály automaticky uložily.
 -
3. Model byl následně vložen do scény a připraven pro připojení na animační systém SALSA a další chování (TTS, emoce).



Obrázek 11 Importovaný model, nastavení modelu

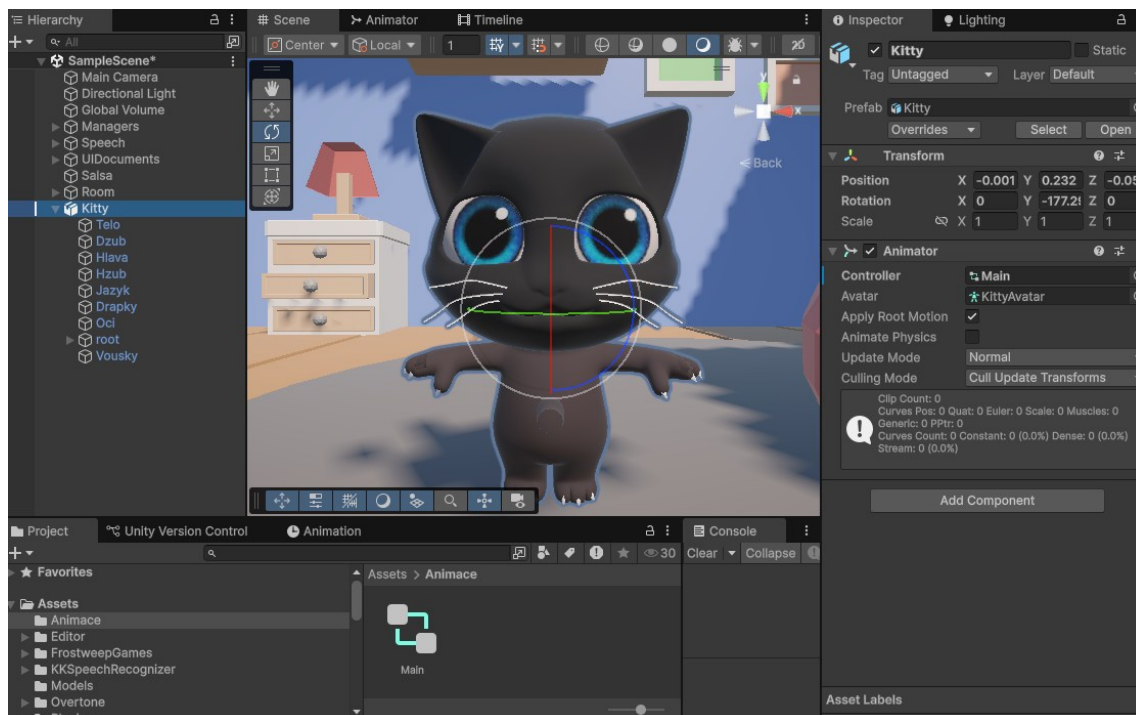
4.4.9 Krok 10: Vytvoření Animator Controlleru a nastavení materiálů pro model kočky

Po úspěšném importu 3D modelu kočky bylo potřeba nastavit animace a vizuální vzhled postavy pro správné fungování ve scéně.

1. V projektu byla vytvořena nová složka Assets/Animace.
2. V této složce byl přes pravé kliknutí vytvořen nový Animator Controller:
 - Create > Animator Controller
 - Pojmenován jako Main
3. Tento Main controller byl následně přetažen do komponenty Animator modelu kočky ve scéně.
4. V Inspectoru modelu kočky bylo zkontrolováno, že:
 - Animator Controller je správně přiřazen.
 - Typ animátoru je nastaven na Humanoid (dle předchozího kroku).
5. Vzhled postavy byl upraven pomocí správně přiřazených materiálů:

Materiály, které byly extrahovány v předchozím kroku, byly přiřazeny do komponenty Skinned Mesh Renderer nebo Mesh Renderer.

Ujistit se, že textury a shader odpovídají požadovanému stylu postavy.



Obrázek 12 Přiřazení Main Animator Controlleru

4.5 Přiřazení skriptů a propojení systémových komponent v inspektoru

4.5.1 Krok 1: Managers

Po vytvoření systémových GameObjectů ve scéně (viz předchozí krok) bylo nutné přiřadit jednotlivé skripty a propojit mezi sebou odpovídající komponenty. Cílem tohoto kroku bylo zajistit, aby klíčové moduly – rozpoznávání řeči, syntéza hlasu, AI interakce, paměť a gamifikace – mohly mezi sebou správně komunikovat.

1. V Unity Editoru byly otevřeny jednotlivé GameObjecty a do jejich komponent byly přidány odpovídající skripty.
2. ChatManager obsahuje:
 - ChatManager.cs – centrální orchestrátor logiky.
 - ChatClientUnity.cs – zajišťuje komunikaci s generativním modelem (ChatGPT, Gemini).
 - FunctionAI.cs – obsluhuje funkční odpovědi AI.
 - V inspektoru byly přiřazeny reference na:
 - IOSSpeech, GoogleSpeech – pro rozpoznávání řeči.

- KittyVoiceManager – pro syntézu hlasu.
- MainUIController, ChatUIController – pro přímé předávání výstupů do UI.
- LevelSystem – kvůli zisku XP za zprávy.

3. KittyVoiceManager obsahuje:

- KittyVoiceManager.cs – hlavní řídicí skript pro práci s TTS.
 - Xi Api Key – API klíč pro službu ElevenLabs
 - Audio Source – reference na vlastní AudioSource komponentu.
 - Overtone Engine / Voice – propojení s offline TTS systémem Piper (skrže plugin Overtone).
 - Chat Manager – reference na ChatManager pro příjem textu k přečtení.
 - Google TTS Manager – reference na Google TTS skript.
 - Azure TTS Manager – reference na Azure TTS skript.
- AudioSource – pro přehrávání syntetizované řeči.
- GoogleTTS.cs – propojený pomocí Google API klíče.
- AzureTTS.cs – obsahuje region a API klíč.
- Tento GameObject přijímá text z ChatManager a přehrává výslednou syntézu hlasu.

4. SpeechModeManager obsahuje:

- SpeechModeManager.cs – řídí přepínání mezi GoogleSpeech.cs a IOSSpeech.cs na základě platformy

5. MemoryManager obsahuje:

- MemoryManager.cs – spravuje krátkodobou paměť konverzace.
- Skripty zde zajišťují správu historie pro AI a případné dotazování na vektory.

6. LevelManager obsahuje:

- LevelSystem.cs – sleduje úroveň uživatele, počet XP za zprávy a barvu progresu.

7. CurrencyManager obsahuje:

CurrencyManager.cs – drží aktuální hodnotu měny, kterou lze využít v minihrách nebo odměnách.

8. AnimationManager obsahuje:

KittyEmoter.cs – komponenta pro vyjádření emocí pomocí animátoru a řízení stavů nudy.

V inspektoru jsou definovány proměnné jako „Bored State Interval“ nebo odkaz na Animator.

9. Ve skriptech byly pomocí SerializeField polí přiřazeny reference na jednotlivé GameObjecty, což umožňuje přímou komunikaci mezi komponentami. Všechny tyto objekty jsou v hierarchii přehledně umístěny pod Managers.

4.5.2 Krok 2: Speech

Po vytvoření stromu Speech bylo nutné propojit jednotlivé komponenty pro rozpoznávání řeči s centrálními částmi systému tak, aby rozpoznáný text mohl být zpracován AI a výstup následně převeden na hlas.

1. GameObject GoogleSpeech obsahuje dvě klíčové komponenty:

- GCSpeechRecognition.cs – komponenta odpovědná za přímou komunikaci s Google Speech-to-Text API.
 - V inspektoru byl zadán platný Google API klíč do pole Api Key.
 - Možnost zapnutí logování chyb je dostupná přes volbu Is Full Debug Log If Error.
- GoogleSpeech.cs – komponenta, která zajišťuje integraci s aplikační logikou.
 - Do pole Chat Manager byla přiřazena reference na objekt ChatManager ze scény.

- Do pole Kitty Voice Manager byla přiřazena reference na objekt KittyVoiceManager, aby po přijetí odpovědi AI mohl být spuštěn výstup syntetizovaného hlasu.
2. GameObject IOSpeech obsahuje skript IOSSpeech.cs se stejnou strukturou:
 - Chat Manager – propojen s hlavním manažerem chatu.
 - Kitty Voice Manager – propojen se systémem syntézy řeči.
 3. Obě tyto komponenty jsou dále řízeny skriptem SpeechModeManager.cs, který vybírá aktivní STT systém podle platformy (např. iOS vs Android).
 4. Po správném nastavení těchto referencí je možné:
 - Automaticky rozpoznat hlas uživatele a převést jej na text.
 - Předat text ChatManageru k odeslání AI modelu.
 - Po obdržení odpovědi spustit syntézu řeči v KittyVoiceManager.

4.5.3 Krok 3: Propojení UI komponent se systémem v inspektoru

Po vytvoření UI struktury ve scéně bylo nutné propojit jednotlivé UI controllery s ostatními systémovými manažery, aby bylo zajištěno správné přepínání panelů, zobrazování dat a interakce mezi uživatelem a AI komponentami.

1. MainUIController:

- Obsahuje UI Document s připojeným souborem MainUI.xml.
- Skript MainUIController.cs slouží jako centrální koordinátor a v inspektoru má přiřazené:
 - SettingsUI, ChatUIDocument, MemoryUI, LVLUIDocument – jednotlivé části UI.
 - SpeechModeManager – pro změnu režimu rozpoznávání řeči.
 - LevelManager, MemoryManager – pro čtení/zápis dat o pokroku a historii.

2. ChatUIDocument:

- Obsahuje UI Document s rozhraním pro konverzaci.

- Skript ChatUIController.cs komunikuje s MainUIController a zobrazuje zprávy od uživatele i AI.
3. SettingsUI:
 - Obsahuje SettingsUI.xml a SettingsUIController.cs.
 - Propojeno se SpeechModeManager, což umožňuje přepínání STT systémů v rámci nastavení.
 4. LVLUIDocument:
 - Zobrazovací rozhraní pro levelovací systém.
 - LevelDetailsUIController.cs načítá aktuální úroveň, XP a zobrazuje průběh.
 5. MemoryUI:
 - Obsahuje přehled historie konverzací a přístup k paměti.
 - MemoryUI.cs napojen na MemoryManager.
 6. PopUI:
 - Slouží k zobrazování vyskakovacích hlášek nebo oznámení.
 - Skript Popup.cs napojen na Popup.xml a aktivován ze systému dle potřeby.
 7. Každý z těchto controllerů má ve svém skriptu SerializeField reference na potřebné komponenty, což umožňuje vzájemnou komunikaci mezi UI a zbytkem aplikace bez nutnosti hledání objektů za běhu.

4.6 Nastavení SALSA

Pro zajištění realistické synchronizace pohybu rtů s mluveným výstupem a výrazů obličeje byl do aplikace integrován systém SALSA LipSync Suite. Ten využívá audio výstup ze syntézy řeči (TTS) k řízení blendshape animací modelu a zobrazení emocí.

4.6.1 Krok 1 Nastavení komponenty SALSA

1. V komponentě SALSA byly nastaveny reference na:
 - Audio Source: přiřazen komponentě KittyVoiceManager (Audio Source) – výstup ze systému TTS.
 - Queue Processor: přiřazen interní objekt Salsa (Queue Processor).

- Emoter: přiřazen interní objekt Salsa (Emoter).
2. V sekci Viseme Configuration bylo přidáno 8 visemů (viseme 1 až viseme 8), každý s vlastním trigger (např. od 0.01 do 0.08) a komponentami řídícími blendshapes pro fonémy.

4.6.2 Krok 2: Nastavení Emoter

1. V komponentě Emoter byla nastavena reference na Queue Processor.
2. V sekci Emote Configuration byl přidán výraz Smile obsahující více komponent, které ovládají blendshapes obličeje.
3. Pole Lipsync Emphasis Chance bylo nastaveno na 1, aby se během mluvení spouštěly přirozené mimické výrazy.

4.6.3 Krok 4: Nastavení očí a mrkání pomocí Eyes

Pro dosažení přirozeného pohybu očí a mrkání byly použity blendshapes modelu:

1. Komponenta Eyes byla propojena s Queue Processor ze SALSA.
2. V rámci konfigurace pohledu byly přiřazeny následující blendshapes:
 - LookDown, LookUp, LookLeft, LookRight byly nastaveny pro pohyb očí
3. Pro simulaci mrkání byly použity blendshapes:
 - EyeBlinkLeft a EyeBlinkRight, které byly přiřazeny ke komponentám hlavy

4.6.4 Krok 5: Propojení s hlasovým výstupem

1. Komponenta SALSA analyzuje výstup z AudioSource připojeného na KittyVoiceManager.
2. Získaný signál je převeden na pohyby rtů pomocí visemů a zároveň aktivuje komponenty Emoter a Eyes.
3. Výsledkem je věrohodná animace mluvení i výrazu obličeje.

4.7 Nastavení Animator Controlleru a import animací

Po importu modelu a přípravě základních materiálů byl v prostředí Unity vytvořen Animator Controller s názvem „Main“. Tento controller umožňuje řídit přechody mezi jednotlivými animacemi postavy v závislosti na vstupních parametrech a aktuálním stavu AI subsystému.

4.7.1 Hlavní kroky:

1. Otevřením souboru „Main“ v okně Animator se zobrazí přehled všech stavů a jejich přechodů.
2. Do tohoto controlleru byly přidány jednotlivé stavy odpovídající animacím, například: IDLE, dance, happy, Lay down, HeadShake, SwingArms, ArmsFolded, KickGround, SlumpBack, FootTap a další.
3. Přechody mezi animacemi jsou řízeny parametry, jako jsou například isOffline, isBored, hands apod., jejichž hodnoty jsou nastavovány skripty podle aktuálního stavu aplikace.
4. Každý stav obsahuje přiřazenou animaci, která byla importována z platforem Mixamo [38], Rokoko [33], případně vytvořena ručně v prostředí Blender.
5. Kompletní vizualizace stromu animací a přechodů je uvedena v Příloze P III.

5 TESTOVÁNÍ APLIKACE

Cílem testování bylo ověřit funkčnost a stabilitu vyvinuté aplikace s důrazem na interakci uživatele, odezvu systému, hlasové vstupy a výstupy a správné chování uživatelského rozhraní. Testování bylo provedeno ve dvou základních rovinách – formou manuálního testování v editoru Unity i na cílových mobilních zařízeních a dále prostřednictvím automatických jednotkových testů (unit testů).

Testování se zaměřilo nejen na jednotlivé komponenty systému (např. správu paměti, vstupy a výstupy, animace), ale i na jejich vzájemnou spolupráci v reálných scénářích. Získané poznatky byly využity pro návrh vylepšení a identifikaci oblastí s potenciálem pro budoucí rozšíření.

5.1 Manuální testování

Funkčnost aplikace byla ověřena manuálním testováním, které se zaměřilo na hlavní oblasti interakce – uživatelské rozhraní, funkčnost 3D modelu a chování hlasových vstupů a výstupů. Testování probíhalo v prostředí editoru Unity, na cílové platformě iOS a v několika případech na Androidu. Hodnoceny byly jak vizuální, tak funkční aspekty chování aplikace z pohledu běžného uživatele.

- Uživatelské rozhraní bylo shledáno jako stabilní a plně funkční. Prvky jako tlačítka, vstupní pole, ikonky a panely správně reagovaly na vstupy uživatele. Přepínání mezi sekcemi (chat, nastavení, paměť) probíhalo plynule a bez chyb.
- 3D model postavy (virtuální kočky) se načítá správně a bez viditelných grafických problémů. Reaguje na události a změny stavu (například spouštění animací, změny výrazu) a zobrazuje se korektně v rámci vytvořeného prostředí.
- Synchronizace rtů (lip sync) prostřednictvím systému SALSA funguje přesně a včasně reaguje na výstup z TTS. Výrazy obličeje jsou při mluvení aktivovány správně, bez výrazného zpoždění nebo chybné detekce pauz.
- Text-to-speech (TTS) výstup byl testován ve více variantách:
 - Piper/Overtone – poskytuje základní offline řešení s dostatečnou srozumitelností, avšak výstup působí monotónně a bez přirozené intonace.
 - Google TTS a Azure TTS – v angličtině velmi kvalitní, pro češtinu však znějí nepřirozeně nebo mechanicky. Použití je proto omezené jazykem.

- ElevenLabs TTS – poskytuje velmi kvalitní výstup i v češtině, s přirozenou intonací a dynamikou projevu. Nevýhodou je však jeho vyšší nákladovost a závislost na internetovém připojení.
- Speech-to-text (STT) moduly:
 - Google STT – rozpoznávání funguje správně, ale s menším zpožděním, které může narušovat plynulost konverzace.
 - Apple Speech Recognition (iOS) – rozpoznávání řeči funguje velmi dobře, s nízkou latencí a vyšší přesností, zejména u češtiny. Je vhodnější pro nasazení na cílové platformě.
- Animace modelu fungují v základní podobě, ale v některých situacích působí strnule nebo málo dynamicky. Bylo by vhodné provést další ladění, zejména u přechodů emocí a gest.

Celkově bylo manuálním testováním potvrzeno, že aplikace je funkční a připravená k dalšímu vývoji. Během testování nebyly identifikovány kritické chyby bránící používání aplikace.

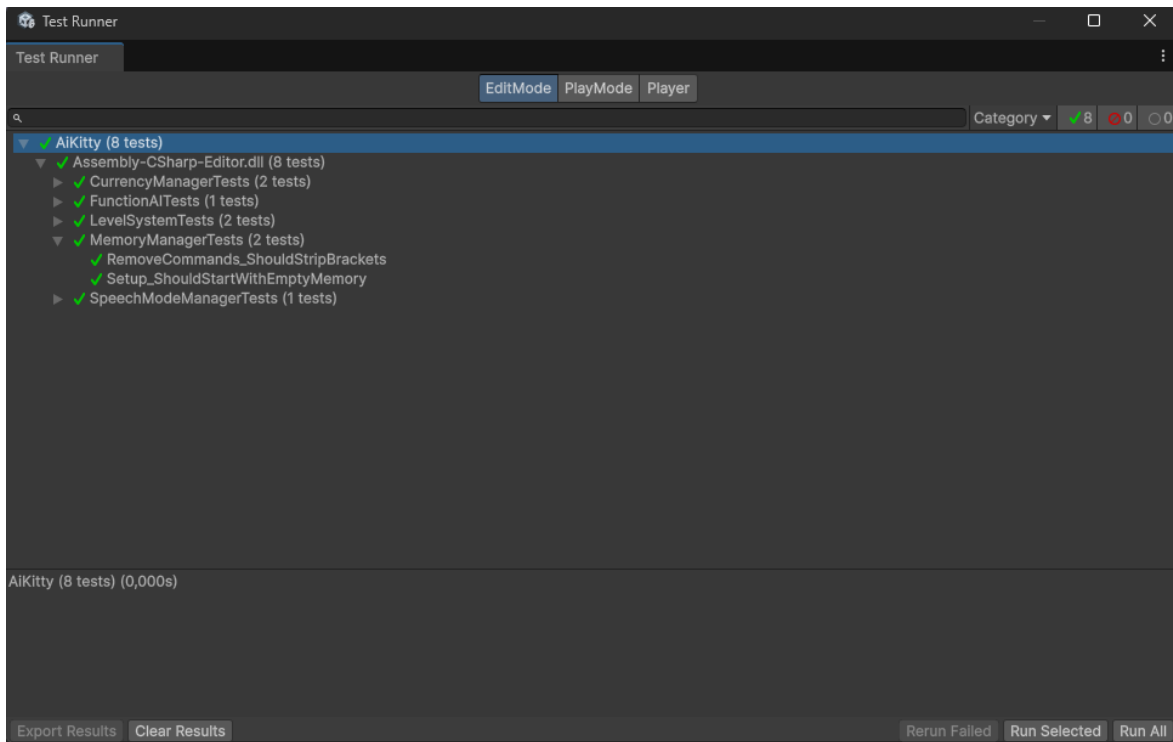
5.2 Unit testy

K ověření funkčnosti jednotlivých částí systému byly vytvořeny jednotkové testy pomocí vestavěného nástroje Unity Test Framework. Tyto testy byly navrženy pro běh v režimu EditMode, což umožňuje testovat logiku skriptů bez nutnosti spouštět celou scénu.

Testování zahrnovalo vybrané logické komponenty systému:

- CurrencyManagerTests – ověřuje správu herní měny, její přičítání a odečítání.
- FunctionAITests – testuje strukturu a obsah výstupu funkce generující AI odpověď.
- LevelSystemTests – ověřuje správné navyšování úrovně na základě získaných bodů.
- MemoryManagerTests – kontroluje práci s pamětí, ukládání, načítání a filtrování textu.
- SpeechModeManagerTests – testuje výběr aktivního rozpoznávače řeči podle platformy.

Všechny testy proběhly úspěšně bez chyb, což potvrzuje správné chování klíčových částí systému v oddělených scénářích.



Obrázek 13 Unit test

6 ZHODNOCENÍ PROVEDENÉHO ŘEŠENÍ A NÁVRH VYLEPŠENÍ

6.1 Zhodnocení provedeného řešení

Výsledná aplikace naplnila hlavní cíle stanovené v úvodu práce. Byl navržen a implementován funkční prototyp virtuálního asistenta, který umožňuje interakci s uživatelem prostřednictvím hlasu a textu. Projekt kombinuje více technologických vrstev – od rozpoznávání řeči a generování odpovědi pomocí AI modelů, přes syntézu hlasu až po zobrazení výstupů pomocí 3D postavy v herním prostředí Unity.;++

Z hlediska architektury byla zvolena komponentově orientovaná struktura, která umožňuje snadnou údržbu, testování a případné rozšiřování o další funkce. Díky modulárnímu přístupu lze jednoduše nahradit nebo přepínat mezi různými poskytovateli TTS a STT služeb, stejně jako mezi AI modely.

Testování prokázalo funkčnost klíčových subsystémů v reálných scénářích. Uživatelské rozhraní je stabilní a umožňuje intuitivní interakci. Výstupy z AI modelů byly relevantní a plynulé. Systém TTS funguje s různými enginy a výstupy byly ve většině případů kvalitní. Rozpoznávání hlasu bylo na platformě iOS spolehlivé a s nízkou latencí, zatímco cloudové STT služby vykazovaly mírné zpoždění.

6.2 Návrh vylepšení

- Vylepšení hlasového výstupu: Výstupy z lokálního systému Piper TTS působí monotónně. Doporučuje se trénink vlastního hlasového modelu s větší dynamikou a přirozenou intonací.
- Vylepšení animací a výrazů postavy: Animace 3D modelu kočky působí v některých situacích strnule. Zavedení přechodových animací a větší palety výrazů by zvýšilo autenticitu projevu postavy.
- Zkvalitnění 3D modelu a prostředí: Model virtuálního asistenta by mohl být vizuálně detailnější a lépe riggovaný. Také prostředí pokoje může být obohaceno o interaktivní prvky, lepší nasvícení a detaily, které zvyšují pohlčení uživatele.
- Vylepšení uživatelského rozhraní: Grafické rozhraní by mohlo být více přizpůsobeno cílové skupině (dětem). Doporučuje se použít hravější design, výraznější barvy, animované prvky a přehlednější ovládací prvky.

- Zvýšení interaktivity: V současné verzi je interakce omezena na komunikaci s AI. Do budoucna je vhodné přidat další typy interakcí s 3D modelem (např. pohazení, společné hraní, změna výrazu postavy na základě emocí).
- Zavedení miniher: Doplnění jednoduchých miniher (např. hádanky, poznávací kvízy, skládání obrázků) může prodloužit dobu strávenou v aplikaci a zvýšit míru zaujetí.
- Vylepšení paměťového systému: Aplikace by mohla uchovávat dlouhodobější informace o uživateli, sumarizovat historii nebo nabízet personalizované reakce na základě předchozí komunikace.
- Zajištění offline použitelnosti: Lokální verze AI a hlasových modulů by snížila závislost na internetovém připojení a rozšířila použitelnost v různých situacích.

Celkově lze řešení hodnotit jako funkční základ s dobře definovanou architekturou, který představuje silný výchozí bod pro budoucí vývoj. Výsledný prototyp demonstruje možnosti hlubší integrace umělé inteligence do interaktivních herních aplikací a poskytuje konkrétní směry pro další zlepšení funkcionality i uživatelského zážitku.

ZÁVĚR

Tato bakalářská práce se zabývá návrhem a vývojem aplikace virtuálního asistenta, který využívá umělou inteligenci k interaktivní komunikaci s uživatelem prostřednictvím textu a hlasu. Cílem bylo vytvořit funkční systém propojující generativní jazykové modely, syntézu řeči (TTS), rozpoznávání hlasu (STT) a vizuální prezentaci v 3D prostředí herního engine Unity.

V teoretické části práce byly analyzovány současné trendy v oblasti umělé inteligence ve hrách, se zaměřením na generativní modely, technologie převodu textu na řeč, rozpoznávání řeči, 3D modelování v Blenderu a možnosti databázového uložení interakcí pomocí Firestore. Dále byly popsány vybrané cloudové a lokální služby (např. ChatGPT, Gemini, ElevenLabs, Piper TTS, Google Cloud, Azure) a vývojové nástroje využité při tvorbě aplikace.

Praktická část práce se zaměřila na návrh architektury aplikace, implementaci hlavních funkčních celků a realizaci stylizovaného 3D prostředí s postavou virtuálního kamaráda. Byl vytvořen systém pro zadávání hlasových a textových vstupů, generování odpovědí pomocí AI modelů a přehrávání syntetizovaného výstupu s doprovodnou animací rtů a emocí postavy.

Aplikace byla úspěšně otestována z hlediska funkčnosti, použitelnosti i výkonu na cílových mobilních platformách. Výsledky ukazují, že kombinací běžně dostupných nástrojů lze vytvořit smysluplný a plynulý systém pro konverzaci v reálném čase s vysokou mírou vizuální přívětivosti.

Přestože byl vytvořen funkční prototyp, byly identifikovány oblasti pro další vylepšení – zejména rozšíření animací a výrazů postavy, integrace miniher, vylepšení paměťového systému a podpora offline režimu. Tyto směry mohou být předmětem dalšího výzkumu či komerčního rozvoje.

Tato práce tak představuje konkrétní příklad hlubší integrace generativní umělé inteligence do herního vývoje a přispívá k rozvoji inovativních interaktivních systémů s využitím AI.

SEZNAM POUŽITÉ LITERATURY

- [1] BUCKLAND, Michael. AI for Games. 3. vyd. Boca Raton: CRC Press, 2017. ISBN 978-1-138-48397-2.
- [2] YANNAKAKIS, Georgios N.; TOGELIUS, Julian. Artificial Intelligence and Games. Springer, 2018. ISBN 978-3-319-63519-4.
- [3] BROWN, Tom B., et al. Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165, 2020. Dostupné z: <https://arxiv.org/abs/2005.14165>. [cit. 2025-04-18].
- [4] INWORLD AI. Inworld AI: Powering the next generation of virtual beings. Online. Dostupné z: <https://inworld.ai>. [cit. 2025-04-18].
- [5] NVIDIA DEVELOPER. Generative AI for Digital Human Technologies and New AI-powered NVIDIA RTX Lighting. NVIDIA Developer Blog. 19. března 2024. Online. Dostupné z: <https://developer.nvidia.com/blog/generative-ai-digital-human-technologies-rtx-lighting>. [cit. 2025-04-18].
- [6] LINARDATOS, Konstantinos, et al. Ethics of AI in gaming: A review. Technology in Society, 67, 101716, 2021. DOI: 10.1016/j.techsoc.2021.101716.
- [7] DABLANDER, Markus. Future Research Avenues for Artificial Intelligence in Digital Gaming: An Exploratory Report. arXiv preprint arXiv:2412.14085, 2024. Dostupné z: <https://arxiv.org/abs/2412.14085>. [cit. 2025-04-18].
- [8] UNITY TECHNOLOGIES. Unity Documentation: Comprehensive resources for getting started, tutorials, and reference guides. Online. Dostupné z: <https://docs.unity.com/>. [cit. 2024-10-17].
- [9] MICROSOFT. Nástroje Unity pro vývoj her | Visual Studio. Online. Dostupné z: <https://visualstudio.microsoft.com/cs/vs/unity-tools/>. [cit. 2025-02-24].
- [10] UNITY TECHNOLOGIES. Unity AI: Create Incredible Experiences with Real-Time 3D. Online. Dostupné z: <https://unity.com/ai>. [cit. 2025-02-24].
- [11] MICROSOFT. Programovací jazyky .NET. Online. Dostupné z: <https://dotnet.microsoft.com/en-us/languages>. [cit. 2025-02-24].
- [12] UNITY TECHNOLOGIES. Visual Studio C# Integration. Online. Dostupné z: <https://docs.unity3d.com/Manual/VisualStudioIntegration.html>. [cit. 2025-02-24].

- [13] MICROSOFT. C# Language Documentation. Online. Dostupné z: <https://learn.microsoft.com/en-us/dotnet/csharp/>. [cit. 2025-02-24].
- [14] UNITY TECHNOLOGIES. .NET profile support. Online. Dostupné z: <https://docs.unity3d.com/6000.0/Documentation/Manual/dotnet-profile-support.html>. [cit. 2025-02-24].
- [15] MUSIOL, Martin. *Generative AI: Navigating the Course to the Artificial General Intelligence Future*. Hoboken, New Jersey: John Wiley & Sons, 2024. ISBN 978-1-394-20591-2.
- [16] GOOGLE. *Gemini AI: The Next Generation of Conversational AI*. Online. Dostupné z: <https://deepmind.google/gemini/>. [cit. 2025-02-24].
- [17] OPENAI. OpenAI introduces ChatGPT-4o Mini: A faster and cost-effective AI model. Online. Dostupné z: <https://www.theverge.com/news/603849/openai-o3-mini-launch-chatgpt-api-available-now>. [cit. 2025-03-11].
- [18] GOOGLE. Google launches Gemini 2: A multimodal AI assistant. Online. Dostupné z: <https://www.wired.com/story/google-gemini-2-ai-assistant-release/>. [cit. 2025-03-11].
- [19] DUTOIT, Thierry. *An Introduction to Text-to-Speech Synthesis*. Springer, 1997. ISBN 978-0-7923-4498-1.
- [20] BLACK, Alan W.; TAYLOR, Paul; CAHILL, Lynne. Festival Speech Synthesis System. Carnegie Mellon University, 1999. Online. Dostupné z: <https://www.cstr.ed.ac.uk/projects/festival/>. [cit. 2025-03-11].
- [21] ELEVENLABS. *ElevenLabs Speech Synthesis API*. Online. Dostupné z: <https://elevenlabs.io/>. [cit. 2025-03-11].
- [22] GITHUB. *Piper TTS: Lightweight and Fast Text-to-Speech*. Online. Dostupné z: <https://github.com/rhasspy/piper>. [cit. 2025-03-11].
- [23] GOOGLE. *Cloud Speech-to-Text API Documentation*. Online. Dostupné z: <https://cloud.google.com/speech-to-text>. [cit. 2025-03-11].
- [24] APPLE. *Speech Framework Documentation*. Online. Dostupné z: <https://developer.apple.com/documentation/speech>. [cit. 2025-03-11].
- [25] BLENDER FOUNDATION. *Blender: The Free and Open 3D Creation Suite*. Online. Dostupné z: <https://www.blender.org/>. [cit. 2025-03-11].

- [26] KERLOW, Isaac Victor. *The Art of 3D Computer Animation and Effects*. Hoboken, New Jersey: John Wiley & Sons, 2012. ISBN 978-1-118-29486-3.
- [27] VAN DEN HERIK, J. H.; WINAND, M.; SPELLER, M. *Inverse Kinematics and Its Application in Character Animation*. Journal of Graphics and Applications, 2021. DOI: 10.1109/TGAA.2021.3069534.
- [28] OSIPA, Jason. *Stop Staring: Facial Modeling and Animation Done Right*. Indianapolis: Wiley, 2010. ISBN 978-0-470-58484-1.
- [29] GOOGLE. Firebase Documentation. Online. Dostupné z: <https://firebase.google.com/docs>. [cit. 2025-03-30].
- [30] GOOGLE. Google Colaboratory. Online. Dostupné z: <https://colab.research.google.com>. [cit. 2025-04-18]
- [31] GOOGLE CLOUD. Text-to-Speech: Voice types and voices. Google Cloud Documentation. Online. Dostupné z: <https://cloud.google.com/text-to-speech/docs/voices>. [cit. 2025-04-18].
- [32] MICROSOFT AZURE. Text-to-speech overview. Azure AI services Documentation. Online. Dostupné z: <https://learn.microsoft.com/en-us/azure/ai-services/speech-service/text-to-speech>. [cit. 2025-04-18].
- [33] ROKOKO. Rokoko Vision. Online. Dostupné z: <https://www.rokoko.com/products/rokoko-vision>. [cit. 2025-04-18].
- [34] LEASTSQUARES LLC. Overtone – Realistic AI Offline Text-to-Speech (TTS). Online. Unity Asset Store, verze 1.5.2, 9. 10. 2023. Dostupné z: <https://assetstore.unity.com/packages/tools/generative-ai/overtone-realistic-ai-offline-text-to-speech-tts-251304>. [cit. 2025-05-26].
- [35] CRAZY MINNOW STUDIO, LLC. SALSA LipSync Suite. Online. Unity Asset Store, verze 2.5.6, 26. 1. 2025. Dostupné z: <https://assetstore.unity.com/packages/tools/animation/salsa-lipsync-suite-148442>. [cit. 2025-05-26].
- [36] FROSTWEEP GAMES. Speech Recognition using Google Cloud [VR / AR / Mobile / Desktop] Pro. Online. Unity Asset Store, verze 5.0, 2. 2. 2025. Dostupné z: <https://assetstore.unity.com/packages/tools/ai-ml-integration/speech-recognition-using-google-cloud-vr-ar-mobile-desktop-pro-72625>. [cit. 2025-05-26].

- [37] KENYS. Mobile Speech Recognizer. Online. Unity Asset Store, verze 1.7.9, 8. 6. 2022.
Dostupné z: <https://assetstore.unity.com/packages/tools/audio/mobile-speech-recognizer-73036>. [cit. 2025-05-26].
- [38] ADOBE. Mixamo. Online. Dostupné z: <https://www.mixamo.com/>. [cit. 2025-05-26].

SEZNAM OBRÁZKŮ

Obrázek 1 Diagram Firestore databáze.....	35
Obrázek 2 Návrh stylizovaného pokoje vytvořeného v Blenderu.	40
Obrázek 3 Model kočky v Blenderu	41
Obrázek 4 Výběr šablony 3D (Mobile)	42
Obrázek 5 Import zakoupených assetů	43
Obrázek 6 Hierarchie objektu Managers	44
Obrázek 7 Hierarchie Speech	45
Obrázek 8 Hierarchie objektu UIDocuments.....	46
Obrázek 9 Komponenty přidáné do GameObjectu Salsa	47
Obrázek 10 Import modelu pokoje	48
Obrázek 11 Importovaný model, nastavení modelu	49
Obrázek 12 Přiřazení Main Animator Controlleru	51
Obrázek 13 Unit test	60

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AI	Umělá inteligence (Artificial Intelligence)
API	Aplikační programové rozhraní (Application Programming Interface)
C#	Programovací jazyk C#
GAN	Generativní adversariální síť (Generative Adversarial Network)
VAE	Variational Autoencoder
TTS	Text-to-Speech (převod textu na řeč)
SDK	Sada pro vývoj software (Software Development Kit)
NPC	Nehratelné postavy (Non-Playable Characters)
Unity	Herní engine pro vývoj her a aplikací
GPT	Generativní model umělé inteligence (Generative Pre-trained Transformer)
CPU	Centrální procesorová jednotka (Central Processing Unit)
FSM	Konečný stavový automat (Finite State Machine)
GPU	Grafická procesorová jednotka (Graphics Processing Unit)
GUI	Grafické uživatelské rozhraní (Graphical User Interface)
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrované vývojové prostředí (Integrated Development Environment)
IK	Inverzní kinematika (Inverse Kinematics)
iOS	Mobilní operační systém společnosti Apple
JSON	JavaScript Object Notation
LLM	Velký jazykový model (Large Language Model)
macOS	Operační systém společnosti Apple pro počítače Mac
.NET	Softwarová platforma společnosti Microsoft
ONNX	Open Neural Network Exchange
PCG	Procedurální generování obsahu (Procedural Content Generation)
RAG	Retrieval-Augmented Generation

- SSM Speech Synthesis Markup Language
- STT Převod řeči na text (Speech-to-Text)
- TPU Tensorová procesorová jednotka (Tensor Processing Unit)
- tvOS Operační systém společnosti Apple pro Apple TV
- UV Souřadnice pro mapování textur
- watchOS Operační systém společnosti Apple pro hodinky Apple Watch
- SALSA Synchronized Animation Layer System for Audio – systém pro lip sync a emoce
- KKSpeech Plugin pro rozpoznávání řeči na iOS
- ARKit Apple AR Kit – platforma pro rozšířenou realitu a rozpoznávání obličejů
- SSML Speech Synthesis Markup Language – značkovací jazyk pro syntézu řeči
- UI Toolkit Systém pro tvorbu uživatelského rozhraní v Unity
- URP Universal Render Pipeline – renderovací pipeline v Unity
- LoRA Low-Rank Adaptation – metoda pro efektivní doladění velkých jazykových modelů
- VITS Variational Inference Text-to-Speech – neuronová architektura pro syntézu řeči
- VRM Voice Response Model – model určený pro generování mluvených odpovědí
- .cs C# Script – soubor obsahující zdrojový kód ve formátu C# pro použití v Unity
- .fbx File Exchange – formát pro výměnu 3D modelů mezi 3D softwary a herními enginy
- .uxml Unity XML – soubor definující strukturu uživatelského rozhraní v UI Toolkit
- .uss Unity Style Sheet – stylopis (podobný CSS), určující vzhled UI prvků v Unity UI Toolkit

SEZNAM PŘÍLOH

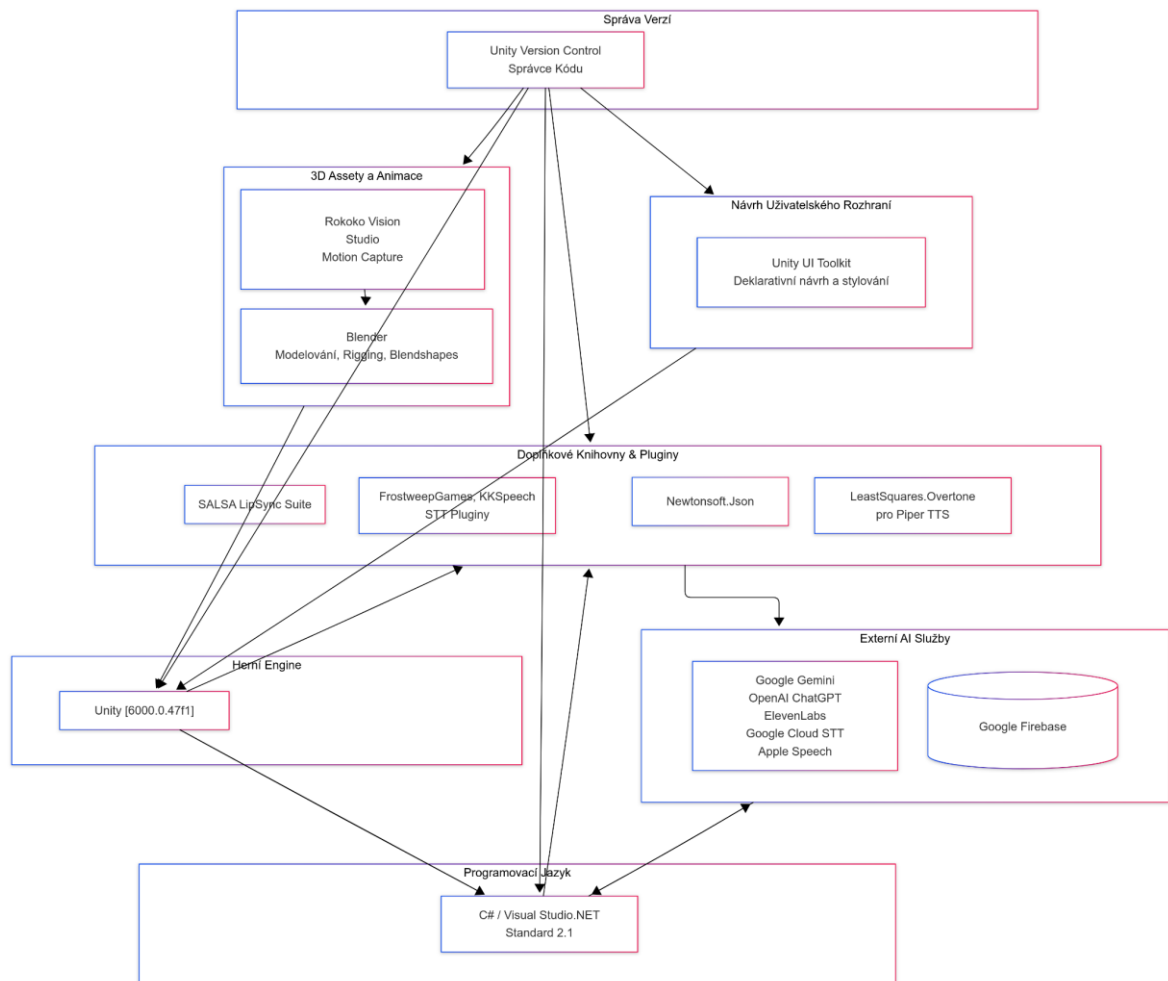
Příloha P I: Diagram vývojového prostředí a nástrojů

Příloha P II: Diagram Blokové schéma softwarové architektury aplikace

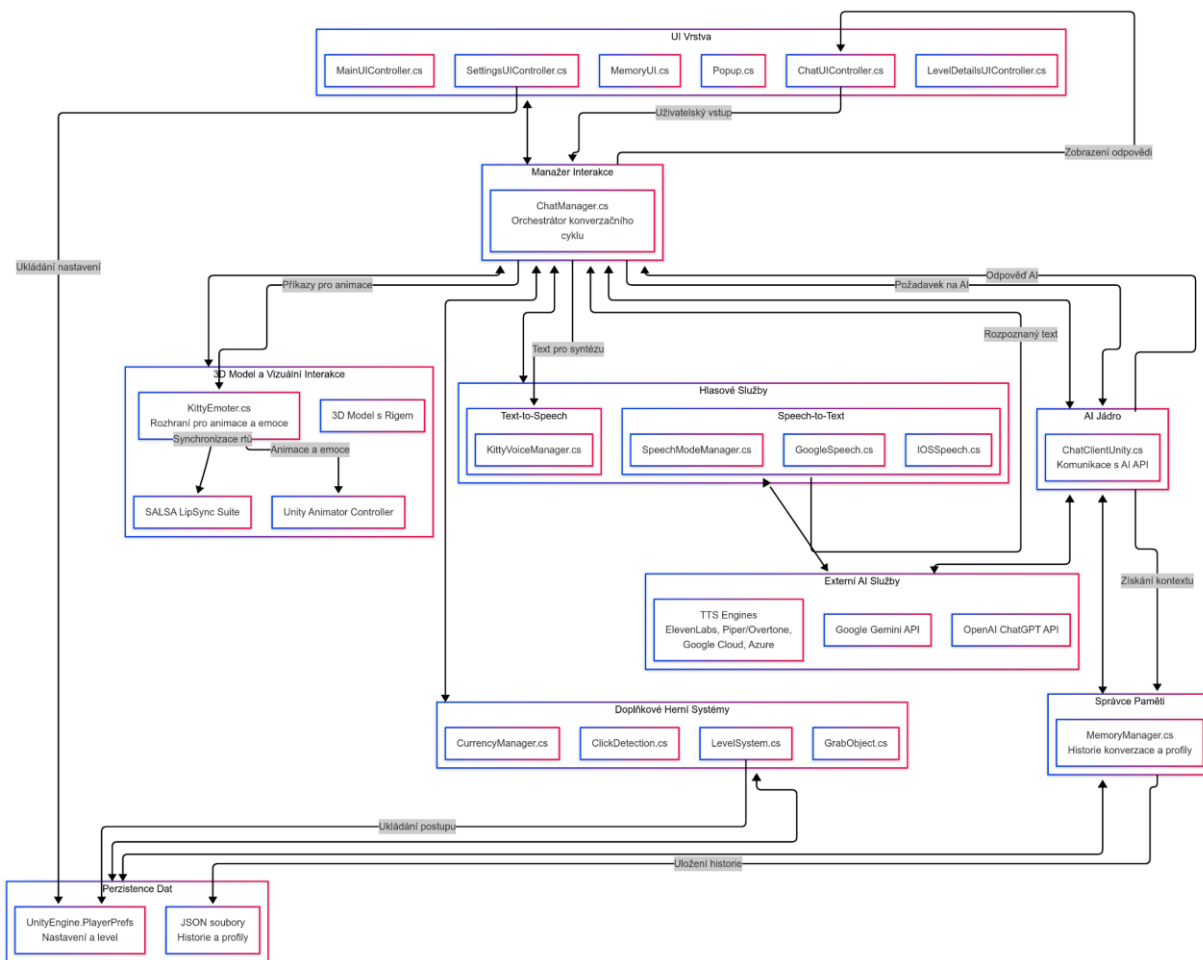
Příloha P III Schéma Animator Controlleru

Příloha P IV: Unity projekt ve formátu .zip

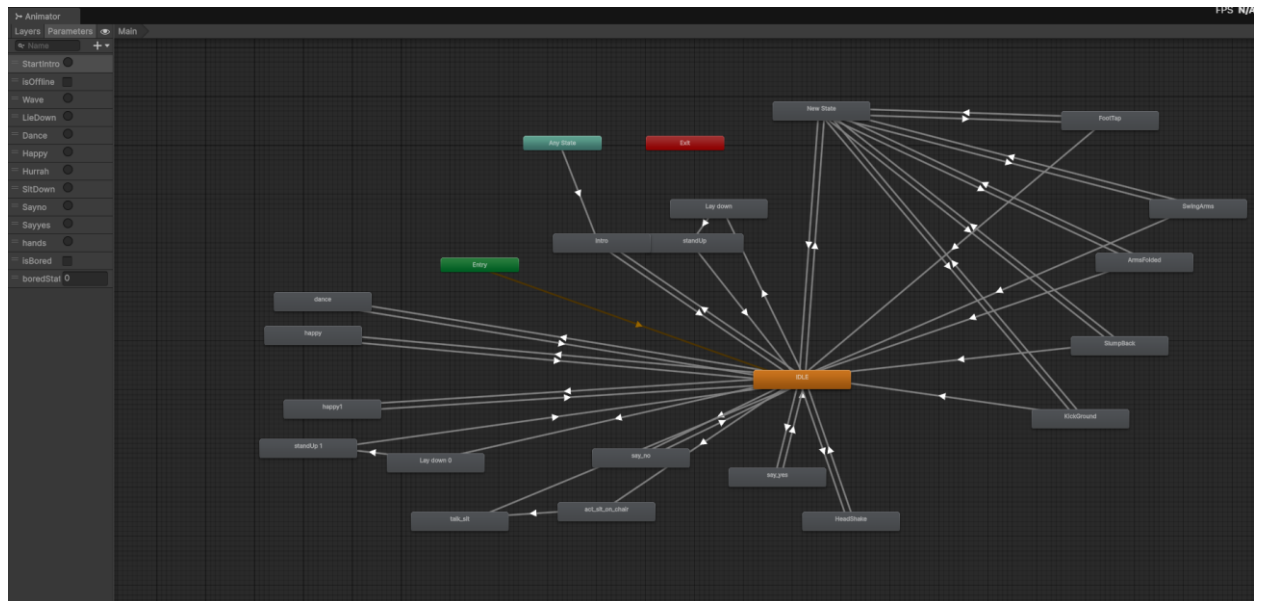
PŘÍLOHA P I: DIAGRAM VÝVOJOVÉHO PROSTŘEDÍ A NÁSTROJŮ



PŘÍLOHA P II: DIAGRAM BLOKOVÉ SCHÉMA SOFTWAREOVÉ ARCHITEKTURY APLIKACE



PŘÍLOHA P III: SCHÉMA ANIMATOR CONTROLLERU



PŘÍLOHA P IV: UNITY PROJEKT VE FORMÁTU .ZIP

Soubor s názvem „Unity.zip“ obsahuje celý projekt spustitelný v enginu Unity je elektronickou přílohou této práce.