UNIVERZITA PALACKÉHO OLOMOUC

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

Pavel Ohryzek

Vývoj všeobecně výukové aplikace pro zrakově postižené

Olomouc 2024

Vedoucí práce: Šaloun Petr, doc. RNDr. Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Vývoj všeobecně výukové aplikace pro zrakově postižené" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

V Novém Hrozenkově dne 18. 6. 2024 ……...

Pavel Ohryzek

Poděkování

Rád bych poděkoval svému vedoucímu panu docentovi Petrovi Šalounovi za ochotu ujmout se takto neobvyklé a specifické práce a za jeho upřímnou, avšak dobře míněnou kritiku, kterou poskytl.

Zároveň bych chtěl poděkovat kolegům z jiných oborů, jejichž názory byly pro práci přínosné, konkrétně Kristýně Vondrákové, která poskytla cenný náhled do vzdělávání osob se zrakovým postižením a byla ochotným prostředníkem pro komunikaci mezi mnou a pracovníky či experty ve speciálně pedagogickém oboru, pokud to v dané chvíli pro mne nebylo možné.

Anotace

Jméno a příjmení:	Pavel Ohryzek
Katedra:	Technické a informační výchovy
Vedoucí práce:	Šaloun Petr, doc. RNDr. Ph.D.
Rok obhajoby:	2024

Název práce:	Vývoj všeobecně výukové aplikace pro zrakově postižené
Název v angličtině:	Development of generally educational app for visually impaired
Zvolený typ práce:	Aplikační práce
Anotace práce:	Tato práce se zaměřuje na vývoj vzdělávací aplikace Blindfold pro osoby se zrakovým postižením s využitím rozhraní enginu Unity a programovacího jazyka C#. Je rozdělena do čtyř kapitol - dvou teoretických, jedné teoreticko-praktické a jedné praktické; v nichž je popsána problematika a následně postup při vývoji, včetně veškerých poznatků.
Klíčová slova:	Programování, Aplikace, Zraková postižení, Herní engine, Unity, C#
Anotace v angličtině:	This Bachelor's thesis focuses on development of an educational application Blindfold for visually impaired using the Unity engine interface and the C# programming language. It is divided into four chapters - two theoretical, one theoretical-practical and one practical; in which the issues and then the

	development process are described, including any information learned.
Klíčová slova v angličtině:	Programming, Application, Visual Impairment, Game Engine, Unity, C#
Přílohy vázané k práci:	1
Rozsah práce:	46 stran
Jazyk práce:	Čeština

Obsah

Úvod

Tato práce se zaměřuje na vývoj vzdělávací aplikace s využitím rozhraní enginu Unity a programovacího jazyka C#. Aplikace by měla být zaměřena na částečně či úplně zrakově postižené a měla by být všestranná v podobě kvízu pro rychlé opakování otázek, souvisejících s vybraným učivem.

Práce je rozdělena na čtyři hlavní části. První část se věnuje teorii, týkající se zrakových onemocnění, na niž navazuje teorie vývoje aplikace. Následovány jsou teoreticko-praktickou částí přípravy a práce na samotné aplikaci a zakončeny částí, věnující se dokončení aplikace.

V první části vymezíme problematiku zrakově postižených a jejich vzdělávání pomocí aplikací, ať už mobilních, počítačových či webových. Dále se zaměříme také na vývoj aplikací a vývojová prostředí, konkrétně na program Unity. Od tohoto tématu přejdeme k přípravě a počátkům práce na samotné aplikaci. Budeme se věnovat především rozvržení, teoretickému fungování aplikace a funkcím, které využijeme. Navážeme zakončením v podobě výsledné aplikace a jejího praktického použití. Cílem je snažit se i o reflexi nad možnými úskalími či výhodami takovéto aplikace.

Cílem je zdokumentovat cestu, kterou bude muset autor zaujmout, aby zdárně vytvořil nástroj, který by měl mít za úkol vzdělávat skupinu, se kterou autor nesdílí formu vnímání světa. Výsledkem by tedy měla být aplikace pro opakování učiva, která je funkční a připravena k použití ať už osobnímu nebo pro práci učitele.

Není důležité, aby vzdělávání bylo velmi odborné či obsáhlé, neboť podobné projekty nejsou projektovány na dobu několika měsíců, jako bakalářská práce. Naopak – snahou by mělo být alespoň vydláždit cestu do budoucna, poznávat a zkusit navrhnout řešení, kombinující prvky her i vzdělávacích aplikací dohromady.

Motivací pro toto téma je hned několik faktorů. Jedním je samotný vztah autora k vývoji, hrám, programování a tvorbě obsahu obecně. Dalším z faktorů jsou výzvy, které tvorba takovéto aplikace přináší, a snaha rozšířit znalosti, které již autor nasbíral. A v neposlední řadě jde o cíl vytvořit aplikaci pro skupinu lidí, na kterou se ve vývoji velmi často zapomíná.

1 Problematika zrakových postižení

Kontakt se zrakově postiženými je v praxi pedagogických pracovníků dnes stále častější. Napomáhá tomu i podpora inkluze a přizpůsobování se jejich specifickým potřebám. Nicméně pojem „zrakově postižený“ nezahrnuje pouze osoby, postižené ztrátou zraku samotného, ale také osoby slabozraké, s poruchou barvocitu či jinak omezené v této oblasti. Podle Růžičkové patří slabozrací mezi nejpočetnější skupinu, čemuž napovídá i povaha samotného postižení – slabozraký jedinec vidí, ale pouze z poloviny tak ostře, jako jedinec zdravý *(Pospíšilová pro ČVUT, 2014)*. Proto se bude práce zabývat přípravou aplikace pro více skupin, tedy – aby text či grafika byly čitelné i pro slabozraké, a současně vizuálně příjemná i pro zdravé uživatele.

1.1 Vymezení zrakových postižení

Každé zrakové postižení se projevuje různě a současně vyžaduje různé zacházení s daným jedincem. Jsou nutné rozmanité pomůcky a subjektivní přístup, a ačkoli existují určitá jednotná pravidla pro práci se zrakově postiženými, je důležité pamatovat na specifickou povahu každého z nich a na různé potřeby, které může mít, stejně jako u zdravých jedinců. V této kapitole si některá postižení přiblížíme a autor se pokusí vymezit přístup, který zaujme ve vývoji, aby těmto potřebám vyhověl a co nejlépe tak našel shodu mezi jeho vlastní představou a potřebami uživatelů.

Pro potřeby této práce se budeme zabývat pouze poruchami, které spadají do skupiny poruch zrakové ostrosti a ztráty zraku. Konkrétně to budou – porucha barvocitu, slabozrakost a nevidomost. Jedním ze zásadních problémů u těchto postižení je problém s orientací v prostoru. Mohlo by se zdát, že toto s vývojem aplikace plně nesouvisí, nicméně pro účely vývoje to znamená celkové minimalizování nutnosti pamatovat si rozložení obrazu či vnímání pohybu myši. V praxi to bude znamenat vytvoření ovládacího systému, který bude intuitivně napomáhat ke snadnému ovládání a výběru možností. Pojďme si nyní jednotlivé skupiny uživatelů s poruchou zraku více přiblížit a uveďme kritéria či fakta, která bude nutné pro tyto skupiny zohledňovat.

1.1.1 Porucha barvocitu (barvoslepost)

Tato porucha se vyznačuje nevyvinutím čípku jedné ze tří základních barev (červená, zelená či modrá). Podle kombinace těchto rovin se může porucha projevovat různě, zpravidla však nerozeznáním, anebo těžším rozeznáváním barev od ostatních. V případě, že jde o kombinaci

všech tří rovin poruchy, mluvíme o achromatopsii. V takovém případě jedinec dokáže rozlišovat pouze černou a bílou *(Růžičková et al., 2006)*.

Podle uvedené definice a samotné povahy poruchy, kdy není jisté, jak vážnou úrovní poruchy bude uživatel trpět, je důležité zaměřit se na nejprimitivnější aspekt grafiky aplikace (detailní plánování grafiky a zdůvodnění jejího užití viz kapitola 3.3). Nemělo by proto být užito barevných prvků, které by uživatelé nemohli vnímat. Barvy se tedy budou vyskytovat převážně v černé a bílé, případně v odstínech šedi. Současně ale tak, aby byly rozdíly barev dostatečně kontrastní a barvy se nepřekrývaly, či nesplývaly, jak Růžičková dále uvádí.

1.1.2 Slabozrakost

Slabozrakost již byla zmíněna v úvodu kapitoly. Pojďme si ji tedy přiblížit a podrobněji popsat. Zpravidla je kvůli praktičnosti dělena na tři stupně – lehká, střední a těžká. Růžičková poté definici dále rozvíjí o speciálně pedagogické znění podle Flenerové, která uvádí, že slabozrakost je orgánová vada zraku, projevující se částečným nevyvinutím, snížením nebo zkreslující činností zrakového analyzátoru očí a tím i poruchou vnímání.

Mezi nejčastějšími chorobami, které můžeme do této kategorie zařadit, jsou – katarakta (šedý zákal), glaukom (zelený zákal) nebo nystagmus (mimovolné kmitání očí). Mezi dalšími, ze skupiny těch více známějších, můžeme uvést dalekozrakost, krátkozrakost (obě ve velmi těžkém stádiu) nebo hemeralopii (šeroslepost). Každá z těchto poruch se projevuje specifickým způsobem, nicméně všechny mají společné to, že se u postiženého jedince vyskytují zbytky zraku a zpravidla se zhoršenou ostrostí.

Na základě uvedené definice je patrné, že je pro vyvíjenou aplikaci důležité, aby obsahovala ostrý obraz, a současně z ní vyplývá ještě silnější důraz na dobrý kontrast barev. Je nežádoucí, aby byla grafika podstatných prvků rozmazaná nebo splývající. Mezi tyto prvky patří hlavně text grafiky či tlačítka, která lze v aplikaci vybrat.

1.1.3 Nevidomost

Poslední skupinou uživatelů ze skupiny zrakově postižených, se kterými má aplikace počítat, jsou nevidomí, případně také osoby se zbytky zraku, které nejsou schopné čtení. Růžičková uvádí data od Michalíka, že tato skupina činí asi jednu čtvrtinu (1/4) všech osob se zrakovým postižením u nás. Podle jeho studie *(Michalík in Renotierová, Ludíková, 2003, s. 33)* celkový počet zrakově postižených dosahoval až 60 000 lidí, z toho bylo nevidomých asi 17 000, nicméně portál Poslepu uvádí modernější data z roku 2018, která zprůměrovali s daty

z Českého statistického úřadu. Podle nich počet dosahuje až 74 000 lidí se zrakovým postižením, z toho by tedy podle statistiky, uvedené Michalíkem, mělo být přibližně 18 – 19 tisíc nevidomých *(Radek Pavlíček pro Poslepu, 2018)*.

Je zřejmé, že jestli má aplikace zohledňovat i tuto skupinu uživatelů, je nutné nejen vhodné grafické zpracování, ale i funkce, které umožní užití nevidomým. V průběhu práce je uvedeno, že právě tento záměr činil nejzásadnější problém. Je totiž nutné užití hlasového výstupu, vhodné uspořádání (seřazení) prvků aplikace a velmi snadné ovládání, které by mělo být intuitivní. Současně je důležité zaměřit se na prvky, které budou podporovat přirozený způsob práce s učivem, kterým je pro nevidomé memorizace.

1.2 Poruchy zraku a jejich souvislost se vzděláváním

V rámci této práce by se autor chtěl konkrétněji zaměřit spíše na specifika, spojená s jejím vývojem. Nicméně je důležité uvést některé prvky z obecného vzdělávání osob s poruchou zraku, které budou určovat obsah aplikace. Nejdůležitější je si uvědomit, že učivo, které je této skupině prezentováno, by mělo být většinou identické s učivem, které je prezentováno zdravým jedincům. Tomu napovídá i článek na webu Šance Dětem, kde PhDr. Kateřina Stejskalová, Ph.D. uvádí citaci od Vágnerové, podle které je dnes rodiči „výrazně preferována individuální integrace“, která představuje „určité potvrzení ‚normality‘ jejich dítěte, jeho pozitivních hodnot, schopností a dovedností navzdory zrakovému postižení“ *(Stejskalová pro Šance Dětem, 2014; Vágnerová, 2001, s. 159)*.

1.2.1 Dostupnost pomůcek pro zrakově postižené

Na základě zkušeností kolegů ze speciálně pedagogického oboru, kteří absolvovali praxe na školách či v zařízeních, kde se dětem s poruchami zraku věnují, je hlavním problémem právě rekapitulace učiva, opakování a rychlé procvičování. Existují nástroje pro nevidomé či slabozraké, jako NVDA, případně převod textu na řeč (dále „TTS“ z angl. „Text-to-speech“), který mívá mnoho moderních systémů, aplikací nebo webů k dispozici (příkladem může být funkce TTS od Microsoft nebo Google). Tyto nástroje jsou sice praktické pro předčítání učiva, nicméně jejich praktičnost klesá v případě, že si chce uživatel zařízení opakovat učivo jinak, než poslechem celých pasáží (např. „testovou“ formou).

Dalším problémem, který u těchto nástrojů vyvstává, je složitost převodu grafických prvků na řeč. Některé z nich mohou mít k dispozici rozšíření právě pro „čtení“ obrázků, mnoho z nich je ale zastaralých či nepodporovaných, jiné zase nepodporují český jazyk. Pro účely práce autor

vyzkoušel některé způsoby TTS, jmenovitě právě zmíněné NVDA, včetně jeho rozšíření pro „čtení“ obrázků, nicméně se mu nepodařilo rozšíření zprovoznit. Tato funkce je velmi podstatná pro samotnou aplikaci, neboť způsob, kterým si ji autor vybral vyvíjet, využívá herního enginu, který využívá grafických výstupů, nikoli textových (podrobněji vysvětleno v kapitole 2).

1.2.2 Konkretizace zaměření aplikace

Jelikož je tedy potřeba aplikací a nástrojů pro rychlé opakování, měla by aplikace obsahovat soubory otázek (dále pouze „okruhy“), které budou obsahovat několik možností, ze kterých má uživatel vybrat tu správnou. Autor se rozhodl pro podobu jakéhosi kvízu s možností přidávat vlastní otázky. Podrobněji se bude jeho výběru věnovat v kapitole 3.

1.2.3 Alternativy řešení pro převod textu na řeč

Naštěstí není nutné se spoléhat pouze na zabudovanou funkci TTS, anebo software třetí strany. Existuje několik alternativ, které lze pro tuto aplikaci využít. Jejich použití bude záležet na tom, jaká bude její finální podoba.

Uveďme si tři alternativy, se kterými se autor rozhodl pracovat, anebo zvážil jejich užití:

- hlasový výstup,
- použití systémové knihovny TTS,
- použití AI nebo pluginu.

2 Problematika vývoje aplikací

Tato kapitola se bude zabývat vymezením důležitých prvků, které se týkají vývoje aplikací, užitých nástrojů, pluginů (rozšíření), programovacího jazyka a dalších věcí, které budou také souviset s touto prací. Jako „aplikaci“ budeme pro potřeby této práce rozumět program, který vyvíjíme pro libovolnou platformu. Konkrétně program, který bude výsledným produktem této práce. Tento program bude vytvářen v rozhraní Unity, kterému se budeme věnovat v kapitole o „enginech“. Pro tuto práci bude ale základním jazykem C#.

2.1 Vývojová prostředí

Vývojové prostředí, z anglického „Integrated Development Environment“ (zkráceně IDE) je počítačový program, nebo obecně software, ve kterém je integrováno mnoho nezbytných nebo velice nápomocných funkcí, které významně pomáhají vývojářům při vývoji softwaru. Jeho hlavní částí je editor zdrojového kódu vyvíjeného softwaru. Příkladem těchto vývojových prostředí mohou být například softwary Miscrosoft Visual Studio, Eclipse, Netbeans a další *(Ing. Michal Strelec na portálu Michal Strelec)*.

Portál Peach-dev také uvádí jako hlavní vlastnosti a prvky těchto prostředí následující:

- **Editor kódu** – umožňující práci s kódem. Mnohé IDE nabízí zvýraznění syntaxe, automatické dokončování kódu a další funkce, které usnadňují jeho psaní.
- **Debugger** – užívaný k identifikování a řešení chyb v kódu tím, že umožňuje sledovat a analyzovat chování programu během jeho provádění.
- **Kompilátor** – překladač zdrojového kódu napsaný vývojářem do spustitelné formy, často do strojového kódu nebo bajtkódu.
- **Integrace s verzovacími systémy** – nástroje pro správu a sledování verzí kódu, což usnadňuje sledování změn a spolupráci s ostatními vývojáři.
- **Podpora pro návrh uživatelského rozhraní** – vizuální návrh rozhraní, který umožňuje vývojářům tvořit rozhraní aplikace pomocí drag-and-drop funkcí.

IDE je dále uváděno jako nezbytný nástroj pro většinu vývojářů *(web peach-dev)*.

Tomuto tvrzení o popularitě a důležitosti IDE v práci vývojáře napovídají i výsledky z webu PopularitY of Programming Language (zkráceně PYPL), který se zabývá aktivním sledováním popularity daných programovacích jazyků či IDE na základě míry vyhledávání pomocí

vyhledávače Google. Právě programovací jazyky, které najdeme na prvních místech, se velice často užívají ve spojení s různými IDE (ať už k práci s nimi určenými nebo alternativními).

Worldwide, Jun 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	29.06 %	+1.4 %
2		Java	15.97 %	+0.2 %
3		JavaScript	8.7 %	-0.6 %
4		C#	6.73 %	-0.0 %
5		C/C++	6.4 %	-0.0 %
6	↑	R	4.75 %	+0.3 %
7	↓	PHP	4.57 %	-0.5 %

Obrázek 1: Přehled sedmi nejpopulárnějších programovacích jazyků podle webu PYPL za červen 2024

Podle průzkumů téhož webu jsou aktuálně nejpopulárnějšími IDE Visual Studio, Visual Studio Code, Eclipse, pyCharm, Android Studio, IntelliJ a NetBeans.

Worldwide, Jun 2024 :

Rank	Change	IDE	Share	1-year trend
1		Visual Studio	28.06 %	-0.1 %
2		Visual Studio Code	13.69 %	-0.2 %
3		Eclipse	11.75 %	+0.2 %
4	↑	pyCharm	10.76 %	+2.1 %
5	↓	Android Studio	9.69 %	+0.5 %
6		IntelliJ	7.6 %	+0.6 %
7		NetBeans	4.07 %	-0.2 %

Obrázek 2: Přehled sedmi nejpopulárnějších IDE podle webu PYPL za červen 2024

Pro účely této práce je nejstěžejnější IDE Visual Studio, neboť právě ono bylo použito k vývoji aplikace.

2.1.1 GitHub Copilot

Pokud již byla zmíněna různá IDE, je v dnešní době také vhodné zmínit jeden z nejrychleji se rozšiřujících nástrojů, které některá IDE dnes nabízejí – GitHub Copilot. Jde o model umělé inteligence, vytvořený ve spolupráci mezi OpenAI, Microsoft a GitHub, určený k asistenci vývojářům a programátorům *(oficiální web GitHub Copilot)*.

Tento nástroj byl již začleněn do IDE Visual Studio a autor některých jeho výhod v rámci práce využil v podobě asistence při opravě kódu nebo při doplňování zamýšleného kódu. Ačkoli tento nástroj umožňuje zadávání požadavků pro psaní komplexních částí kódu, nebyl v této podobě v rámci práce nijak využit.

GitHub uvádí, že tento nástroj dnes využívá více než 50 000 firem, a že od jeho uvedení na trh vzrostla efektivita programování až o 55 %. Forma jeho využití se může různit od pouhé asistence při opravách nebo urychlení opakovaných činností, kdy se Copilot učí od vývojáře a sleduje jeho aktivitu, aby v případě opakovaných akcí nabídl okamžitou úpravu pouhým zmáčknutím klávesy tabulátor, až po zadávání textových požadavků a příkazů, pomocí kterých může vývojář nechat Copilot vygenerovat celé části kódu o mnoha řádcích v závislosti na přesnosti požadavku.

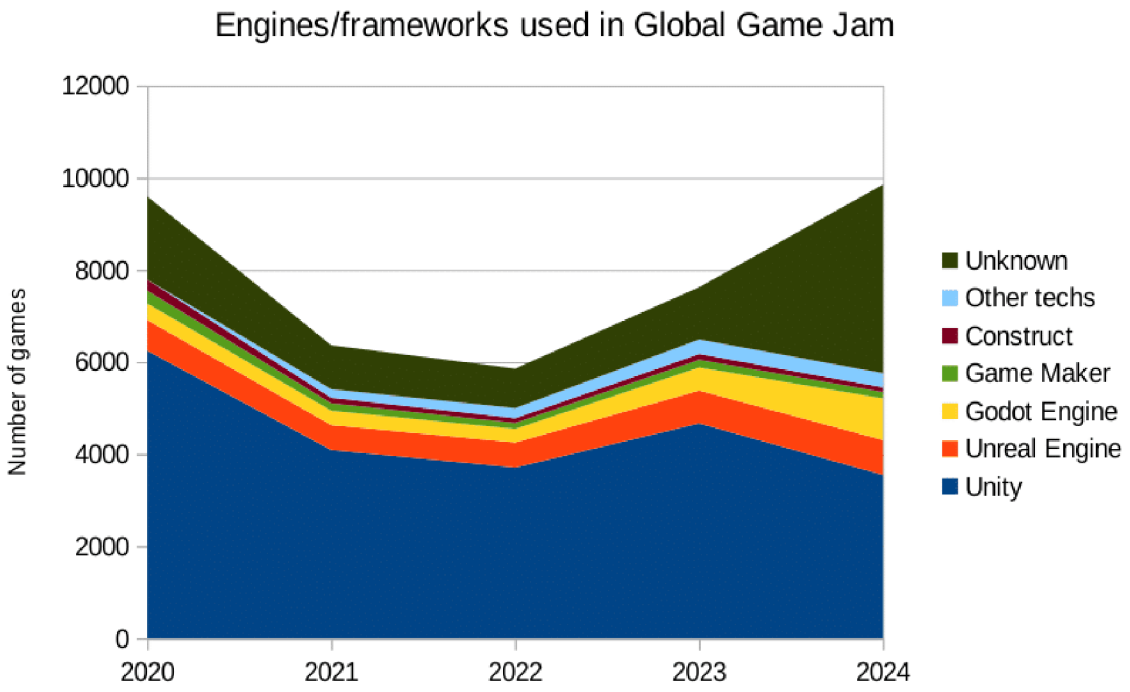
V rámci práce se Copilot projevil jako efektivní nástroj, který je velmi přizpůsobivý a je schopný chápat i možnosti, které nabízí Unity a nejsou obsaženy v základu jazyka C#.

2.2 Herní enginy

Podobně jako u IDE nebyly definice a popisy jednotné, avšak měly vždy podobné znění, je tomu tak i u enginů. Enginy můžeme v určité míře k IDE i připodobnit, protože jsou opět nástrojem, který vývojáři používají k usnadnění práce, urychlení procesu a podobně. Mnoho enginů má v sobě také zabudovány funkce IDE, jako je debug a okamžitá zpětná vazba při testování kódu.

Na oficiálním webu společnosti ARM, předního světového vývojáře CPU, najdeme definici znějící v překladu asi takto: „Herní engine je softwarové vývojové prostředí s nastaveními a konfiguracemi, které optimalizují a zjednodušují vývoj videoher napříč programovacími jazyky. Mohou obsahovat 2D nebo 3D grafický vykreslovací engine, engine fyziky, umělou inteligenci, zvukový engine, animační engine a další prvky.“ *(oficiální web ARM Holdings)*

Hlavní výhodou pro účely práce je fakt, že komerční herní enginy mívají často podrobnou dokumentaci a širokou uživatelskou základnu, což je činí snazšími k použití, než vytváření aplikací „od nuly“ či vytváření vlastního enginu. Mezi dnes nejpopulárnější enginy patří například Godot (nabírající na popularitě hlavně po „pádu Unity“, popsaném v kapitole 3.1.3), Unreal Engine (velmi silný engine pro vývoj 3D her), Game Maker nebo právě Unity, užitý v rámci této práce. Tato data vychází z průzkumu celosvětové soutěže Game Jam 2024. Byla uveřejněna na oficiálním webu a účtu platformy X organizátorů.



Obrázek 3: Přehled nejpopulárnějších herních enginů z ledna 2024

2.2.1 Unity Engine

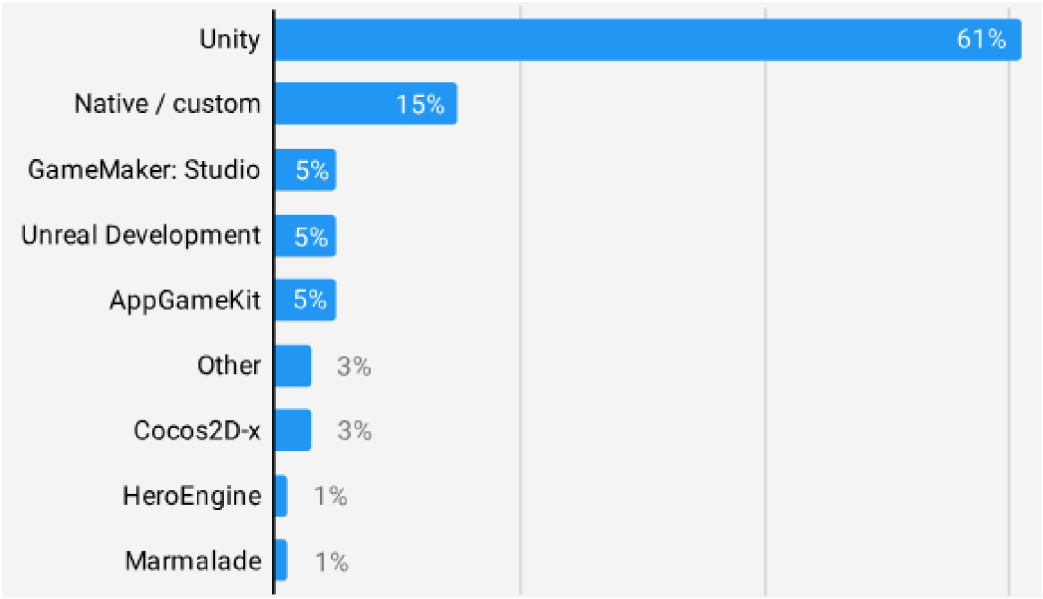
Samotný engine Unity má v komunitě vývojářů dlouholetou tradici díky několika faktorům – jeho naučení je velice snadné, práce s ním je přehledná, je zdarma a mimo dokumentaci má početnou skupinu tvůrců, kteří tvoří blogy či videonávody pro práci s tímto enginem.

Unity dnes aktivně podporuje pouze programovací jazyk C#, nicméně dříve podporoval i jiné jazyky a tuto podporu do jisté míry zachoval, příkladem u jazyku Java, JavaScript (a některých dalších), případně jde pro tuto podporu využít rozšíření *(oficiální web Unity)*.

Na oficiálním webu se také můžeme dočíst, že Unity byl původně zamýšlen jako 3D engine, který má ale také podporu 2D. O jeho objektivní efektivitě či neefektivitě z hlediska těchto dvou grafických pohledů nenajdeme spolehlivé informace, avšak ve vývojářské komunitě obecně převládá názor, že „existují lepší 3D enginy, než Unity“, a současně že „Unity není

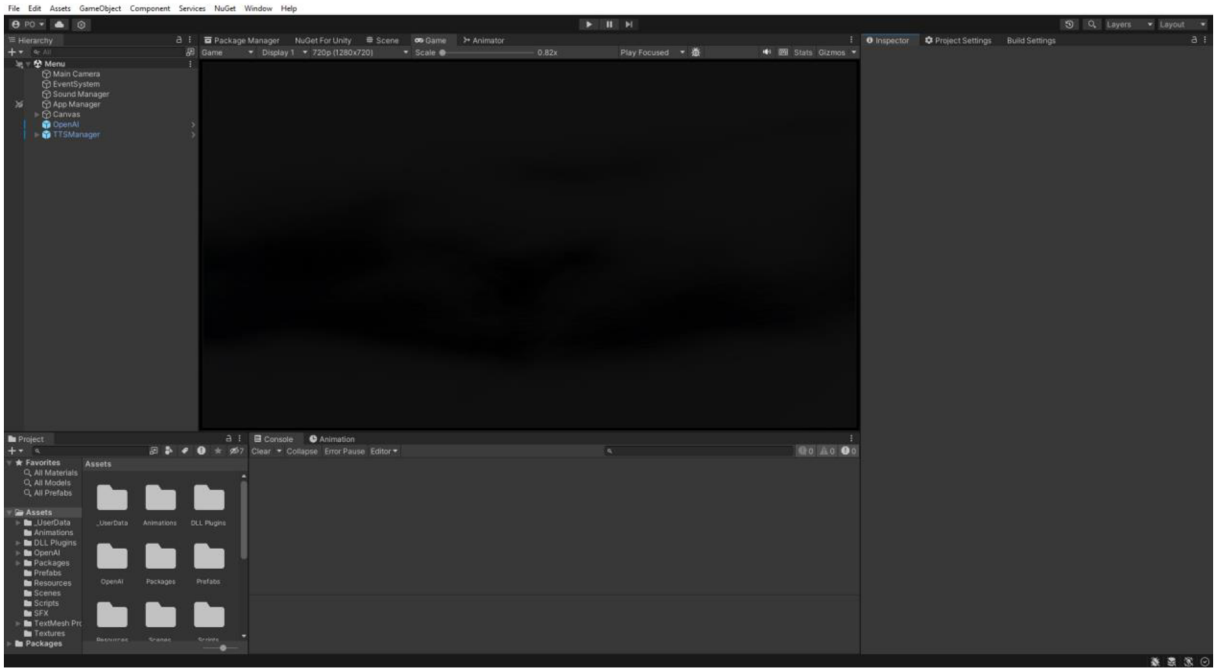
dělaný s myšlenkou efektivní funkčnosti 2D“. Nicméně se opět ve vývojářské komunitě obecně uvádí, že různé enginy vyhovují různým uživatelům a různým účelům, a že nelze nalézt univerzálně „nejlepší“ engine.

Unity podporuje tvorbu aplikací pro nejrůznější platformy, jako je standalone, Android, iOS, MacOS, PlayStation, Xbox, tvOS nebo WebGL. To ho činí velice všestranným a praktickým pro nejrůznější sortu vývojářů. Jeho největší využití najdeme hlavně u mobilních her, kde drtivá většina mobilních titulů byla vytvořena v Unity, včetně těch nejpopulárnějších (příkladem stojí za zmínku karetní hra Hearthstone nebo celosvětový fenomén Pokémon GO), jak i naznačují data průzkumu za rok 2021 přímo na webu Unity:



Obrázek 4: Přehled nejpopulárnějších herních enginů pro mobilní hry za rok 2021

Unity nabízí velice přehledné prostředí, ve kterém vývojář najde veškeré potřebné nástroje snadno a rychle. Prostředí si lze současně upravit včetně ukotvení a rozvržení doků nebo klávesových zkratek. Pro vyhledávání různých prvků slouží vnořené uživatelské rozhraní, ve kterém se uživatel snadno zorientuje. Na uvedeném obrázku můžeme vidět aktuálně otevřenou scénu. V levé části se nachází přehled všech objektů ve scéně, uprostřed její vizuální podoba, v pravé části je umístěn dok pro přehled vlastností vybraných objektů. V levé dolní části najdeme přehled souborů projektu a vedle nich dok s konzolí.



Obrázek 5: Ukázka vývojového prostředí Unity

Jak už bylo uvedeno, každý dok lze upravit a přemístit dle potřeby, případně je lze umístit jako záložky stejného doku (tedy najdeme dva a více doků v podobě záložek, umístěných u sebe).

Jako výchozí IDE pro práci s kódem Unity používá Visual Studio, nicméně lze upravit cesty ke většině souborů a nástrojů, které uživatelův klient využívá, tudíž lze zvolit i jiné IDE dle výběru uživatele.

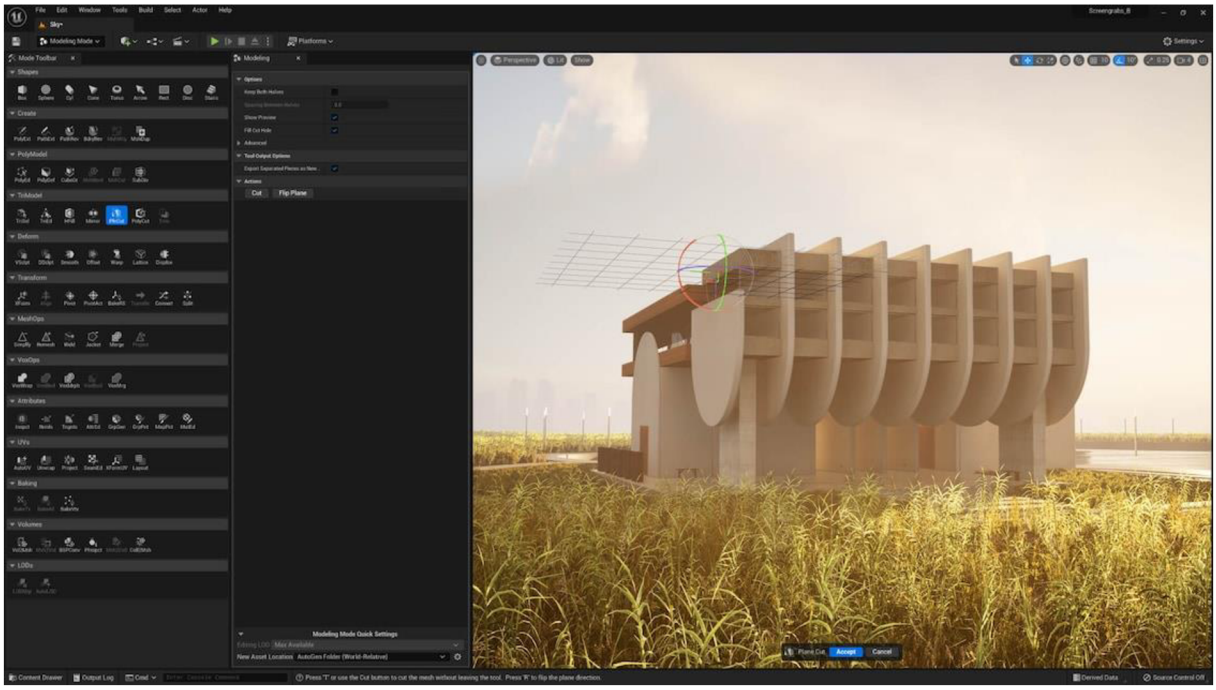
Současně nabízí širokou podporu knihoven, balíčků a pluginů, včetně spolupráce a podpory mnohých velkými firem, jako je Google, Microsoft či jiné. V případě potřeby se námi vybraná rozšíření dají snadno umístit do souborů projektu a Unity za nás odvede veškerou práci s jejich instalací a nastavením. K tomu je přímo v Unity zabudován správce balíčků, přes který můžeme snadno nalézt většinu nejpoužívanějších pluginů, anebo můžeme navštívit oficiální obchod Unity s komunitou vytvořenými balíčky, objekty, modely či dalšími potřebnými prvky.

2.2.2 Unreal Engine

O Unity se nedá skoro bavit bez zmínění jeho potenciálně největšího konkurenta, Unreal Enginu, vyvíjeného společností EpicGames. Autor již uvedl, že je Unreal velice silným 3D enginem, nicméně v online komunitě je toto často bráno jako velmi hrubé podcenění jeho možností, a mnoho vývojářů klade důraz na Unreal jako na „nejlepší 3D engine“. Názor na jeho vhodnost pro začátečníky bývá ale velice smíšený. Někteří tvrdí, že je vhodný a lze si na něj zvyknout, jiní tvrdí, že je pro úplného nováčka v oboru velmi nepřehledný, náročný a obsahuje příliš

mnoho funkcí, které začínající vývojář nevyužije. Součástí tohoto dohadu ale může být také fakt, že oproti Unity nemá Unreal tak silnou komunitu tvůrců, kteří by pro něj vytvářeli návody a blogy. Z grafu popularity enginů také můžeme vyčíst, že jeho popularita mírně poklesla ve prospěch Godotu a jiných nástrojů.

Hlavní výhodou Unreal je kvalita. I mezi balíčky dodatečného obsahu tvořeného komunitou nalezneme v Unreal mnoho velmi kvalitního obsahu. Společnost EpicGames také dává pravidelně část obsahu zdarma nebo se značnými slevami. Současně je samotné prostředí do jisté míry velmi podobné Unity, například upravitelnými doky.



Obrázek 6: Ukázka prostředí Unreal z jejich oficiálního webu

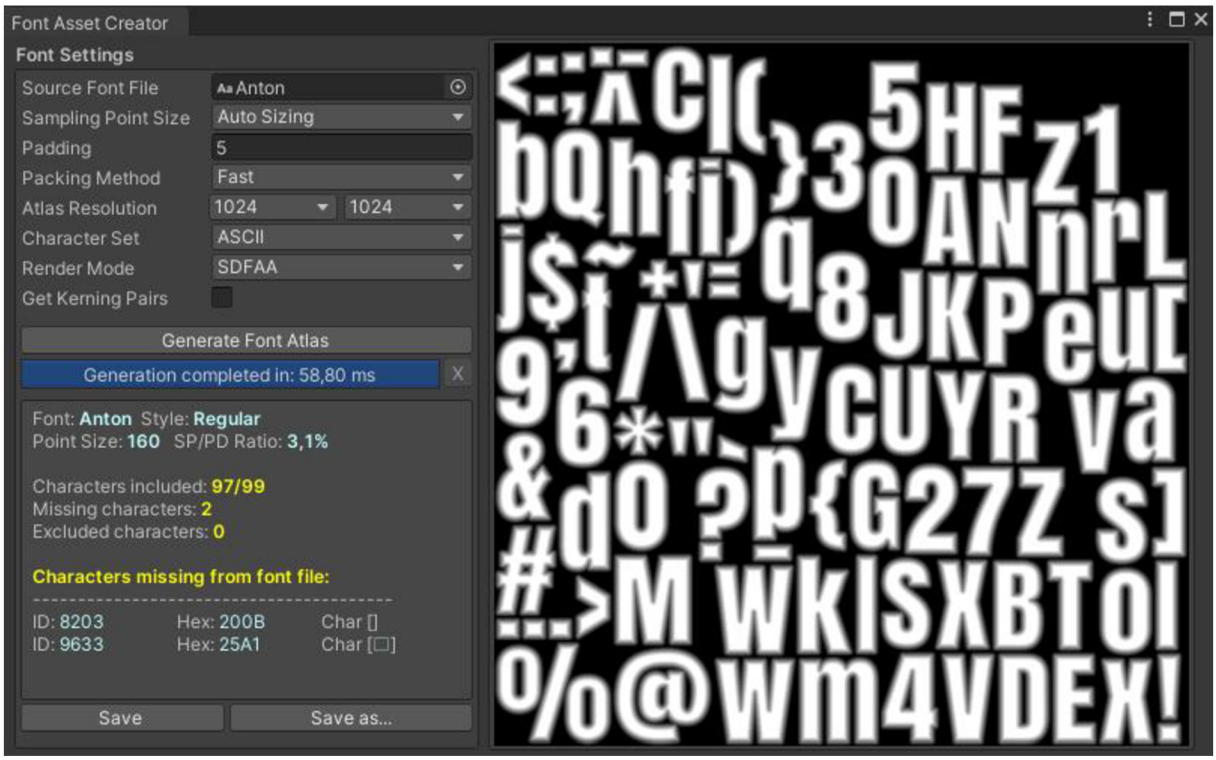
Ačkoli mírně ztratil na popularitě, obdržel v rámci posledních měsíců značné množství aktualizací a nových nástrojů, které umožňují tvorbu velmi realistických světů, prostředí a animací *(oficiální web Unreal, 2023)*.

2.2.3 Problematika grafického převodu textu u herních enginů

Už bylo uvedeno, že herní enginy využívají širokou škálu nástrojů, které je odlišují od jiného softwaru. Jako jedno ze specifik byly uvedeny vykreslovací enginy. Tyto vykreslovací enginy jsou použity prakticky pro veškeré úkony spojené s grafikou, současně tedy i s prvky uživatelského rozhraní, jako je právě text.

Olle Alvin ve své práci uvádí v překladu zhruba následující: „Nejpřímější způsob pro vykreslení textu ve 3D herních enginech je před-rastrovat jednotlivé glyfy/znaky (daného fontu)

do jednoho obrázku, obsahujícího všechny z nich. Tento obrázek je nazýván atlas textur.“ Podobný atlas textur využívá i již zmíněný engine Unity, například pro práci se spritovými animacemi, případně v rámci rozšíření TextMeshPro, podrobněji popsaném v kapitole 3.5.1.



Obrázek 7: Ukázka nástroje Unity pro vytvoření atlasu textur pro fonty

Olle Alvin dále uvádí: „Každý znak je obvykle rozdělen na dva trojúhelníky, tvořící čtyřúhelník. Tento postup je velmi efektivní a utváří velmi kvalitní statický text. Nicméně pokud bychom text zvětšili, moderní GPU často vyprodukují rozmazaný obraz. Abychom zachovali dobrou kvalitu textu, všechny velikosti (znaků) by musely být uloženy v atlasu textur, případně by musely být rastrovány za běhu na CPU, což bývá pro hry často příliš pomalé.“ *(Olle Alvin et al., 2020, s. 10)*

Z toho vyplývá, že text v herních enginech je převáděn na grafickou texturu, která je přizpůsobována požadované velikosti. Proto je nemožné text číst asistenčními aplikacemi pro čtení textu.

2.3 Programovací jazyky (jazyk C#)

Pavel Bory, vývojář senior pro projekty společnosti Unicorn a. s. ve své knize definuje programovací jazyky jako „jazyk, srozumitelný počítači, pomocí kterého zapíšeme algoritmus počítačového programu“ *(Bory et al., 2022, s. 17)*.

Tuto definici můžeme rozšířit o definici z webu vlašimského gymnázia, které ve svých materiálech pro žáky uvádí následující: „Programovací jazyky jsou jazyky sloužící k tvorbě počítačových programů (programování). Programování je proces algoritmizace dané úlohy, tj. vytváření postupu, jenž vede k řešení dané úlohy;“ a dále: „XML, HTML, WML apod. nejsou programovací jazyky ale jazyky značkovací.“ *(web gymnázia Vlašim, 2014)*.

Na zmíněném webu jsou dále programovací jazyky rozděleny do různých skupin podle jejich funkcionalit, užití a principu fungování. Mezi uvedenými příklady se objevují i nejpopulárnější jazyky, uvedeny v kapitole 2.1.

Z těchto programovacích jazyků je pro práci nejstěžejnější již zmíněný jazyk C#. Jde o vyšší programovací jazyk z rodiny jazyku C, který je objektově orientovaný a běží na platformě .NET Framework. Mezinárodní vzdělávací web W3schools zároveň uvádí, že je velice podobný jazykům C++ nebo Java, což je vzájemně činí snadnými na uchopení při znalosti některého z nich. Současně uvádí, že poslední aktualizaci obdržel v roce 2023, a že je velice snadný k naučení a použití s širokou komunitní podporou *(web W3schools)*.

Jazyk C# má současně širokou podporu nejrůznějších platforem a je tak velice všestranný. Užívá se pro tvorbu desktopových aplikací, ale také aplikací webových, mobilních, k vývoji her a dá se také využít pro virtuální realitu, webové stránky či služby. Jednou z jeho výhod a specifik je využití abstraktního datového typu List<>, který je do jisté míry velmi podobný klasickému poli. Poskytuje ale některé možností, které pole přímo nenabízí, jako například jednoduché přidání konkrétního prvku do seznamu pomocí List<>.Add() nebo odebrání pomocí List<>.Remove(). Není tak nutné prvek v celém seznamu vyhledávat, jak je tomu u polí, protože tento úkon C# provede za vás. Další výhodou seznamu je možnost utvářet seznamy uvnitř seznamů, které mohou být i jiného typu, než je tomu u polí, u kterých můžeme sice vytvořit multidimenzionální pole, avšak pouze jednoho daného typu. Tato výhoda je uvedena hlavně kvůli jejímu využití v rámci aplikace.

Mezi dalšími výhodami můžeme uvést přehlednost, vzájemnou kompatibilitu a komunikaci mezi jazykem, IDE a enginem Unity, případně podporu u zmíněného Github Copilot.

3 Příprava a počátky práce

Autor by rád v této části popsal svoji představu, týkající se vývoje aplikace. Hlavní nápady, body a prvky, které by měla obsahovat, již autor naznačil v předešlých kapitolách. Tato část má za cíl některé z nich konkretizovat a přiblížit.

Jedná se převážně o návrh možné podoby, která by v ideálním případě následně prošla měsíci testování, sbíráním zpětné vazby a úpravami na základě těchto připomínek, hlavně ze strany uživatelů zrakově postižených. Tato možnost, bohužel, není zcela k dispozici, a proto se autor pokusí co nejvíce upravit již počáteční verzi aplikace tak, aby byla co nejvhodnější pro tuto skupinu uživatelů. Aplikace by zároveň měla být přizpůsobena i „zraku zdravého uživatele“, neboť podobné aplikace bývají zpravidla graficky nepřitažlivé, což je věc, která by mohla všedního uživatele odradit. Aplikace by tedy měla být i vizuálně přívětivá.

3.1 Proč Unity a C#?

V předešlých kapitolách se již objevil popis těchto dvou nástrojů. Unity je, jak už bylo zmíněno, převážně herní engine. Nicméně jeho využití je široké a je velice výhodný i z hlediska možností, které vývojářům nabízí. Má širokou komunitu s velmi aktivními fóry a je celkově snadný k užití.

3.1.1 Vhodnost Unity

Ačkoli je jeho použití pro tento typ aplikace velmi neobvyklé, jeho podpora různých platforem jak při vývoji, tak při stavění aplikace, ho činí praktickým pro tvorbu pomůcek a nástrojů. Zřídka se setkáme s problémem, kdy by aplikace vyvíjená pro systém Windows nefungovala i na produktech firmy Apple, případně na mobilních zařízeních nebo v prohlížeči. Je tak tedy možné aplikaci vyvíjet pro jednu platformu a poté ji pouze zaměnit před vydáním aplikace.

Nicméně je důležité pamatovat na riziko nekompatibility z hlediska ovládání, rozlišení a podobně. Zde naštěstí přichází na řadu možnosti, které umožňují nastavovat pevné či případně přizpůsobivé rozlišení, které zajistí kvalitní obraz u různých poměrů displejů. Další možností je využít kód, který by upravoval rozlišení napevno. Může se totiž stát, že námi zvolené rozlišení a omezení nebudou zcela vhodně umístěny na mobilních zařízeních, která mívají obraz přes celou velikost displeje. Zde přichází C# a jeho možnostm vymezení oblastí kódu, které můžeme regulovat a ignorovat podle námi stanovených podmínek. Nyní nejsou myšleny klasické podmínky „if“ a „else“, které by byly obvykle uvnitř struktur, ale podmíněné

kompilace, které vypadají velmi podobně, ale ve výsledku se chovají jinak ve smyslu toho, že umožní ignorovat celý úsek kódu při kompilaci. Příklad užití je uveden na obrázku 8.

```
#if PLATFORM_STANDALONE
quitButton.SetActive(true);
#endif
```

Obrázek 8: Příklad užití podmíněné kompilace k vymezení části kódu pro určitou platformu

Při spojení této možnosti s kódem pro úpravu rozlišení tak můžeme vytvořit úseky kódu, které budou efektivní z hlediska toho, že nebudou na námi vybraných platformách obsaženy a nehrozí tak výskyt chyby, která by způsobila, že se rozlišení nevhodně upraví pro nezamýšlenou platformu.

3.1.2 Vhodnost C#

Již bylo zmíněno, že C# je aktuálně hlavním programovacím jazykem, podporovaným Unity. Jeho užité je tedy na jednu stranu logické, na druhou stranu ale také snadné a efektivní. Zmíněná podmíněná kompilace tohoto jazyka může být jedním z důvodů. Dalším důvodem je přehlednost a snadné užití. C# je velice snadný na pochopení a má jasnou strukturu. Každá část kódu je řádně oddělena a pro dodatečné zpřehlednění můžeme zkombinovat „srolování“ sekcí podmínek či struktur s komentáři, což zajistí, že se u dlouhého kódu budeme snáze orientovat.

```
// Úkony k provedení před vypnutím aplikace
// a její následné ukončení.
public void QuitApp()...
```

Obrázek 9: Příklad kombinace srolování a komentáře

Mezi dalšími důvody by mohla být široká podpora tohoto jazyka různými platformami. U jazyků jako Java bývá nutností nainstalovat i prostředí, aby aplikace fungovala. Zatímco ji to činí praktickou, protože není závislá na platformě, ale na přítomnosti prostředí, pro všedního uživatele to nemusí být žádoucí a autor se již setkal s uživateli, neochotnými prostředí instalovat, nebo případně neznalými toho, kde prostředí získat.

3.1.3 Kontroverze ohledně Unity

Ačkoli je Unity silný nástroj, nese si s sebou i řadu negativ. Jelikož jde o komerční nástroj, vlastněný nadnárodní firmou, je zde riziko, že se mohou změnit podmínky užití, celkové fungování nástroje nebo jeho dostupnost. K dispozici je sice bezplatná verze pro osobní užití a drobné projekty, nicméně se již objevily situace, kdy mělo dojít k zásadním změnám, které by ohrozily drobné vývojáře v jejich svobodě.

Příkladem může být nedávná kontroverze, ke které došlo v roce 2023, kdy se podmínky používání a míra poplatků měly změnit z bezplatných na placené podle počtu uživatelů aplikace. Tyto změny mohly zásadně ohrozit bezplatné a drobné projekty, nicméně jsou v aktuální chvíli upraveny a firma některé změny vzala zpět *(oficiální web Unity, 2023)*. Ačkoli se to přímo nedotýká tématu práce a samotného vývoje, je to skutečnost, na kterou je nutné brát zřetel a je nutné si ji uvědomovat kvůli možné budoucí politice firmy.

3.2 Problém převádění textu na obraz

Tato kapitola navazuje na problém, týkající se herních enginů z hlediska grafického výstupu, který byl naznačen v kapitole 2. Jelikož je veškerý text, který v Unity zobrazujeme, převáděn na obraz, tvoří to bariéru mezi nevidomým uživatelem a aplikací, jejíž překonání je tedy na vývojáři. Jedinou výjimkou je pole pro zápis textu, které ale při testování nedokázaly aplikace pro čtení snímat.

3.2.1 Alternativy pro čtení textu

Text bude nutné uživateli předčítat v rámci aplikace, k čemuž existuje hned několik možných řešení. V rámci práce autor uvažoval nad těmito možnostmi:

- **Hlasový výstup** – nahraný autorem aplikace, případně člověkem, který vytvořil okruh otázek. Negativem je zdlouhavější příprava otázek, pozitivem ale je mnohem větší flexibilita v jejich formulaci a také přítomnost emocí v mluveném slovu.
- **Užití systémové knihovny TTS** – tuto knihovnu mívá k dispozici mnoho systémů a zařízení by tak bylo schopné samo převádět text, který by mu aplikace poskytla. Nicméně pokud by se autor rozhodl aplikaci exportovat i v podobě pro web, nelze se na tuto funkci spoléhat. Současně implementace této knihovny se může lišit s každou platformou, nemusela by tedy být konzistentní.
- **Užití umělé inteligence nebo pluginu** – tato možnost se nabízí obzvlášť v dnešní době. Některé firmy poskytují zdarma rozhraní pro AI, která je schopná převádět text na velmi

lidsky znějící řeč. Negativem je nutnost připojení k internetu, protože mnoho těchto nástrojů odesílá přečtený text ze svých serverů, na kterých je AI umístěna. Současně mohou vyvstat problémy s převodem některých znaků či specifického textu. Příkladem může být matematický zápis „$\pi \times r^2$“, který měla testovaná AI tendenci číst jako „pí krát er čtvereční“. Velikým pozitivem je ale zproštění nutnosti nahrávat ke každé otázce a odpovědi zvukovou stopu. Tato skutečnost by byla přínosem i z hlediska datového objemu aplikace.

3.2.2 Užití umělé inteligence

Pro své řešení autor vybral poslední možnost. Užití umělé inteligence je velmi efektivní a v dnešní době velmi populární. Jelikož tvorba umělé inteligence zabere mnoho času a zdrojů, byl autor nucen vyhledat již hotový model, který zároveň poskytuje vhodné rozhraní a současně má podporu českého jazyka jak ve čtené, tak mluvené formě.

Jednou z prvních možností byla AI Polly od firmy Amazon, která jej poskytuje bezplatně pro vývojáře, kteří se přihlásí do jejich vývojářského programu. Při testování nebyla nijak těžká k implementaci, neboť Amazon k Polly nabízí i balíček, vytvořený přímo pro užití v Unity. AI dokáže přečíst české texty, ale jejich výslovnost zatím není v češtině podporována. I přes velmi přirozeně znějící hlasy byla tato možnost nevhodná.

Další možností bylo užití velmi populárního modelu od společnosti Google. S tímto modelem se můžeme setkat v mnoha dnešních aplikacích s podporou TTS. Poznáme jej podle typického hlasu, známého třeba z webu Google Translate. Při testování dosáhl tento model asi nejlepších výsledků, co se týče výslovnosti a přesnosti mluvy. Nicméně pro jeho fungování bylo nutné nainstalovat na zařízení autentifikační aplikaci, bez které model nemohl data odesílat. Při snaze vytvořit jiné řešení, které by odesílalo data přímo na server bez nutnosti autentifikační aplikace, autor nedosáhl pozitivního výsledku, a proto i od této možnosti upustil.

Posledním a také autorem vybraným modelem je model společnosti OpenAI. Jejich produkty se staly z poslední dobu velmi populárními i díky jejich službě ChatGPT, jak napovídá i průzkum z webu Visual Capitalist z letošního roku (uvedeno v miliardách):

AI Tool	Total Web Visits (Sept 2022 to Aug 2023)	Share of Industry Total
ChatGPT	14.6B	60.2%
Character.AI	3.8B	15.8%
QuillBot	1.1B	4.7%

Obrázek 10: Srovnání nejpopulárnějších AI nástrojů (Niccolo Conte pro web Visual Capitalist, 2024)

Jejich model sice nemá k dispozici přímou podporu Unity, jako předchozí dva modely, nicméně jejich rozhraní je velmi snadné k implementaci v podobě web requestů. Současně se autorovi podařilo najít veřejně dostupný balíček jiného uživatele Unity, který implementaci OpenAI modelu značně zjednodušuje. Hlavním negativem je nedostupnost bezplatné verze, nicméně účtované poplatky nejsou nikterak vysoké. Při testování měl model velmi přívětivě znějící hlas a velmi obstojnou češtinu. Má ale tendenci k anglickému přízvuku při výslovnosti hlásek a slabik R, HA a dlouhých písmen. Tyto problémy ale nebyly natolik markantní, že by měly zásadně ovlivnit užití aplikace.

3.2.3 Implementace umělé inteligence

Jak již bylo uvedeno – OpenAI neposkytuje žádný balíček k přímé implementaci jejich modelu v Unity. Nicméně lze jejich model využít pomocí web requestu, který odešle veškeré informace, jako model AI, input, typ hlasu, formát a rychlost řeči, včetně autorizačního klíče v Json formátu. Zde je příklad implementace, dostupné v balíčku OpenAI Text-To-Speech Integration for Unit. Proměnné jsou obsaženy v argumentech funkce, která tento kód obsahuje.

```
using var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", openAIKey);

TTSPayload payload = new TTSPayload
{
    model = model.EnumToString(),
    input = text,
    voice = voice.ToString().ToLower(),
    response_format = this.outputFormat,
    speed = speed
};

string jsonPayload = JsonUtility.ToJson(payload);

var httpResponse = await httpClient.PostAsync(
    "https://api.openai.com/v1/audio/speech",
    new StringContent(jsonPayload, Encoding.UTF8, "application/json")
);

byte[] response = await httpResponse.Content.ReadAsByteArrayAsync();
```

Obrázek 11: Příklad implementace requestu v C#

Následně získaná data načteme v bajtové podobě a až následně s nimi budeme pracovat v podobě mp3 formátu. Tato získaná data uložíme pomocí C# funkce File.WriteAllBytes(). Tímto získáme mp3 soubor. Data následně z uloženého souboru přečteme pomocí dalšího web requestu, který ale tentokrát odkážeme na umístění v našem zařízení. Po přečtení můžeme soubor buďto vymazat a uvolnit tak místo pro další, nicméně tato možnost by nebyla zcela efektivní z hlediska neustálého odesílání requestů při každém čteném textu. Tím by docházelo k nadměrnému využívání a tedy i k vyšším nákladům.

Namísto toho soubor ponecháme uložený v příslušném souboru otázky, aby jej bylo možné načíst opakovaně, případně jej poskytnout i dalším uživatelům či v případě nedostupnosti internetu. Soubor můžeme ukládat buďto do takzvané Application.persistentDataPath, která data uloží do složky LocalLow v počítači uživatele, případně do příslušné složky v mobilním zařízení. Mnoho uživatelů ale samo není obeznámeno s existencí této složky a tak je vhodnější využít cestu Applicaton.dataPath, která odkazuje na samotnou složku aplikace. Tento odkaz by mohl mít podobu, uvedenou na obrázku 12.

```
string path = $"{Application.dataPath + Path.AltDirectorySeparatorChar + "Custom"}/audio.mp3";
```

Obrázek 12: Příklad cesty k souborům aplikace

Tato cesta je uložena jako string, který je pro přehlednost uveden ve většině hlavních scriptů, které se neodkazují na sebe navzájem. Pro jistotu uvedu, že symbolem $ v C# označujeme formátovaný textový řetězec. V našem případě je to tedy řetězec, který bude mít podobu Blindfold/Custom/audio.mp3.

S pomocí kombinace AI, webrequestů a propojení s kódem, který umožňuje ovládání pomocí klávesnice, tak můžeme vytvořit vlastní, spolehlivý TTS systém, který řeší problém s grafickým převodem textu.

3.3 Výběr a návrh grafiky

Již bylo uvedení řešení textového předčítání. Nicméně mimo tento grafický prvek (myšleno text) je potřeba vybrat vhodnou grafiku aplikace, která bude splňovat již v první kapitole uvedené požadavky pro jednotlivé kategorie zrakových onemocnění. Pro jistotu si připomeňme, že by grafika měla být pokud možno kontrastní, hrany objektů a textu by měly být zřetelné a s ohledem i na osoby s poruchou barvocitu není zcela vhodné používat příliš barevných kombinací. Barvy tedy budou ponechány ve stupních šedi s takovými odstíny, které nebudou příliš splývat.

3.3.1 Panely a tlačítka

Jelikož nebude pro aplikaci zapotřebí vytvářet náročnou vlastní grafiku a bude stačit vycházet pouze z jednoduchých základních tvarů, je možné upotřebit základní možnosti, které nabízí Unity. Jsou v něm již v základu dostupné některé návrhy, ať už prázdné obdélníky, které je možno obarvovat, anebo některé sprity, jako textura pro panely s oblými roky. Jejich kombinací

můžeme dosáhnout překvapivě vizuálně přívětivých výsledků. Jako příklad uveďme dvě různé značky pro vybrané objekty, znázorněné na obrázku 13:

Obrázek 13: Dva návrhy zobrazení výběru objektu

Na uvedených obrázcích je uveden návrh výběru odpovědi při zodpovídání otázky (vlevo) a návrh výběru možnosti v menu, popřípadě v nastavení (vpravo). Pro tyto dva různé návrhy oproti jednomu jednotnému návrhu se autor rozhodl z vizuálních důvodů, neboť každý návrh v celkovém rozvržení aplikace vypadá lépe právě v této podobě, nicméně jde i o prvek předcházení pocitu, že je design repetitivní.

3.3.2 Logo, font a dodatečné sprity

Mimo uvedené druhy grafických prvků se také autor rozhodl vytvořit jednoduché logo, které bude současně i ikonou samotné aplikace. Bude se jednat o pouhé slovo Blindfold s náznakem pásky přes první písmeno. Návrh je minimalistický a kontrastní, aby příliš nerozptyloval, ale současně značil smysl aplikace.

Z dostupných fontů autor vybral font Anton, který je dostupný v základu pluginu TextMeshPro, který aplikace využívá pro zobrazování textu (více v kapitole 3.5). Tento font je velmi přehledný, má ostré a zřetelné hrany a po konzultaci s kolegy ze speciálně pedagogického oboru byl vyhodnocen jako nejvhodnější i díky jeho úzké podobě, která šetří místo v textových polích.

K dostupným spritům autor vytvořil ještě několik vlastních návrhů pro zpestření vizuální stránky, jako obrázek v pozadí, který vznikl náhodným kombinováním přechodů černé a tmavě šedé. Není tak příliš výrazný, ale slouží jako skvělý podklad, který dodává příjemnější nádech. Mimo něj je zde i rám pro „scény“ s výběrem otázek, návrh spritu tlačítka (viz Obrázek 7: Ukázka nástroje Unity pro vytvoření atlasu textur pro fonty), ikona složky a další drobné sprity.

Obrázek 14: Ukázka vlastních spritů

3.4 Rozvržení aplikace

Ačkoli návrh rozložení aplikace předcházel samotnému grafickému zpracování, je uveden až v této kapitole kvůli návaznosti na technickou přípravu. Nejdůležitějším rozhodnutím v rozvržení bylo, zda aplikaci dělat s různými scénami, mezi kterými bude uživatel přenášen, anebo využít pouze jednu scénu, ve které následně budou umístěny jednotlivé panely, mezi kterými se bude dát přepínat.

Výhodou i nevýhodou první možnosti je, že data se mezi scénami sama od sebe neukládají. Můžeme tak přirozeně pročistit paměť a můžeme také omezit počet načtených objektů pouze na ty, které jsou dostupné v dané scéně. Nicméně by potřebná data musela být přenášena „ručně“, tedy – muselo by být přesně vybráno, která data se mají mezi scénami uložit, a která ne. Zároveň vícero scén znamená opakované načítání při přechodech mezi nimi, což může u pomalejších zařízení trvat delší dobu a může to být z hlediska uživatele nežádoucí. Proto bude

aplikace stavěna s jedinou scénou, nicméně pro účel přehlednosti bude termín „scéna“ používán jako výraz pro jednotlivé sekce aplikace – menu, výběr témat, výběr otázky a podobně.

S ohledem na různé poruchy zraku dbal autor při rozvržení jednotlivých scén na vhodné umístění a hlavně na dostatečné mezery mezi jednotlivými tlačítky a panely, aby nedocházelo k vizuálnímu splývání.

Hlavní menu bude rozvrženo do tří hlavních tlačítek, umístěných v levém horním rohu, pro spuštění a pro nastavení, společně s jejich náležitými indikátory výběru, zatímco v pravém dolním rohu bude umístěno logo aplikace.

Obrázek 15: Rozvržení hlavního menu

Scéna nastavení bude obsahovat pouze tlačítka pro základní úpravy, jako je změna hlasitosti čtení či úplné vypnutí mluveného slova. Původně měly být k dispozici i možnosti pro výběr hlasu nebo rychlosti čtení, tyto možnosti byly ale z finální aplikace odebrány z důvodu ať už kapacity na uživatelském zařízení, neboť by aplikace musela obsahovat kombinaci každé volby s každou další volbou, ale také z kapacitních důvodů samotného API umělé inteligence.

Poslední dvě scény – scéna výběru témat a scéna samotného kvízu – jsou rozvrženy staticky, avšak se aktualizují dynamicky. Ve scéně výběru témat najdeme pevně nastavený panel s rolovacím obdélníkem v pravé části, se kterým lze rolovat nahoru či dolů skrz seznam okruhů. Samotné okruhy jsou ale jednotlivé objekty, které jsou vytvořeny z předlohy až na základě

načtených souborů aplikace. Každý objekt reprezentuje jeden okruh. Jméno odpovídá jménu složky okruhu a počet otázek odpovídá počtu složek uvnitř každé složky okruhu.

Obrázek 16: Rozložení scény výběru témat

Scéna kvízu se skládá pouze z panelů pro jednotlivá textová pole, v horní části pro otázku a v dolní části pro jednotlivé odpovědi s příslušícími indikátory. Tato textová pole jsou aktualizována dynamicky v závislosti na vybrané otázce, kterou systém načte ze souborů.

Obrázek 17: Rozložení scény kvízu

3.5 Příprava potřebných rozšíření a nastavení

Mimo základní funkce Unity a C# bylo zapotřebí stáhnout, nainstalovat či povolit několik dodatečných rozšíření a balíčků. Některé z nich byly podstatné pro samotný chod aplikace, jiné jsou vhodné pro usnadnění práce či zefektivnění. Základními balíčky, které je vhodné zmínit, jsou: již zmíněný balíček propojení s AI, propojení s IDE Visual Studio, rozšíření pro 2D uživatelské rozhraní (UI) pro zobrazování textu, obrázků a podobně, balíčky pro systémovou diagnostiku, správu a kompilaci a některé další, které by měly být automaticky přítomny při vytváření projektu.

3.5.1 Plugin TextMeshPro

Hlavní součástí aplikace je přehledné zobrazení textu, které je podstatné pro osoby se zbytky zraku či osoby s poruchou barvocitu. Unity v tomto ohledu dříve nabízel pouze základní text, který se skládal ze samotného fontu a základních prvků, jako tučné písmo, kurzíva a dalších. Nicméně tento základní balíček způsoboval ztrátu ostrosti při větším přiblížení či zvětšení písma. Současně zde chyběly prvky pro zvýraznění jako stínování nebo nastavení zvýraznění.

Tento problém vyřešil plugin, který nastoupil na Unity trh s rozšířeními a dodatečným obsahem. TextMeshPro se stal natolik populárním a praktickým nástrojem, že jej Unity později začlenil jako hlavní nástroj pro práci s textem a poskytl jej zcela zdarma *(Brian Winter pro oficiální blog Unity, 2017)*.

Ve článku, který o tomto začlenění mluví, je také stručný popis, který zmiňuje využití shaderů, mimo jiné vysvětlující důvod, proč není text detekovatelný aplikacemi pro čtení textu, ale také zde najdeme zmínku o funkcích, jako zvýraznění, efekt svícení, využití textur a masek. Současně je rozšíření optimalizováno k použití pro počítače i mobilní zařízení, což z něj dělá skvělého kandidáta pro vývoj aplikace s potenciálním portem pro různé platformy.

3.5.2 LeanTween

Pro vizuálně uspokojivější pocit z aplikace, oddělení jednotlivých scén a pro vytvoření prostoru mezi přechody pro načtení dat a aktualizaci jednotlivých nastavení bylo vhodné využít drobné animace, které tyto požadavky zajistí. Unity v základu poskytuje systém animací, který má ale značnou nevýhodu, která vychází na povrch až při hlubším pochopení informací z dokumentace. Touto nevýhodou je fakt, že jeho animace jsou neustále aktivní v určitém stavu, který čeká na příští příkaz. Tímto zatěžuje zařízení neustálým udržováním animací „v chodu“.

Knihovna LeanTween se na tento problém zaměřuje a řeší ho tím, že animace provádí pomocí takzvaného „tweenování (anglicky tweening)“. Podle oficiální webové příručky Adobe: „Tweening je zkrácená podoba inbetweening, což je proces generování obrázků mezi jednotlivými klíčovými snímky. Těmi jsou počáteční a koncový snímek plynulé animace. Například animovaná postavička může vypadat, že chce skočit z jednoho místa na druhé…“ Dále je zmíněno, že mezi těmito dvěma body jsou jednotlivé snímky dokreslovány *(Oficiální web Adobe, 2024)*.

Z této informace a ze samotného fungování a projevu LeanTween je patrné, že knihovna na rozdíl od Unity animací nečeká v daném stavu, nýbrž pouze zadá dva body pro libovolný objekt, mezi kterými má proběhnout požadovaná animace, kterou vykoná. Tímto šetří výkon a je vhodná pro drobné animace, jako jsou přechody. Současně má podporu většiny platforem.

3.6 Úprava struktury souborů

Tato podkapitola se zaměřuje na úvahu nad strukturováním a rozvržením souborů aplikace, jak v podobě projektu, tak v podobě programu. Správná struktura napomáhá přehlednosti a jasnosti, což spadá do obecně uznávaného pohledu na počítačovou gramotnost. Současně by struktura měla být v rámci aplikace srozumitelná a pokud možno co nejpřívětivější pro uživatele.

3.6.1 Uspořádání vlastních souborů uživatele

Vlastní soubory byly již zmíněny v předešlých kapitolách. Skládají se z dat, která uživatel chce sám do aplikace vkládat. Tato funkce je vhodná jak pro vyučující, kteří by chtěli svým žákům sdílet kompletní sadu materiálů i se zvukovými nahrávkami, tak je vhodná i pro jedince, který si chce sám přidat či upravit otázky na dané téma. Pro lepší představu o struktuře a fungování zpracování těchto dat budou v souborech aplikace přiloženy prvotní příklady, podle kterých lze snadno vytvořit vlastní okruhy a jejich otázky. Aplikace data přijímá a dekóduje pouze v podobě neformátovaného textu a zvuku, takže je tímto mimo jiné i předcházeno případnému zneužití spouštěním externích scriptů přes aplikaci.

Složka s uživatelskými daty je pojmenována „_UserData“ a je umístěna ve složce „Blindfold_Data“, která obsahuje hlavní soubory aplikace, a do které snadno a efektivně odkazuje cesta, popsaná v kapitole 3.2.3. Jak již bylo zmíněno – každá složka uvnitř uživatelské složky je jedním okruhem. Její název je také názvem okruhu, který se v aplikaci zobrazí. Jednotlivé otázky okruhu jsou umístěny v příslušných složkách společně s případným zvukovým souborem pro přečtení jména okruhu, pojmenovaným „topicName.mp3“. Jméno

okruhu je jednorázově vygenerováno při načítání okruhů v aplikaci, pokud už tento soubor neexistuje. V případě změny jména okruhu je tedy potřeba vymazat a znovu nechat vygenerovat zvukový soubor. Oproti tomu jména otázek ale nejsou nijak podstatná pro chod aplikace. Aplikace je pouze čte a pamatuje si je pro případy, kdy by bylo potřeba znovu přečíst data z konkrétní složky. Jména tedy mohou být libovolná, avšak při jejich změně za průběhu kvízu by mohlo dojít k chybě.

Každá složka otázky musí vždy obsahovat textový soubor, který obsahuje text otázky a až tři možné odpovědi, a je pojmenován „data.txt“. Je formátovaný tak, aby jej chápal kterýkoli uživatel: každý řádek reprezentuje input, kdy aplikace detekuje řádek s textem „otázka“ nebo „question“, který je převáděn do malých písmen, aby byla eliminována chyba v případě opomenutí správného formátu. Na tento řádek ihned navazuje text otázky. V případě opakovaného výskytu řádku „otázka“ je tento řádek přeskočen. Aplikace poté hledá řádek „odpovědi“ nebo „answers“, na nějž by měly ihned navazovat tři řádky s odpověďmi, kdy první z odpovědí je považována za správnou. Další soubory ve složce otázky nemusí být, jako v případě zvukového souboru s názvem okruhu, přítomny a jsou v případě potřeby vygenerovány za běhu aplikace. Nicméně je lze opět přiložit pro plynulejší chod a pro ušetření času. Tyto soubory jsou poté pojmenovány „question.mp3“, „correctAnswer.mp3“ a „incorrectAnswer_1 (potažmo 2).mp3“.

3.6.2 Uspořádání souborů projektu

Ačkoli na samotné struktuře souborů projektu funkčně nezáleží, jsou pro přehlednost kvůli případnému pokračování ve vývoji nebo v případě předání zdrojových souborů jinému vývojáři organizovány podle účelu. Hlavní výjimkou je opět složka „_UserData“, která je přítomna i v projektu a chová se identicky se složkou uživatelských dat aplikace. Data těchto složek tedy lze navzájem zaměňovat a přenášet. Důvodem může být testování, anebo psaní otázek přímo v projektu a jejich následné přenášení do hotového programu.

Mimo složku s uživatelskými daty jsou ve složce „Assets“ roztřízeny soubory do složek s animacemi (Animations), knihovnami a pluginy (DLL Plugins), AI soubory (OpenAI), balíčky (Packages), předlohami (Prefabs), Unity scénami (Scenes), skripty (Scripts), vestavěnými zvukovými nahrávkami (SFX) a obrázky a sprity (Textures). Tato nebo obdobná struktura bývá v komunitě Unity vývojářů velmi populární a lze si jí všimnout i u zdokumentovaných projektů, zpravidla na streamovacích platformách. Proto lze předpokládat, že by se jiný uživatel, který by k projektu získal přístup, v souborech okamžitě orientoval.

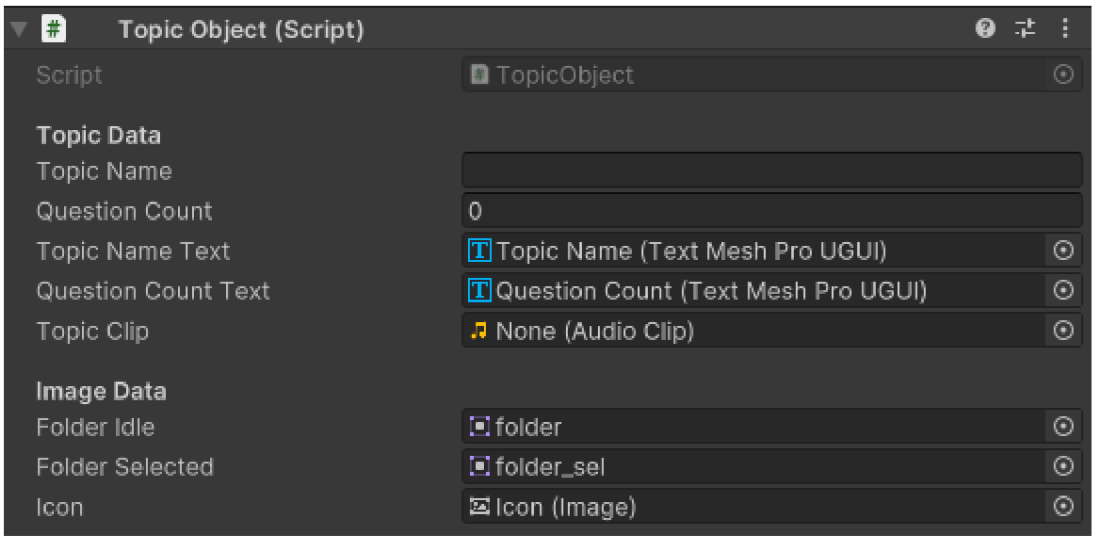
Složka s obrázky také obsahuje soubory grafického programu, ve kterém byly některé textury a sprity vytvářeny pro případ úpravy, které jsou ve formátu .kra open-source aplikace Krita.

3.7 Plánování a struktura kódu

Ačkoli existují obecně uznávaná pravidla pro formátování kódu v různých programovacích jazycích, zpravidla si každý tvůrce vytváří svůj osobní, vyhovující formát kódu. V této kapitole bude nastíněna hlavní myšlenka za organizací kódu a jeho strukturou s případným odůvodněním, ačkoli nemusí kód vždy odpovídat nejefektivnějšímu nebo nejstručnějšímu zápisu.

3.7.1 Vlastní typy a objekty

C# umožňuje tvorbu vlastních programovatelných objektů, respektive vlastních serializovatelných tříd. Unity na tuto vlastnost navazuje a rozšiřuje ji o definici „MonoBehviour“ pro třídy, která umožňuje propojení s dalšími funkcemi a nabízí také přiřazení těchto tříd objektům uvnitř editoru. Tímto způsobem můžeme vytvářet programovatelné objekty s požadovanými vlastnostmi. Ačkoli třídy s definicí MonoBehaviour nemusí být vždy žádoucí, pokud jde pouze o jakési upřesnění či vytvoření vlastního typu, je velmi efektivní pro vytváření předloh. V rámci aplikace byla tato funkce využita pro vytvoření předlohy pro objekt okruhu otázek nebo pro nastavení indikátorů.



Obrázek 18: Přiřazení scriptu objektu

V případě, kdy bylo potřeba vytvořit novou serializovatelnou, byla vždy vepsána do scriptu, který obsahem odpovídal požadované třídě. Příkladem – script pro správu otázek obsahoval třídu, ve které byla obsažena jednotlivá data, která každá otázka musí mít.

```
[System.Serializable]
public class QuestionData
{
    public string topic;
    public string tag;

    public string question;
    public AudioClip questionClip;
    public string correctAnswer;
    public AudioClip correctAnswerClip;
    public string incorrectAnswer_1;
    public AudioClip incorrectAnswerClip_1;
    public string incorrectAnswer_2;
    public AudioClip incorrectAnswerClip_2;
}
```

Obrázek 19: Příklad serializovatelné třídy

3.7.2 Umístění a využití proměnných

Ačkoli je možné zapisovat a číst proměnné prakticky z kterékoli části kódu, kvůli přehlednosti jsou umístěny vždy na začátku scriptu. Veřejné proměnné se současně zobrazují v editoru Unity, a proto jsou pro přehlednost také setřízeny do skupin podle jejich využití, případně označeny pomocí „Header(<string>)“, což v editoru umožní přidání nadpisu. Pořadí proměnných je většinou následující: nejprve statické proměnné dostupné napříč scripty a soukromé proměnné, následně veřejné či viditelné soukromé proměnné, které se zobrazí v editoru, a nakonec seznamy a pole. V případě nutnosti se může pořadí lišit, příkladem mohou být odkazy na ostatní scripty, které bývají umístěny jak mezi soukromými proměnnými, tak na úplném konci seznamu proměnných.

```
public GameObject topicPrefab;
public GameObject topicPanel;
public ScrollRect topicScroll;
public List<QuestionData> questionData = new List<QuestionData>();
public List<TopicObject> topics;

[Header("Text fields")]
public TextMeshProUGUI question;
public TextMeshProUGUI answer1;
public TextMeshProUGUI answer2;
public TextMeshProUGUI answer3;
```

Obrázek 20: Příklad seřazení proměnných

3.7.3 Struktura funkcí a metod

Opět v principu není umístění funkcí a metod nijak důležité. Nicméně je pro lepší orientaci a také znovu kvůli návyku vývojářské komunity Unity dodrženo konkrétní umístění a pořadí některých z nich: metody Awake() (provádí se před načtením celé aplikace), Start() (provádí se ihned po spuštění, ale až po metodě Awake()) a Update() (aktualizuje se každý snímek) jsou umístěny přesně v tomto pořadí vždy na začátku scriptu, protože jsou z mého pohledu považovány za klíčové a měly by být co nejdříve dostupné.

Následně až navazují vlastní metody a funkce, které jsou vždy uspořádány v pořadí, které alespoň teoreticky odpovídá jejich logickému pořadí využití. Příkladem může být: nejprve metoda pro nalezení indikátoru (AddIndicator()), poté metoda pro odstranění (RemoveIndicator()), následně metoda aktivace indikátoru, na kterou navazuje metoda deaktivace všech indikátorů. Obdobně pro všechny ostatní metody.

```
private void Awake()...

void Start()...

public void AddIndicator(IndicatorObject newIndicator)...

public void RemoveIndicators(string tag)...

public void EnableFirstIndicator(string tag)...

public void DisableAllIndicators()...

public void SelectNextIndicator()...

public void SelectPreviousIndicator()...
```

Obrázek 21: Příklad struktury vlastních metod

4 Dokončení, export a testování aplikace

S navržením vizuální hlavní podoby a určením organizace, struktury a provedením kódu, souborů a zbylých částí aplikace přecházíme k samotnému propojení jednotlivých částí, nastavení objektů, nastavení parametrů a omezení aplikace, jako je grafika, rozlišení a podobně, a celkovému zakončení a exportu práce. V této kapitole autor popíše hlavní úskalí, na která v průběhu práce narazil, možná řešení, výsledky testování aplikace autorem a několika vybranými uživateli a dodatečné poznatky, postřehy, omezení a fakta, se kterými se setkal společně s testery, kterými byly dvě slečny. První trpí vrozenou poruchou barvocitu a druhá je povoláním testerka aplikací. Byly tedy vhodnými kandidáty pro testing.

4.1 Povaha aplikace

Aplikace funguje na jednoduchém zadávání vstupů pomocí klávesnice či myši. Po spuštění uživatele uvítá zvuková nahrávka, která mu popíše základní ovládání. Tuto nahrávku lze přeskočit libovolnou interakcí, ať už pohybem myší nad objektem či zmáčknutím šipky/W/S. Pokud není předčítání vypnuto, předčítá automaticky veškerý text, který je obsažen v nebo souvisí s jednotlivými objekty. Pomocí mezerníku, entru či myši lze potvrdit výběr. V módu myši stačí jedno kliknutí na objekt, v případě použití klávesnice jsou potřeba dvě zmáčknutí klávesy, kde první značí výběr a je doprovázeno zvukovým efektem, a druhé zmáčknutí značí potvrzení výběru.

Při výběru náhodného tématu aplikace zobrazí načítací obrazovku a oznámí, že načítá data. V tomto čase vybere náhodný okruh a z něj vylosuje otázku. Otázky se mohou opakovat. Následně nás přenese do scény kvízu, popsané v kapitole 3.4. V případě zvolení výběru témat se opět zobrazí načítací obrazovka a aplikace v průběhu načte veškeré okruhy. V obou případech během zobrazení načítací obrazovky aplikace ověřuje dostupnost zvukových nahrávek a v případě potřeby ověří dostupnost spojení mezi serverem a poté nahrávky zpracuje. Během toho je doba trvání načítací obrazovky prodlouženo, dokud nejsou zpracovány všechny soubory.

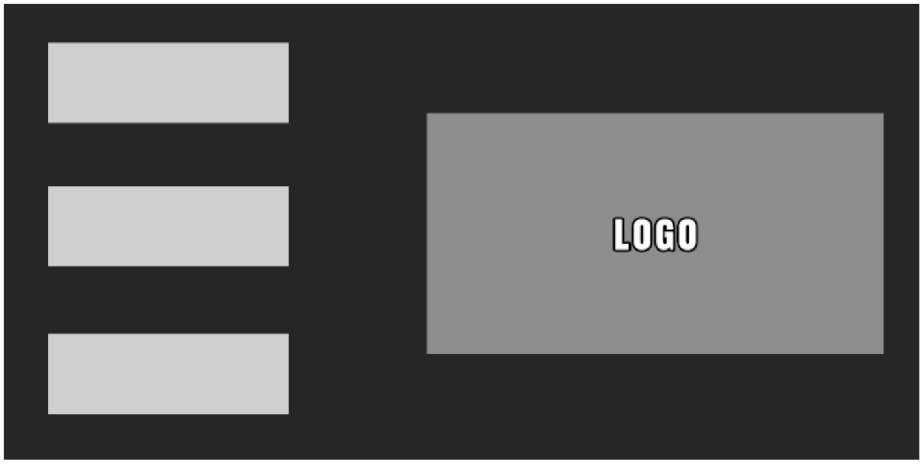
Během kvízu je vždy zobrazena otázka se třemi odpověďmi, jejichž pořadí je náhodně určeno. Pokud uživatel nerozuměl otázce, může ji nechat znovu přehrát pomocí klávesy tabulátor.

4.2 Pořadí a organizace práce

Tato krátká podkapitola reflektuje pořadí jednotlivých úkonů, které bylo pro dokončení uskutečnit. Hlavním počátečním krokem bylo samotné grafické rozložení.

4.2.1 Grafické rozvržení

Umístění jednotlivých tlačítek a objektů napomohlo setřídit priority a logickou posloupnost navazujících úkonů. Pomocí jednobarevných obdélníků bylo navrženo menu a scéna kvízu. Ačkoli bývá vhodnější v podobných projektech zajišťovat grafickou podobu až mezi posledními kroky, byla veškerá grafická stránka pro okamžitou přesnost a konkrétnost prokonzultována prostřednictvím kolegyně ze speciálně pedagogického oboru s paní prof. Ludíkovou, která je v tomto oboru expertkou, a také se zaměstnanci brněnského tyflopedického centra. Společně s poznatky, uvedenými v teoretické části, byla grafika průběžně přiřazována jednotlivým objektům ihned na počátku a v průběhu práce.



Obrázek 22: Prvotní rozložení aplikace

4.2.2 Přechody a interakce s objekty

Na grafickou stránku navazovalo nastavení funkčnosti jednotlivých objektů. Ačkoli kompletní funkčnost scén a veškerých funkcí zatím nebyla k dispozici, kvůli okamžitému a snadnému testování byla každému z tlačítek v menu přiřazena akce, kterou má provést, současně s jednoduchou animací pro aktivaci či deaktivaci jednotlivých objektů. Tento postup zajistil, že nebylo nutné ručně aktivovat jednotlivé scény v případě nutnosti, anebo umožnil nastavení testování odezvy každého tlačítka v případě, že by došlo k chybě, kterou by bylo potřeba debutovat.

4.2.3 Nastavení umělé inteligence

V průběhu práce na ostatních prvcích bylo také nutné vyhledávat a zkoušet dříve zmíněné modely AI. Okamžité upřednostnění vyhledávání vhodného modelu se ve výsledku projevilo velmi pozitivně, neboť zajistilo nejprve řádné otestování a implementaci před pokračováním v dokončení zbývajících kroků. V případě, že by se nepodařilo najít vhodný model, bylo by nutné zaujmout alternativní řešení pro funkčnost, jako je osobní namluvení potřebných textů. Během konzultací ohledně grafické podoby byly autorovi také poskytnuty odkazy ke stažení a otestování některých volně dostupných aplikací pro čtení textu, užívaných v brněnském tyflocentru. Jejich testování a následní zjištění nekompatibility s Unity aplikacemi byly nakonec důvodem k využití umělé inteligence.

4.2.4 Programování zbývajících funkcí

V rámci nastavování umělé inteligence již také probíhala práce na dalších částech aplikace, které byly v danou dobu potřebné pro testování a následně i pro funkčnost. Nejprve proběhlo propojení serveru AI s aplikací přes kód, který následně data ukládal. V návaznosti na to bylo nutné dokončit kód pro čtení těchto dat, s ním vznikl i kód pro správu samotných otázek, neboť umělé inteligenci bylo nutné poskytnout data. V této části došlo k rozhodnutí vynechat možnost výběru hlasu a rychlost řeči.

Posledním krokem byla úprava některých funkcionalit, hlavně na základě zpětné vazby od testerů. Proběhla oprava chyb v kódu a úprava částí, které nebyly vhodné. Současně bylo nastaveno pevné rozlišení aplikace v podobě okna 1280×720 pixelů, byla zablokována změna rozlišení a kód byl doplněn o část, která při spuštění nastaví rozlišení dodatečně, kdyby došlo k chybě, která by rozlišení nějakým způsobem ovlivnila. Důvodem k nastavení pevného rozlišení je případný export pro jiné platformy. Převážně u mobilních zařízení s jiným poměrem rozlišení, než 16:9, se autor setkal s nevhodným posunutím aplikace. Nastavení pevného rozlišení zajistí umístění aplikace na střed zařízení a předejde tak například situaci, kdy by obrazu bránil přední fotoaparát zařízení. Také proběhlo nastavení aplikace pomocí možností, které Unity nabízí. Jedná se o ikonu, název, podpis, kvalita grafiky nebo také úprava detekovaného vstupu klávesnice.

4.3 Komplikace a nedostatky při vývoji a používání

Již byly zmíněny úpravy některých funkcionalit a eliminace nežádoucích bugů či nedostatků a nepřesností. Během vývoje se však vyskytly další problémy, které bylo potřeba adresovat nebo s nimi počítat. Zde jsou uvedeny některé z nich.

4.3.1 Limity umělé inteligence

Ačkoli je umělá inteligence velmi silným nástrojem, její funkce jsou místy stále nedokonalé, anebo limitované. Jedním z hlavních problémů při výběru modelů byl již zmíněný český jazyk. Ten, ačkoli je mnoho modelů schopných jej přečíst, je stále velice málo podporovaným jazykem. Tento parametr eliminoval jednoho z kandidátů AI a zanechal pouze modely od společnosti Google a OpenAI, nicméně byl model Google později vyřazen kvůli zmíněné nutnosti autentifikační aplikace. Jejich hlavní nevýhodou je omezení počtu web requestů, které mohou zpracovat. U zbývajícího modelu OpenAI je to aktuálně padesát (50) requestů za minutu pro obyčejný model a pouze tři (3) za minutu pro HD model. Jelikož každá otázka aplikace potřebuje až čtyři zvukové nahrávky, limituje toto omezení počet možných zpracovaných otázek na dvanáct (12) za minutu. Tomuto je ale možné aplikaci přizpůsobit tak, aby mezery mezi jednotlivými requesty vyhovovaly tomuto omezení.

Dalším z limitů je schopnost AI číst žádaný text. Během testování jsme se s testery setkali s problémy čtení krátkých textových úryvků. Problémová byla současně krátká slova na začátku otázek či odpovědí. Jeden ze způsobů jak tento nedostatek co nejvíce eliminovat bylo přidání dodatečných znaků, které AI nebude číst, ať už na začátek či na konec. Těmito znaky mohou být pomlčky či tečky, ačkoli při používání pomlček dělala AI v mluvě automatické pomlky, což může být způsobeno jejím trénováním, a hlavní využití slouží k oddělení čteného slova. Nicméně komolení či přímá degenerace slov místy přetrvává i v náhodných intervalech. Toto může být způsobeno zatížením serveru, na kterém je AI umístěno, avšak přímá souvislost při testování nebyla zjištěna.

4.3.2 Problémové detekce kurzoru

Během testování se také objevily situace, kdy může pozice kurzoru myši na obrazovce ovlivnit předčítání textu. Text je automaticky spouštěn při interakci s objektem, avšak v případě, že je myš na daném objektu přítomna ve chvíli, kdy se na obrazovce objeví, může se stát, že se nepřečte systémem určený výchozí text, ale místo něj text, umístěný pod kurzorem myši. Tento, ač drobný a funkčnost aplikace neovlivňující, nedostatek se, bohužel, nepodařilo efektivně

eliminovat tak, aby řešení neovlivnilo zbývající fungování. Jediným řešením je dobrovolnost uživatele odstranit kurzor z okna aplikace či z umístění objektu v době, kdy dochází k přechodům mezi scénami či ke čtení otázek.

4.3.3 Falešně pozitivní upozornění antiviru

Kvůli povaze aplikace číst soubory a ukládat do nich data může tato její funkce vyvolat reakci antiviru na zařízení uživatele. Tento problém byl během testování nepravidelný, kdy se na některých zařízeních reakce antiviru objevila a na jiných ne. Způsobem zabránění tomuto efektu by byla registrace nebo širší rozšíření aplikace tak, aby byla umístěna do seznamu důvěryhodných aplikací v databázích antivirů či systémové ochrany (v případě Windows). V aktuální situaci není ani jedno možné a proto je nutné, aby tento problém uživatel vyřešil sám povolením aplikace. Jakékoliv omezení či zabránění některých funkcí antivirem by mělo za následek neschopnost aplikace číst soubory a tedy plnit svou funkci.

4.4 Možnosti pokračování práce

Výsledná aplikace nabízí pouze základní funkci opakování a je navržena ve velmi jednoduchém duchu. Je zde však prostor pro případné rozšíření nebo prohloubení práce, kterým mohou být dodatečné funkce, bližší rozlišení okruhů na menší podokruhy, využití efektivnějšího modelu AI nebo vytvoření vlastního modelu.

Současně je aktuální podoba aplikace určena pouze pro použití na počítačových systémech, byla však testována a je funkční i na dalších platformách, primárně na platformě Android. Nicméně tomu tak je pouze u verze pro osoby vidomé. Pro tyto další platformy není aplikace ještě zcela přizpůsobena a bylo by nutné její funkce upravit i pro osoby nevidomé.

Další z možných úprav či vylepšení je vytvoření pomocné aplikace či scény ke generování otázek, aniž by je musel uživatel vytvářet pomocí souborů.

Mimo to je zde i prostor k rozšíření načítací obrazovky, která by mohla obsahovat informaci o aktuálním stavu načítání.

Závěr

Cílem práce bylo vytvořit aplikaci v prostředí enginu Unity, která je zaměřena na částečně či úplně zrakově postižené. Její využití je určeno ke vzdělávání v podobě kvízu k opakování učiva. Autor stanovené cíle splnil a zdokumentoval veškerý postup, který k jejich splnění vedl.

Jednotlivé části práce popisovaly problematiku zrakových postižení, problematiku vývoje aplikací a také postup při samotném vývoji aplikace, která po dokončení obdržela název Blindfold. Práce také rozebírá různá úskalí či nedostatky, se kterými se autor setkal při vývoji nebo během testování.

Aplikace byla primárně otestována autorem, slečnou, trpící poruchou barvocitu, slečnou, pracující jako testerka aplikací a byla představena, včetně poskytnutí k vyzkoušení, několika kolegům autora a všedním uživatelům z řad rodičů, žáků či studentů, z nichž někteří trpí vrozenou degenerací zraku nebo ochabnutím zraku v závislosti na jejich věku.

Na základě zpětné vazby byly koncept i samotné provedení a jednoduchost aplikace hodnoceny pozitivně s návrhy pro zdokonalení a rozšíření. Aplikace však nebyla poskytnuta širší veřejnosti.

Zdroje

Internetové zdroje a publikace

1. CARBONNELLE, Pierre, 2024. *PYPL PopularitY of Programming Language. Pypl.github* [online]. Dostupné z: https://pypl.github.io/PYPL.html. [cit. 2024-06-14].
2. CONTE, Niccolo, 2024. *Ranked: The Most Popular AI Tools. Visual Capitalist* [online]. Dostupné z: https://www.visualcapitalist.com/ranked-the-most-popular-ai-tools/. [cit. 2024-06-14].
3. PAVLÍČEK, Radek, 2018. *Kolik je v České republice zrakově postižených lidí? Poslepu.cz* [online]. Dostupné z: https://poslepu.cz/kolik-je-v-ceske-republice-zrakove-postizenych-lidi/. [cit. 2024-06-14].
4. POSPÍŠILOVÁ, Magdalena. [b. d.]. *Definice, dělení (slabozrakost a slepota). Fyzioterapie* [online]. Dostupné z: https://fyzioterapie.utvs.cvut.cz/document/show/id/30/. [cit. 2024-06-14].
5. PROCHÁZKA, Matěj, [b. d.]. *IDE. Peach-dev* [online]. Dostupné z: https://peach-dev.cz/wiki/ide/. [cit. 2024-06-14]
6. STEJSKALOVÁ, Kateřina, 2023. *Vzdělávání dětí se zrakovým postižením. Šance dětem* [online]. Dostupné z: https://sancedetem.cz/vzdelavani-deti-se-zrakovym-postizenim. [cit. 2024-06-14].
7. STŘELEC, Michal, [b. d.]. *Co je to IDE - vývojové prostředí? Michal Střelec* [online]. Dostupné z: https://www.strelec.pro/slovnik-vyvojare/co-je-to/ide-vyvojove-prostredi. [cit. 2024-06-14].
8. WAGNER, Bill, [b. d.] *Operátory a výrazy jazyka C# (referenční dokumentace jazyka C#). Microsoft* [online]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/operators/. [cit. 2024-06-14].
9. WINTER, Brian, 2017. *TextMesh Pro Joins Unity. Unity* [online]. Dostupné z: https://blog.unity.com/games/textmesh-pro-joins-unity. [cit. 2024-06-14].
10. *Creating animated action with tweening,* [b.d.]. *Adobe.* [online]. Dostupné z: https://www.adobe.com/creativecloud/video/discover/tweening.html. [cit. 2024-06-14].
11. *Gaming Engines,* [b. d.]. *ARM* [online]. Dostupné z: https://www.arm.com/glossary/gaming-engines. [cit. 2024-06-14].

12. *Global Game Jam, Inc.*, 2024. [online]. Dostupné z: https://globalgamejam.org/. [cit. 2024-06-14].
13. *Programming in Unity*, [b. d.]. *Unity Technologies* [online]. Dostupné z: https://unity.com/solutions/programming. [cit. 2024-06-14].
14. *Programovací jazyky*, 2024. *Stránka k výuce informatiky* [online]. Dostupné z: https://ivt.mzf.cz/seminar/14-programovaci-jazyky/. [cit. 2024-06-14].
15. *The Unity Runtime Fee,* 2023. *Unity Technologies* [online]. Dostupné z: https://unity.com/products/pricing-updates. [cit. 2024-06-14].
16. *The world's most widely adopted AI developer tool,* [b. d.]. *GitHub* [online]. Dostupné z: https://github.com/features/copilot. [cit. 2024-06-14].
17. *What's new in Unreal Engine 5.3,* 2023. *Epic Games* [online]. Dostupné z: https://www.unrealengine.com/en-US/updates. [cit. 2024-06-14].
18. *2021 Gaming Report,* 2021. *Unity Technologies* [online]. Dostupné z: https://create.unity.com/2021-game-report. [cit. 2024-06-14].

Knižní zdroje

1. BENEŠ, Pavel, 2019. *Zraková postižení: behaviorální přístupy při edukaci s pomůckami*. Praha: Grada. Pedagogika (Grada). ISBN 978-80-271-2110-6.
2. BORY, Pavel, 2022. *C# bez předchozích znalostí*. 2. vydání. V Brně: Computer Press. ISBN 978-80-251-5061-0.
3. RŮŽIČKOVÁ, Veronika, 2006. *Integrace zrakově postiženého žáka do základní školy*. Olomouc: Univerzita Palackého v Olomouci. ISBN 80-244-1540-2.

Jiné zdroje

ALVIN, Olle, 2020. *Rendering Resolution Independent Fonts in Games and 3D Applications*. Magisterská práce. Lund University. Dostupné z: https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9024910&fileOId=9024911. [cit. 2024-06-14].

Seznam obrázků