Návrh a implementace systému umožňujícího vývoj her pro nevidomé a slabozraké

Michal Hudák

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Michal Hudák**
Osobní číslo:	**A20007**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Návrh a implementace systému umožňujícího vývoj her pro nevidomé a slabozraké**
Téma práce anglicky:	**Design and Implementation of Game Development System for Visually Impaired People**

Zásady pro vypracování

1. Vypracujte literární rešerši na zadané téma.
2. Navrhněte systém pro tvorbu her, které budou určeny pro nevidomé a slabozraké.
3. Realizujte navržený systém.
4. Navrhněte a implementujte krátkou demonstrační hru vytvořenou v tomto systému.
5. Zpracujte dokumentaci k vytvořenému systému a iplementované hře.

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

Seznam doporučené literatury:

1. Jan JIRKOVSKÝ a kolektiv. Game industry: vývoj počítačových her a kapitoly z herního průmyslu. Praha, 2011. ISBN 978-80-904387-1-2.
2. Unity User Manual 2021.3 (LTS) [online]. San Francisco, USA: Unity Technologies, 2022 [cit. 2022-11-28]. Dostupné z: https://docs.unity3d.com/Manual/index.html.
3. HOLAN, Tomáš. Unity: první seznámení s tvorbou počítačových her. Praha: CZ.NIC, z. s. p. o., 2020, 1 online zdroj (176 stran). CZ.NIC. ISBN 978-80-88168-60-7. Dostupné také z: https://knihy.nic.cz/files/edice/Unity_prvni_seznameni_s_tvorbou_pocitacovych_her.pdf.
4. ŘEHÁČEK, Denis. Development of games for blind users. Prague, 2017. Bachelor Project. Czech Technical University in Prague. Vedoucí práce Doc. Ing. Daniel Novák, Ph.D.
5. NETO, Luiz Valério, Paulo H.F. FONTOURA JUNIOR, Rogério A. BORDINI, Joice L. OTSUKA a Delano M. BEDER. Design and implementation of an educational game considering issues for visually impaired people inclusion. Smart Learning Environments [online]. 2020, 7(1) [cit. 2022-11-28]. ISSN 2196-7091. Dostupné z: doi:10.1186/s40561-019-0103-4.
6. MARGOLIS, Michael. Arduino cookbook. Second edition. Sebastopol: O'Reilly, 2011, xx, 699 s. ISBN 9781449313876.

Vedoucí bakalářské práce: **Ing. Tomáš Vogeltanz, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

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 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že 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;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že 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 samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 25.5.2023 Michal Hudák, v.r.

ABSTRAKT

Bakalárska práca sa venuje vývoju softvéru, v ktorom môžu byť vytvárané počítačové hry pre nevidiacich a slabozrakých ľudí. V teoretickej časti je popísaná situácia týchto ľudí a ich vzťah k počítačom a hrám. Sú v nej tiež predstavené technológie vývoja aplikácie. V praktickej časti je popísaný podrobný vývoj a štruktúra aplikácie, čo môže slúžiť aj ako jej dokumentácia. Ako príloha k práci je demonštračná hra vytvorená v tomto systéme.

Kľúčové slová: nevidiaci, slabozrakí, hra, aplikácia

ABSTRACT

The bachelor's thesis is devoted to the development of software in which can be created computer games for blind and partially sighted people. The theoretical part describes the situation of these people and their relationship to computers and games. It also introduces application development technologies. The practical part describes the detailed development and structure of the application, which can also serve as its documentation. As an attachment to the work, there is a demonstration game created in this system.

Keywords: blind people, visually impaired people, game, application

Moja vďaka patrí vedúcemu tejto bakalárskej práce, pánovi Ing. Tomášovi Vogeltanzovi, Ph.D., ktorému som prvotný nápad práce predstavil už v prvom ročníku a už vtedy som dúfal, že túto rolu prijme.

Ďakujem aj zamestnankyni Únie nevidiacich a slabozrakých slovenska, pani Riečickej a taktiež Eme, nevidiacej študentke za podporu, rady a možnosť poznania ich situácie.

Za pomoc pri tvorbe demonštračnej hry (scenár a dabing) ďakujem Matúšovi, Tadeášovi a kolektívu z TRUNI.

Za zapožičanie 3D tlačiarne a 3D skenera, tiež za konzultácie ohľadom hardwarovej časti vyjadrujem vďaku Romanovi, a Ing. Rastislavovi Hollému.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

ÚVOD

Od ich vstupu do popkultúry sa počítačové hry stali neoddeliteľnou súčasťou života mnohých ľudí. Príbehy, kompetetívnosť a komunity fanúšikov sú len niektoré z prvkov, ktoré formovali celosvetovú popularitu videohier. Tieto hry sú však často navrhnuté tak, aby predpokladali, že hráč vidí obrazovku, čo sa môže zdať samozrejmé, no v skutočnosti to znamená, že z tohto sveta je vylúčená istá skupina ľudí: nevidiaci.

Podľa štúdií, žije na svete približne 295 miliónov ľudí so zrakovým postihnutím, z ktorých je 43 miliónov úplne nevidiacich. Táto populácia nemá taký prístup k počítačovým hrám ako ľudia bez zrakového postihnutia a pri vývoji počítačových hier je často prehliadaná.

Bakalárska práca prináša softvér, v ktorom takéto hry môžu vznikať jednoduchým, lacným a rýchlym spôsobom. V teoretickej časti sa zameriava aj na analýzu súčasnej situácie v oblasti počítačových hier pre nevidiacich.

Je potrebné zvážiť, ako vytvoriť otvorenejšie herné prostredie. Ako je možné prekonať prekážky, ktoré bránia nevidiacim hrať videohry. Ako môžeme vytvoriť hry, ktoré budú rovnako zábavné aj pre ľudí, ktorí nemajú schopnosť vidieť. Tieto otázky predstavujú základné výzvy, ktorým sa táto práca venuje.

Nevidiaci ľudia majú obmedzený prístup do herného sveta. Táto práca môže pomôcť k tomu, aby bol herný svet naozaj pre každého.

Práca sleduje štyri ciele, v tomto poradí podľa priority:

1) Aplikácia na vývoj hier pre nevidiacich
    a) Samotná aplikácia
    b) Šablóna hry
    c) Dokumentácia a literárna rešerš
2) Demonštračná hra vytvorená v tejto aplikácii
3) Ovládač na ovládanie týchto hier

Okrem tohto produktu s praktickým využitím práca predstaví aj nevšednú techniku interakcie naprieč technológiami, menovite generovanie a následné spúšťanie kódu jazyka Python pomocou C# a .NET.

I. TEORETICKÁ ČASŤ

1 PROBLEMATIKA NEVIDIACICH A SLABOZRAKÝCH

Niektorí ľudia s vážnou poruchou zraku majú schopnosť čiastočne vidieť a vďaka tomu môžu používať počítač a vidieť obsah na obrazovke pomocou lupy (vizuálneho zväčšovača). Ale úplne nevidiaci, alebo ľudia s najvážnejšími poruchami zraku sa spoliehajú na asistenčné technológie, aby mohli interagovať s počítačom. Buď môžu používať Braillov displej, ktorý dynamicky mení text na Braillovo písmo, alebo môžu použiť zvukových asistentov, ktorí nahlas čítajú text na obrazovke [1][2]. Existujú rôzne asistenčné programy, ale väčšina moderných webových prehliadačov už v sebe takýto modul má.

1.1 Zásady jasnej tlače

Či už tlačený alebo digitálny text, určený slabozrakým ľuďom by mal spĺňať zásady jasnej tlače, aby taký text mohli čítať s čo najmenšími komplikáciami. Zoznam dôležitých zásad:

a) Písmo musí mať vhodnú veľkosť

b) Písmo musí mať vhodný font (bezpätkový)

c) Musí byť vysoký kontrast medzi farbou pozadia a farbou písma

d) Písmená, slová a riadky musia mať medzi sebou vhodný rozostup a riadkovanie [3]

1.2 Nevidiaci a počítačových hry

Za uplynulú dobu sa digitálne hry stali populárnymi vďaka technologickému pokroku a zvyšujúcemu sa prístupu k technológiám pre ľudí. Pre aktuálnu dobu to platí ešte viac, keďže ľudia sa už rodia do digitálneho sveta [4].

Čísla na úvod: Na svete je približne 3,24 miliardy hráčov počítačových hier [5]. Na svete je taktiež približne 43 miliónov úplne nevidiacich ľudí z 295 miliónov ľudí so stredne ťažkou alebo ťažkou poruchou zraku [6]. Čo je dokopy 10,43% z počtu hráčov a 4,2% z celkového počtu ľudí na svete (v prvej polovici roku 2023 bol počet ľudí na svete 8,046 miliardy [7]). To nie je finančne veľmi lukratívna cieľová skupina, dôvody sú napísané v nasledujúcich odstavcoch:

Podľa knihy “Game industry – vývoj počítačových her a kapitoly herního průmyslu“, ktorá vyšla v roku 2011, sa herný priemysel najviac zameriava na kvalitu produktov, napriek tomu, že hlavným motivátorom obchodníka je zisk [8]. To už je mierne v rozpore s pohľadom na dnešnú dobu (2023). Spoločnosti aj kvôli väčším ziskom vydávajú hry so

zameraním na veľké masy ľudí (do popredia sa dostávajú hry na mobilné telefóny [9], robia sa ústupky a cenzúra, aby mohli byť vydané vo veľkej krajine v ktorej platia rôzne reštrikcie [10][11], hry čoraz až znepokojivo často bývajú "family friendy" – teda s ratingom under 18, či sa vkladá úsilie na ryžovanie z nostalgie ľudí a prerábajú sa aj hry (tzv. remasters alebo remakes), ktoré to nepotrebujú [12]).

Cena vývoja softvéru je vysoká. Indie hry (nezávislé hry tvorené malým tímom vývojárov) stoja priemerne od 50 000 USD (v závislosti na časti sveta) a môže trvať aj rok na dokončenie [13], pre nadšencov, ktorí hru tvoria len vo svojom voľnom čase to môže trvať aj viac. Audiohra je síce príklad jednoduchšieho softvéru a (takmer vôbec) nedisponuje grafickou stránkou, stále je nutné mať pri tvorbe hry obsadených vo vývojárskom tíme niekoľko rolí (napríklad dizajnéra hry + scenáristu, programátora, testera, producenta, audio dizajnéra + skladateľa a ďalšie - niekedy jednotlivý vývojár vykonáva viacero rolí naraz) [14].

Niekoľko hier určených pre nevidiacich a slabozrakých ľudí existuje [15]. Vo väčšine prípadov však ide o adaptáciu iných jednoduchých hier, ako je napríklad dáma, karty alebo kocky [16]. Vítaným krokom sú aktivity štúdií, ktoré do svojich AAA hier (hier s vysokým rozpočtom, tvorené veľkým tímom a s marketingovou propagáciou) určených pre klasickú cieľovú skupinu, pridávajú aj prispôsobenie hier pre hráčov so zrakovým postihnutím [17]. Avšak takéto prispôsobenie hry stojí tiež nemalé zdroje, dalo by sa povedať že je to zatiaľ len v rozkvete a takýchto projektov veľa neexistuje. Taktiež nie je možné očakávať, že si priemerný nevidiaci človek kúpi hernú konzolu alebo herný počítač, aby si na tom mohol zahrať pár jednotiek takýchto hier. Celkovo platí, že nevidiaci a slabozrakí ľudia nemajú príliš veľký výber [18]. To sa taktiež týka jazykových variácii toho, čo už existuje na to, aby si mohli zahrať aspoň niekoľko titulov za rok. Aktuálna forma virtuálnej zábavy pre nevidiacich ľudí pozostáva z audio komentára filmov a seriálov, alebo z predčítaných audiokníh [19][20]. Výstup tejto práce im prinesie podobný zážitok, ale stým, že vývoj príbehu budú mať vo svojich rukách a budú ho môcť ovplyvňovať.

1.3 Nástroje na tvorbu systému

Výber správnych nástrojov na tvorbu systému bol starostlivý proces. Parametre použitých technológii sú (samozrejme okrem toho, že musí umožniť splnenie cieľa danej časti systému):

a) Ľahká čitateľnosť a rozšíriteľnosť kódu.

b) Pohodlnosť tvorby. Do istej miery rozhodovali aj osobné preferencie autora na základe jeho skúseností s danými technológiami a ako dobre sa dajú navzájom prepojiť.

c) Kvalita oficiálnej dokumentácie, ako aj aktivita komunity používateľov technológie (napríklad diskusie na fórach, tvorba knižníc a podobne) a taktiež aktivita tvorcov technológie (v prípade open-source technológií sa do tohto zahŕňa aj spomínaná aktivita komunity). Toto udáva predpoklady o budúcnosti technológie, či bude napríklad ešte dlhodobo podporovaná alebo používaná.

d) Výstup práce (program) musí byť rozbehnuteľný aj na menej výkonných počítačových zostavách a musí mať jednoduchú inštaláciu. Použitie nástroja musí byť tiež jednoduché, aby neodradilo používateľov bez počítačových zručností.

Vývoj nástroja, ktorý umožní tvoriť audio hry sa dá robiť z dvoch pohľadov:

1) Vývoj v engine: Engine je nástroj, ktorý sa vo videohernom priemysle spája s prácou s grafikou, teda vykresľovaním 3D a 2D objektov. Obsahuje hernú slučku a nástroje na vytváranie a úpravu herného sveta pridávaním a nastavovaním herných objektov. Väčšina enginov tiež podporuje programovanie v programovacích jazykoch. V prípade prístupného Unity, je to jazyk C# [21]. Pre vývoj aplikácie na tvorbu hier pre nevidiacich je však grafika bezpredmetná, engine ale podporuje aj rozšírenú prácu so zvukom [20]. Keďže cieľom je vytvoriť nevidiacim hráčom kvalitný audio zážitok, hlavnou výhodou enginu je priestorový zvuk [22]. Unity umožňuje vyvinúť si vlastný modul a sprístupniť ho ostatným používateľom [23]. Autor práce by vyvinul modul, ktorý by si používatelia – tvorcovia hier vložili do Unity enginu na svojom počítači a ňom by hry tvorili.

2) Vývoj vlastného programu, nie enginového charakteru: Výstup programu (hra) by týmto pádom tiež nebežala na engine, muselo by ísť o samostatný program vygenerovaný aplikáciou. Vyhovujúcou voľbou je Python, vďaka jeho knižniciam na prácu so zvukom. Tvorba hier na mobilnom telefóne či tablete nepripadá do úvahy kvôli nepohodlnosti tvorby, možné sú nasledujúce voľby:

    2.1) Webová aplikácia: Išlo by o online nástroj, tvorba hier by prebiehala vo webovom prehliadači. Vyvíjaná by bola buď klasickými webovými technológiami (spolupráca jazykov HTML, CSS, JavaScript, PHP, +

frameworky ako Bootstrap, vue.js a ďalšie), alebo prácou v ASP.NET. K tomu prináleží hosting domény, databázy a iných nákladov [24].

2.2) Desktopová aplikácia: Takto vyvinutý nástroj by sa inšaloval priamo lokálne na používateľských počítačoch. Umožnilo by to offline použitie. Vyvinutý by mohol byť napríklad v jazyku C, Python, C++, Java alebo C#. Program musí disponovať grafickým používateľským rozhraním, ktoré podporuje generovanie elementov do okna a prístup k nim. Ak by bol zvolený jazyk Python, na GUI by mohol byť použitý nástroj PyQT6. V prípade C# by to mohla byť knižnica WPF (spolupráca s jazykom XAML). V tom prípade by bol zrejme zvolený .NET Framework, ktorý má svoje výhody ale aplikácia by bola uzamknutá na platforme Windows [25]. Ak by bol zvolený jazyk C++, GUI by mohlo byť vytvorené pomocou Qt, v prípade Javy by to mohol byť napríklad by to mohol byť napríklad Swing, ktorý je súčasťou klasickej JFC knižnice [26]. V tomto momente odpadá jazyk C, ktorý je jazyk nižšej úrovne a natívne nepodporuje GUI, čo by zapríčinilo náročný vývoj, i keď niektoré GUI toolkity pre C++ podporujú aj rozhranie pre C, bolo by to zbytočné a nie vhodné [27].

Z vyššie uvedených bodov (a-d), ktoré určujú podmienky voľby technológie, Unity nepĺňa bod d. Webová aplikácia bola pre tentoraz vylúčená, kvôli nákladom spojeným s hostingom. C++ ani Java nespĺňajú bod b. Python bol vylúčený, pretože nástroj PyQT nie je stavaný na jednoduché generovanie tak masívneho množstva elementov do UI a následný prístup k nim. WPF (a teda aj .NET a C#) bola v tomto prípade lepšia voľba. Autor práce tiež viac preferuje výsledne zvolenú technológiu pred Pythonom, pretože na takúto rozsiahlejšiu a štruktúrovanú prácu je pohodlnejší a prehľadnejší. Taktiež v počiatočných fázach vývoja bolo z pohľadu vývoja zmätočné, keď v Pythone vznikal softvér na tvorbu hier a aj samotná hra. Jazyk Python bol ponechaný len na hru. Všetky IDE, v ktorých sa vyvíja aplikácia majú prístup zadarmo na nekomerčné účely.

1.3.1 Ovládač

Súčasťou systému je aj hardwarový prvok – ovládač. Bol skonštruovaný špeciálne k ovládaniu týchto hier. Nevidiaci ľudia bývajú síce učení k práci s počítačom a teda aj klávesnicou, kvôli ich pohodliu bol však zostrojený ovládač, aby nemuseli premýšľať nad prstokladom a ovládanie hry bolo pre nich intuitívnejšie. Avšak ovládanie klávesnicou bude

systém stále podporovať, aby bolo možné hry hrať aj ak používateľ – hráč tento ovládač nemá. Ovládač musí mať jednoducho a jednoznačne uchopiteľný tvar, pretože hmat je jeden z najdôležitejších zmyslov nevidiacich ľudí, pomocou neho identifikujú objekty a ich správny úchop [28].

Jadro ovládača musí poháňať mikropočítač. Ten musí mať malé rozmery, aby sa do obalu ovládača zmestil. Vhodný by mohol byť plošný spoj z rady *Arduiono*, ktorý disponuje mnohými senzormi na zachytávanie okolia [29]. Nutné sú iba tlačidlá. So svojim mikrokontrolérom ATmega32U4 a rozmermi 18 x 48 milimetrov vyhovuje Arduino micro, ktorý podporuje priamo komunikáciu s počítačom USB rozhraním. Jeho cena je 27 € (632 CZK) (- s DPH) [30]. V prípade ovládača síce ide len o prototyp – v prípade hromadnej alebo sériovej výroby by bolo vhodnejšie vyvinúť si vlastný mikropočítač na mieru (kde cena za kus môže byť v jednotkách CZK), ale je vhodnejšie sa poobzerať aj po lacnejšej alternatíve. Tou je nízkorozpočtový mikropočítač *STM Blue Pill*, ktorý je možné priviesť napríklad z Číny za približne 1 € (25 CZK) aj s programátorom, ktorý simuluje originálny ST-Link V2, ktorý je nutný na prístup a úpravu zdrojového kódu z počítača [31]. Program sa píše v jazyku C, v prostredí STM32CubeIDE, ktoré priamo podporuje pripojenie mikropočítača a písanie či úpravu jeho zdrojového kódu [32].

1.3.2 Hra

Výstup softvéru (hra) je naprogramovaný v jazyku Python. Tento jazyk bol zvolený pre jednoduché písanie a čítanie kódu a je obľúbený [33]. To sú dôležité parametre s ohľadom na fakt, že systém ponúka aj svojpomocné doprogramovanie hry tvorcom, ak by im niečo v základnej sade nástrojov softvéru chýbalo. Python disponuje aj veľkým množstvom funkcionalít prostredníctvom knižníc [33]. V tomto prípade ide o jednoduchú a efektívnu manipuláciu so zvukovými súbormi, ktorá využíva dvoma prístupmi: knižnicou Pydub a knižnicou PyGame. Dôležitými parametrami pri práci so zvukom bolo okrem jeho prehrania aj dynamické nastavenie hlasitosti, nastavenie stereo kanálu alebo prehrávanie súboru v nekonečnom cykle na inom vlákne (počas prehrávania súboru sa môžu vykonávať iné výrazy, napríklad prehrávanie iného zvukového súboru súčasne). Toto knižnica Pydub v spolupráci s knižnicou PyGame umožňuje [34][35]. Titulky hry sa zobrazujú v jednoduchom GUI, prostredníctvom knižnice PySimpleGUI. Použité IDE: Visual Studio Code od Microsoftu, neskôr Pycharm od JetBrains.

1.3.3 Aplikácia na tvorbu hier

Aby mala aplikácia prívetivé používateľné rozhranie (UI), bola zvolená knižnica .NET (Frameworku) - WPF. Dokáže vytvárať rozhranie a jeho elementy pomocou značkovacieho jazyka XAML a to formou drag-and-drop (chytenie myšou názov elementu z ponuky a jeho položenie ho do okna), alebo priamo písaním kódu [36]. Aby mal tvorca hry voľnosť pri tvorbe, je mnoho elementov generovaných dynamicky. To sa dá zariadiť jednoducho cez kód jazyku C# [37]. Jazyk C# slúži aj na ostatný backend okrem obsluhy tvorby elementov. V aplikácii je použitý aj na dopísanie Python súboru a generovanie spustiteľného súboru hry z Python skriptu. Použité IDE: Visual Studio Community 2022 od Microsoftu.

1.3.4 Návrh systému

Návrh systému je realizovaný cez jazyk UML, ktorý slúži aj k vizualizácii systému prostredníctvom diagramov. Je možné v ňom vytvoriť modely požiadaviek (requirements), prípadov použítia (use cases), tried (classes), sekvenčný diagram a iné [38]. V tomto prípade bolo potrebné vypracovať funkčné požiadavky systému. Použité prostredie: Enterprise Architect od Sparx.

II. PRAKTICKÁ ČASŤ

2 OVLÁDAČ

Nevidiaci ľudia potrebujú jasne identifikovať, ako uchopiť predmet (napríklad potrebujú vedieť že určitý výklenok alebo tlačidlo istej veľkosti sa nachádza na určitej strane predmetu). Najlepší spôsob ako toto dosiahnuť, je vytvarovať ovládač priamo na ruku, aj s drážkami na uloženie prstov, aby bol úplne jednoznačný spôsob úchopu ovládača. Nie každý má rovnakú veľkosť ruky, ale aj keď bol obal ovládača vymodelovaný na mieru na ruku autora práce, všeobecne na to veľmi nezáleží (pokiaľ používateľ nemá naozaj krajnú veľkosť ruky). Ovládač je *HID* (Human Interface Device).

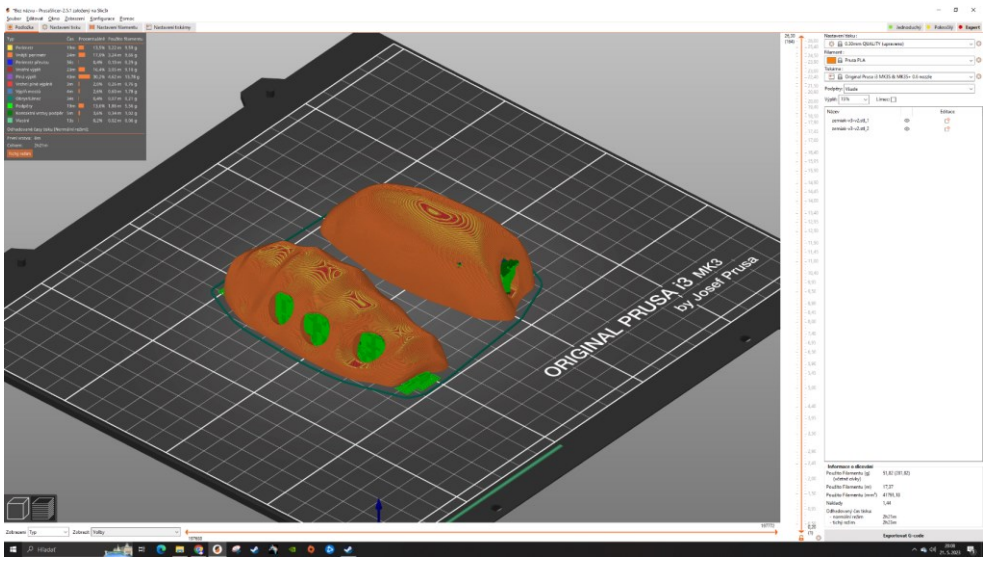
2.1 Hardvér a obal ovládača

Z toho dôvodu bola maketa ovládača vymodelovaná z klasickej umeleckej plastelíny, ktorá bola následne naskenovaná 3D skenerom (Creaform's Go!SCAN 3D). Ovládač disponuje štyrmi tlačidlami, každé na jeden z prvých štyroch prstov (palec – ukazovák – prostredník – prstenník, malíček slúži len na držanie ovládača).

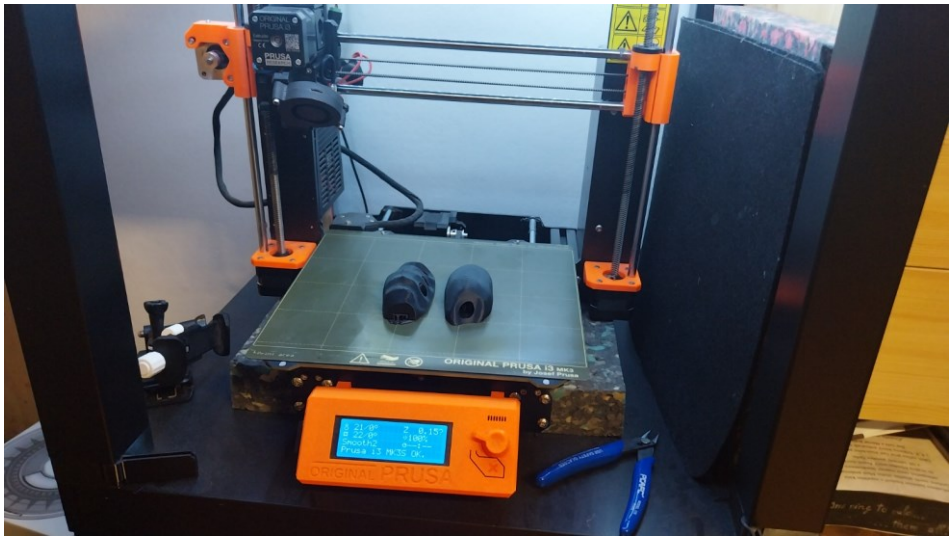
Obrázok 1: Maketa ovládača z plastelíny

Obrázok 2: Skenovanie makety 3D skenerom

Model bol potom v programe Blender 3.4 mierne upravený (vyhladený pomocou Smooth modifiera a spravené diery na tlačidlá pomocou boolean modifiera), rozdelený na 2 polovice, exportovaný do stl formátu (nasledujúci obrázok) a napokon vytlačený na 3D tlačiarni (Prusa MK3S, prostredníctvom jej softvéru Slicer, s pridanými podperami). Po vytlačení sa ručnou brúskou dotváralo vnútro a zväčšovali otvory na tlačidlá.



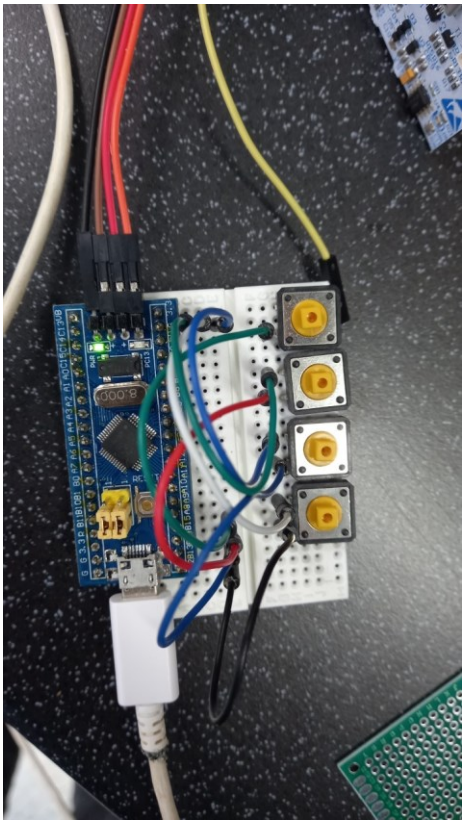
Obrázok 3: Naskenovaný model v programe Slicer



Obrázok 4: Vytlačený model na Pruse

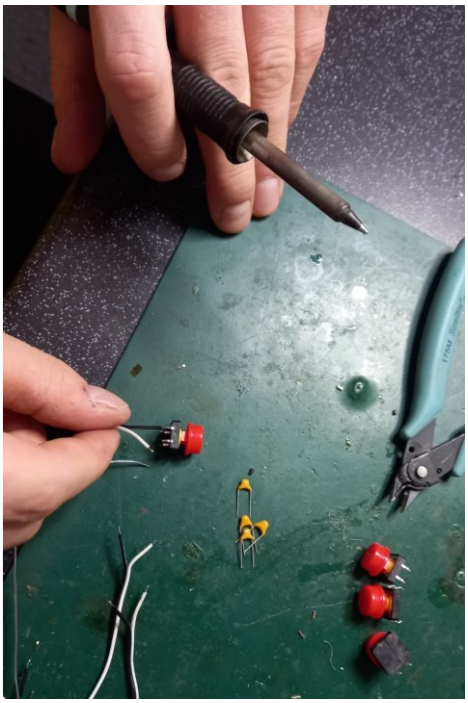
Pred spájkovaním bolo všetko (mikropočítač, káble a tlačidlá) napojené na nepájivé pole, aby sa otestovala funkčnosť.

To je možné vidieť na nasledujúcom obrázku. Dole je pripojené micro USB, ktoré smeruje do počítača (v tejto fáze slúži na napájanie mikropočítača). Hore sú pripojené na 4 piny kábliky, ktoré vedú do programátora (časť z neho je vidieť v pravom hornom rohu), ktorý sprostredkoval v počítači napísaný kód a nahral ho do mikropočítača. Štyri tlačidlá sú pripojené na svoje piny mikropočítača a taktiež na zem (GND):



Obrázok 5: Mikropočítač dočasne zložený na nepájivom poli

Až potom mohlo prebehnúť spájkovanie. Mikropočítač, 4 tlačítka, vodiaca doska a všetky kábliky boli k sebe pevne spojené (klasickým letovacím drôtom).



Obrázok 6: Spájkovanie tlačidiel ku káblikom

Aby ovládač fungoval správne a nereagoval napríklad aj na falošné stlačenie, je potrebné odstrániť problém, takzvaný bouncing (otrasy), ktorý reaguje aj na šum, respektíve rušenie, malé elektrické impulzy. Na debouncing sa na každé tlačidlo pripojí paralelne malý kondenzátor. Ten sa nabije keď je tlačidlo stlačené, a pomaly vybije, keď stlačené nie je. Toto zariadi, že sa stlačenie tlačidla vyhodnotí až keď je tlačidlo naozaj stlačené. Ako bolo spomínané, ide o "na kolene robený" prototyp, pre ostrú výrobu je potrebné podrobnejšie analyzovať hardwarovú časť a vytvoriť automatizovanejší spôsob výroby.

2.2 Softvér ovládača

Ovládač simuluje klasickú klávesnicu. Nejde o žiadne špeciálne tlačidlá, ale o klávesy "x", "1", "2", "3" v poradí od palca po prstenník (je to z dôvodu, aby sa hra dala ovládať aj klasickou klávesnicou). Autor práce nie je autorom kódu ovládača, ten je prebraný. Autor nasledoval návod na instructables.com [39]. (Je uvedený v zdrojoch, ale keďže čo sa tohto týka ide o smerodajný link, kde je celý postup práce na softvéri ovládača, autor ho umiestnil aj sem: https://www.instructables.com/STM32-As-HID-USB-Keyboard-STM32-Tutorials/ (projekt STM32_Keyboard_Tutorial)). Po pripojení mikropočítača k programátoru, ktorý sa pripojí k počítaču a po otvorení IDE sa nastavia aktívne piny, na ktorých budú tlačidlá. Potom sa zoberal predom vytvorený kód určený pre myš a modifikoval sa pre potreby klávesnice. V súbore *usbd_hid.c* sa hodnota parametra nInterfaceProtocol prepíše z "0x02" – protokol určený pre myš, na "0x01" – určený pre klávesnicu. Ďalej sa pole Report Descriptora myši upraví na pole Descriptora klávesnice, podľa návodu. Report Descriptor klávesnice má 74 bytov, preto sa musí nahradiť pôvodných 63 pre myš v súbore *usbd_hid.h* (74U - unsigned). Potom sa už jen v hlavnom súbore *main.c* pridá kód, ktorý spôsobí že mikropočítač bude rozpoznávať stlačené tlačidlo, spojí si ho s konkrétnym klávesom a tú pošle počítaču. Na konci sa kód nahrá do mikropočítača (odtiaľ už nie je možné ho spätne vytiahnuť) [39]. Ovládač sa do počítača pripojí pomocou USB káblu (USB micro male – USB A male).

Po splnení všetkých troch krokov hardvér/obal/softvér, sa vloží mikrokontrolér do obalu (s vytŕčajúcim USB káblom), tlačidlá sa vložia a zalepia do svojich otvorov a ovládač sa zalepí (ďalšia manipulácia už nie je možná)

2.3 Komponenty, materiál, nástroje, náklady

Bol vytvorený prototyp, vytlačením modelu 3D tlačiarňou. Pre prípad hromadnej alebo sériovej výroby však bude vhodnejšie robiť plastové odtlačky alebo odliatky modelu. Taktiež bude viac vhodné vyvinúť mikropočítač na mieru.

Náklady autora na výrobu prototypu: *materiál – cena* (Drôtiky a tlačidlá išlo kúpiť len hromadne, viac kusov než bolo potrebných).

a) Materiál (filament) na obal - 1.00 €

b) Doska STM Blue Pill + programátor - 1.00 € [31]

c) Tlačidlá pre Arduino – 3.79 € [40]

d) Drôtiky (2x - MM a FF) – 4,58 € [41][42]

e) 3D skener Creaform's Go!SCAN 3D (jednorazové použitie) – 19000.00 € [43] (už nie je zohnateľný, zdroj odkazuje na podobný; autor si ho zapožičal zadarmo)

f) 3D tlačiareň Prusa MK3S (jednorazové použitie) – 719 € [44] (autor si ju zapožičal zadarmo)

g) Plastelína (jednorazové použitie, musela mať bielu farbu pre potreby skenera) – 3.20 € [45]

h) Nepájivé pole (jednorazové použitie) – 1.90 € [46]

i) Pájivé pole – 0.90 € [47]

j) 2x USB kábel (Micro USB male – USB A male, jeden musí byť súčasťou hotového výrobku, druhý bol len na jednorazové použitie pri vývoji) – 10.68 € [48]

k) Elektrická energia, lepidlo a použitie uhľovej brúsky na rezanie pájivého poľa a brúsky na opracovanie obalu – nezaznamenané, nebrané do úvahy

3 SOFTVÉR

Cieľom práce je systém (softvér), ktorý umožňuje jednoducho vytvárať hry (audiohry), aj bez znalosti programovania alebo iných technických znalostí (stačí zvládať základné ovládanie počítača). V tomto prípade sa rozlišuje Používateľ – tvorca hry a Používateľ – (nevidiaci) hráč hry. S týmto nástrojom môže tvorca vytvárať hry na pravidelnej báze s nízkym rozpočtom na dabingových hercov a hudobný sprievod (prípadne aj scenáristu, v závislostí od spisovateľských schopností tvorcu) a zásobovať nimi konzumentov.

Cieľom tvorcu hry by malo vytvoriť zábavný audio (nevizuálny) zážitok. Aj keď softvér bol pôvodne vyvíjaný k tvorbe príbehových adventúr (na štýl gamebookov alebo tabletop RPG), je možné v ňom vytvoriť aj hry iných žánrov (napríklad logické, hádankové, náučné alebo detektívne a iné). Hry vytvorené týmto nástrojom budú mať síce podobnú šablónu, záujem hráčov však dlhodobo udrží pútavosť scenárov a príbehov, ktoré si tvorca bude pripravovať. Taktiež tento softvér je stavaný k tomu, aby sa hra vetvila a spájala, čo môže mať vždy iný priebeh alebo zakončenie a môže viesť k znovuhrateľnosti, teda priviesť hráča k opätovnému zapnutiu hry a skúsiť iné cesty. Pojem “Videohra“ v tomto prípade nie je presný, jedná sa o “Audiohru“.

V aplikácii (v používateľskom rozhraní) tvorca vytvára hru jednoduchou interakciou s elementami: tlačidlami pridáva nové elementy a vkladá zvukové a textové súbory, do textových polí zadáva text, zo zoznamov vyberá možnosti a nakoniec jedným kliknutím hru exportuje do spustiteľného súboru. Potom môže hru distribuovať. Napriek tomu, že systém je stavaný tak, aby tvorca hry mohol vytvoriť funkčnú hru jednoducho a rýchlo, bez technických znalostí, môže si hru upraviť nad rámec toho, čo mu systém ponúka. Stačia mu k tomu základné až pokročilé znalosti jazyku Python.

Keďže primárna cieľová skupina hráčov nevidí, alebo má vážne problémy so zrakom, hra je takmer bez akejkoľvek grafiky a vizuálnej stránke hry nie je venovaná skoro žiadna pozornosť – jediným telesným zmyslom na ktorý hra cieli je sluch a v grafickom používateľskom rozhraní sa zobrazujú iba titulky, určené pre slabozrakých hráčov (ktorí majú aspoň malú schopnosť vidieť).

3.1 Návrh a implementácia hry tvorcom

3.1.1 GameCase

Hra vytvorená v systéme predstavuje implementáciu scenára, ktorý si autor hry pripravil. Scenár si tvorca hry môže pripraviť v textovom editore, napríklad v Microsoft Office Word v LibreOffice Writer, poznámkovom bloku alebo inom programe.

Scenár pozostáva z:

a) jednotlivých scén, ktoré by mali mať uvedené aký soundtrack v danej scéne hrá (hudba na pozadí)

b) postáv, ktoré v nej vystupujú - poradie ich dialógov (vrátane rozprávača)

c) podmienok na to, aby mohla byť scéna prehraná (aké možnosti musel hráč zvoliť v predchádzajúcich scénach, aby aktuálna scéna mohla nastať)

d) možností voľby – kam sa bude hra (príbeh) ďalej uberať. Tieto možnosti môžu byť maximálne tri, minimálne žiadna (respektíve jedna – hráč nevolí nič, ak aktuálna časť scenára má len jednu líniu)

V systéme zodpovedá jednej scéne jeden *GameCase* (herný prípad). Názov je inšpirovaný názvom TestCase (testovací prípad) z testovania softvéru. Tiež ide o sériu jednoduchých udalostí, ktoré sa spúšťajú, keď sú zavolané a v rámci svojej skupiny (GameCasu) na seba nadväzujú, avšak samotné skupiny logicky na seba nadväzujú iba slabou väzbou. V tomto dokumente znamenajú pojmy “GameCase“ a “herný prípad“ rovnakú vec.

Scény v scenári, a teda aj samotné GameCasy majú svoj identifikátor (číslo, v systéme sa ID nastavuje automaticky). Úvodnému GameCasu sa nastaví ID s číslom 0, a po jednotkách sa ID inkrementuje (zvyšuje) podľa toho, v akom poradí sa GameCase pridal. Jedna zo zásad tvorby scenára je, že GameCase s istým číslom (napríklad 10) môže zavolať niektorý z predchádzajúcich GameCasov (napríklad s číslom 6), ak hráč splnil podmienky toho GameCasu až potom v jednom z neskorších GameCasov. Dôležité však je, že sa môže spustiť iba GameCase, ktorý ešte v minulosti nehral (počas jednej hry nemôže byť spustený rovnaký herný prípad dva alebo viackrát).

3.1.2 Titulky

Jediným grafickým prvkom hry sú titulky určené pre slabozrakých ľudí, ktorí majú aspoň minimálnu možnosť vidieť. Titulky, tak ako akékoľvek iné súbory môže tvorca hry pridať dobrovoľne. Samotné titulky dodržujú zásady jasnej tlače.

Implementácia zásad jasnej tlače v systéme je nasledovná:

a) Veľkosť písma: v tomto prípade je to veľkosť 30

b) Font písma: v tomto prípade je to font Arial

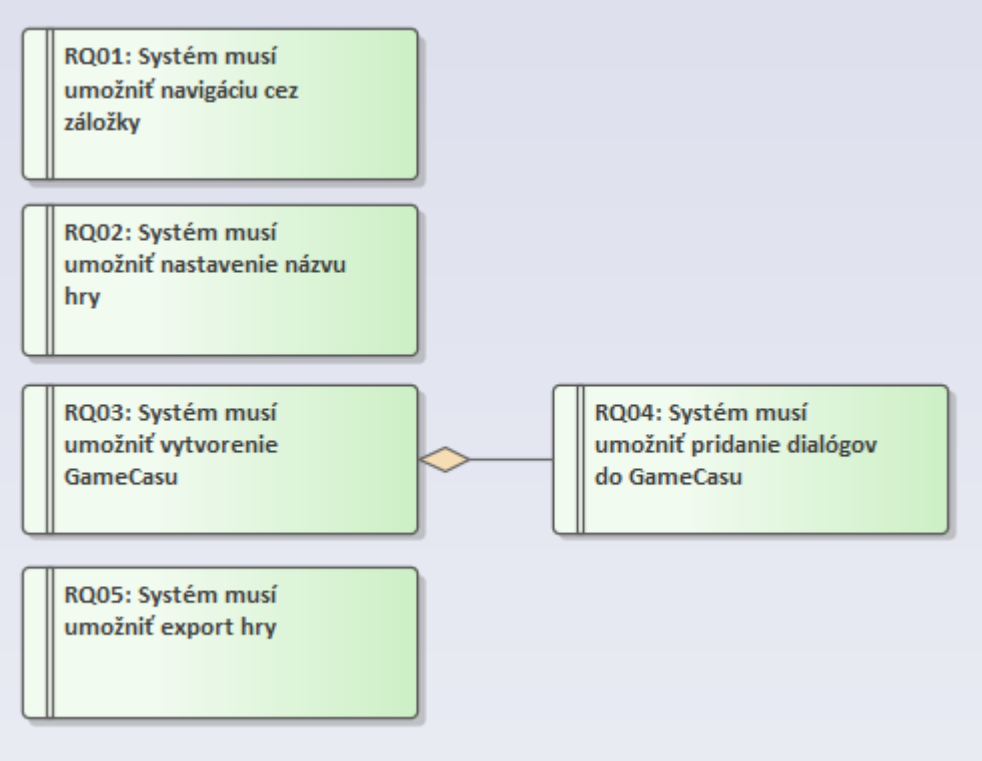
c) Kontrast textu a pozadia: hra má farbu pozadia čiernu a farbu písma bielu

d) Rozostup a riadkovanie textu: tento bod autor práce v systéme nešpecifikoval, základné nastavenia sú postačujúce.

4 NÁVRH SYSTÉMU

4.1 Model požiadaviek

Vytvorený bol model funkčných požiadaviek (requirements) pre aplikáciu (RQ) a pre hru (RQG):



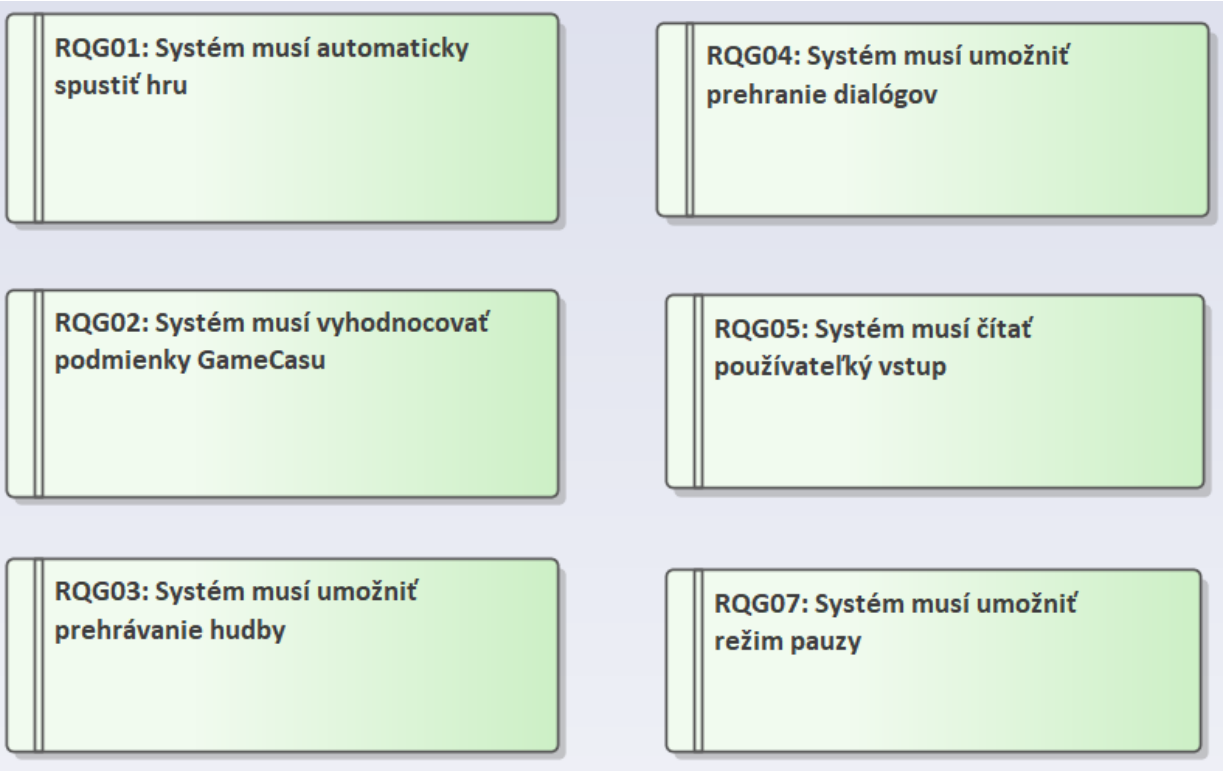
Obrázok 7: Model funkčních požiadaviek pre aplikáciu

Číslo RQ	Názov RQ	Popis RQ
1	RQ01: Systém musí umožniť navigáciu cez záložky	Systém musí umožniť navigáciu medzi tromi záložkami
2	RQ02: Systém musí umožniť nastavenie názvu hry	Systém musí umožniť nastavenie názvu hry v záložke hry
3	RQ03: Systém musí umožniť vytvorenie GameCasu	Systém musí umožniť vytvorenie skupiny elementov, niektorým je možné nastaviť hodnotu alebo obsah
4	RQ04: Systém musí umožniť pridanie dialógov do GameCasu	Systém musí umožniť vytvorenie skupiny elementov do svojej nadriadenej skupiny elementov, niektorým je možné nastaviť obsah alebo hodnotu.

5	RQ05: Systém musí umožniť export hry	Systém musí umožniť exportovanie hry do zvoleného adresára. Systém vygeneruje Python script a súbor, ktorý ju dokáže spustiť

Tabuľka 1: Funkčné požiadavky aplikácie

Medzi 3 a 4 je väzba "Aggregation to whole", 4 je súčasťou 3.



Obrázok 8: Model funkčných požiadaviek hry

Číslo RQ	Názov RQ	Popis RQ
1	RQG01: Systém musí automaticky spustiť hru	Systém musí spustiť hru, ktorej predchádza uvitanie hneď po spustení programu
2	RQG02: Systém musí vyhodnocovať podmienky GameCasu	Systém musí vyhodnotiť podmienky spustenia GameCasu a určiť, či môže byť spustený alebo nie
3	RQG03: Systém musí umožniť prehrávanie hudby	Systém musí umožniť automatické prehrávanie hudby na pozadí v nekonečnom cykle

4	RQG04: Systém musí umožniť prehranie dialógov a zobrazenie tituliek	Systém musí umožniť automatické prehranie dialógov a zobrazenie tituliek príslušného GameCasu
5	RQG05: Systém musí čítať používateľký vstup	Systém musí čítať používateľský vstup z klávesnice v dobách na to určených
6	RQG07: Systém musí umožniť režim pauzy	Systém musí umožniť hru uviesť do režimu pauzy, kedy sa neprehrávajú ďalšie GameCasy

Tabuľka 2: Funkčné požiadavky hry

5 IMPLEMENTÁCIA

Implementácia návrhu je rozdelená na aplikáciu na tvorbu hier, ktorá pozostáva z GUI, backendu a zo šablóny hry, ktorá taktiež obsahuje GUI, nie je veľmi prepracované nakoľko v ňom ide len o vypísanie tituliek, ktoré istá časť cieľovej skupiny aj tak neuvidí.

5.1 Používateľské rozhranie

Softvér (jeho UI) je tvorený aj pomocou knižnice Windows Presentation Foundation (WPF), čo vytvára predpoklad pre čisté, moderne vyzerajúce a prehľadné GUI (Graphic User Interface – Grafické Používateľské Rozhranie). Vzhľad aplikácie v spolupráci s balíčkom ModernWpfUI (vzhľad tlačidiel, TextBoxov a iných elementov) pripomína vzhľad bežnej Windows 10 či Windows 11 aplikácie od firmy Microsoft (napríklad Windows kalkulačky alebo Nastavení systému).

Všetky UI prvky sú XAML elementy združené do sekcii. Každú sekciu zahrňuje jeden súbor typu XAML (subor.xaml), ku ktorému patrí jeden súbor backendu, ktorý je programovaný v jazyku C# (subor.xaml.cs).

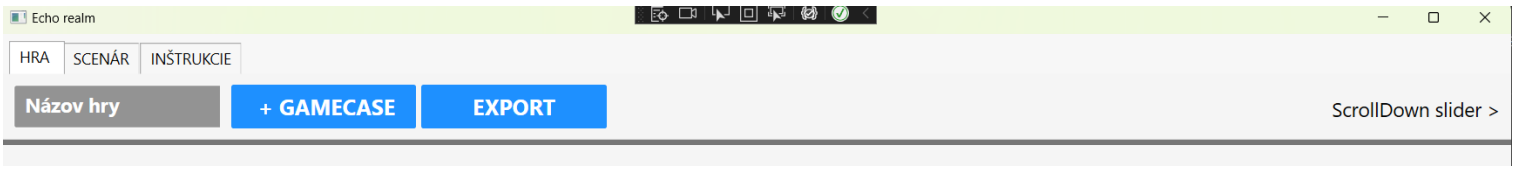
5.1.1 Hlavné okno

V hlavnom okne (MainWindow.xaml) sú tri záložky (takzvané Taby):

a) HRA

b) SCENÁR

c) INŠTRUKCIE



Obrázok 9: Taby v hlavnom okne, vybraný tab HRA

Párový element <*TabControl*> ohraničuje sekciu záložiek.

V *TabControle* sú tri skupiny párových elementov <*TabItem*>, ktorý predstavuje jednu záložku. Jeho atribút "*header*" je text, predstavuje názov záložky.

TabItem volá priamo konkrétny *UserControl* záložky.

Tieto záložky sú zobrazené neustále na vrchu aplikácie, sú viditeľné a kedykoľvek je možné sa prekliknúť z jednej na druhú. Záložky sú zahrnuté do párového elementu <*ScrollViewer*>, ktorý zobrazí posúvač (*scrollbar*), ak je v okne viac elementov ako sa zmestí na obrazovku, aby bolo možné stránku posúvať smerom nahor a nadol (*scrollovať*).

V tomto súbore sa nachádzajú aj nastavenia štýlov pre elementy (tlačidlá a *TabItemy*). Primárne dotvárajú vzhľad týchto elementov, ale aj niektoré iné ich atribúty (napríklad ako sa majú zmeniť tlačidlá po ich zacielením kurzorom myši (*hover*), alebo ako má vyzerať práve zvolená záložka). Toto je zahrnuté v párovom elemente <*Window.Resources*>. Pre nastavenie štýlu pre konkrétny element sa používa párový element <*Style*>, ktorého atribút "*TargetType*" predstavuje názov elementu (resp. jeho typ), ktorý má byť upravený. Do elementu <*Style*> patria alementy <*Setter*>, ktoré v atribúte "*Value*" nastavujú hodnotu pre konkrétnu propertu (napríklad hodnota "biela" pre propertu "*background*" nastaví bielu farbu pre pozadie elementu). Do elementu <*Style*> patrí taktiež párové elementy <*Style.Triggers*> ("spúšťače" štýlov), do ktorého patria samotné <*Trigger*> párové elementy, ktoré nastavujú hodnoty pre špecifické "*property* akcií", napríklad pre "*isMouseOver*" (hover – zacielenie kurzorom myši) alebo "*isSelected*" (bolo na to kliknuté, je to vybrané).

5.1.1.1 Záložka - HRA

Ide o najdôležitejšiu záložku. V tejto záložke používateľ – tvorca vytvára hru pre nevidiacich alebo slabozrakých ľudí a nastavuje všetky jej parametre. Jej obsahom sú elementy:

a) TextBox (pole na zadávanie textu) *gamename*, slúži na zadanie názvu hry. Jeho predvolený text je "*Názov Hry*". Dôležité: z názvu sa pri exporte automaticky odstránia medzery, inak by to nebolo možné spustiť.

b) Tlačidlo *addGroupButton*. Po kliknutí naň sa vygeneruje nový GameCase a všetky elementy, ktoré k nemu patria. Tieto GameCasy sa generujú pod seba, prakticky nie je limit na ich množstvo. Keď ich v okne bude viac ako sa zmestí na obrazovku, objaví sa posúvač a v okne bude možné "scrollovať". Toto tlačidlo volá funkciu *AddButton_Click( )* z triedy *GameSection*

c) Tlačidlo *exportbutton*, slúži na export hotovej hry, v zvolenom adresári sa vygeneruje (okrem iných) aj spustiteľný .exe súbor. Toto tlačidlo volá funkciu *CreateGameFolder( )* z triedy *GameSection*.

d) Ostatné prvky (deliaca čiara alebo štítok na označenie *scrolldown slideru*)

Všetky tieto elementy majú vo svojich parametroch nastavenú farbu textu, farbu pozadia, hrúbku textu, veľkosť písma, prípadne veľkosť elementu. Tieto elementy sú na vrchu okna, pod nimi je voľné miesto na generované GameCasy.

5.1.1.2 Záložka – SCENÁR

V tejto záložke sa nachádza *TextBox* (pole na zadávanie textu), do ktorého si tvorca hry môže písať poznámky k scenárom alebo scénam

5.1.1.3 Záložka – INŠTRUKCIE

V záložke Inštrukcie je textový *Label* (štítok), ktorého obsahom je stručná príručka k obsluhe programu a celého systému. Inštrukcie sa berú zo súboru *erinstructions.txt*, ako obsah labelu ho nastavuje .xaml.cs kód tejto záložky.

5.2 Štruktúra programu

V tejto podkapitole je rozpísaná adresárová a súborová štruktúra programu. Vypísané sú len hlavné adresáre a súbory, týkajúce sa tejto práce, ostatné (References, Properties, App.config a iné) vygenerovalo prostredie Visual Studio automaticky a autor do nich nezasahoval. Pracovný názov softvéru je EchoRealm.

EchoRealm (koreňový adresár)
- |- Backend (adresár pre sekcie – taby a ďalšie triedy)
  - |- CopySound.cs (kopírovanie súborov z Resources)
  - |- ElementGroup.cs (generovanie GameCasov)
  - |- GameSection.xaml
    - |- GameSection.xaml.cs
  - |- InstructionSection.xaml
    - |- InstructionSection.xaml.cs
  - |- ScenarioSection.xaml
    - |- ScenarioSection.xaml.cs
- |- Resources (systémové súbory (zvukové a jeden Python súbor))
  - |- EchoRealmGameStarter.exe
  - |- end.wav
  - |- endText.txt

|- erinstructions.txt
|- game_in_python.py
|- optionsSound.wav
|- pauseMsg.wav
|- pauseMsgText.txt
|- site-packages.zip
|- three_options.wav
|- three_optionsText.txt
|- two_options.wav
|- two_optionsText.txt
|- welcome_instructions.wav
|- welcome_instructionsText.txt
|- MainWindow.xaml
|- MainWindow.xaml.cs

5.3 Priebeh programu

Od začiatku až po koniec programu sa dbá na intuitívnosť a jednoduchosť jeho používania. Hlavný životný cyklus programu má 15 krokov. Jednotlivé kroky sú podrobnejšie rozpísané vo svojich podkapitolách:

1) Používateľ spustí softvér, nastaví názov svojej hry
2) V záložke “HRA“ používateľ klikne na tlačidlo “+ GameCase“
3) Systém vygeneruje skupinu elementov patriacich jednému GameCasu
4) Používateľ naplní elementy hodnotami (cesta k hudobnému pozadiu, počet možností, podmienky spustenie GameCasu, omeškanie po skončení GameCasu, popis GameCasu)
5) Používateľ klikne na tlačidlo "Pridať dialóg"
6) Systém vygeneruje skupinu elementov patriacich jednému Dialógu, ktorý patrí jednému GameCasu
7) Používateľ naplní elementy dialógu hodnotami (cesta k zvukovému súboru dialógu, stereo kanál (pan) zvukového súboru, omeškanie spustenia dialógu, cesta k textovému súboru tituliek k dialógu)

8) Takýchto dialógov si používateľ môže do GameCasu naklikať ľubovoľné množstvo
9) Takýchto GameCasov si môže používateľ vytvoriť ľubovoľné množstvo
10) Používateľ klikne na tlačidlo “Export“ a zvolí cieľový priečinok
11) Systém vytvorí adresárovú a súborovú štruktúru
12) Systém vygeneruje súbor typu “.py“ (Python), na základe GameCasov vytvorených používateľom
13) Systém vygeneruje (respektíve skopíruje) systémové zvukové súbory do cieľového adresára
14) Systém skopíruje taktiež program v jazyku C#, ktorý dokáže vytvorené hry spúšťať.
15) Používateľ (tvorca hry) môže hru následne distribuovať. Hra “nazovhry.exe“ je pripravená na otvorenie a po spustení sa hra okamžite začne. Používateľ môže taktiež upravovať vygenerovaný Python súbor a ručne si exportovať upravenú verziu hry, ak má na to znalosti.

Každý .xaml súbor má svoj .xaml.cs súbor. Kým súbor .xaml definuje aké elementy sú v okne, prípadne ako vyzerajú, .xaml.cs súbor definuje ich logiku pomocou programovacieho jazyku C#. Takýto súbor sa nazýva aj “code behind“, dalo by sa povedať, že ide o dve strany jednej mince. Preto .xaml súbor môže pristupovať aj k metódam s modifikátorom prístupu nastaveným na “*private*“, svojho pridruženého .xaml.cs súboru.

Metódy, ktoré sa volajú z okna aplikácie a obsluhujú .NET udalosti, majú parametre (object sender, RoutedEventArgs e). Taká udalosť môže byť napríklad kliknutie na tlačidlo v okne WPF. Parameter *sender* odkazuje na objekt, ktorý spustil udalosť (tlačidlo). Cez tento parameter je možné pristupovať k niektorým vlastnostiam objektu a upraviť ich. *RoutedEventArgs* je trieda, ktorá obsahuje informácie o udalosti. Tieto parametre môžu byť užitočné, ale v rámci tejto práce neboli zvláštnym spôsobom využité.

5.3.1 GameSection

Trieda *GameSection* (súbor GameSection.cs) obsahuje metódy, ktoré spracovávajú ich volania zo záložky “HRA“. Trieda dedí z triedy *UserControl*, pretože táto samotná sekcia je ovládací prvok v rámci záložky.

Na začiatku sú vytvorené inštancie iných tried a deklarujú sa členské premenné (fieldy, s modifikátorom prístupu “private“). Nasleduje konštruktor a private metódy na pridanie

GameCasu, vytvorenie adresárovej štruktúry vo zvolenom priečinku, na kopírovanie systémových súborov a metódy na generovanie (respektíve dokončenie) Python súboru s hrou.

5.3.2 Generovanie GameCasov

Generovanie jedného herného prípadu nastáva v momente, keď používateľ v používateľskom rozhraní v záložke "HRA" klikne na tlačidlo *gamename*, s textom "+ GAMECASE". Toto tlačidlo zavolá funkciu "*AddButton_Click( )*" z triedy *GameSection*.

5.3.2.1 Metóda AddButton_Click( )

Metóda, patriaca triede *GameSection* vygeneruje identifikátor (*int id*), na základe doterajšieho počtu GameCasov v okne (to je docielené tak, že v triede je deklarovaný list ElementGroups – teda skupín elementov – teda GameCasov (inak povedané elementov, ktoré patria jednému GameCasu), do ktorého sa GameCase pridá pri jeho vytvorení). Identifikátor začína od čísla 0. Následne sa vytvorí inštancia triedy ElementGroup, do ktorej konštruktora sa vloží vygenerované ID, aby mohlo byť priamo generované do samotného GameCasu. Objekt sa pridá do zoznamu *ElementGroups* a zavolá sa metóda *GetVisual*( ) triedy *ElementGroup*, ktorá vygeneruje elementy. Táto trieda sa zavolá do stackpanelu, ktorý je nachystaný v okne, v súbore GameSection.xaml. Tam sa elementy vygenerujú.

5.3.3 Skupina elementov

Aby sa oddelila logika generovania elementov od zbytku programu, má svoju vlastnú triedu *ElementGroup* v súbore *ElementGroup.cs*, ktorý patrí priečinku *Backend*. V rámci triedy sú definované elementy (vďaka namespacu System.Windows.Controls;), ktoré sa potom budú vo svojom poradí pridávať do mriežky (grid). Všetky elementy GameCasu tvoria jednu skupinu elementov (*ElementGroup*).

V konštruktore sa preberá identifikátor (int id), ktorý sa rovno vkladá do textu hlavného textového štítku (*label*), ktorý sa zobrazí na vrchu GameCasu. Jediná funkcia identifikátora je informatívna – aby používateľ vedel ktorý GameCase práve vytvára a vedel si ich spárovať so scénou v scenári.

5.3.3.1 Generované elementy

Slovník pojmov, ktoré sú použité v tomto dokumente (môže byť použitý originálny tvar a anglickom jazyku alebo jeho slovenský ekvivalent), vedľa popis elementu:

Anglický názov	Slovenský názov	Popis
Label	Štítok/ textový štítok	Text nachádzajúci sa v okne je needitovateľný a nekopírovateľný
Button	Tlačidlo	Po kliknutí naň nastane istá udalosť
TextBox	Textové pole	Pole na zadávanie (alebo vkladanie) textu používateľom
Border	Rám	Slúži na orámovanie skupiny elementov
Slider	Posúvač	Slúži na nastavenie hodnoty posúvaním uchopovača (grapper) na ohraničenej úsečke
Grid	Mriežka	Slúži na organizované umiestňovanie elementov, ktoré sú v nej
ComboBox	Rozbaľovací zoznam	V ňom sa vyberá jedna z prednastavených možností, je podobný klasickému DropDown Listu
StackPanel	StackPanel	Kontainerová trieda, slúži na jednoduché rozmiestnenie prvkov vnútri. Je jednoduchšia, ale má menej možností ako pri ukladaní elementov do Gridu

Tabuľka 3: Slovník pojmov

Súčasťou jedného *GameCasu* sú nasledovné elementy, ktoré sú aj definované v triede *ElementGroup*.

(pozn. autora: v skutočnosti sú to inštancie svojich tried, ale keďže reprezentujú statické či dynamické objekty v okne aplikácie, budú v tomto dokumente tiež nazývané ako “elementy“, podobne ako u pôvodných xaml elementov):

1) Mriežka *grid*: hlavná mriežka, v ktorej sú ostatné elementy (okrem rámu)
2) Label *id_label*, zobrazuje sa na vrchu herného prípadu a obsahuje jeho číslo. Jeho textom je "GAMECASE: {číslo}".

3) Textové pole *soundtrack_file_textbox*, slúži na zadanie cesty k zvukovému súboru hudby, ktorá má hrať v tej scéne na pozadí. Môže byť do neho napísaný aj text "go", v prípade že používateľ chce aby hudba pokračovala z predošlého herného prípadu, alebo "*off*", ak používateľ chce aby žiadna hudba v danej scéne nehrala. Predvolený text je "*off*". Cesta k súboru sem môže byť napísaná alebo skopírovaná ručne, prípadne sa sem vloží automaticky ak bude používateľ vyhľadávať súbor pomocou tlačidla *soundtrack_file_button*.

4) Tlačidlo *soundtrack_file_button*, po kliknutí naň sa otvorí dialógové okno do systémového súborového prehliadača, v ktorom môže používateľ vybrať zvukový súbor hudby, ktorá má hrať na pozadí. Jeho textom je "Soundtrack file...".

5) Tlačidlo *add_dialogue_button*. Po kliknutí naň sa v hernom prípade vygeneruje ďalšia skupina elementov, obsluhujúca jeden dialóg.

6) Rám *border* vizuálne ohraničuje ostatné elementy, respektíve celý GameCase.

7) Label *conditionsLabel* je štítok, ktorý vizuálne informuje užívateľa, že vo vedľajšom elemente si môže nastaviť podmienky spustenia tohto herného prípadu. Jeho text je "Podmienky: ".

8) Textové pole *conditionsTextBox*. Jeho predvoleným textom je "none". Do tohto poľa môže používateľ vkladať podmienky spustenia aktuálneho GameCasu formou textu. Viac v kapitole 5.6.

9) Textové pole *descriptionTextBox* slúži na vloženie vloženie popisu herného prípadu, používateľovými vlastnými slovami. Je to na používateľovu lepšiu orientáciu vo svojom projekte, taktiež sa tento popis vloží do vygenerovaného Python súboru vo forme komentára k danému hernému prípadu. Predvolený text je "popis vlastnými slovami..."

10) Label *gcDelayLabel* (gamecaseDelayLabel) informuje používateľa o tom, že vedľajší TextBox slúži na zadávanie omeškania konca GameCasu, respektíve koľko sekúnd po prehratí posledného dialógu sa ešte nespustí nasledujúci herný prípad.

11) Textové pole *gcDelayTextBox* slúži na zadávanie čísel, čo znamenajú čas v sekundách, ktorý predstavuje, ako dlho po skončení aktuálneho herného prípadu sa môže spustiť nasledujúci herný prípad. Ak chce používateľ spraviť medzeru medzi posledným dialógom tohto GameCasu a prvým dialógom nasledujúceho GameCasu,

môže síce nastaviť omeškanie spustenia konkrétneho dialógu, ale v prípade že používateľ chce aby nejakú dobu ešte nerušene hrala hudba aktuálneho GameCasu, môže nastaviť toto omeškanie.

12) Textový štítok *gcSLabel*: jeho textom je "s", čo používateľovi značí, že do textového poľa na zadávanie omeškania sa píše číslo, ktoré reprezentujú čas v sekundách.

13) Textový štítok *optionsLabelpoužívateľovi* značí, že vedľajší ComboBox vyjadruje počet možností, ktorý aktuálny herný prípad má.

14) Rozbaľovací zoznam *optionsComboBox* vyjadruje koľko ciest (možností) má aktuálny GameCase. Na výber je z "1", "2", "3" alebo "END" - v prípade, že tento herný prípad značí koniec hry.

Každý z týchto elementov je objekt svojej triedy. Ako napríklad tlačidlo soun*dtrack_file_button* je triedy *Button*. Definícia tlačidla teda vyzerá nasledovne:

```
private Button soundtrack_file_button = new Button( );
```

Každému elementu sa nastavajú rôzne hodnoty pre rôzne premenné (*fields*), napríklad veľkosť písma, odsadenie, predvolený text a podobne. Tieto hodnoty sa dajú nastaviť kdekoľvek v jeho *scope* (v bloku kódu, kde bola inštancia vytvorená, ohraničený zloženými zátvorkami { }), pristúpením ka danej hodnote objektu. Napríklad farba pozadia (*Background*) tlačidla môže byť nastavená kódom:

```
soundtrack_file_button.Background = Brushes.LightGray;
```

Keďže jeden objekt môže mať tých vlastností nastavených viac, a obyčajne sa už v priebehu kódu nebudú meniť (meniť sa bude iba text priamou manipuláciou používateľa s daným elementom), tak všetky potrebné vlastnosti boli už nastavené pri vytváraní objektu (technikou objektovej inicializácie – *object initializer*), aby mohlo byť všetko nastavené v jednej deklarácii. Prispieva to k prehľadnosti kódu. Jednotlivé vlastnosti sú oddelené čiarkami. Napríklad takto:

```
private Button soundtrack_file_button = new Button( )
{
    Content = "Soundtrack file...",
    Foreground = SystemColors.WindowTextBrush,
    Background = Brushes.LightGray,
    Margin = new Thickness(10, 10, 0, 0)
};
```

Tlačidlu na vyhľadávanie súboru pre hudbu bol nastavený predvolený text na "Soundtrack file", farba textu (*Foreground*) bola nastavená na predvolenú, farba pozadia bola nastavená na svetlo-šedú a odsadenie je zľava a zhora je hodnota 10.

Margin je vonkajšie odsadenie elementu, teda predstavuje aká veľká medzera bude okolo elementu. To zabezpečí trieda "*Thickness*", teda "hrúbka", ktorá berie maximálne štyri parametre v poradí:

1) Hodnota zľava
2) Hodnota zhora
3) Hodnota sprava
4) Hodnota zdola

Ak sa uvedú všetky parametre, uvedené hodnoty sa nastavia na svoju stranu. Ak sa uvedú len dve hodnoty, tak prvá hodnota bude rovnaká pre vľavo a vpravo a druhá bude platiť pre hore a dole. Ak sa uvedie len jedna hodnota, bude platiť pre všetky štyri strany.

Margin aj všetky ostatné vlastnosti, ktoré používajú *Thickness( )* (*konštruktor*) štruktúry *Thickness*, majú jednotku hodnoty *DIP* (*device-independent pixel*), ktorý je ekvivalentný 1/96 palca. Jednotka DIP platí aj pri niektorých ostatných vlastnostiach, uvedených nižšie.

Zoznam parametrov nastavovaných inštanciám tried elementov GameCasu objektovou inicializáciou:

1) Content: Nastavenie textu labelom alebo tlačidlám. Príklad: `Content = "Zobrazený text".`
2) FontSize: Nastavenie veľkosti textu. Prijíma hodnotu typu double. Príklad: `Fontsize = 20`.
3) FontWeight: Nastavenie hrúbky textu. Príklad (nastavenie textu na hrubý): `FontWeight = FontWeights.Bold`.
4) ForeGround: nastavenie popredia (textu). Príklad (nastavenie farby textu na čiernu): `Foreground = Brushes.Black`.
5) Text: prednastavenie textu pre textové pole (tento text sa dopredu vyplní ale je možné ho v aplikácii modifikovať). Prijíma hodnotu string. Príklad: `Text = "prednastavený text"`.

6) Width: Nastavenie šírky elementu (TextBoxu). Jednotka je DIP. Prijíma hodnotu typu double. Príklad: `Width = 410`.
7) Heigth: Nastavenie výšky elementu (TextBoxu). A aplikácii je volená približná výška jedného riadku. Jednotka je DIP. Prijíma hodnotu typu double. Príklad: `Height = 30`.
8) Margin: Nastavenie odsadenia (medzery) zo štyroch smerov elementu. Jednotky sú DIPs. Príklad: `Margin = new Thickness(0, 10, 0, 0).`
9) HorizontalAlignment: Nastavenie horizontálneho zarovnania. To môže byť napríklad vľavo, centrované, natiahnuté na strany alebo vpravo. Príklad (zarovnanie doľava): `HorizontalAlignment = HorizontalAlignment.Left`.
10) BorderBrush: Nastavenie farby orámovania pre inštanciu triedy Border. Príklad (nastavenie farby na šedú): `BorderBrush = Brushes.Gray`.
11) BorderThickness: Nastavenie hrúbky orámovania. Podobne ako pri Margine, aj tu sa volá trieda Thickness. Príklad: `BorderThickness = new Thickness(2)`.
12) Padding: Nastavenie odsadenia vnútorného obsahu prvku od jeho orámovania. Príklad: `Padding = new Thickness(10)`.
13) CornerRadius: Nastavuje zaoblenie rohov (napríklad orámovania). Volá sa na to konštruktor CornerRadius( ), ktorý prijíma hodnotu typu double.
14) Orientation: Určí, či sa elementy zoradia zľava doprava alebo zhora dole. Príklad (vertikálne zoradenie): `Orientation = Orientation.Vertical`.
15) TextWrapping: Nastavuje u TextBoxov správanie textu po tom, čo dosiahne koniec riadku. "Wrap" nastaví to, že text sa zalomí a bude pokračovať do druhého riadku. "NoWrap" je nastavenie, ktoré zaručí že riadok bude pokračovať donekonečna smerom vpravo. Príklad: `TextWrapping = TextWrapping.Wrap`.
16) VerticalScrollBarVisibility: Nastavuje viditeľnosť vertikálneho posúvača, v tomto prípade u TextBoxu. Keď je presiahnutá výška obsahu u elementu, v danom textovom poli bude možné posúvač "okno" TextBoxu aby bol zameraný určitý text (napríklad kolieskom myši alebo šípkami na klávesnici). Ak je nastavené "Auto", posúvač sa zobrazí až keď obsah (výška obsahu) presiahne hranice elementu. Ak je nastavené "Visible", posúvač je viditeľný stále, aj keď ešte nie je potrebný. Ak je nastavené "Hidden", posúvač nie je viditeľný (aj keď tam je). Ak je nastavené

"Disabled", tak výška obsahu textového poľa nemôže byť väčšia, ako pôvodná výška textového poľa a text nad rozsah výšky tam nebude môcť byť pripísaný. Príklad: `VerticalScrollBarVisibility = ScrollBarVisibility.Auto` .

Vlastnosti ako *Height*, *Width* alebo *FontSize* prijímajú hodnoty typu double (reálne číslo aj s desatinnou časťou). Sú to však takzvané *DependencyProperty* ktorú sú v getteri svojej property (napr. property FontSize) svojej triedy pretypované na double. (Napríklad FontSize je deklarované v triede Control, ktorá dedí z FrameworkElement. Vlastnosti Height a Width sú definované priamo v triede FrameworkElement a sú tiež definované ako "property"). WPF dokáže sledovať niektoré akcie (napríklad dynamická zmena okna), a tomu prispôsobiť vlastností elementov cez DependencyProperty. Tieto property sú typu double.

Vlastnoť *Text* pre TextBox prijíma hodnotu typu string, ide však o TextProperty, ktorá je pretypovaná na string vo svojej property Text, ktorá patrí priamo triede TextBox (property Text je dátového typu string).

Vlastnosť *Content* pre textové štítky alebo texty tlačidiel tiež prijíma text, ale property ako taká nie je typu string, ale je to object (môže obsahovať akúkoľvek hodnotu alebo inštanciu). Do vlastnosti Content sa dá okrem textu vložiť aj napríklad obrázok.

(pozn. autora: ak by sa elementy vkladali priamo v XAML, tak text v párových atribútoch Button alebo TextBox povkladá *XAML pars*er do svojich *propertií*, podobne ako sa to tu robí v C#).

Príklad:

```
<Button>Text tlačidla</Button>
```

Text "Text tlačidla" XAML *parser* automaticky umiestni do property *ContentProperty*. Taktiež text elementu <TextBox> umiestni do property *TextProperty*.

Vlastnosti však môžu byť aj iných typov, napríklad *HorizontalAlignment* či *ScrollBarVisibility* sú typu *enum*, *Thickness* je konštruktor štruktúry *Thickness*, *FontWeights* je trieda a konkrétne *Bold* je jedna z jej propertií, nastavená cez *expression-bodied* member (používa sa v prípade, ak má property iba jeden výraz). Ak si čitateľ chce pozrieť typy aj ostatných vlastností, môže tak urobiť cez "klávesa CTRL + kliknutie na hľadanú vlastnosť", prípadne nahliadnuť do dokumentácie. Z pohľadu tvorby aplikácie ide o menej podstatné skutočnosti, o ktoré sa C# a kompilátor stará sám (volanie predom vytvorených tried a iných prvkov knižnice).

5.3.4 Hlavná mriežka

Metóda *GetVisual( )* vráti rám, v ktorom bude vytvorená mriežka so všetkými elementami GameCasu, prostredníctvom metódy *GetGrid( )*.

5.3.4.1 Metóda GetGrid( )

Pri každom volaní tejto funkcie sa vždy vytvorí nová inštancia mriežky grid (respektíve sa stará inštancia prepíše na novú), a nastaví sa jej vonkajšie odsadenie (najmä preto, aby tie GameCasy neboli na sebe úplne nalepené).

Vytvoria sa a pridajú do mriežky dva riadky (cez property *RowDefinitions* triedy ExtendedData triedy Grid, typu RowDefinitionCollections) a na podobný spôsob sa do mriežky pridá päť stĺpcov.

Pridanie riadku:

```
grid.RowDefinitions.Add(new RowDefinition( ) { Height =
GridLength.Auto });
```

Pridanie stĺpca:

```
grid.ColumnDefinitions.Add(new ColumnDefinition( ) { Width =
GridLength.Auto });
```

Následne sa do mriežky pridávajú vyššie spomínané elementy (inštancie) v správnom poradí.

Do zoznamu optionsComboBox sa pridajú možnosti (“1“, “2“, “3“, “END“) a prvá hodnota sa nastaví na predvolene vybranú. Ukážka kódu:

```
// nastavenie možností GameCasu
optionsComboBox.Items.Add("1");
optionsComboBox.Items.Add("2");
optionsComboBox.Items.Add("3");
optionsComboBox.Items.Add("END");
optionsComboBox.SelectedItem = optionsComboBox.Items[0];
```

Aby boli niektoré elementy v skupinách a jednoduchšie sa manipulovalo s ich pozíciami (napríklad ak dva elementy majú byť stále pri sebe), tak sa vkladajú do ďalších kontajnerov prostredníctvom mriežok alebo stackpanelov. V metóde bola teda vytvorená ďalšie mriežka *soundtrack_and_options_GRID*, do ktorej sú vložené elementy *soundtrack_file_button*,

soundtrack_file_textbox, *optionsLabel* a *optionsComboBox*. Táto mriežka bola vložená do hlavnej mriežky.

Príklad nastavenia elementu (soundtrack_file_textbox) v mriežke:

Nastavenie elementu na prvý riadok v mriežke:

```
Grid.SetRow(soundtrack_file_textbox, 1);
```

Nastavenie elementu do prvého stĺpca mriežky:

```
Grid.SetColumn(soundtrack_file_textbox, 1);
```

Pridanie elementu do podpornej mriežky:

```
soundtrack_and_options_GRID.Children.Add(soundtrack_file_textbox);
```

Podobným spôsobom bol vytvorený aj stackpanel *gc_controls_SP*, do ktorého boli pridané elementy add_dialogue_button, conditionsLabel, conditionsTextBox, descriptionTextBox, gcDelayLabel, gcDelayTextBox a gcMsLabel. Stackpanel bol taktiež pridaný do hlavnej mriežky.

Tlačidlám, k ich udalostiam (kliknutie) sú priradené *lambda* funkcie ako delegáti (pridanie k odozvám na udalosť). V prípade *soundtrack_file_button* je to funkcia *FileButton_Click( )*, ktorá otvorí dialógové okno do prieskumníka súborov, aby si používateľ mohol vybrať súbor:

```
soundtrack_file_button.Click += (s, e) => FileButton_Click(s, e, soundtrack_file_textbox, "audio");
```

Parameter “s“ predstavuje tlačidlo (ako zdroj udalosti) a parameter “e“ obsahuje dodatočné informácie o kliknutí (ako o udalosti). Toto sú štandardné parametre. Ale metóda *FileButton_Click( )* berie aj dva dodatočné parametre. Jeden je TextBox, do ktorého sa vybraná cesta uloží a druhý je typ súboru, ktoré môžu byť vyhľadávané. V tomto prípade sa vybraná cesta zapíše do textového poľa *soundtrack_file_textbox* a vyhľadávané môžu byť iba audio súbory (.wav alebo .mp3).

Pri udalosti kliknutia na tlačidlo *add_dialogue_button* sa zavolá metóda *add_dialogue_button_Click( )*, ktorá vygeneruje elementy potrebné pre dialóg.

5.3.5 Dialóg

Používateľ môže v hernom prípade kliknúť na *add_dialogue_button*, na čo sa vygenerujú nasledujúce elementy. Všetky elementy jedného dialógu sú v jednom riadku a nie je žiaden limit na počet dialógov v hernom prípade. Dialógy sa ukladajú pod seba s tým, že orámovanie GameCasu mení svoju veľkosť (zväčšuje sa, rozširuje sa smerom nadol).

Zoznam elementov pre dialóg:

1) Label *dialogue_label*: tento štítok označuje riadok dialógu. Button
2) Button *dialogue_file_button*: tlačidlo na výber zvukového súboru dialógu v dialógovom okne.
3) TextBox *dialogue_file_textbox*: textové pole, kde sa vloží cesta k zvukovému súboru dialógu (ručne alebo automaticky po vybraní súboru pomocou tlačidla). V textovom poli môže byť aj text "none", v prípade že dialóg (zatiaľ alebo vôbec) nemá zvukový súbor. Predvolený text je "none".
4) Label *pan_label*: štítok označuje, že nasledujúci element (slider) nastaví stereo kanál (pan). Jeho textom (contentom) je "Pan:".
5) Slider *pan_slider*: horizontálny posúvač, ktorým sa nastaví stereo kanál. Rozsah posúvača je <-1, 1>, posúvanie ide po desatinách. Predvolene je nastavený na hodnotu 0 (stred), a posúvaním vľavo alebo vpravo sa nastavuje, či bude a ako veľmi bude zvuk počuť v ľavom alebo pravom uchu (slúchadle).
6) Label *delay_label*: tento štítok označuje, že nasledujúce textové pole reprezentuje omeškanie dialógu. Jeho textom (contentom) je "Delay:"
7) TextBox *delay_textbox*: do tohto textového poľa sa píše číslo, ktoré značí omeškanie spustenia dialógu (aká pauza bude pred spustením dialógu). Predvolený text je "0".
8) Label *s_label*: štítok je umiestnený za delay TextBoxom, značí že číslo zadané v tom textovom poli majú byť sekundy (môžu to byť aj zlomky sekúnd, napríklad 1.25). Jeho textom je "s". Ako desatinný oddeľovač sa použije bodka (".")
9) Button *dialogue_text_file_button*: po kliknutí na toto tlačidlo sa otvorí dialógové okno do prieskumníka súborov, vyhľadávané môžu byť súbory s príponou .txt.
10) TextBox *dialogue_text_file_textBox*: do tohto textového poľa sa vloží cesta k textovému súboru (ručne vložená alebo automaticky pomocou tlačidla), ktorý

predstavuje titulky k aktuálnemu dialógu. Text môže byť aj "none", ak žiadne titulky (zatiaľ alebo vôbec) nie sú. Predvolený text je "none".

Elementom sa nastavujú vlastnosti vymenované v podkapitole Generované elementy. Okrem nich sa špeciálne nastavujú aj vlastnosti posúvača (slideru):

Minimum: nastavuje minimálnu hodnotu (u horizontálneho slideru je táto hodnota úplne vľavo). Prijíma hodnotu double (môže to byť desatinné číslo, môže ísť aj do mínusu). Príklad: `Minimum = -1`.

Maximum: nastavuje maximálnu hodnotu (u horizontálneho slideru je táto hodnota úplne vpravo). Prijíma hodnotu double (môže to byť desatinné číslo, môže ísť aj do mínusu – za predpokladu že minimálna hodnota je ešte menšia). Príklad: `Maximum = 1`.

TickFrequency: určuje jeden "krok" slideru, teda hodnotu, o ktorú uchopovač (grapper) skočí pri jednom posunutí. Prijíma hodnotu double (môže to byť desatinné číslo). Príklad: `TickFrequency = 0.1`.

IsSnapToTickEnabled nastaví, či sa kroky pri posúvaní budú lepiť na svoje značky, jedného ticku (čitateľ si môže predstaviť funkciu "snap to grid" pri nejakom inom programe). Ak je hodnota nastavená na true, grapper sa bude lepiť na svojej značky (nastavené vlastnosťou TickFrequency). Príklad: `IsSnapToTickEnabled = true`.

Value: hodnota, ktorá bude nastavená čerstvo pri vygenerovaní slideru (musí to byť hodnota z daného rozsahu). Príklad: Value = 0.

Slider, ktorý sa generuje v aplikácii:

```
// Slider pre nastavenie stereo kanálov:
Slider pan_slider = new Slider( )
{
    Minimum = -1,
    Maximum = 1,
    TickFrequency = 0.1,
    IsSnapToTickEnabled = true,
    Width = 150,
    Height = 30,
    Value = 0,
    Margin = new Thickness(5, 0, 0, 0)
```

```
        };
```

Po kliknutí na tlačidlo *dialogue_file_button* sa zavolá funkcia, ktorá otvorí dialógové okno na výber súboru a cestu k nemu vloží do textového poľa *dialogue_file_textbox*:

```
dialogue_file_button.Click += (s, ev) => FileButton_Click(s, ev, dialogue_file_textbox, "audio");
```

Po kliknutí na tlačidlo dialogue_text_file_button sa zavolá rovnaká funkcia:

```
dialogue_text_file_button.Click += (s, ev) => FileButton_Click(s, ev, dialogue_text_file_textBox, "text");
```

Vytvorený bol stackpanel dialogue_controls_STACKPANEL, do ktorého sa povkladajú všetky elementy dialógu:

```
dialogue_controls_STACKPANEL.Children.Add(dialogue_label);
dialogue_controls_STACKPANEL.Children.Add(dialogue_file_button);
...
```

Všetky elementy sa pridávajú do stackpanelov a gridov ako "*Children*". Čiže ak treba pridať stackpanel s elementami dialógu do hlavnej mriežky, je nutné k nej pristúpiť. Element add_dialogue_button, ktorý patrí GameCasu, teda priamo hlavnej mriežke, je dieťa (Children) hlavnej mriežky (autor si nie je istý, či by slovenský ekvivalent mohol byť "potomok", tak ako sa podobná vec nazýva inde v objektovo orientovanom programovaní). A keďže spomínané tlačidlo volá funkciu na generovanie dialógu, je to hlavný prostredník medzi GameCasom ako takým a jeho dialógom. Preto sa pristupuje cez tlačidlo (ako sender – je to jeden z parametrov tejto funkcie [*object sender*]), kedy sa volá jeho rodič, teda "*parent*", teda hlavná mriežka. K tomu je nutné pretypovať sendera na tlačidlo. V programe sa vytvorí nový stackpanel "parentStackPanel", do ktorého sa vloží hlavná mriežka (teda parent, ktorý sa tu z gridu pretypuje na stackpanel). Vytvorí sa tu grid, do ktorého sa zase vloží "*parentStackPanel*", opäť naspäť pretypovaný na Grid, čo je jeho pôvodná podoba. Keď sa toto stane, do hlavnej mriežky je možné pridať stackpanel so všetkými elementami dialógu.

Do mriežky sa pridá jeden riadok (všetky elementy dialógu sú len na jednom riadku) a do toho riadku sa pridá spomínaný stackpanel s elementami dialógu. Napriek tomu, že sa tu práve ten riadok pridal, nejde ho ako taký spoznať, najnovší nie je označený ako posledný. Je nutné si zistiť počet riadkov v GameCase a elementy vložiť do posledného (keďže tých

dialógov tu môže byť pridaných niekoľko). K tomu slúži premenná “ *int newRow*“. Ukážka kódu, kde sa toto vykonáva:

```
  // Prístup k rodičovi
        StackPanel parentStackPanel =
(StackPanel)((Button)sender).Parent;
        Grid grid = (Grid)parentStackPanel.Parent;

        // Pridanie nových riadkov do Gridu pre všetky
elementy
        grid.RowDefinitions.Add(new RowDefinition( ) { Height
= GridLength.Auto });
        // Nastavenie riadkov pre nové elementy:
        int newRow = grid.RowDefinitions.Count - 1;
        Grid.SetRow(dialogue_controls_STACKPANEL, newRow);
```

Všetky elementy, kde používateľ manipuluje s hodnotou, sa pridajú do zoznamu (list) *dialogueSection*, aby bol k týmto elementom zjednodušený prístup, keď ich hodnoty bude chcieť program neskôr čítať pri exporte. V tomto súbore má tento svoj list aj svoju metódu Get (public), cez ktorý sa k tejto sekcii program dostane.

V hernom prípade sa vytvára taktiež stackpanel *generatedElementsStackPanel*, do ktorého sa pridávajú niektoré elementy (vrátane elementov dialógu), aby bola jednoduchšia manipulácia s ich umiestnením v hlavnej mriežke. Ak používateľ v aplikácii nechce nastaviť špecifickú hodnotu, nemusí. V tom prípade ale musí ponechať (alebo znovu nastaviť) hodnotu “none“, “0“ prípadne “off“ či “go“, aby program fungoval správne.

5.3.6 Výber súboru

Tlačidlá na výber súborov soundtracku, dialógu a tituliek dialógu (teda *soundtrack_file_button*, *dialogue_file_button* a *dialogue_text_file_button*) volajú metódu *FileButton_Click( )*, ktorá okrem nutných parametrov “sender“ a “e“, prijíma aj parameter TextBoxu, do ktorého sa vybraná cesta k súboru vpíše a taktiež parameter “*filetype*“, ktorý určí či sa budú v dialógovom okne zobrazovať iba zvukové alebo iba textové súbory.

Metóda otvorí dialógové okno za pomoci inštancie *OpenFileDialog* (za pomoci knižnice Microsoft.Win32), zobrazí len súbory, ktoré sa môžu zobraziť a následne skontroluje, či

nejaký súbor bol vybraný (ak nebol – a používateľ iba zavrie okno, nestane sa nič). Metóda je private, jej volanie je potrebné robiť len v tejto triede.

Kód metódy:

```
private void FileButton_Click(object sender, RoutedEventArgs
e, TextBox textbox, string fileType)
        {
            var file_dialog = new OpenFileDialog( );

            if (fileType == "audio")
                file_dialog.Filter = "Audio
Files|*.mp3;*.wav";
            else if (fileType == "text")
                file_dialog.Filter = "Text Files|*.txt";

            Nullable<bool> result = file_dialog.ShowDialog( );
            if (result == true)
            {
                string file_path = file_dialog.FileName;
                textbox.Text = file_path;
            }
        }
```

Obrázok 10: Okno s GameCasmi a dialógmi

5.3.7 Export hry

Okrem zvukových súborov, ktoré do hry pridal používateľ – tvorca hry, musí hra obsahovať aj niekoľko systémových súborov:

end.wav: zvukový súbor, ktorý hráčovi oznámi koniec hry, keď ju hráč dohrá.

optionsSound.wav: jedná sa o zvukový súbor, ktorý signalizuje hráčovi to, že môže vyberať možnosť a počet možností, z ktorých vyberať môže.

pauseMsg.wav: zvukový súbor, ktorý hráčovi oznámi, že hra je v režime pauzy a povie mu, aké má teraz možnosti.

three_options.wav: zvukový súbor, ktorý oznámi hráčovi, že môže vyberať z troch možností, keď skončí režim pauzy.

two_options.wav: zvukový súbor, ktorý oznámi hráčovi, že môže vyberať z dvoch možností, keď skončí režim pauzy.

welcome_instructions.wav: zvukový súbor, ktorý hráča privíta v hre a vysvetlí mu inštrukcie ohľadom ovládania hry.

Všetky tieto systémové zvukové súbory sú nahovorené zatiaľ len v slovenskom jazyku. Do dvojice majú svoj textov7 súbor s titulkami. Sú pridané v priečinku *Resources*, každému jednému z nich boli nastavené vlastnosti (*Properties*):

a) Build Action: Resource

b) Copy to Output Directory: Do not copy

Tieto vlastnosti zahrnú súbory do výsledného zostavenia ako "zdroje - *resources*" a taktiež zapríčinia to, že so súbormi nebude možné manipulovať, ani ich vidieť. Všetky budú zabalené do jedného balíka. Toto v aktuálnej fáze komplikuje multi-jazykovosť (všetky systémové zvukové súbory sú v slovenčine), avšak v budúcich verziách softvéru možno pribudne možnosť pridania vlastných systémových súborov, alebo autor práce vytvorí tieto súbory aj v iných jazykoch a pridá do softvéru s tým, že používateľ – tvorca hry si jazyk vyberie.

Okrem uvedených systémových súborov je tam ešte jeden- a to game_in_python.py, čo je predpripravený kód v jazyku Python, na ktorom celá hra bude bežať. Všetky funkcie sú v ňom už napísané, súbor sa dotvorí na základe elementov (GameCasov), ktoré si tvorca hry vytvoril. Systém automaticky vygeneruje tento Python kód a vloží ho do toho súboru.

5.4 Vytvorenie adresárovej štruktúry

Export hry započne kliknutím na tlačidlo "*exportButton*", s textom "EXPORT" v tabe "*GameSection*". Zavolá sa metóda *CreateGameFolder( )* triedy GameSection.

Táto metóda otvorí dialógové okno pomocou inštancie WPFFolderBrowserDialog, v ktorom si používateľ vyerie priečinok, kam chce svoju hru exportovať. V tom priečinku sa vytvorí hlavný adresár, ktorý nesie názov podľa mena, ktoré zadal používateľ v textboxe "gamename", v tabe GameSection. Ak už taký priečinok v zvolenom priečinku existuje, používateľovi sa objaví správa (vyskakovacie okno - MessageBox), ktorá sa ho spýta, či si bude priať ten priečinok nahradiť týmto novým, alebo nie. Ak zvolí nie, v zadanom priečinku sa vytvorí priečinok, ktorému sa k názvu pridá podčiarnik a číslo, podľa toho koľko takých priečinkov sa tam už nachádza, počnúc jednotkou s automatickou inkrementáciou. Príklad: "nazovhry_2". O to sa stará metóda string *FileWithNumberedSuffix(string oldFilePath)*, ktorá vráti nový názov priečinka.

Do tohto priečinka sa vytvoria ďalšie priečinky, až to bude mať nasledujúcu adresárovú štruktúru:

-Hlavný priečinok

|

|- soundtracks

|

|- dialogues

|

|- dialogueTexts

|

|- *všetky systémové súbory a rozbalený priečinok site-packages*

Do podpriečinka soundtracks sa prekopírujú všetky súbory, ktoré uviedol používateľ ako soundtracky, z ich pôvodnej cesty (v danom TextBoxe) sem.

Do podpriečinka dialogues sa prekopírujú všetky súbory, ktoré uviedol používateľ ako dialógové zvukové súbory, z ich pôvodnej cesty (v danom TextBoxe) sem.

Do podpriečinka dialogueTexts sa prekopírujú všetky súbory, ktoré uviedol používateľ ako dialógové textové súbory, z ich pôvodnej cesty (v danom TextBoxe) sem.

Ak sú v daných textboxoch hodnoty "none" (v prípade soundtracku "*go*" alebo "*off*"), neskopíruje sa nič.

Generovanie podpriečinkov sa vykonáva vďaka *Path.Combine( )* (knižnica Path), ktorá dokáže jednoducho vytvárať a spájať fragmenty ciest, bez toho aby musel vývojár ručne dopĺňať napríklad lomítka či spätné lomítka medzi dva názvy priečinkov a podobne.

V programe je vytvorená premenná "*private bool isError = false;*", ktorá kontroluje či je všetko v poriadku. V prípade akéhokoľvek problému sa nastaví na true, spraví sa takzvaný "revert" – vrátenie zmien a celá adresárová štruktúra sa zmaže.

Následne sa do hlavného adresára nakopírujú všetky systémové súbory, vymenované vyššie.

V tejto metóde sa potom na konci zavolá metóda na vygenerovanie Python súboru.

Pred tým, ako sa vygeneruje kód v programovacom jazyku Python, vygeneruje sa textový súbor s názvom "elementValues.txt", do ktorého sa vo svojom poradí vypíšu hodnoty všetkých elementov. Až tento súbor sa následne číta, aby sa vygeneroval Python kód. Je to z dôvodu väčšej prehľadnosti kódu.

V metóde *CreateGameFolder( )* sa zavolá metóda *CreateElementValuesFile( )*, ktorý v hlavnom priečinku vytvorí textový súbor pre vypísanie hodnôt z elementov, tým že sa prechádza zoznam *ElementGroups*.

Do toho súboru sa zapíšu elementy v tom to poradí: popis, id (extrahuje sa z id_labelu), cesta k súboru soundtracku, omeškanie ukončenia GameCasu, podmienky spustenia GameCasu, počet možností a pre každý dialóg: cesta k zvukovému súboru dialógu, pan, omeškanie prehratia dialógu a cesta k textovému súboru dialógu. Každý element je označený vzorom "názov_elementu: hodnota elementu /nový riadok/". Príklad "id: 2" (nový riadok).

V prípade, že GameCase neobsahuje žiadne dialógy, vpíše sa text "no dialogue". V prípade že hra neobsahuje ani žiaden GameCase, vpíše sa text "no Gamecases". Po vypísaní všetkých elementov GameCasu (a jeho dialógov), vpíše sa do textu "SECTION ENDS", keď sa vypíšu všetky GameCasy, vpíše sa text "GAMECASES END". Popri generovaní textového súboru, sa rovno s tým kopírujú všetky súbory z ich pôvodnej cesty do ich novo-vygenerovaných adresárov. Stará sa o to metóda *CopyFileToNewDirectory(string zdrojovaCesta, string cielovyPriečinok)*. Do textového súboru sa vpisujú už tieto nové cesty k súborom. Dôležité je, aby dané súbory vo svojich pôvodných cestách existovali (prípadne aby tie TextBoxy obsahovali hodnotu, ktorá dáva najavo že žiadna cesta k súboru tam nie je).

Metóda *GeneratePythonGame( )* generuje kód v jazyku Python do predom vytvoreného a pripraveného systémového súboru "game_in_python.py" na základe práve vygenerovaného súboru "elementValues.txt". Hlavný kód hry je už napísaný, teraz sa do neho dopisuje iba zoznam GameCasov s jeho elementami. Metóda číta súbor po riadkoch, vyhodnotí čo sa v riadku nachádza a podľa toho to implementuje do Python kódu. Každému GameCasu ešte nastaví hodnotu "0" pre premennú *played* (nie je definovaná tu, až v Python súbore).

Na konci sa vytvorí kópia Python súboru, keďže aby mohla byť hra spustená, musí sa ešte dodatočne automaticky modifikovať. Toto zabezpečí "čistú verziu" pre toho, kto by do nej chcel nahliadnuť.

6 HRA V JAZYKU PYTHON

V súbore "*game_in_python.py*" je celý kód, na ktorom hra beží. Časť z neho sa generuje dynamicky na základe GameCasov.

Nosným pilierom programu je funkcia *process_instance( )*, ktorá volá väčšinu ostatných funkcii: na prehrávanie hudby, na prehrávanie dialógov, volá funkciu na spracovanie používateľského vstupu, funkciu na pauzu apod.

V programe je zoznam stringov *choices[ ]*, do ktorého sa ukladajú všetky voľby, ktoré hráč počas hrania zvolil.

V aplikácii (v tomto kontexte treba rozoznať pojem aplikácia – systém na tvorbu hier a aplikácia - hra) sa riešia vlákna, to len z dôvodu vypisovania tituliek na okno v GUI (vytvorené pomocou knižnice PySimpleGui).

Je nutné udržiavať si pri sebe všetky elementy jedného GameCasu, aby sa so samotným herným prípadom dalo dobre pracovať. Keďže sa počíta s tým, že GameCasov je v hre niekoľko, najlepšia možnosť je mať na ne zoznam. Lenže jeden GameCase pozostáva z niekoľkých hodnôt rôznych dátových typov. A taktiež dialóg GameCasu pozostáva z niekoľkých hodnôt rôznych dátových typov. Keďže dialógov môže jeden GameCase obsahovať tiež väčšie množstvo, je nutné ich tiež udržiavať v zozname. To znamená, že jeden GameCase obsahuje svoje hodnoty, ale aj zoznam ďalších hodnôt. Optimálnym riešením je vytvoriť si v Pythone triedu (s názvom Inštancia, čo môže byť zmätočný názov, avšak logicky pasuje do kontextu), v ktorej je vytvorená inicializačná funkcia. Týmto bude možné vytvoriť si zoznam inštancií, kde každá inštancia bude obsahovať niekoľko hodnôt a ešte k tomu ďalší zoznam.

Vyzerá to nasledovne:

```
class Instancia:
    def __init__(self, id, hudba, delay, conditions, options,
played, dialog):
        self.id = id
        self.hudba = hudba
        self.delay = delay
        self.conditions = conditions
        self.options = options
```

```
        self.played = played
        self.dialog = dialog #zoznam dialogov
```

Väčšina hodnôt patriacich GameCasu je už z tohto dokumentu známa, okrem predposlednej, ktorá značí či daný GameCase už nebol prehraný. Predvolene je hodnota nastavený na 0, teda že ešte nehral, avšak po prvom prehraní sa nastaví na 1, čo mu zabráni opätovnému prehraniu. Posledná hodnota "dialog" predstavuje celý zoznam dialógov.

Zoznam GameCasov je vo funkcii "*create_instance_list( )*", kde sa naplní hodnotami a inicializuje sa zavolaním tejto funkcie. Táto funkcia sa dotvorí až pri exporte hry, môže ale vyzerať nasledovne:

```
def create_instance_list( ):
    gcList = [
#toto je gamecase 0 - (vložený popis s GameCasu):
        Instancia("0", "none", 0.5, "none", "3", 0, [
           {"subor": "none", "pan": 1.0, "diDelay": 0.2,
"text": "none
           {"subor": "none", "pan": -1, "diDelay": 0.4,
"text": "none"},
        ]),
# toto je gamecase 0:
        Instancia("1", "off", 1.2, "none", "2", 0, [
           {"subor": "none", "pan": 0.3, "diDelay":
0.5,"text": "none"}
        ]),
    ]
    return zoznam
```

Autor nastavil všetkým cestám príkladu hodnotu na "none", aby sa jednalo o prehľadnejšiu ukážku. Ako vyplýva z príkladu, zoznam dialógov je v skutočnosti zoznam slovníkov (*dictionary*) Pythonu, kde každá *hodnota* má svoj *kľúč* (napríklad stringová hodnota cesty k súboru má kľúč "subor").

6.1 Priebeh funkcie process_instance( )

Hlavné parametre, ktoré táto funkcia prijme, sú "*inštancia*" (a s ňou aj prístup k všetkým elementom GameCasu a jeho dialógov), "*choices*" (všetky voľby, ktoré hráč doteraz v hre zvolil) a hudba (pred volaním funkcie je nutné inicializovať hudbu vo funkcii *main( )*).

Aby mohol byť GameCase prehraný, je nutné skontrolovať či hráčove voľby spĺňajú podmienky prehrania tejto scény. To zabezpečuje funkcia *conditionsMet( )* (viac v podkapitole 5.6). Ak žiadne podmienky nie sú, herný prípad môže byť prehraný. Súčasne s tým sa musí aj skontrolovať, či už daný herný prípad v minulosti nehral (môže iba raz za hru). Informácia o tom sa uchováva v premennej "played" danej inštancie.

Hneď na začiatku prehrávania scény sa premenná "*instancia.played*" nastaví na hodnotu 1, aby sa dalo najavo, že scéna bola prehraná (resp. teraz bude).

Pred prehraním dialógov a iných akcii je nutné spustiť hudbu na pozadí (music), teda soundtrack (samotná funkcia pre hudbu je popísaná v podkapitole 5.3). Najprv sa v podmienke skontroluje, aký text sa nachádza v ceste pre súbor soundtracku:

a) Ak je v premennej pre cestu k súboru text "off", hudba sa vypne pomocou príkazu *music.stop( )* (premenná "hudba" je jeden z parametrov funkcie. Hudba musela byť inicializovaná pred volaním funkcie pre prípad, ak by hneď prvý GameCase mal nastavenú na hudbu na vypnutú - inak by nebolo čo vypínať, ak by hudba neexistovala vôbec. Mimochodom, je možné vypnúť už vypnutú hudbu - (v prípade, že dva po sebe idúce GameCasy majú nastavenú hudbu na vypnutú - programu to nevadí)). Dialógy sa potom pustia do ticha. Premenná "*soundtrackIsOff*" sa nastaví na hodnotu 1 (premenná sleduje, či je hudba vypnutá).

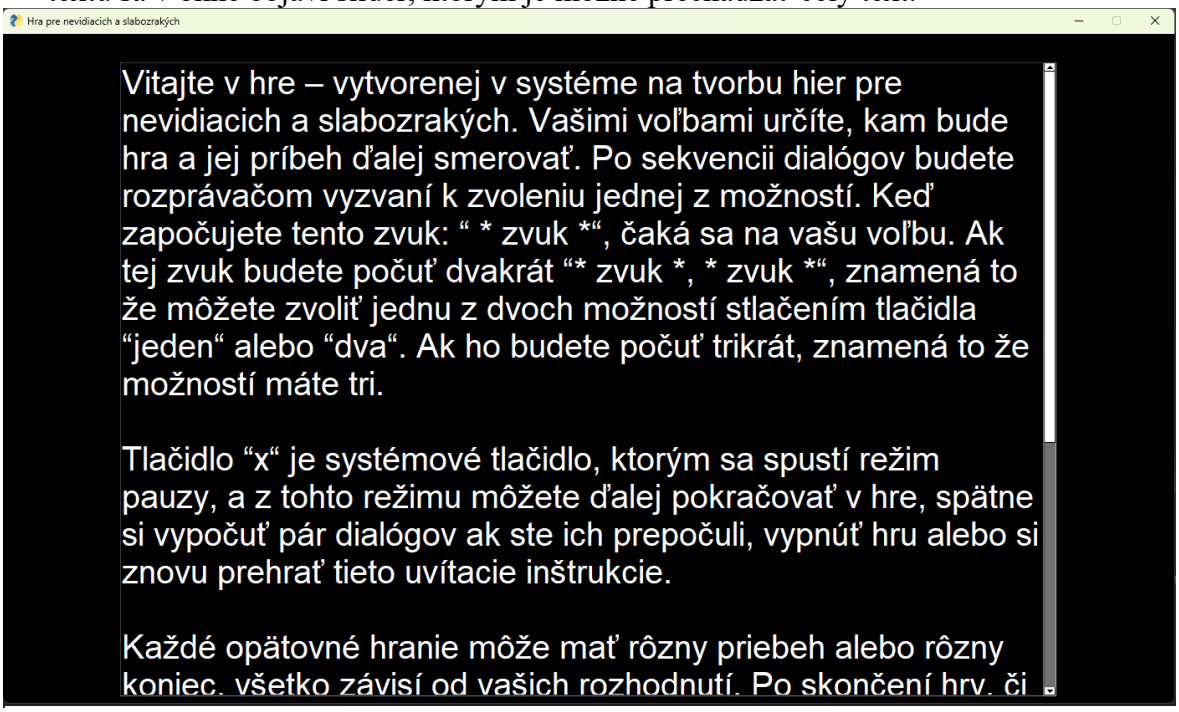
b) Ak je v premennej pre cestu k súboru skutočne cesta k zvukovému súboru, tak doteraz hrajúca hudba je vypnutá (vypnúť možno aj už vypnutú hudbu) a zavolaním funkcie *play_soundtrack( )* sa spustí aktuálne nastavená hudba. Premenná "*soundtrackIsOff*" je nastavená na hodnotu 0.

c) Ak je v premennej pre cestu k súboru text "go", tak sa skontroluje premenná "*previous_soundtrack*", čo je jeden z parametrov funkcie "*processInstance( )*", predstavuje hudbu, ktorá hrala v predchádzajúcej scéne (respektíve hodnotu tej hudby). Predvolene je tento parameter nastavený na "*None*", pretože nie je povinný (v prípade prvého GameCasu sa tento parameter nevkladá, pretože predchádzajúci

GameCase neexistuje, a v jazyku Python nie je podporované klasické preťažovanie funkcii, ako je tomu napríklad v jazyku C++). Čiže ak tento parameter existuje, ale jeho hodnota je "off", (hudba v predchádzajúcej scéne bola vypnutá), hudba sa štandardne vypne aj v tejto scéne (premenná "*soundtrackIsOff*" sa nastaví na hodnotu 1). Nakoniec sa do hodnoty pre cestu k súboru nastaví hodnota cesty pre predchádzajúci soundtrack. V prípade, že hudba v predchádzajúcej scéne vypnutá nebola, sa do premennej "hudba" vloží hudba, ktorá hrala v predchádzajúcej scéne (tiež ako parameter funkcie, nastavený predvolene na "None" z rovnakého dôvodu). Avšak nevloží sa tam ako cesta k súboru, ktorý má funkcia *play_soundtrack( )* prehrať, pretože by to spôsobilo to, že soundtrack sa začne prehrávať od začiatku a zároveň by prekryla aktuálne hrajúcu, tú istú hudbu (hrala by dvojmo). Je to vyriešené tak, že hudba bude plynulo pokračovať bez prerušenia alebo resetovania. V tomto prípade sa premenná "*soundtrackIsOff*" nastaví na hodnotu 0. Pojmom "prechádzajúci soundtrack" sa myslí hudba, ktorá hrala v scéne pred scénou, ktorá hrá teraz. Nemyslí sa tým scéna, ktorá je číselne pred aktuálnou scénou (scény môžu hrať aj mimo svojho poradia na základe ID, ktoré im bolo nastavené v GameCase).

Po prehraní hudby v pozadí sa prehrajú dialógy, v tom poradí ako boli používateľom pridané do scény. Zabezpečuje to cyklus for, ktorý prechádza celý zoznam dialógov (for dialog in instancia.dialog:). S dialógom sa zobrazia aj jeho titulky, ktoré používateľ vložil do GameCasu ako textový súbor. V Python kóde je to nazvané ako "*instancia.text*". V prípade, že cesta k tomuto súboru je "none", namiesto tituliek sa v okne zobrazí text "NEPRIDANÝ TEXT DIALÓGU". Ak tam bol pridaný skutočný súbor, tak sa tu otvorí a jeho obsah sa zobrazí v GUI, pomocou funkcie "*display_text( )*". Pri prehrávaní zvukového súboru dialógu nachádza svoje jediné využitie premenná "*soundtrackIsOff*". Pretože ak hudba v tejto scéne hrá, tak počas prehrávania dialógov sa automaticky stíši. Počas doby, kedy sa dialóg neprehráva, sa hlasitosť vráti opäť na svoju pôvodnú hodnotu. Aby sa to dosiahlo, musí sa pri volaní funkcie na pridanie dialógu vložiť ako parameter hudba, ktorá sa má stíšiť. V prípade, že hudba nehrá (soundtrack is off), nemá sa čo vložiť. Pôvodný zámer autora bol, že v prípade keď je hudba vypnutá, bude sa "prehrávať" úplne tichý zvukový súbor (napríklad 5 sekundový). Avšak od tohto nápadu sa upustilo, aby program nekonal zbytočné inštrukcie (prehrávanie tichého súboru), napriek tomu že to viedlo k omnoho dlhšiemu a zložitejšiemu kódu (neustále overovanie, či hudba je vypnutá alebo nie je) (je nutné rozlišovať "*None*" a "*none*", poznatok autora).

Ak cesta k zvukovému súboru nie je nastavená na “none“, prehrá sa pomocou vláknovania (konkrétne ide o vlákno “*audio_thread*“). Cieľová funkcia na prehranie dialógu je *play_dialogue( )* a ako argumenty berie všetky hodnoty dialógu. Vlákno na prehranie jednotlivého dialógu sa spustí a aby sa čakalo na dokončenie (a nezačal sa prehrávať ďalší dialóg, než ten pred ním skončí), vlákno sa “pripojí na koniec k ostatným“. V prípade dlhého textu sa v okne objaví slider, ktorým je možné prechádzať celý text.



Obrázok 11: Obrazovka hry – zobrazujú sa titulky uvítacích inštrukcii

6.1.1 Možnosti

Na konci funkcie sa spracujú voľby hráča, na základe možností, ktoré daná scéna má. Inicializuje sa premenná pre klávesu (*key*). V prípade, že miesto počtu možností bol v scéne nastavený koniec (END), program spustí zvukový súbor, ktorý oznámi koniec hry (end.wav) a vypne hru, späť do systému Windows (nastaví *exit_program.set( )*, čo sa neskôr vyhodnotí).

V prípade, že scéna má iba jednu možnú cestu, premenná pre klávesu sa automaticky nastaví na hodnotu 1, na používateľský vstup sa nečaká a hra pokračuje bez prerušenia ďalej.

V prípade, že scéna má dve alebo tri možnosti, prehrá sa zvukový súbor, ktorý informuje hráča o počte možností, z ktorého môže vyberať (optionsSound.wav). V nekonečnej slučke sa čaká na vstup (input) od používateľa. Slučka sa preruší a program bude pokračovať ďalej, až keď hráč zvolí platnú (valídnu) možnosť. To znamená, že ak má aktuálna scéna dve

možnosti, akceptovaný vstup od používateľa je iba klávesa "1" alebo "2" (ak má scéna tri možnosti, akceptované klávesy sú "1", "2", "3"). Okrem týchto kláves môže hráč spustiť režim pauzy, stlačením klávesy "x". V tomto prípade sa zavolá funkcia *pause_button( )*, ktorá sa vyhodnotí samostatne. Po skončení pauzy sa program dostane späť na toto miesto, kedy sa čaká na používateľov vstup (v prípade, že počas pauzy hráč nevypol hru).

Hráčova voľba sa vloží do zoznamu choices (v správnom formáte, teda číslo_gamecasu - bodka - číslo_voľby, napríklad 2.3). Pred tým, než scéna skončí, vyčká sa omeškanie (delay), ktorý tvorca hry pre túto scénu nastavil. Hráčove voľby z klávesnice zaobstaráva funkcia get_key_input( ).

Funkcia vráti (return) hodnotu 0, v prípade že scéna bola úspešne prehraná (ak boli splnené podmienky volieb, ako aj podmienka o tom, že scéna ešte nemohla byť prehraná). Ak podmienky spustenia scény splnené neboli, vráti sa hodnota 1. Spolu s hodnotou 0 alebo 1 sa vráti aj premenná hudby (*music*). Toto celé je pre to, aby sa poznala hodnota "*previous soundtrack*" pre nasledujúcu scénu.

6.2 Funkcia používateľského vstupu

O vstup z klávesnice sa stará funkcia *get_key_input( )*, ktorá neberie žiadne parametre. V nekonečnej slučke sa stará o to, či bola stlačené klávesa, ktorú hrá pozná ("1", "2", "3", "x") – za pomoci knižnice *keyboard*. Následne tú klávesu vráti.

6.3 Funkcia na prehrávanie hudby v pozadí

Hudbu spúšťa funkcia *play_soundtrack( )*, ktorá prijíma jeden parameter cesty k zvukovému súboru. Ten súbor následné pustí za pomoci knižnice *Pygame*. Prehrávanie súboru sa deje na pozadí, v inom vlákne, takže počas prehrávania hudby sa môžu prehrávať dialógy, alebo robiť akékoľvek iné operácie – nečaká sa na dokončenie prehrávania. Toho by sa program ani nedočkal, keďže parameter "-1" vo funkcii *play( )* zabezpečí, že súbor sa bude prehrávať v cykle donekonečna (až kým sa nenastaví nevypnutie niekde v programe). Funkcia sa spustí n + 1 krát (v prípade 0 sa spustí 1 krát, v prípade 4 sa hudba prehrá 5 krát), iba parameter -1 (alebo iné záporné číslo) zabezpečí, že po dokončení prehrávania súboru sa má prehrať znova.

6.4 Funkcia na prehranie dialógu

Dialóg (prípadne iný zvukový súbor okrem hudby) prehrá funkcia *play_dialogue( )*. Ako parametre funkcia prijíma cestu k zvukovému súboru dialógu (*filepath*), hodnotu stereo kanálu (*pan*), omeškanie spustenia (*delay*) a práve hrajúcu hudbu (*soundtrack*), ktorá je predvolene nastavená na "*None*", ako nepovinný parameter (ak žiadna hudba nehrá). Premenná dialógu sa nastaví ako *AudioSegment* (z knižnice Pygame) a počká sa požadovaná doba nastavená ako omeškanie. V prípade, že funkcia prijala parameter pre hudbu, tak ju stíši na 20%. Prehrá sa dialóg a hudba sa vráti na 100%. Ak parameter hudby nebol prijatý, iba sa prehrá dialóg. Dialóg sa prehrá pomocou Pygame funkcie *play( )*, ktorá prijme ako parameter nastavený audiosegment, ktorý zase zavolá funkciu *pan( )*, ktorý prijme ako parameter hodnotu stereo kanálu. Stereo kanál je v rozsahu -1 (zvuk bude úplne vľavo), až po 1 (zvuk bude úplne vpravo).

6.5 Funkcia režimu pauzy:

Do režimu pauzy sa hráč dostane keď pri voľbe možností bude jeho vstup "x". Vtedy sa zavolá funkcia *pause_button( )*, ktorá ako parametre prijíma dialóg, ktorý hral v scéne, počas ktorej bola hra prerušená, prijíma aj premennú, ktorá kontroluje či je hudba vypnutá alebo nie, prijíma samotnú hudbu a počet možností.

Automaticky sa prehrá zvukový súbor (pause_message.wav), ktorý hráčov informuje o režime pauzy a aké možnosti v ňom majú. Potom sa čaká opäť na vstup používateľa:

a) Ak hráč stlačí klávesu "x", pauza sa ukončí a hráč sa vráti do hry na miesto, kde skončil. Predtým ale bude informovaný o počte možností, medzi ktorými môže voliť (podľa toho sa prehrá súbor two_options.wav alebo three_options.wav). Na toto sa zavolá funkcia *pause_ends( )*, ktorá bola vytvorená na elimináciu duplicity kódu.

b) Ak hráč stlačí klávesu "1", prehrá sa znovu ten istý dialóg, ktorý sa prehral pred pauzou. Potom bude hráč vrátený do hry.

c) Ak stlačí klávesu "2", hráčovi sa opäť prehrajú úvodné inštrukcie o ovládaní hry (*welcome_instructions.wav*). Po tomto bude znovu hráč informovaný o počte možností ktoré má (funkcia *pause_ends( )*) a bude vrátený do hry.

d) Ak hráč stlačí klávesu "3", prehrá sa súbor "*end.wav*" a hra sa vypne

Všetky systémové zvukové súbory majú svoj súbor s titulkami, ktoré sa zobrazia v okne.

6.6 Funkcia na kontrolu podmienok

Pred prehraním inštancie sa musí skontrolovať, či sa prehrať môže. Na to sa volá funkcia *check_conditions( )*, ktorá prijíma parametre string, čo je reťazec podmienok danej scény a zoznam choices, teda zoznam možností, ktoré hráč počas hrania vykonal. Keďže podmienok môže byť v rámci jednej scény viac a môžu byť oddelené buď kľúčovým slovom "and" (a zároveň), alebo "or" (alebo), podľa toho, či musia byť splnené všetky podmienky alebo stačí iba jedna z nich, musí sa reťazec string rozdeliť (kde oddeľovačom je dané kľúčové slovo *and* alebo *or*), do zoznamu (*conditions_by_one*) – podmienky po jednom. Potom podľa toho, aké bolo kľúčové slovo, sa každá podmienka porovná s každou voľbou v zozname *choices*, či sa tam nachádza (v prípade or stačí len jedna z nich, v prípade and musia všetky). V prípade že podmienka je len jedna, jednoducho sa skontroluje či v zozname je. Podľa toho sa vráti premenná *isOk*, ktorá môže mať hodnotu 0 (nespĺňa podmienky), alebo 1 (spĺňa podmienky).

6.7 Funkcia na pridávanie tituliek

Funkcia *display_text( )* prijíma parameter text, ktorý pridá do predom pripravenej fronty textov. Samotné zobrazovanie textu sa vykonáva vo funkcii *thread_master( )*.

6.8 Funkcia sledujúca vlákna

Funkcia *thread_master( )* kontroluje, či bolo zavreté GUI okno. Ak áno, program sa vypne (avšak aktuálne hrajúci dialóg musí skončiť). Funkcia taktiež kontroluje frontu (*text_queue*), či je v nej nejaký text, ktorý by mohla zobraziť v GUI okne. Funkcia čaká na udalosť 100 milisekúnd.

6.9 Funkcia main( )

Funkcia *main( )* vytvorí list inštancií (GameCasov), zavolaním funkcie *create_instance_list( )*. Vytvorí zoznam hráčskych volieb (zatiaľ prázdny), inicializuje hudbu a pre každú inštanciu zavolá funkciu *process_instance( )*. Keď sa scéna úspešne prehrá, vráti sa na začiatok zoznamu inštancií a kontroluje, či niektoré staršie scény, pre ktoré hráč ešte nespĺňal podmienky, ich už nespĺňa aby mohla byť prehraná. V prípade, že scéna nebola prehraná (process_instance vráti 1), index sa zvýši a nasleduje ďalšia inštancia.

6.10 Používateľské rozhranie hry

GUI (grafické používateľské rozhranie) existuje len kvôli titulkám dialógov, inak by nebolo nutné. Používa sa k tomu rozhranie z knižnice *PySimpleGUI*. Farba pozadia sa nastaví na čiernu, farba textu na bielu, podľa zásad jasnej tlače. Z okrajov sa nastaví odsadenie, aby text nebol na okraje veľmi nalepený. *Text_element* je zatiaľ prázdny, v priebehu hry sa do neho budú vkladať texty tituliek. Jeho font je nastavený na Arial veľkosti 30. *Text_element* a odsadenia (*top* a *bottom_spacer*) sa pridajú do *layoutu* vo svojom poradí.

6.11 Spustenie funkcii

Na samom konci kódu sa mimo akýchkoľvek funkcií hra spustí. Nastaví sa hlavné vlákno jednej inštancie, počas ktorého bežania sa zavolá funkcia *thread_master( )*, ktorá všetko odštartuje.

6.12 Spustenie hry

Cieľom je, aby hra (jej Python skript) mohol byť spustený na počítači, kde Python nie je nainštalovaný (je nemysliteľné čakať od nevidiacich používateľov, aby si ho nainštalovali). Toto bol jeden z najťažších orieškov tohto systému. V .NETe existujú nuget balíčky (napríklad *Iron Python*), ktoré umožnia z Python kódu vytvoriť spustiteľný .exe súbor, k tomu je však potrebné mať na počítači nainštalovaný Python interpreter. Tento nápad musel byť zavrhnutý. V .NETe existujú aj balíčky (*nuget packages*), ktoré umožnia spustenie Python skriptu bez *Python interpreteru*, pretože ten je už obsiahnutý v balíčku (a musí podporovať Python verzie 3). Takýto nuget balíček sa volá "*Python Included*" Keďže ale Python skript vzniká dynamicky na počítači používateľa, nie je možné spraviť program, ktorý by napevno predvolený skript spúšťal. Taktiež potrebuje mať dodatočne v sebe obsiahnuté všetky balíčky, ktoré Python skript volá.

Z tohto dôvodu boli na počítači autora všetky Python balíčky stiahnuté regulérne cez príkaz "*pip install názovBalíčku*". Následne boli všetky skopírované a zabalené do archívneho .zip súboru (*site-packages*), ktorý bol nakopírovaný do zložky hlavnej aplikácie ako "*Resource*". Bolo rozhodnuté, že tento Python skript musí spúšťať separátny C# a .NET program, ktorý bol vyvinutý a jeho "releastnutá" .exe verzia bola taktiež nakopírovaná do hlavnej aplikácie ako "*Resource*". Ten program nesie meno EchoRealmGameStarter.exe, ale po exportovaní hry bude mať názov podľa názvu hry, pretože toto je práve ten súbor, ktorý si hráč spustí.

Balíček *site-packages* a spúšťací súbor hry sú pri exporte nakopírované do hlavného adresára hry. Python skript má všetky cesty k súborom nastavené ako relatívne cesty. Ale keďže spustený C# program spúšťa Python skript tiež z pohľadu relatívnej cesty a program si vytvára svoj pracovný adresár mimo adresára s hrou, bolo nutné, aby program, ktorý hru spúšťa ešte pred tým ako ju spustí, nastavil všetky cesty z Python skriptu na absolútne cesty a vytvorí kópiu Python skriptu s týmito úpravami. Keďže adresár s hrou môže byť kdekoľvek a na používateľskom počítači sa môže presúvať, je potrebné aby sa toto vykonávalo dynamicky, tesne pred spustením skriptu.

Spomínaný program disponuje dvomi funkciami:

a) Funkcia na úpravu Python skriptu – na nahrádzanie všetkých relatívnych ciest absolútnymi cestami

b) Funkcia na spustenie Python skriptu

7 DEMONŠTRAČNÁ HRA A ĎALŠÍ VÝVOJ

7.1 Implementácia demonštračnej hry

Súčasťou tejto práce je príloha v podobe skrátenej verzie vytvorenej jednej hry v tomto systéme (iba na ukážku toho, ako to zhruba vyzerá, hra končí predčasne na niekoľkých zvolených cestách). Autor nie je vlastníkom hudby, tá pochádza z videohry Gothic II (2002; developer: Piranha Bytes, composer: Kai Rosenkranz). Autor s pomocou dobrovoľníkov vytvoril scenár a dabing postáv. V rámci tvorby testovacích hier autor zaznamenával dĺžku trvania tvorby (práca na hre bola sporadická, nie je reprezentovaná striktne v *manday*-och):

a) Tvorba scenára: 7 dní

b) Dabing postáv (zorganizovanie hercov a nahranie): 7 dní

c) Implementácia hry do systému a jej export: 1 hodina

Príbeh hry je zasadený do stredoveku, sleduje vývoj hlavnej postavy pred vstupe do armády. Trvanie hry závisí na zvolenej ceste.

7.2 Ďalší vývoj

Zoznam nerealizovaných nápadov, ktoré budú doplnené v ďalších verziách systému:

a) Systém uloženia a otvorenia projektu: Aby bolo možné pokračovať v práci, ak počas nej musel tvorca aplikáciu alebo počítač vypnúť.

b) Väčšia interakcia systému s používateľom: Pridať viac oznámení (napríklad o úspešnom exporte apod.), ale aj upozorňovať na chyby (ak je v TextBoxe neočakávaná hodnota, napíše sa aká a kde).

c) Rozšírenie aplikácie na ďalšie platformy: Aplikácia je zatiaľ uzamknutá na platforme Windows, cieľom je expandovať aj na ďalšie platformy.

d) Rozšírenie exportu hry na ďalšie platformy: Cieľom je, aby mohla byť hra spustená na mobilných telefónoch, teda hlavne na operačných systémoch iOS a Android.

e) Zo záložky “Scenár“ vytvoriť plnohodnotný textový editor (aspoň so základnými funkciami inšpirovanými z MS Wordu).

f) Implementácia umelej inteligencie: V záložke “Scenár“ pribudne možnosť generácie scenáru jazykovým modelom, prípadne rovno predpripravenie GameCasov v záložke “GameSection“ umelou inteligenciou.

g) Rozšírenie podporovaných jazykov: Jazyk systému ako takého, tak aj systémových súborov.

h) Pridanie viac funkcií na tvorbu: Napríklad umožnenie používateľovi manipulovať s hlasitosťou súborov, nastavenie efektov ako fade in / fade out, možnosť pridania viacero zvukov naraz na pozadí na spôsob soundtracku, dynamické nastavovanie stereo kanálu, odstraňovanie pridaných elementov a ďalšie.

i) Zjednodušenie a optimalizácia skriptu hry v Pythone, hlavne čo sa týka threadingu, GUI a podobne, aby bol kód ešte viac prístupnejší a porozumiteľnejší pre programátorov, ktorý si chcú hru upraviť. Text tituliek sa tiež nebude rýchlo prepisovať ako je tomu doteraz, ale bude pribúdať k už zobrazenému textu a postupným posúvaním sa bude ukazovať nový.

j) Bezdrôtový ovládač: Miesto pripojenia ovládača s počítačom pomocou USB by bol ovládač pripojený pomocou technológie Bluetooth, do vnútra ovládača by sa vložila nabíjateľná batéria, nabíjaná pomocou USB.

k) Viac funkcií ovládača: Pridanie pohybového senzoru, prípadne vibračný motorček.

l) Automatické predčítavanie textov pomocou *Text-to-speech*, teda prevod textu na reč: Takouto technológiou bežne disponujú nevidiaci ľudia používajúci počítač, či už formou asistenčného softvéru alebo zabudovanou funkciou v prehliadači pri surfovaní na internete. Táto funkcia bola pôvodne zamýšľaná, ale nakoniec zavrhnutá, pretože generované vety by boli bez akéhokoľvek nádychu, prejavu a intonácie, čo by v hrách, kde sa dbá o atmosféru a vtiahnutie do deja bol problém. Táto funkcia sa však možno do systému raz implementuje, ak by mal tvorca hier problém zohnať hercov, alebo by umelecký prejav hra nevyžadovala (nebola by to "príbehovka")

m) Tvorba webovej stránky, kde bude k dispozícii tento softvér a na ktorú môžu ľudia pridávať svoje hry alebo ich sťahovať

ZÁVER

Výstupom tejto práce je softvér, ktorý umožňuje jednoduché vytváranie hier dostupných pre nevidiacich a slabozrakých hráčov. Tento nástroj je navrhnutý tak, aby bol intuitívny a vstupný prah pre jeho použitie bol čo najnižší, čím otvára dvere pre vytváranie mnohých nových hier.

Softvér má potenciál prispieť k zvýšeniu kvality života nevidiacich ľudí. Avšak, jeho vývoj je len začiatkom. Budúce úsilie by malo byť zamerané na šírenie povedomia o tejto problematike, ako aj na ďalšie zlepšovanie prístupnosti a otvorenosti počítačových hier, taktiež záujmom autora je prácu ďalej rozvíjať a vylepšovať aj po skončení štúdia, aby to nezostalo len pri bakalárskej práci.

Autor týmto konštatuje aj splnenie všetkých bodov zadania. V teoretickej časti je spravovaná literárna rešerš, systém je navrhnutý pomocou modelu funkčných požiadaviek v UML v spolupráci s textom na začiatku praktickej časti, navrhnutá a implementovaná demonštračná hra je súčasťou prílohy a dokumentácia systému je obsiahnutá v jadre praktickej časti.

ZOZNAM POUŽITEJ LITERATÚRY

[1] 5 facts about blindness and the blind. Envision [online]. March 2, 2023 [cit. 2023-04-16]. Dostupné z: https://www.letsenvision.com/blog/5-facts-about-blind-people-and-blindness

[2] How can people who are blind operate computers?. DO-IT, University of Washington [online]. 2021, 04/09/21 [cit. 2023-04-16]. Dostupné z: https://www.washington.edu/doit/how-can-people-who-are-blind-operate-computers

[3] Jasná tlač. Únia nevidiacich a slabozrakých Slovenska [online]. 2023 [cit. 2023-04-17]. Dostupné z: https://www.washington.edu/doit/how-can-people-who-are-blind-operate-computers

[4] NETO, Luiz Valério, Paulo H.F. FONTOURA JUNIOR, Rogério A. BORDINI, Joice L. OTSUKA a Delano M. BEDER. Design and implementation of an educational game considering issues for visually impaired people inclusion. Smart Learning Environments [online]. 2020, 7(1) [cit. 2023-04-10]. ISSN 2196-7091. Dostupné z: doi:10.1186/s40561-019-0103-4

[5] CLEMENT, Jessica. Number of video gamers worldwide in 2021, by region. Statista [online]. Oct 25, 2022 [cit. 2023-04-11]. Dostupné z: https://www.statista.com/statistics/293304/number-video-gamers/

[6] Global blindness was slowing prior to pandemic study reveals. Orbis [online]. 2021 [cit. 2023-05-09]. Dostupné z: https://www.orbis.org/en/news/2021/new-global-blindness-data

[7] World population. Countrymeters [online]. 2023 [cit. 2023-05-03]. Dostupné z: https://countrymeters.info/en/World

[8] JIRKOVSKÝ, Jan. Game industry: vývoj počítačových her a kapitoly z herního průmyslu. [Praha]: D.A.M.O., 2011. ISBN 978-80-904387-1-2.

[9] CLEMENT, Jessica. Mobile gaming market in the United States - Statistics & Facts. Statista [online]. Oct 18, 2022 [cit. 2023-05-03]. Dostupné z: https://www.statista.com/topics/1906/mobile-gaming/

[10] HOLMES, Oliver. No cults, no politics, no ghouls: how China censors the video game world. The guardian [online]. 15 Jul 2021 [cit. 2023-05-03]. Dostupné

z: https://www.theguardian.com/news/2021/jul/15/china-video-game-censorship-tencent-netease-blizzard

[11] LOBER, Andreas. A short history of banned games in Germany. GamesIndustry [online]. March 17, 2020 [cit. 2023-05-02]. Dostupné z: https://www.gamesindustry.biz/a-short-history-of-banned-games-in-germany

[12] ZAMORA, Gabriel. The Last of Us Remake Is Wholly Unnecessary. PCMag [online]. June 11, 2022 [cit. 2023-05-04]. Dostupné z: https://www.pcmag.com/opinions/the-last-of-us-remake-is-wholly-unnecessary

[13] What Is the Average Cost of Game Development?. Guru [online]. Jan 11, 2022 [cit. 2023-05-04]. Dostupné z: https://www.guru.com/blog/what-is-the-average-cost-of-game-development/

[14] 7 Key Roles in Video Game Development. Indeed [online]. March 10, 2023 [cit. 2023-05-05]. Dostupné z: https://www.indeed.com/career-advice/finding-a-job/game-development-roles

[15] LITTLE, Daniel. Audio Games: Exploring 6 Videogames for the Blind and Visually Impaired [online]. January 30, 2022 [cit. 2023-05-06]. Dostupné z: https://previewlabs.com/audiogames/

[16] The 10 Best Adapted Games for the Blind or Visually Impaired. Orcam [online]. 2022-10-20 [cit. 2023-05-06]. Dostupné z: https://www.orcam.com/en-us/blog/best-adapted-games-for-the-blind-or-visually-impaired

[17] STABLEY, Justin. Why developers are designing video games for accessibility. Pbs [online]. May 11, 2023 [cit. 2023-05-17]. Dostupné z: https://www.pbs.org/newshour/arts/why-developers-are-designing-video-games-for-accessibility

[18] ŘEHÁČEK, Denis. Development of games for blind users. Prague, 2017 [cit. 2023-04-02]. Bachelor Project. Czech Technical University in Prague. Vedoucí práce Doc. Ing. Daniel Novák, Ph.D.

[19] Audio Description (AD). Royal National Institute of Blind People [online]. [cit. 2023-04-05]. Dostupné z: https://www.rnib.org.uk/living-with-sight-loss/assistive-aids-and-technology/everyday-tech/tv-audio-and-gaming/audio-description-ad/

[20] Audiobooks: A Boon for People with Visual Impairment. Wecapable [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://wecapable.com/audiobooks-blind-people/

[21] HOLAN, Tomáš. Unity: první seznámení s tvorbou počítačových her. Praha: CZ.NIC, z. s. p. o., 2020 [cit. 2023-03-29], 1 online zdroj (176 stran). CZ.NIC. ISBN 978-80-88168-60-7.

[22] Audio overview. Unity Documentation [online]. 2023 [cit. 2023-05-19]. Dostupné z: https://docs.unity3d.com/Manual/AudioOverview.html

[23] Creating a custom module details panel. Unity Documentation [online]. 2023 [cit. 2023-05-19]. Dostupné z: https://docs.unity3d.com/Manual/Profiler-customizing-details-view.html

[24] Ukážte svoje nápady celému svetu. Websupport [online]. [cit. 2023-05-14]. Dostupné z: https://www.websupport.sk/webhosting/

[25] The Good and the Bad of .NET Framework Programming. Altexsoft [online]. 24 Dec, 2021 [cit. 2023-05-22]. Dostupné z: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/

[26] KHURPADERUSHI143. Introduction to Java Swing. GeeksforGeeks [online]. [cit. 2023-05-16]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-java-swing/

[27] MITCHELL, Robin. Which Programming Language Should I Choose? Graphics and GUIs. MakerPro [online]. Julz 20, 2018 [cit. 2023-04-11]. Dostupné z: https://maker.pro/custom/tutorial/which-programming-language-should-i-choose-graphics-and-guis

[28] GROOPMAN, Cynthia. The Importance of Touch. American Council of the Blind [online]. Feb 2022 [cit. 2023-05-01]. Dostupné z: https://www.acb.org/importance-touch

[29] MARGOLIS, Michael. Arduino cookbook. Second edition. Sebastopol: O'Reilly. 2011 [cit. 2023-05-07], , 699 s. ISBN 978-14-49313-87-6.

[30] ARDUINO MICRO S KONEKTORY. HWKitchen [online]. 2023 [cit. 2023-05-10]. Dostupné z: https://www.hwkitchen.cz/arduino-micro/

[31] STM32F103C8T6 ARM STM32 Minimum System Development Board Module For Arduino DIY Kit ST-Link V2 Mini STM8 Simulator Download. Aliexpress [online]. 2023 [cit. 2023-05-1000]. Dostupné z: https://www.aliexpress.com/item/32278016818.html?spm=a2g0o.productlist.main.5.1f0d7ae0KMnz1c&algo_pvid=b3339370-6ca6-4166-978c-3cdca946a47a&algo_exp_id=b3339370-6ca6-4166-978c-3cdca946a47a-2&pdp_npi=3%40dis%21EUR%214.55%211.01%21%21%21%21%21%4021224e9b16845799531893950d07d5%2167106876827%21sea%21SK%210&curPageLogUid=4bFjZeknFi3r

[32] Integrated Development Environment for STM32. STMicroelectronics [online]. 2023 [cit. 2023-02-20]. Dostupné z: https://www.st.com/en/development-tools/stm32cubeide.html

[33] FINLEY, Klint. Python Is More Popular Than Ever. Wired [online]. Mar 2, 2020 [cit. 2023-02-10]. Dostupné z: https://www.wired.com/story/python-language-more-popular-than-ever/

[34] AMIN, Faizan. Read And Process Audio Files Using Pydub In Python. MLHive [online]. November 22, 2022 [cit. 2023-03-21]. Dostupné z: https://mlhive.com/2022/11/read-and-process-audio-files-using-pydub-in-python

[35] FINCHER, Jon. PyGame: A Primer on Game Programming in Python: Sound Effects. RealPython [online]. 2016 [cit. 2023-02-17]. Dostupné z: https://realpython.com/pygame-a-primer/#sound-effects

[36] Tutorial: Create a new WPF app with .NET. Learn.Microsoft [online]. 02/08/202 [cit. 2023-02-01]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/get-started/create-app-visual-studio?view=netdesktop-7.0

[37] How to create a Grid in WPF Dynamically?. C-sharpcorner [online]. May 27 2012 [cit. 2023-02-03]. Dostupné z: https://www.c-sharpcorner.com/Resources/676/how-to-create-a-grid-in-wpf-dynamically.aspx

[38] What is Unified Modeling Language (UML)?. Visual-paradigm [online]. 2023 [cit. 2023-05-02]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

[39] GHEVARIYA, Ankit. DIY HID USB Keyboard Using STM32 [STM32 Tutorials] [HAL]. Instructables [online]. 2021 [cit. 2023-04-10]. Dostupné z: https://www.instructables.com/STM32-As-HID-USB-Keyboard-STM32-Tutorials/

[40] Https://www.drotik-elektro.sk/arduino-platforma/51540-sada-25-tlacidiel-s-klobucikom-pre-arduino.html. Drotik-elektro [online]. 2023 [cit. 2023-04-22]. Dostupné z: https://www.drotik-elektro.sk/arduino-platforma/51540-sada-25-tlacidiel-s-klobucikom-pre-arduino.html

[41] Vodiče samec a samec - 65 kusů. Dratek [online]. 2023 [cit. 2023-04-22]. Dostupné z: https://dratek.cz/arduino/827-vodice-samec-samec-65-kusu.html

[42] DuPont kabel F-F - 40x, 20 cm. Dratek [online]. 2023 [cit. 2023-04-22]. Dostupné z: https://dratek.cz/arduino/835-eses-40-x-f-f-dupont-kabel.html

[43] Shining 3D EinScan HX– Inspection Bundle. Eshhop.Edmasys [online]. 2023 [cit. 2023-04-11]. Dostupné z: https://eshop.admasys.sk/3d-skenery/shining-3d-einscan-hx-inspection-bundle/?gclid=CjwKCAjw36GjBhAkEiwAKwIWyc4_g7PIesbQ34nmKsjOJBwJNsdIlOMTLJNNdvE2TKGDtb9KxTWAmhoCwyMQAvD_BwE

[44] Stavebnice 3D tiskárny Original Prusa i3 MK3S+. Prusa3D [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://www.prusa3d.com/cs/produkt/stavebnice-3d-tiskarny-original-prusa-i3-mk3s-3/?gad=1&gclid=CjwKCAjw36GjBhAkEiwAKwIWyRCJGkAPbt2IIPrmCN2_wmW_vJQjmPcn5tUpBjzFB8R71i4qyX2TyxoCjRsQAvD_BwE

[45] ASTRA Plastelína 500g Biela, 303117002. Leoness [online]. 2023 [cit. 2023-03-21]. Dostupné z: https://leoness.sk/astra-plastelina-500g-biela-303117002/p9778?gclid=CjwKCAjw36GjBhAkEiwAKwIWyS-UuIc7-D6GhoZSha76abBrGZ0yCZAk1h_WW_r7LdU-C7UOl9jWrRoCk-gQAvD_BwE

[46] Nepájivé pole 400 bodov. TechFun [online]. 2023 [cit. 2023-04-24]. Dostupné z: https://techfun.sk/produkt/nepajive-pole-400-bodov/

[47] Univerzální vrtaný plošný spoj 5x7 cm, 432 pin, oboustranný. MyArduino [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://www.myarduino.cz/kontaktni-pole-pajive/univerzalni-vrtany-plosny-spoj-5x7-cm-432-pin-oboustranny

[48] Vention Luxury USB 2.0 -> micro USB Cable 3A Gray 1 m Aluminum Alloy Type. Alza [online]. 2023 [cit. 2023-04-10]. Dostupné z: https://www.alza.sk/vention-luxury-usb-2-0-microusb-cable-3a-gray-1m-aluminum-alloy-type-d5858212.htm?o=9

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

GUI Grafické používateľské rozhranie (z anglického Graphical User Interface)

RQ Požiadavka (z anglického Requirement)

RQG Požiadavka hry (z kombinácie anglických slov Requirement a Game - hra)

UI Používateľské rozhranie (z anglického User Interface)

WPF Windows Presentation Foundation

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha P I: CD disk

PRÍLOHA P I: CD DISK

Priložený CD disk obsahuje:

- elektronickú verziu bakalárskej práce vo formáte PDF/A -fulltext.pdf

+Balíček prilohy.zip:

- elektronickú verziu bakalárskej práce vo formáte .pdf - fulltext.pdf
- zdrojové kódy hlavnej aplikácie zabalené v balíčku - aplikacia.zip
- zdrojové kódy spúšťacieho programu zabalené v balíčku - starter.zip
- zdrojové kódy ovládača zabalené v balíčku - ovladac.zip
- naskenovaný 3D model ovládača vo formáte stl - ovladac.stl
- návrh systému vo formáte .eapx - navrh.eapx
- demonštračnú hru zabalenú v balíčku - vzorova_hra.zip