Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra

bakalářská práce

Interaktivní didaktická aplikace pro výuku základů ovládání počítače v nástroji Unity 3D

Vypracoval: Ondřej Strmiska
Vedoucí práce: doc. PaedDr. Jiří Vaníček, Ph.D.

České Budějovice 2017

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej STRMIŠKA**
Osobní číslo:	**P120266**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Interaktivní didaktické aplikace pro výuku základů ovládání počítače v nástroji Unity 3D**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Unity 3D je flexibilní a výkonná vývojová platforma pro vytváření multiplatformní 3D a 2D her s interaktivními prvky zážitkového charakteru. Jde o komplexní systém pro vytváření high-endového obsahu, má řadu funkcí, umožňující pokročilé simulace reálných situací jako interaktivitu objektů, gravitaci apod, matematické enginy apod.

Student vytvoří herní didaktickou interaktivní aplikaci určenou pro žáky 1. stupně základní školy, sloužící k naučení se základnímu ovládání počítače. Aplikace bude vytvářena pomocí vývojového prostředí Unity 3D.

Aplikace bude tematicky rozdělena na několik témat (např. psaní na klávesnici, používání ovladačů, dvojklik, vyplňování apod.), každá se bude skládat ze série aktivit vzhledu miniher. Výběr a scénář těchto miniher bude určen na základě výsledků anylýzy průzkumu mezi budoucími učiteli 1. stupně a jejich hodnocení podobných aplikací používaných v současných učebnicích informatiky.

K úlohám bude vytvořen stručný manuál a metodické komentáře (co která minihra učí a jak se ovládá). Vytvořená aplikace bude vyzkoušena v praxi na 1. stupni ZŠ a poté bude upravena a odstraněny případné chyby a nedostatky.

Rozsah grafických prací:	**CD ROM**
Rozsah pracovní zprávy:	**40**
Forma zpracování bakalářské práce:	**tištěná**
Seznam odborné literatury:	**viz příloha**

Vedoucí bakalářské práce:	**doc. PaedDr. Jiří Vaníček, Ph.D.** Katedra informatiky
Datum zadání bakalářské práce:	**22. března 2016**
Termín odevzdání bakalářské práce:	**30. dubna 2017**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 22. března 2016

Prohlášení

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

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

Poděkování

Na tomto místě bych rád poděkoval doc. PaedDr. Jiřímu Vaníčkovi, Ph.D. za cenné připomínky a odborné rady, kterými přispěl k vypracování této diplomové práce. Zároveň bych chtěl poděkovat studentům oboru Učitelství pro 1. stupeň základních škol za otestování aplikace, nejrůznější připomínky a návrhy na vylepšení.

Abstrakt

Cílem této bakalářské práce je vytvoření herní didaktické aplikace určené pro žáky 1. stupně základní školy, která slouží k naučení se základům ovládání počítače. Aplikace bude vytvořena pomocí vývojového prostředí Unity 3D.

Unity 3D je herní vývojářská platforma umožňující vytvořit 2D nebo 3D aplikace pro počítače, konzole, mobilní zařízení, webové stránky atd.

V teoretické části byl proveden průzkum již vytvořených podobných aplikací používaných v současných hodinách informatiky. Na základě tohoto průzkumu byla provedena analýza a dle výsledků této analýzy byla vybrána témata, která byla v aplikaci vytvořena.

V praktické části proběhlo samostatné programování jednotlivých témat v programu Unity 3D, z kterých byla později vytvořena jedna kompletní aplikace. Vytvořená aplikace byla otestována studenty oboru Učitelství pro 1. stupeň základních škol a podle jejich reakcí byla aplikace doladěna.

K aplikaci byly vytvořeny webové stránky s odkazem na spuštění nebo stažení aplikace. Tyto webové stránky jsou umístěné na školním portálu a jsou dostupné z odkazu: http://home.pf.jcu.cz/jop/.

Klíčová slova

Unity 3D, didaktická hra, základy ovládání počítače, výuka ICT, základní škola

Abstract

Aim of this bachelor thesis is creating educational game application, designed for pupils of the first grade of elementary school, used to teach the basic of computer control. Application will be created using the Unity 3D engine.

Unity 3D is gaming development platform that allows to create 2D or 3D applications for computers, consoles, mobile devices, websites, etc.

In the theoretical part of a survey was conducted of already created similar applications used in contemporary computer classes. Based on the analysis results were chosen themes, which were created in application.

In the practical part were programmed individual topics in the Unity 3D, of which were later made one complete application. Created application was tested by students of Teacher 1st grade primary school and application was modified by their response.

For application were created website, with a link to launch or download the application. These websites are located on the school portal and they are accessible from the link: http://home.pf.jcu.cz/jop/.

Keywords

Unity 3D, didactic game, basic of computer control, ICT education, elementary school

Obsah

1 Úvod

Počítačové hry či videohry slouží převážně k zábavě, ovšem velké množství her slouží také k výuce a rozvoji schopností, jako jsou postřeh, přesnost, okamžité rozhodování atd.

Uživatelé počítačových her si často ani neuvědomují, že během hraní se učí novým věcem a zlepšují se v nich. Velké množství her, které zdánlivě slouží především pro zábavu, bylo vytvořeno k edukačním účelům, zářným příkladem takových her jsou hry od společnosti Microsoft, která tyto hry zabudovala do operačních systémů Windows[1].

První hra tohoto typu přišla do operačního systému Windows 3.0 v roce 1990, jedná se o hru Microsoft Solitaire. Kromě toho, že tato hra ukázala uživatelům možnost hraní karetních her bez fyzického balíčku karet, také sloužila k tomu, aby si uživatelé osvojili práci s myší.

Uživatelé do té doby pracovali většinou s příkazovým řádkem a neznali pojem jako je drag and drop, který je už řadu let běžným základem ovládání počítače.

O 2 roky později, s operačním systémem Windows 3.1, byla představena hra Minesweeper, v českém překladu Hledání min. Účelem této hry bylo ukázat lidem koncept klikání levým a pravým tlačítkem. Microsoft chtěl, aby se uživatelé operačního systému Windows naučili přirozeně klikat oběma tlačítky a naučit je to formou hry jim připadalo jako nejlepší nápad. [1]

[1] Od roku 2012, kdy vyšla Windows 8, byly tyto hry odebrány a hry byly oficiálně k dostání pouze z Windows Store. Ovšem s verzí Windows 10 se vrátila hra Solitaire.

1.1 Východiska práce

K učebnicové předloze Informatika pro 1. stupeň základní školy[1] byly vytvořeny aplikace v programu Imagine Logo. Ovšem program Imagine Logo nenabízí takové možnosti tvorby aplikace jako jiná vývojářská prostředí. Proto aplikace vytvořené v Imagine Logu mohou vypadat jednotvárně, ovládání aplikace je celkem omezené a v neposlední řadě Imagine Logo vykazuje různé chyby, jako je špatné zobrazení při určitých rozlišení atd.

Z toho důvodu se začala jednotlivá témata vytvářet v moderním enginu Unity 3D, který nabízí mnohem více možností vývoje herní aplikace, jako jsou různé animace, skriptování, práce s fyzickými prvky atd.

Unity 3D patří mezi nejpopulárnější herní vývojářské nástroje současnosti a to z důvodu možnosti programovat pro všechny nejpoužívanější herní platformy, počítače, konzole, mobilní zařízení, webové stránky atd. Unity 3D umožňuje vytvářet jednoduché i složitější 2D nebo 3D aplikace. Program umožňuje programovat v jazyce C# a Javascript.

V tomto programu se již v rámci naší školy naprogramovalo několik témat, ze kterých bude tato bakalářská práce vychází. Témata těchto již naprogramovaných aplikací jsou Tahání myší, Klikání myší a Psaní na klávesnici.

Tato témata již byla vytvořena v předmětu Tvorba didaktického software. Zatímco aplikace na témata Tahání myší a Klikání myší byla vytvořeny Alešem Velkem a Kateřinou Márovou. Aplikaci na téma Psaní na klávesnici jsem vytvořil já.

Každá z těchto aplikací je souborem několika úloh, které mají za úkol pomocí pěkného vizuálního zpracování naučit děti 1. stupně na základních školách dané téma. Obtížnost her a grafická stránka hry je určena pro děti ve věku 7–11 let. Mezi úlohami se uživatel pohybuje pomocí několika tlačítek. v každé hře má uživatel instrukce, jak má v úloze postupovat.

V některých úlohách si může uživatel vybrat obtížnost a úlohu libovolně opakovat, tím se zlepšovat v daném aspektu ovládání počítače. Pro příklad uvedu: v aplikaci na téma Píšeme na klávesnici, se v jedné hře uživateli zobrazují slova, kterých musí napsat za 30 vteřin co nejvíce, tato slova jsou vybrána podle obtížnosti, kterou si uživatel vybral. Po uplynutí 60 vteřin se uživateli zobrazí hodnocení.

1.2 Cíle práce

Jedním z cílů práce je analyzování aktuálně používaného softwaru pro výuku ovládání počítače na základních školách. Dalším cílem je analyzování již vytvořených aplikací v enginu Unity 3D (seznámení se strukturou aplikací) a seznámení se s potřebnými funkcemi Unity 3D pro vytvoření úloh na téma Ovladače (posuvníky, tlačítka, přepínače), Kreslení čar (rovné čáry, libovolné čáry), Psaní na klávesnici (textová pole) atd.

Hlavním cílem práce je vytvořit herní didaktickou interaktivní aplikaci určenou pro žáky na 1. stupni základní školy. Pro aplikaci vytvořit úlohy dosud nezpracovaných témat (Doplňování a úprava textu, Kreslení rovných čar, Kreslení a Obarvování, Ovladače). Vytvořit úlohy buď kompletně na základě učebnicové předlohy, nebo touto předlohou inspirovány, případně vytvořit kompletně nové úlohy na určité téma.

Vytvořit úlohy tak, aby hráč měl instrukce, co má v dané úloze dělat a je hráči poskytovaná zpětná vazba. V některých úlohách umožnit volbu obtížnosti úlohy. Vytvořit úlohy tak, aby je hráč mohl hrát v libovolném pořadí.

Výstup z enginu Unity 3D je portable aplikace (aplikace se pustí bez nutnosti jakékoliv instalace).

Po dokončení aplikaci vyzkoušet buď přímo mezi žáky nebo mezi studenty učitelského oboru pro 1. stupeň základní školy. Na základě jejich reakcí, návrhů a připomínek odstranit případné chyby.

1.3 Metoda práce

Na počátku práce jsem analyzoval hry, které byly vytvořené k učebnici Informatika pro 1. stupeň základní školy[1], podle kterých byly vytvářeny jednotlivé aplikace v Unity 3D. Následně jsem se kompletně seznámil s funkcionalitou již naprogramovaných aplikací (spouštění úloh, pohyb kamery, tlačítka, skripty atd.) a naučil jsem se pracovat s potřebnými funkcemi programu Unity, kterými jsou například posuvníky, tlačítka, čáry, textová pole atd.

v druhé fázi práce jsem začal vytvářet jednotlivé úlohy, které byly rozděleny do jednotlivých aplikací podle tematické sady, kterými jsou Ovladače, Kreslení rovné čáry atd. Zároveň jsem aktuální stav jednotlivých úloh i aplikací pravidelně konzultoval s vedoucím práce a domlouval se na dalších postupech tvorby aplikace.

V momentě, kdy byly vytvořeny všechny úlohy, zkompletoval jsem všechny aplikace (včetně aplikací od jiných autorů) do jedné aplikace. K této aplikaci jsem vytvořil webové stránky, ve kterých je možné hru spustit ve webovém prohlížeči nebo z nich hru stáhnout, případně se dozvědět o hře některé informace. Tyto stránky jsem nahrál na školní portál.

Finální verze aplikace byla předána studentům oboru Učitelství pro 1. stupeň základních škol, kteří aplikaci vyzkoušeli a ohodnotili jednotlivé úlohy. Na základě jejich návrhů a připomínek byly některé úlohy upraveny, případně kompletně přepracovány.

2 Učebnicová předloha

Témata úloh této bakalářské práce vychází z první kapitoly knižní předlohy Informatika pro 1. stupeň základní školy[1]. K této učebnici jsou dostupné přílohy v podobě aplikací, které obsahují úlohy zmíněné v učebnici, tyto přílohy jsou dostupné na stránkách vydavatele.

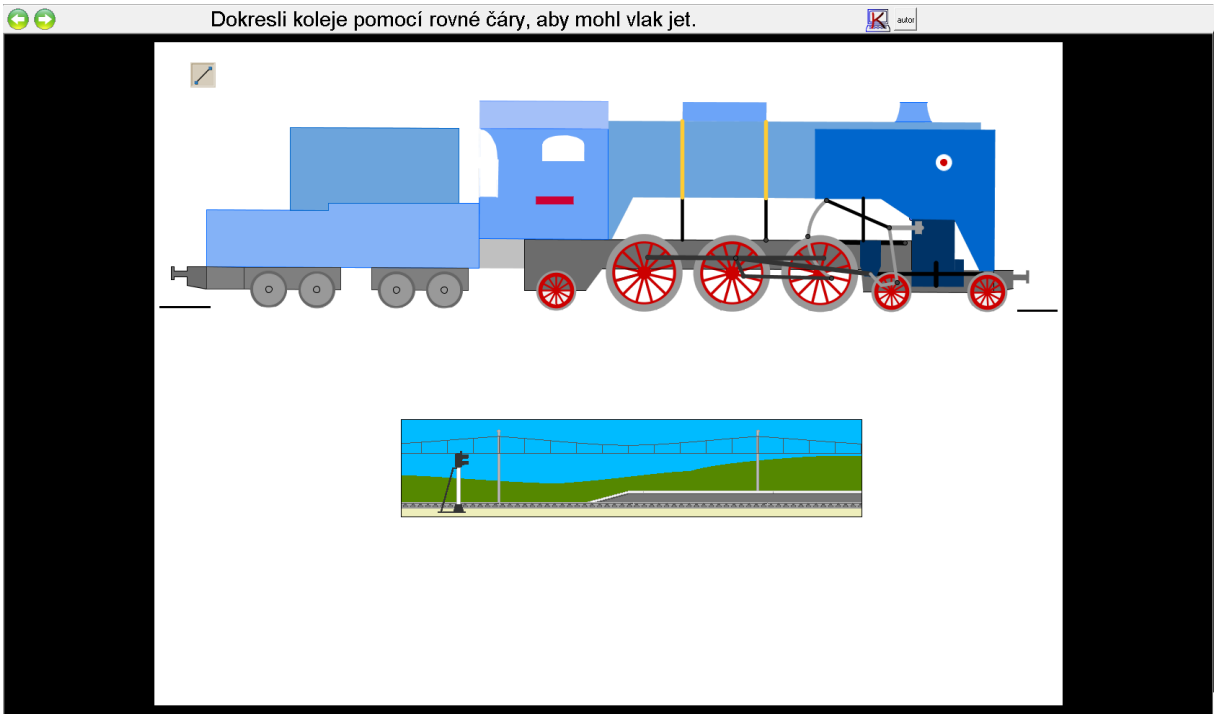
V celé této první kapitole učebnice jsou popsány úplné základy ovládání počítače. Kapitola je rozdělena do jednotlivých částí, z nichž každá věnuje některému aspektu ovládání počítače a zároveň obsahuje metodické doporučení pro učitele.

Tato doporučení se týkají nejen konkrétních aspektů ovládání, ale i například spustitelnosti aplikací, např. „Je nezbytné umístit soubory tak, aby byly co nejsnáze dostupné (například na plochu nebo z Plochy spustitelné). s úlohami budou pracovat začátečníci. Jestliže dítě ještě neumí pořádně klikat myší, těžko bude hledat ve složkách nebo v nabídce Start“[1].

2.1 Aplikace

Jednotlivá témata úloh jsou rozdělená do jednotlivých aplikací (některá témata jsou ve více aplikacích), všechny tyto aplikace mají podobný koncept. v každé aplikaci se nachází „uvítací stránka“, ve které jsou informace o aplikaci. Každá aplikace má horní lištu, ve které jsou umístěné šipky, pomocí kterých se uživatel pohybuje mezi jednotlivými úlohami. Uprostřed herní lišty je text, ve kterém se vyskytuje zadání jednotlivé úlohy, případně je tento text změněn jako zpětná vazba pro hráče. v pravé části horní lišty se nachází tlačítka s informacemi o tom, zda aplikace kontroluje danou úlohu a tlačítko s informacemi o autorovi aplikace.

Všechny tyto aplikace byly vytvořeny programem Imagine Logo – jedná se o kompletně objektový jazyk, který je řízen událostmi, podporuje paralelní programování a má též propracovanou ideu obrázkových tvarů želv. Má některé nové prvky, které jsou typické pro programy pod Windows, např. překrývající se grafické plochy (jako listy papíru), tlačítka i s obrázky, posuvné lišty, textová pole, lišty tlačítek apod. [3]



Obrázek 1 – Náhled na aplikaci, sloužící jako příloha učebnice

2.2 Jednotlivá témata úloh

Jezdíme po obrazovce

V tomto tématu jsou úlohy zaměřené na pouhé posouvání myší bez nutnosti klikání na tlačítka. Nachází se zde například úloha na rozpoznání jednotlivých součástí počítače (myš, klávesnice atd.), úloha kontroluje při najetí kurzorem myši nad vyžadovaný obrázek součásti počítače.

Klikáme myší

Jak už název napovídá, v tomto tématu jsou úlohy zaměřené na klikání myší. v úlohách se procvičuje klikání levým i pravým tlačítkem. Jedním z příkladů je úloha, ve které musí hráč kliknout na zvířátka, která mají přesně 4 nohy.

Taháme věci

Toto téma učí děti funkci jménem Drag and drop, tedy uchycení předmětu a přesunutí předmětu na jiné místo. Jednou z úloh je přesunutí zvířátka na místo, na které patří (vyznačené siluetou zvířátka).

Dvojklik

U tématu dvojklik se děti učí dvojité kliknutí levým tlačítkem myši. Úlohy na toto téma například spustí animaci při kliknutí na obrázek nebo měří rychlost dvojkliku apod.

Kreslíme čáry

V tomto tématu se učí děti kreslit čáry. v aplikaci se začnou kreslit čáry v momentě, kdy hráč stiskne levé tlačítko myši a táhne myší. Například se kreslí nejkratší cesta mezi kostičkou a psem.

Kreslíme a vybarvujeme

Toto téma je rozděleno na 2 aplikace a děti se v něm učí kreslit a vybarvovat obrázky. v jedné aplikaci se pouze vybarvují obrázky, například se mají vybarvit všechny hvězdy žlutě, obdélníky hnědě atd. v druhé aplikaci je možné i kreslit, například zakreslit mezeru a následně pomocí nástroje „plechovka“ obarvit obrázek.

Kreslíme čáry a tvary

Toto téma je také rozděleno do 2 aplikací a děti se v něm učí kreslit rovné čáry nebo různé geometrické tvary. v jedné aplikaci se kreslí pouze rovné čáry, například má dítě za úkol nakreslit pavučinu pro pavouka. Ve druhé aplikaci hráč kreslí pomocí nástrojů, například hráč pomocí obdélníků kreslí dům.

Ovladače

V tomto tématu se učí dítě pracovat s posuvníky, zaškrtávacími políčky a tlačítky. Příkladem takové úlohy je zaškrtávání obrázků, které patří do určitého ročního období.

Psaní na klávesnici

Toto téma v učebnici seznamuje dítě s rozložením kláves na klávesnici a vysvětluje funkce hlavních kláves. v jednotlivých úlohách dítě například hledá požadovaná písmenka na klávesnici.

3 Unity 3D – o programu

V celé této kapitole jsou popsány veškeré základní informace o programu Unity (licencování, výstupní platformy atd.), základy práce v Unity Editoru, dále jsou zde popsány jednotlivé aspekty, pomocí kterých je tvořena aplikace. Je zde uveden Layout celé aplikace, základy skriptování v Unity, včetně uvedení základních příkazů.

3.1 Základní informace o programu

Unity 3D je multiplatformní herní engine vyvinutý společností Unity Technologies, který umožňuje vývoj 2D i 3D her libovolného žánru. Unity 3D je nabízeno v různých plánech pro nekomerční i komerční využití. Již v základní bezplatné verzi jsou umožněny téměř všechny možnosti programu. Unity 3D nabízí nejrůznější funkce od tvorby propracovaných animací, práci s grafickými objekty s 2D i 3D fyzikou, přes skriptování v různých jazycích.

Ve všech verzích programu je možné pracovat jak samostatně, tak s týmem pomocí služby Unity Cloud, která umožňuje práci více lidí na projektu současně. Engine nabízí zároveň službu Unity Ads, která slouží primárně pro vývojáře mobilních her, tato služba nabízí vývojářům možnost zpeněžit svůj projekt a oslovit nové publikum pomocí videoreklamy.

Unity 3D je nabízen v různých edicích. Podstatné je, že pokud aplikace nevydělává více než 100 000 dolarů za rok, je možné využít personální edici, která je kompletně zdarma a kromě několika málo funkcí obsahuje veškeré možnosti programu. Zároveň v programu vytvořeném personální edicí se při startu aplikace zobrazuje logo Unity. Další edice obsahují lepší skiny uživatelského rozhraní, rychlejší cloud atd.

V tomto enginu již bylo vytvořeno mnoho her od menších i velkých vývojářů. V enginu je možné vybudovat hry různých žánrů, od budovatelských strategií, přes logické nebo karetní hry, po akční adventury. Nejznámější hry (Hearthstone, Pokémon Go, Angry Birds 2, Cities: Skylines atd.).

3.1.1 Unity Platformy

Velikou výhodou využití Unity 3D oproti jiným herním enginům je veliká multiplatformní podpora. Samotní vývojáři používají motto „Build once, deploy anywhere“, v překladu „Vytvoř jednou, použij kdekoliv“. Ve výsledku to zjednodušeně vypadá tak, že si vývojář zvolí platformy, pro které chce aplikaci vytvořit, k těmto platformám si stáhne a nainstaluje optimalizační balíčky a následně už si jenom vývojář v Unity vybere, pro kterou platformu si přeje aplikaci vytvořit.

Proces vytváření aplikace může zabírat až několik hodin (v závislosti na různých nastavení projektu a výběru platformy). Pokud se proces vytváření aplikace úspěšně dokončí, stačí už jenom aplikaci nahrát na požadované zařízení a tam aplikaci spustit.

Seznam jednotlivých podporovaných platforem[4]:

Desktopová zařízení

1. Windows – operační systém od společnosti Microsoft.
2. Windows Store Apps – digitální distribuční platforma pro aplikace na systémy Windows.
3. Mac – operační systém pro počítače od firmy Apple.
4. Linux / Steam OS – operační systém založený na linuxovém jádru / Steam OS – linuxová distribuce založena na distribuci Debian, kterou vyvíjí společnost Valve Corporation.
5. Facebook Gameroom – aplikace určená pro Windows 7 a vyšší.

Webová rozhraní

6. WebGL – JavaScriptové API pro zobrazování interaktivní 3D grafiky. WebGL programy jsou vykonávány na grafické kartě počítače. WebGL je vyvíjeno a spravováno neziskovou organizací Khronos Group. WebGL plugin je již implementovaný ve všech prohlížečích.

Mobilní platformy

7. iOS – mobilní operační systém vytvořený společností Apple Inc.
8. Android – mobilní operační systém založený na linuxovém jádru.
9. Windows Phone – mobilní operační systém firmy Microsoft.
10. Tizen – operační systém založený na linuxovém jádře, využívá se v některých zařízeních vyrobených převážně od firmy Samsung.
11. Fire OS - Android – mobilní operační systém vytvořený firmou Amazon.

Konzolová zařízení

12. PlayStation 4 – herní konzole od společnosti Sony Computer Entertainment.
13. PS Vita – herní kapesní konzole firmy Sony Computer Entertainment.
14. Xbox One – herní konzole od společnosti Microsoft.
15. Wii u – herní konzole vyráběná společností Nintendo.
16. Nintendo 3DS – přenosná herní konzole vyrobená firmou Nintendo.
17. Nintendo Switch – herní konzole vyráběná společností Nintendo.

Zařízení s virtuální realitou

18. Oculus Rift – headset pro vstup do virtuální reality vytvořený společností Oculus VR.
19. Google CardBoard – platforma vyvíjená společností Google.
20. Steam VR – platforma vyvíjená společností Steam.
21. PlayStation VR – platforma vyvíjená společností Sony Computer Entertainment.
22. Gear VR – platforma vyvíjená společností Samsung Electronics.
23. Microsoft Hololens – mix rozšířené reality a chytrých brýlí od společnosti Microsoft.
24. DayDream – platforma vyvíjená společností Google.

Chytré televize

25. Android TV – smart TV vyvíjená firmou Google.
26. Samsung Smart TV – smart TV vyvíjená firmou Samsung.
27. tvOS – smart TV vyvíjená firmou Apple Inc.

3.2 Základní pojmy v Unity

Celá hra v Unity je tvořena z jednotlivých scén, každá hra musí mít minimálně jednu scénu, běžné je využít minimálně jednu scénu pro hlavní menu a jednu scénu pro danou hru. v jednotlivých scénách je hra vytvořena z tzv. objektů (anglicky Game Object).

3.2.1 Projekt

Projekt představuje hru jako celek. Každý projekt má několik parametrů:

- Název projektu – název hry.
- Lokaci – umístění na pevným disku, při tvorbě projektu se automaticky vytvoří složka s názvem projektu, ve které se automaticky vytvoří několik složek (Assets, Library, ProjectSetting, Temp).
- Organizaci (volitelné) – umožnuje použití některých Unity služeb.

V každém projektu se nastavují využívané scény, platforma hry a další nastavení, které jsou určeny platformou hry.

3.2.2 Scény

Scény jsou používány pro vytvoření jednotlivých úrovní, hlavního menu a všeho dalšího, co hra obsahuje. Každá scéna by měla být unikátní část hry. v každé scéně se umisťuje prostředí, dekorace, v podstatě se ve scéně navrhuje a tvoří hra. Obsah scény je vytvořen z objektů.

3.2.3 Objekty

Představují ve hře veškeré postavy, rekvizity a scenérie[5]. Samotné objekty toho moc nedělají, ale slouží jako kontejnery pro komponenty, které zajišťují veškeré funkcionality, více informací o komponentech je uvedeno v kapitole Komponenty objektu. Tyto objekty tvoří ve hře prakticky všechno, každý objekt musí mít svoje jméno (jméno nemusí být unikátní), může mít tag a být zařazen v některé vrstvě.

Objekty mohou mít své vlastní pod-objekty, tyto objekty jsou ve vztahu rodič – dítě. Některé objekty ani nemusí mít žádné komponenty (kromě komponenty zajišťující pozici objektu) a mohou sloužit pouze jako složky pro ostatní objekty. Každá scéna musí mít minimálně 1 objekt, tento objekt musí obsahovat komponenty zajišťující funkce hlavní kamery. Další objekty ve scéně jsou již na vývojáři.

Každý objekt může být aktivní/neaktivní a tento stav může být nastaven jak při startu aplikace, tak za běhu aplikace pomocí skriptu. Tento stav se může za běhu aplikace změnit nesčetněkrát. Když se deaktivuje objekt, deaktivují se také jeho veškeré komponenty a pod-objekty.

3.2.4 Assety

Assety představují veškeré předměty, které mohou být použity ve hře. Assety mohou být vytvořeny mimo Unity (3D modely, zvukové soubory) i přímo v Unity (Animator Controller, Prefab).

Veškeré assety vytvořené mimo Unity musí být uložené ve složce Assets projektového adresáře (například C:*název projektu*\Assets). Celá tato složka je zobrazena v okně Project, více informací v kapitole Okno Project.

V průzkumníku souborů mohou být viděné soubory s koncovkou .meta, které nejsou viděny v okně Project, tyto soubory jsou vytvořeny pro každý asset a složku v projektu, obsahují informace o tom, jak je asset použitý v projektu.

Z assetů lze vytvořit takzvané Asset Packages, což jsou balíčky assetů, které obsahují různé soubory a data z projektu, které jsou kompresovány a uloženy v jednom souboru. Tyto balíčky mohou být importovány do různých projektů.

Základní typy assetů:

- Obrázkové soubory – jsou podporovány všechny nejběžnější formáty pro obrázky (.bmp, .tif, .tga, .jpg, pgn, psd atd.).
- 3D modely – jsou podporovány všechny soubory z nejběžnějších softwarů pro tvorbu 3D modelů (.max, .blend, .mb, .ma atd.).
- Audio soubory – jsou podporovány běžné zvukové formáty (.mp3, .ogg, wav atd.).
- Animace – ty mohou být vytvořeny přímo v Unity, případně v externích programech jako jsou Maya, Cinema 4D, 3D Studio Max atd.
- Další assety – Unity podporuje i spoustu dalších typů assetů, jako jsou skripty, písma atd.

Ve všech případech Unity nijak neupravuje původní zdrojový soubor.

Většina assetů vytvořených externím programem (obrázky, zvukové soubory) musí před použitím v projektu projít importujícím nastavením, v tomto nastavení se nejčastěji určuje typ assetu, komprimace a další různá nastavení v závislosti na typu assetu.

Standart Assets jsou balíčky assetů, které lze stáhnout do projektu přímo z aplikace, Standart Assets obsahují spoustu základních již vytvořených assetů, jako jsou různé 3D modely (postavy, vozidla, prostředí), kamery, 2D assety, efekty atd.

Další oficiální možností stažení již hotových assetů je použití obchodu jménem Asset Store, zde je možné si do svých projektů stáhnout assety placené i ty, které jsou zdarma. Tyto assety jsou vytvářené nejen společností Unity Technologies, ale i od členů komunity, kteří si mohou tímto způsobem svou tvorbou vydělat.

3.2.5 Prefaby

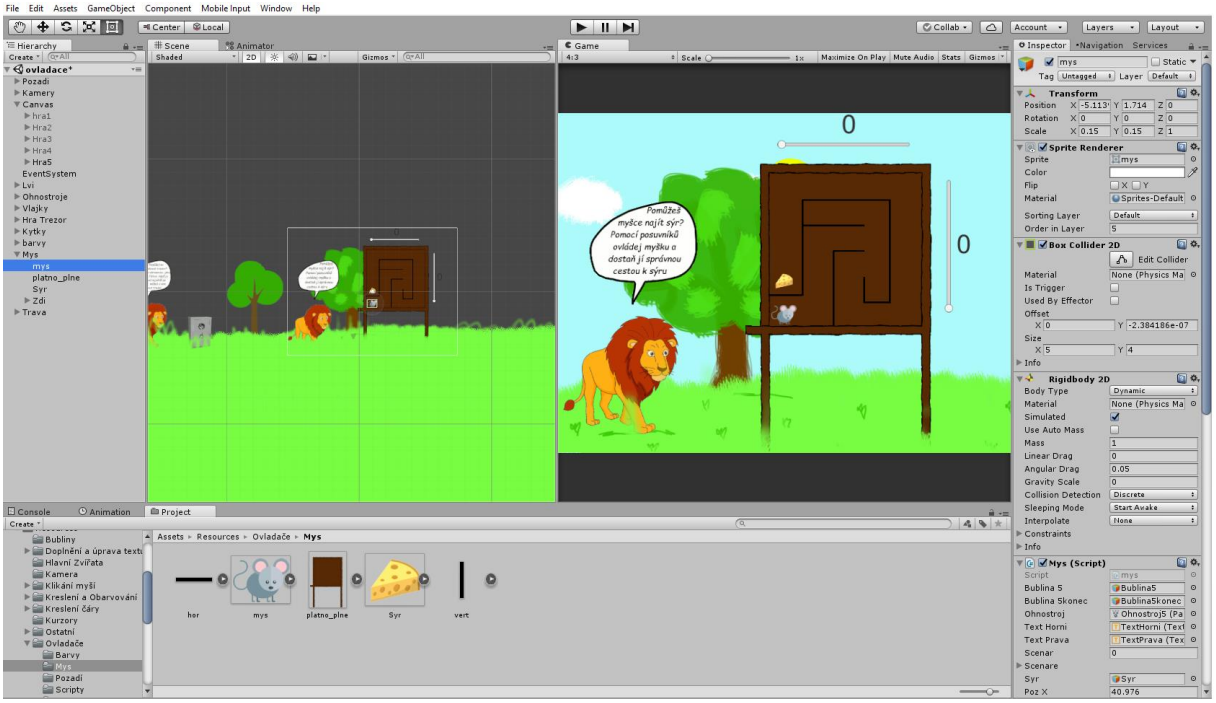
Prefab je asset, který umožňuje uložit objekt se všemi jeho komponenty a nastaveními, včetně všech jeho pod-objektů. Instance těchto prefabů je následně možné používat jako objekty ve scéně v libovolném počtu. Změna, která se udělá na prefabu, se automaticky projeví ve všech jeho instancích ve scéně. Každou instanci je možné následně upravovat.

Prefab lze vytvořit vybráním **Assets > Create > Prefab**. v místě, kde je otevřeno okno Project, se vytvoří prázdný prefab. K dokončení tvorby prefabu je nutné přetáhnout z okna Hierarchy některý objekt, který poslouží jako šablona prefabu (tento objekt se automaticky stane instancí prefabu). Tím je dokončena tvorba prefabu a následně se může prefab upravovat a tvořit jeho instance.

Prefaby se používají převážně v případě potřeby využít ve scéně větší množství objektů, které mají mít stejné nebo podobné parametry. Tato možnost se může použít například pro objekty NPC (nehráčská postava). Případně pro opakující se prostředí (stromy, tráva atd.). Prefaby také slouží jako šablona pro nové objekty, které se vytvoří v průběhu běžící hry.

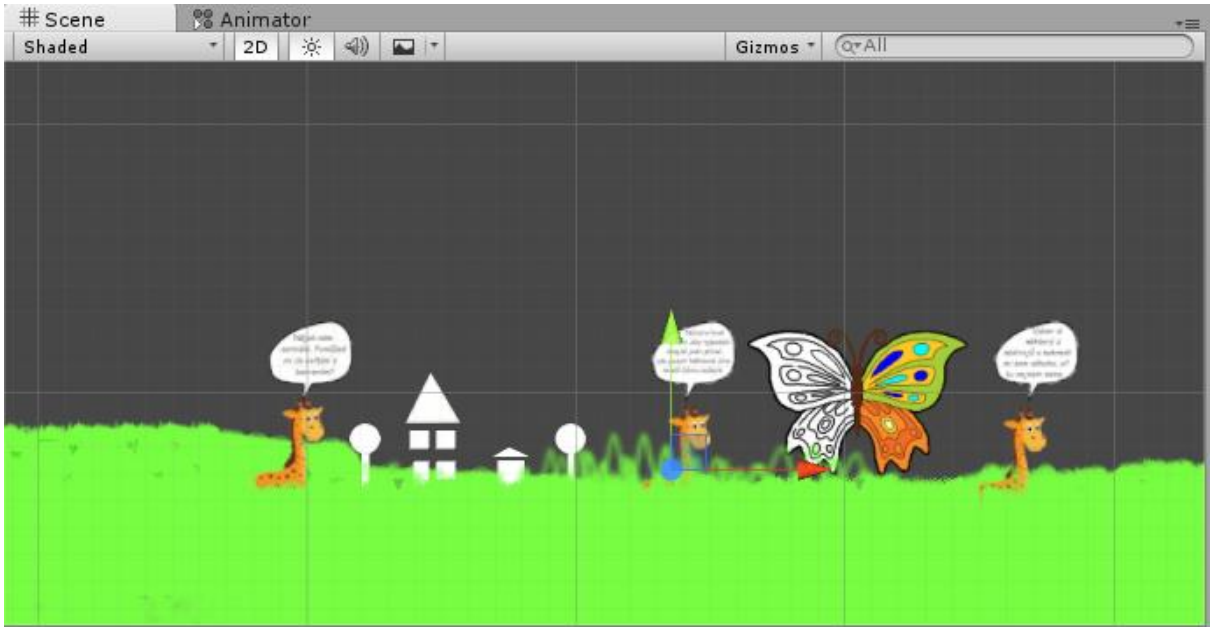
3.2.6 Rozhraní Unity

V Unity Editoru stráví vývojář během vývoje hry nejvíce času, proto je důležité, aby byl editor pro vývojáře přehledný. Samotný editor je rozdělen do několika oken, tato okna si může vývojář sám rozvrhnout buď pomocí specifického Layoutu, nebo nastavením vlastního rozvržení, ve kterém je možné využít více monitorů. Jednotlivá okna lze maximalizovat.



Obrázek 2 – Náhled na Unity Editor

Okno Scene



Obrázek 3 – Okno Scene

V okně Scene se pracuje s polohou všech objektů, které jsou ve scéně. v okně je možno zapnout 2D náhled, který usnadňuje práci s 2D objekty. Přesouvat se po scéně je možné buď pomocí panelu nástrojů tlačítka view, viz Obrázek 4 – Panel nástrojů(číslo 1) (defaultní klávesová zkratka Q), nebo pomocí stisknutí prostředního tlačítka myši a pohybu myší. Pokud je scéna ve 3D (2D náhled není aktivní), kamera se otáčí pomocí pravého tlačítka myši a pohybu myší.

Obrázek 4 – Panel nástrojů

- Pomocí panelu nástrojů je možné s objekty manipulovat. Nejdříve je nutné vybrat objekt, ten lze vybrat v okně hierarchie, viz kapitola

Okno Hierarchy nebo kliknutím na objekt (pokud je vybraný některý z manipulačních nástrojů).
Manipulační nástroje jsou:

- Move – nastavuje pozici objektu Obrázek 4 – Panel nástrojů (číslo 2).
- Rotate – nastavuje rotaci objektu Obrázek 4 – Panel nástrojů (číslo 3).
- Scale – nastavuje velikost objektu Obrázek 4 – Panel nástrojů (číslo 4).
- Rect handles – nastavuje pozici i velikost objektu Obrázek 4 – Panel nástrojů (číslo 5).

Dalšími nástroji jsou:

- „*Pivot / Center*“ – při volbě „Pivot“ se osy protínají v původní nulové pozici objektu, volba „Center“ znamená, že se osy protínají ve středu objektu. Obrázek 4 – Panel nástrojů (číslo 6).
- „*Local / Global*“ – při volbě „Local“ budou osy natočeny podle rotace objektu. Při volbě „Global“ budou osy natočené vždy stejně Obrázek 4 – Panel nástrojů (číslo 7).

Okno Hierarchy

V okně hierarchy je možné vidět všechny objekty ve scéně, včetně jejich hierarchické spojitosti s jinými objekty. Pokud se nějaký objekt přidá, nebo vymaže ve scéně, tato změna projeví i v okně hierarchy. V tomto okně je možné objekty kliknutím vybírat, přejmenovávat, mazat, duplikovat, kopírovat, vyhledávat, případně vytvářet. Pomocí běžného Drag and drop je možné měnit pořadí, hierarchii objektů.

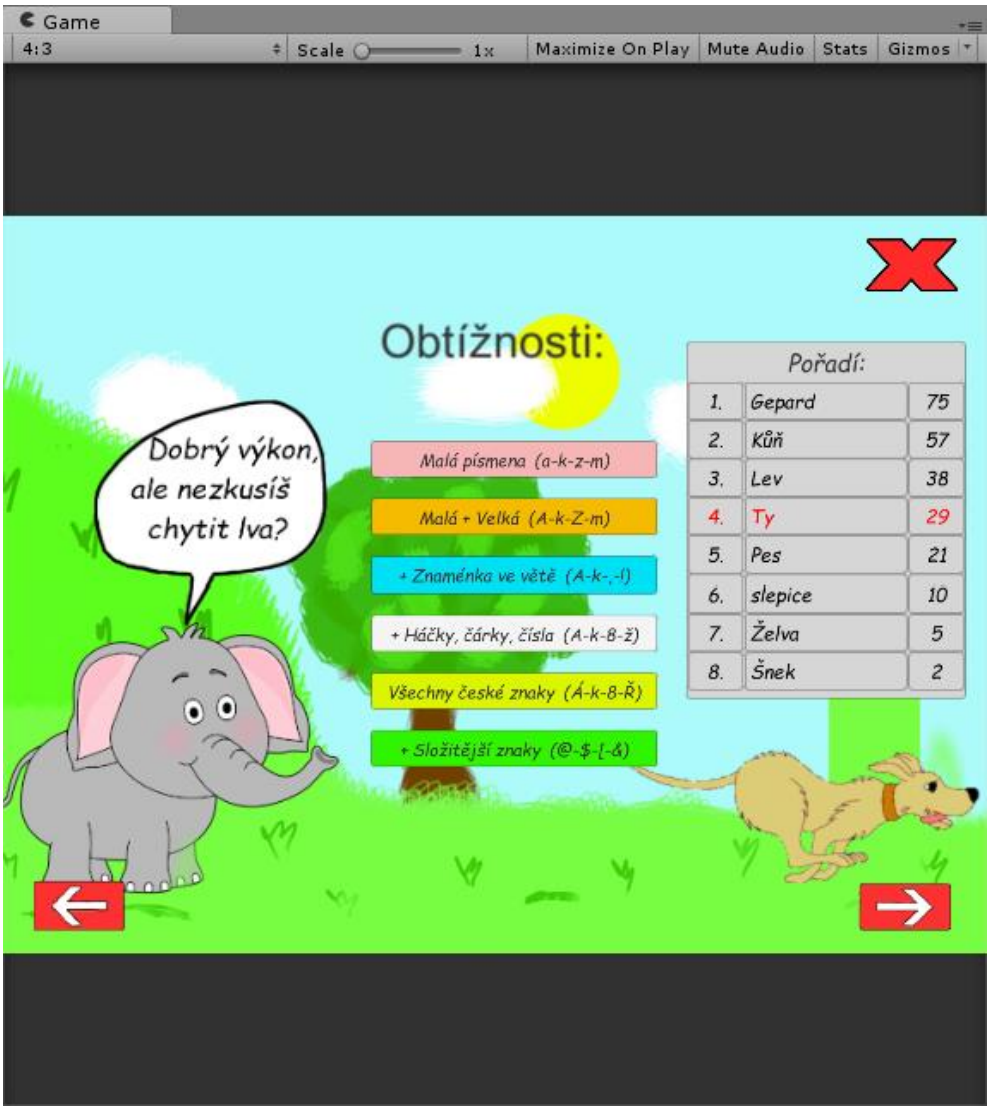
Převážně v tomto okně se nastavuje veškeré „rodičovství“ objektů, jednoduchým přesunutím prvního objektu do druhého objektu se z prvního objektu objektu stane child objekt druhého objektu.

Obrázek 5 - Okno Hierarchy

Okno Game

V okně Game je možné sledovat hru z pohledu hráče, v tomto okně se nijak nezasahuje do vývoje hry, ale pouze se zde hra „hraje“. Hra se zde spouští převážně pro testovací účely.

Nabídky okna Game jsou: vybrat rozlišení, případně poměr stran, ve kterém hra poběží. Případně přiblížit si náhled na hru nebo maximalizovat okno při spuštění hry.

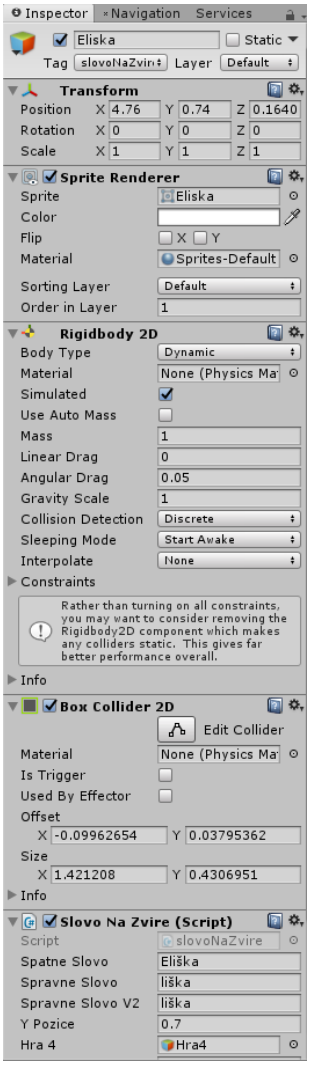


Obrázek 6 – Okno Game

Hra se spustí pomocí tlačítka „play“ z panelu nástrojů, viz Obrázek 7 – Spuštění/pozastavení hry.

Obrázek 7 – Spuštění/pozastavení hry

Okno Inspector



Obrázek 8 - Okno Inspector

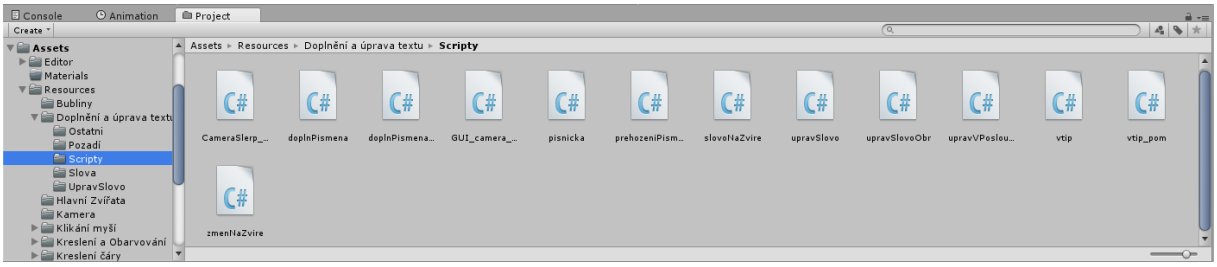
V okně Inspector se nastavují všechny vlastnosti právě vybraného objektu. Každý objekt má několik základních vlastností:

- Zapnutí – je objekt zapnutý „Ano/Ne“.
- Název objektu – slouží k identifikaci mezi ostatními objekty.
- Statický – je objekt statický „Ano/Ne“ – zaškrtává se u nepohybujících se objektů (například zeď), vylepšuje optimalizaci aplikace.
- Tag – používá se pro označení skupiny objektů, například „zbraně“, „nepřátelé“, „kytky“ atd. K tagům lze přistupovat ze skriptů (např. „najdi objekty s tagem „zbraň“ a zařaď objekty do kolekce“).
- Vrstvy – další možnost rozřazení objektů.

V okně Inspector jsou veškeré informace o komponentech objektu, více informací v kapitole Komponenty objektu.

Okno Project

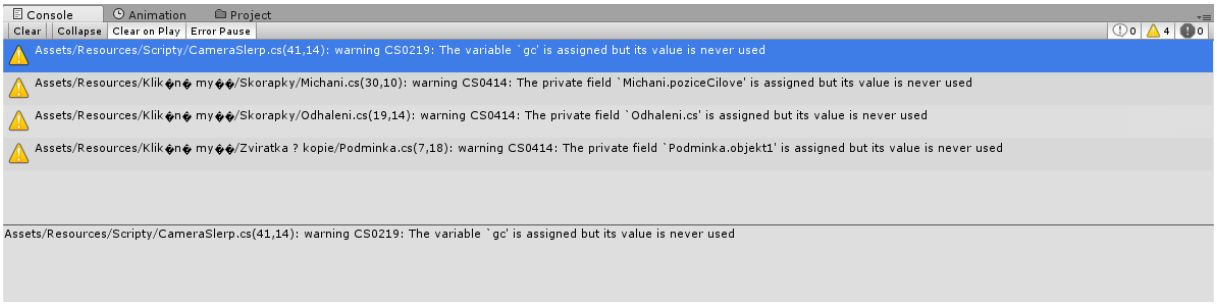
V okně Project můžeme vidět Unity průzkumníka souborů, kde máme rozřazené veškeré soubory k projektu ve složkách. v tomto okně můžeme do projektu importovat/vkládat různé soubory, které chceme v projektu využít. Zároveň zde můžeme vyhledávat, kopírovat, mazat či vytvářet soubory. Veškeré předměty, které jsou viditelné v okně Project, představují soubory v počítači, smazáním těchto souborů v Unity je smažeme i v počítači.



Obrázek 9 – Okno Project

Okno Console

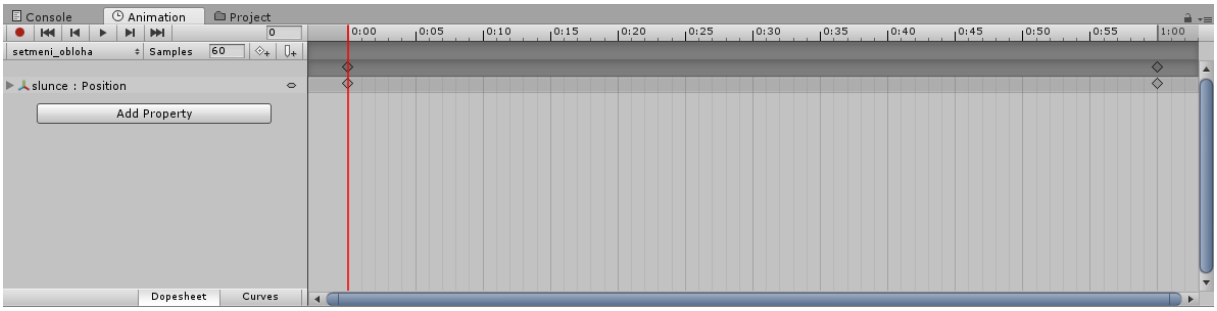
V konzolovém okně se ukazují převážně různé chyby/upozornění v projektu, většinou se jedná o chyby v kódu, ale může se jednat i o špatné nastavení komponent na některém objektu, případně problém s některou částí projektu. Kromě chyb a upozornění se zde vypisují i výstupy z aplikace, které například pomáhají vývojáři s napsáním kódu.



Obrázek 10 – Okno Console

Okno Animation

V okně Animation se vytváří animace objektů. Nejdříve je nutné vybrat objekt, pro který se má vytvářet animace. Poté vytvořit soubor. Pomocí tlačítka Record je možné začít animaci tvořit.



Obrázek 11 – Okno Animation

Okno Animator

V okně Animator se nastavují objekty, které mají více druhů animací, typickým příkladem takového objektu může být postava, která chodí/běhá, skáče, skrčuje se atd.

3.3 Komponenty objektu

Komponenty jsou „šrouby a matice“ objektů, které určují chování hry, jsou to funkcionální části každého objektu.[5]

Každý objekt obsahuje minimálně 1 komponentu (Transform nebo Rect Transform). Ovšem většina objektů obsahují více komponent. Komponentů je v Unity celá řada, můžou to být komponenty pro vykreslování, uživatelské rozhraní, efekty, fyziku, animace, navigaci, kameru, vlastní skripty. Každá komponenta slouží zároveň jako třída objektu a ke komponentám lze přistupovat pomocí skriptu. v několika následujících kapitolách je výčet některých komponent.

Transform

Komponenta Transform, zajišťuje umístění objektů, které je určeno pomocí 3 os, osy X, osy Y a osy Z, pokud vývojář vytváří 2D hru, osa Z má u většiny objektů hodnotu 0.

Komponenta Transform patří mezi nejzákladnější komponenty, tato komponenta je součástí všech objektů (kromě objektů sloužících pro komponenty uživatelského rozhraní). v této komponentně se nastavují lokální pozice, rotace a velikost[2] objektu na všech 3 osách.

3.3.2 Komponenty vykreslující objekt

Každý objekt může mít maximálně 1 komponentu, která zajišťuje vykreslování objektu.

[2] Lokální pozice, rotace a velikost je brána vůči rodičovskému objektu. Pokud má rodičovský objekt dvojnásobnou velikost, jeho pod-objekty budou mít také dvojnásobnou velikost, i když budou mít hodnotu velikosti nastavenou na 1. Pokud Objekt nemá rodiče, lokální hodnoty jsou stejné jako globální.

Mesh Filter

Komponenta zajišťuje nastavení tvaru objektu, k této komponentě se automaticky vytvoří komponenta Mesh Renderer, která zajišťuje vykreslení a stínování objektu na obrazovce, pokrytí objektu materiálem.

Mesh Text

Komponenta vkládající do scény text. Možnosti komponenty jsou nastavení obsahu textu, velikosti písma, nastavení fontu, zarovnání atd., stejně jako u Mesh Filter. Vykreslení objektu zajišťuje komponenta Mesh Renderer.

Sprite Renderer

Komponenta zajišťující vykreslení obrázků na obrazovce. Těmto obrázkům lze nastavit různou barvu i pořadí vykreslení.

Pokud se má na některém místě vykreslit více objektů, pořadí vykreslování určují „Sorting Layers“. Pokud jsou objekty ve stejné Sorting Layer, rozhoduje pořadí ve vrstvě.

Trail Renderer

Komponenta vytvářející stopu za objektem, který se pohybuje po scéně. v komponentě lze nastavit tloušťku stopy, barvu stopy, vytváření stínů od světla, uběhnutou dobu, než stopa začne mizet, případně materiál stopy.

Line Renderer

Komponenta vytvářející čáru. Délka a pozice čáry se nastavuje v nastavení Positions, kde se nastavuje počet bodů a pozice jednotlivých bodů (pozice na X, Y, Z osách), které lajna propojuje. Lajna může být tvořena minimálně ze 2 bodů (úsečka), ale může být vytvořena i z libovolného počtu bodů. Dalšími možnostmi nastavení komponenty jsou: materiál čáry, tloušťka čáry, barva čáry atd.

Particle System

Komponenta Particle System napodobuje entity fyzických jevů jako jsou kapaliny, mraky nebo plameny pomocí generování velkého množství obrázků ve scéně. Komponenta má velké množství různých nastavení. v této komponentě lze tvořit velké množství efektů.

3.3.3 Fyzikální Komponenty

Collider Komponenty

Collider komponenty definují tvar objektu pro fyzické účely. Každý objekt, který je určený pro fyzické akce, by měl mít některý z collider komponentů. Tyto komponenty se rozdělují na 2D a 3D. Collider komponenty jsou:

- Box Collider (2D) – tvar kostky
- Sphere (Circle) Collider – tvar koule
- Capsule Collider (2D) – tvar kapsle
- Edge Collider 2D – umožňuje nastavit vlastní tvar
- Mesh Collider – kopíruje tvar Mesh modelu
- Polygon Collider – kopíruje tvar Sprite Rendereru (obrázku)
- Terrain Collider – kopíruje tvar terénu

Rigidbody

Objekty s komponentou Rigidbody začnou podléhat fyzice Unity Enginu. Na objekt působí veškeré fyzické síly, včetně gravitační. 2D objekty musí mít komponentu Rigibody 2D.

3.3.4 Komponenty uživatelského rozhraní

Objekty s komponenty uživatelského rozhraní (dále jen UI) neobsahují komponentu Transform, ale komponentu Rect Transform.

Některé UI komponenty mají v nastavení možnost „Transition“, která zajišťuje grafické přechody komponenty při různých interakcí myší (najetí na tlačítko, stisknutí tlačítka, opuštění tlačítka). Tyto přechody můžou být vytvořeny buď změnou barvy obrázku nebo změnou obrázku.

Většina „aktivních“ UI komponent obsahuje funkci, která přistupuje k jinému objektu (nejčastěji objekty se skriptem) a vykonává nějakou činnost. Například některé komponenty obsahují funkci onValueChanged(), které dynamicky mění hodnotu některé proměnné (textové pole, posuvník, přepínač apod.). Další funkce, kterou obsahují komponenty Button (tlačítka), se nazývá OnClick(), tato funkce může fungovat tak, že při kliknutí na komponentu se vykoná nějaký příkaz (například se spustí některá funkce v skriptu). Tyto funkce není nutné využívat, ke komponentám lze přistupovat také přímo ze skriptu, více info v kapitole Skriptování v Unity.

Pokud je více objektů s UI komponentou na tom samém místě, tak se blíž ke kameře vykresluje objekt, který je v okně hierarchy níže a pouze ten objekt na daném místě reaguje na veškeré uživatelovy vstupy myší. Například pokud je tlačítko ukryto za obrázkem, na tlačítko není možné kliknout.

Canvas

Canvas je prostor pro komponenty s UI. Veškeré objekty s komponenty s UI musí být pod-objekty objektu s Canvas komponentou.

Komponenta Canvas má 3 základní módy:

- Screen Space – Overlay – vykresluje uživatelské rozhraní na vrchu scény.
- Screen Space – Camera – vykresluje uživatelské rozhraní v určené vzdálenosti od kamery.
- World Space – Canvas – může být umístěn kdekoliv ve scéně a uživatelské rozhraní je vykresleno podle umístění objektů.

Pokud je Screen Space – Overlay nebo Screen Space – Camera, otevře se nastavení Canvas Scaler, kde se nastavuje umístění objektů vzhledem k různým rozlišením hry:

- Constant Pixel Size – pozice a velikost uživatelského rozhraní je konstantní, nastavuje se podle pixelů.
- Scale With Screen Size – pozice a velikost uživatelského rozhraní se mění společně se změnou rozlišení.
- Constant Physical Size – pozice a velikost uživatelského rozhraní je konstantní, nastavuje se podle fyzických jednotek (centimetry, milimetry).

Obrázek 12 – Constant Pixel Size 640x480

Obrázek 13 – Constant Pixel Size 1024x768

Obrázek 14 – Scale With Screen Size 640x480

Obrázek 15 – Scale With Screen Size 1024x768

Obrázek 16 – Constant Physical Size 640x480

Obrázek 17 – Constant Physical Size 1024x768

Event System

Objekt s komponentou Event System se vytvoří automaticky v momentě, kdy se vytvoří první UI komponenta ve scéně. Tato komponenta funguje pasivně. Ve scéně by neměl být více než 1 objekt s touto komponentou. Komponenta pasivně zpracovává veškerou manipulaci s událostmi ve scéně. Komponenta zajišťuje, který objekt je zrovna uživatelem vybrán, který vstupní modul je aktuálně používán atd.

Event Trigger

Tato komponenta slouží ostatním UI komponentám. Objektu se může přidat jedna nebo více specifických funkcí. Jsou to funkce, které při určité činnosti s UI komponentou vykonají nějakou činnost, touto činností může být například PointerEnter/PointerExit (při najetí myší nad objekt/při opuštění objektu), select/deselect (při vybrání/odvybrání objektu) atd.

Rect Transform

Komponenta Rect Transform je protějškem komponenty Transform. Rozdíl mezi těmito komponenty je, že Transform reprezentuje jediný bod, ale Rect Transform vyznačuje obdélník, ve kterém může být UI umístěno.

Text

Komponenta zajišťující zobrazení textu. Funguje jak samostatně, tak je součástí spousty jiných komponent, nastavení komponenty jsou: obsah textu, nastavení fontu, velikost písma, zarovnání, barva písma atd.

Image

Zajišťuje funkci zobrazení obrázků v UI. Podobné jako Sprite Renderer.

Button

Zajišťuje funkci tlačítek. Využívá komponentu Image a komponentu Text. Samotná komponenta Button obsahuje funkci On Click(), která zajišťuje vykonání akce při kliknutí na tlačítko. Dále komponenta Button obsahuje možnosti Transition.

Toggle

Komponenta zajišťující funkci přepínače. Komponenta využívá dvakrát komponentu Image, jednou pro obrázek pozadí přepínače a jednou pro obrázek zaškrtnutí. Komponenta využívá ještě komponentu Text. Tato komponenta má 2 stavy, zapnuto/vypnuto. Obsahuje funkci OnValueChange(), která zajišťuje vykonání akce při kliknutí na přepínač. Další možnosti komponenty jsou Transition. Případně můžou být zařazeny v některé Togggle Group.

Togggle Group

Pasivní komponenta sloužící komponentě Toggle. Komponenta zajišťuje, že pouze jeden přepínač ve skupině může být zapnutý. Jediným atributem této komponenty je umožnění, že můžou být všechny přepínače ve skupině vypnuté.

Slider

Komponenta sloužící jako posuvník. Komponenta využívá třikrát komponentu Image. Jednou pro rukojeť, jednou pro „vyplněnou“ část a jednou pro „nevyplněnou“ část. Samotná komponenta Slider obsahuje možnosti Transition, směr posuvníku, minimální a maximální hodnotu posuvníku, možnost pouze celých čísel a defaultní hodnotu posuvníku. Komponenta ještě obsahuje funkci OnValueChanged(), která umožňuje dynamicky měnit některou hodnotu při použití posuvníku.

InputField

Komponenta sloužící jako textové pole. Využívá komponentu Image jako pozadí textového pole a svakrát komponentu Text, jednou pro text samotný napsaný hráčem a jednou pro placeholder (text, který se může ukazovat, pokud uživatel ještě nezadal žádný text, např. „Zde začněte psát“). Samotná komponenta obsahuje funkce OnValueChanged(), která zajišťuje vykonání akce při jakékoliv změně a funkci OnEndEdit(), která vykoná akci, když uživatel ukončí psaní v textovém poli (textové pole se odznačí). Dále komponenta obsahuje možnosti Transtition, maximální počet znaků, typ obsahu (standardní, jenom celá čísla, jenom čísla, jenom písmena, e-mailová adresa, heslo, pin atd.), tloušťka caretu (kurzoru), rychlost blikání caretu.

Dropdown

Komponenta Dropdown je další z „menu“ možností. Komponenta se skládá ze spousty jiných UI komponent. Samotná komponenta obsahuje možnosti a funkci OnValueChnaged(), která může měnit int hodnotu některé proměnné. Další možnosti jsou Transition a nastavení původní hodnoty.

Scrollbar a Scroll Rect

Tyto 2 komponenty zajišťují scrollování velkého textu. Komponenta Scroll Rect zajišťuje prostor, který je třeba scrollovat. Možnosti komponenty jsou typ pohybu, sensitivita scrollování a přidání (horizontální, vertikální) komponenty Scrollbar. Dále je potřeba přidat komponentu Mask, která zajišťuje, že text mimo Scroll Rect bude skrytý.

Komponenta Scrollbar se nastavuje automaticky z komponenty Scroll Rect. Pokud se Scrollbar používá samostatně, tak se nastavuje směr Scrollbaru, hodnota Scrollbaru, velikost, počet kroků. Komponenta obsahuje funkci OnValueChanged(), která umožnuje dynamicky měnit hodnotu při použití Scrollbaru.

3.3.5 Další komponenty

Camera

Každá scéna musí mít alespoň jeden objekt s komponentou Camera, tato komponenta umožnuje hráčům vidět hru. Objektů s komponentou Camera může být ve scéně víc. To umožňuje přepínat mezi kamerami, případně split screen.

Tato komponenta má 2 základní módy:

Ortografický mód – častěji využívané pro 2D hry, velikost objektu na kameře je stejná bez ohledu na vzdálenost od objektu, to znamená, že pokud je objekt velký na obrazovce 100 pixelů ve vzdálenosti 10 bodů od kamery, zabere 100 pixelů, i když bude vzdálený od kamery 1000 bodů.

Perspektivní mód – kamera vnímá objekty podobně jako lidské oko a velikost objektů na obrazovce se mění společně se vzdáleností od kamery, i když je skutečná velikost objektu stále stejná.

Animation

Umožňuje objektu provádět animaci, jako hlavní parametr je soubor s animací, další možnosti komponenty jsou: počet animací, spustit animaci automaticky nebo animovat fyziku.

Animator

Přidá objektu Animator, hlavním parametrem je soubor s Controllerem.

Light

Unity nabízí práci se světly a stíny a Komponenta Light nabízí zdroj světla.

Tipy světla:

- Bodová světla – zdroj je v určitém bodě a vrhá světlo do všech směrů od bodu, čím větší vzdálenost, tím menší intenzita světla (funguje jako lampa).
- Reflektory – zdroj je v určitém bodě, ale vysílá světlo pouze do určitého úhlu (funguje jako svítilna).
- Směrová světla – funguje jako efekt slunce ve scéně.
- Prostorová světla – jsou definována jako obdélníky v prostoru, světlo je stejné ve všech bodech v obdélníku, ale pouze z jedné strany.

Audio Source

Zajišťuje přehrávání zvukového souboru. Hlavním nastavením je zvukový soubor. Další nastavení komponenty jsou: opakované přehrávání, přehrát při startu, ztlumení, hlasitost atd.

Terrain

Pokud scéna obsahuje terén, tato komponenta se vytvoří automaticky. Komponenta obsahuje různé nástroje pro úpravu terénu.

Script

Jedna z nejdůležitějších komponent pro tvorbu celé hry. Všechny skripty, které ve hře jsou, jsou použity jako komponenty objektů. Nastavení těchto komponent spočívá ve změně veřejných proměnných ve skriptu.

3.4 Skriptování v Unity

Skriptování je podstatnou součástí ve všech hrách, i ta nejjednodušší hra potřebuje skripty. Například na detekování hráčových akcí, zajištění událostí atd. Skripty mohou být použity na vytvoření grafického efektu, kontrolu fyziky, implementace umělé inteligence a mnoho dalšího.

V Unity má vývojář možnost programovat hru buď v jazyku **C# (C Sharp)** nebo v jazyku UnityScript, tento jazyk je vytvořen přímo pro Unity a je modelován podle jazyku **Javascript**. Přímo v programu Unity je jazyk UnityScript nazýván rovnou jako Javascript.

Unity podporuje ladění za běhu programu. Tím pádem je možné editovat skript za běhu aplikace a změny ve skriptu se projeví ihned v běžící hře.

Aby byl skript ve scéně aktivní, musí být skript použitý jako komponenta některého objektu ve scéně, skript může být na více objektech současně a zároveň každý skript může obsahovat více objektů.

3.4.1 Vytváření Skriptu

Skripty v Unity lze přidat 2 základními způsoby:

- Přidat v okně **Inspector** objektu nový komponent (**add component**), zvolit **New Script**, vybrat jazyk skriptu a název. Skript se vytvoří v základní složce Assets.
- V okně **Project**, kliknout pravým tlačítkem myši a zvolit **Create,** C# Script (Javascript). Skript se vytvoří v aktuálně otevřeném adresáři.

Po vytvoření skriptu se spustí nástroj pro editaci skriptů, defaultní nástroj pro editaci je program MonoDevelop, který se instaluje společně s Unity. v nastavení Unity je možné tento nástroj změnit například na Visual Studio, které obsahuje sofistikovanější vývojové C# prostředí.

Rozdíl mezi C# a JS (Javascript) je mimo jiné v tom, že zatímco veškeré proměnné, funkce atd. jsou v JS veřejné (pokud není uvedeno jinak), tak v C# jsou privátní. To znamená, že pokud je nutné k těmto proměnným, funkcím atd. přistupovat i mimo skript, je nutné před to v C# uvést modifikátor přístupu *public*.

Počáteční obsah každého skriptu při vytvoření vypadá přibližně takto:

C#

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class nazevScriptu : MonoBehaviour
{

	// Use this for initialization
	void Start () {

	}

	// Update is called once per frame
	void Update () {

	}
}
```

Javascript

```
#pragma strict

function Start () {

}

function Update () {

}
```

3.4.2 Základní funkce skriptů

Programování v Unity je jako u každého vývojového programu trošku specifické. v Unity jsou specifické například právě funkce, které se vykonají automaticky při splnění některých podmínek. Tyto funkce musí být nadefinovány podle syntaxe daného jazyka.

V C# je to:

```
void nazevFunkce()
{
	//seznam příkazů, které se provedou, když se splní podmínky funkce.
}
```

V JS je to:

```
function nazevFunkce()
{
	//seznam příkazů, které se provedou, když se splní podmínky funkce.
}
```

Zde je výčet některých základních funkcí:

Start()

Patří mezi nejzákladnější funkce. Vykoná příkazy uvnitř funkce v momentě prvního snímku, kdy je skript aktivní (konkrétně když je aktivní objekt, který má skript jako komponentu). Tato funkce se používá například na přidělení objektu proměnné.

Update()

Stejně jako funkce Start() je toto jedna z nejzákladnějších funkcí. Tato funkce zpracuje veškeré příkazy uvnitř funkce v každém snímku hry. Tato funkce má spoustu využití, například detekování určitých hráčových vstupů (je/není stisknuté tlačítko), odečet času, neustálá kontrola splnění požadavku atd.

Awake()

Funkce podobná funkci Start(), ovšem vykoná se na všech objektech dříve, než se začne vykonávat funkce Start().

OnGUI()

Funkce která vykresluje GUI elementy a zajišťuje GUI události, funkce OnGUI() je zavolána každý snímek, stejně jako funkce Update(). Tato funkce už se dnes často nepoužívá, jelikož byla nahrazena UI komponenty.

OnMouseDown()

Funkce, která se vykoná jednou v momentě, kdy hráč klikne myší na objekt s colliderem nebo na GUI Element. Funkce se používá například na zničení objektu kliknutím na něj, na stisknutí GUI tlačítka atd.

OnMouseEnter()

Funkce, která se vykoná jednou v momentě, kdy hráč najede myší na objekt s colliderem nebo na GUI Element. Využívá se například na zvýraznění objektu/GUI tlačítka při najetí myší.

OnMouseExit()

Opak funkce OnMouseEnter(), tato funkce se zavolá v momentě, když myš opustí objekt s colliderem nebo GUI Element. Využívá se například na zrušení zvýraznění objektu/GUI tlačítka při opuštění myši objektu/GUI Elementu.

OnMouseOver()

Funkce, která se vyvolá každý snímek, když je myš nad objektem s colliderem nebo GUI Elementem.

OnCollisionEnter(2D)(Collision collision)

Funkce, která se jednou vyvolá, pokud se jeden objekt dostane do kolize s jiným objektem, oba tyto předměty musí mít komponentu collider, viz kapitola Collider Komponenty, a minimálně jeden z nich musí mít komponentu Rigibody, viz kapitola Rigidbody. Prvek collision je objekt, s kterým došlo ke kolizi. Pokud je tato funkce požadována v 2D, je nutné, aby tyto fyzikální prvky byly ve 2D.

Funkce se používá například na zničení jiného objektu při doteku. Dalším příkladem může být, pokud je tato funkce na hráčově postavě a prvek collision je „země“, že funkce neudělá nic, ale pokud je prvek collision například „láva“, charakter umře.

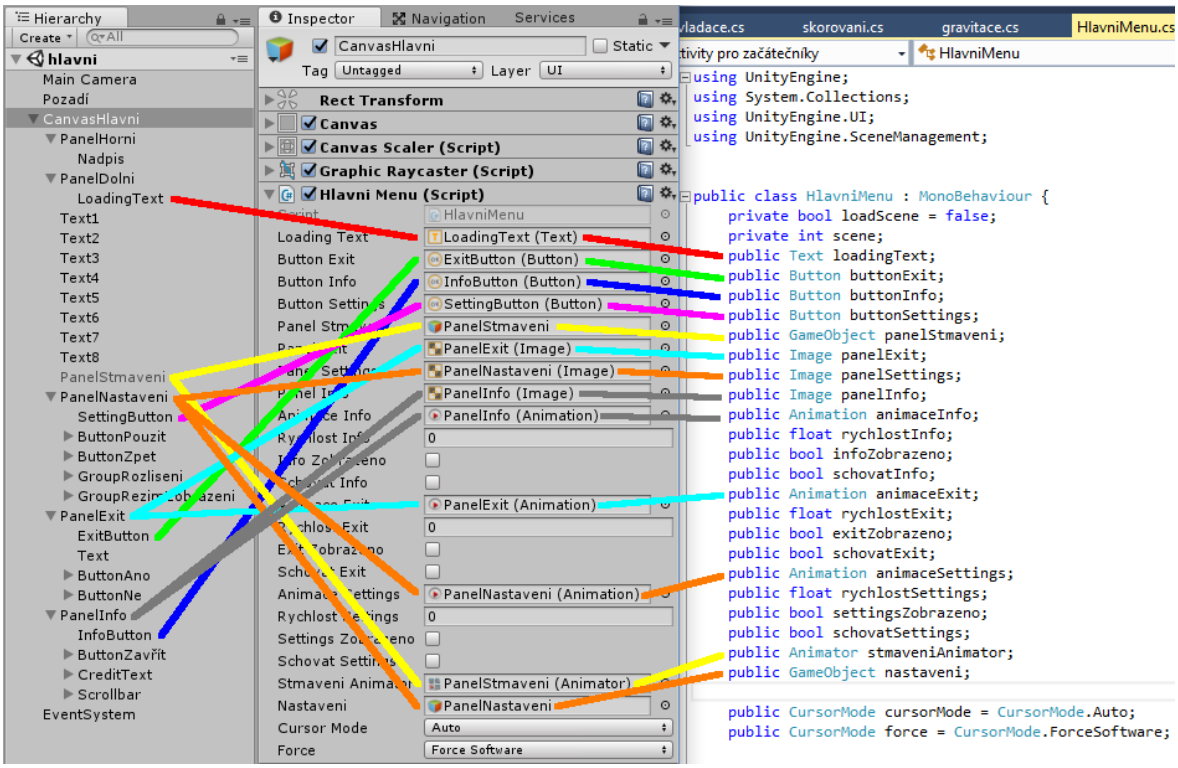
3.4.3 Proměnné a jejich přiřazení

V Unity jsou ve skriptech často používány běžné proměnné, jako jsou Int, Float, String, Bool/Boolean, listy, pole atd. Ale zároveň jsou v Unity často využívané proměnné, které vyžadují přístup k jinému objektu, ke komponentě, případně ke komponentně na jiném objektu. Těchto proměnných mohou být na skriptu desítky i stovky a je nutné všem těmto proměnným přiřadit hodnoty.

Přiřazení těchto hodnot může probíhat 2 základními způsoby:

Manuální přiřazení v Unity

V momentě, kdy je skript připojen jako komponenta k objektu, se v okně **Inspector** ukážou všechny veřejné proměnné ve skriptu. Pokud je proměnná jiný objekt, případně komponenta jiného objektu, je možné přiřadit hodnotu přetažením objektu z okna **Hierarchy**, viz Obrázek 18 – Ukázka propojení proměnných.



Obrázek 18 – Ukázka propojení proměnných

Tato metoda přiřazení proměnných je poměrně jednoduchá, ovšem není praktická, pokud je daný skript s mnoha proměnnými na více objektech, případně když je ve skriptu větší množství proměnných.

V okně **Inspector** se zároveň nastavují všechny veřejné proměnné objektu. To umožnuje nastavit každému objektu unikátní hodnoty proměnných.

Přiřazení pomocí skriptu

Přiřazení hodnot proměnných je možné udělat přímo ve skriptu. Proměnné, jako jsou Int, Float, String, Bool/Boolean atd., se nastavují podle běžné syntaxe programovacího jazyka.

Přiřazení pomocí skriptu je účinnější, například v momentě, kdy je skript použitý na více objektech a bylo by zdlouhavé hodnoty proměnných nastavit manuálně pomocí unity. Navíc je tato metoda nezbytná v tom případě, že se některý objekt se skriptem vytvoří za chodu aplikace.

Přiřazení proměnné GameObject.

V případě potřeby přiřadit hodnotu proměnné GameObject je nutné využít některý z „hledacích“ příkazů. Většinou se tyto příkazy využívají ve funkci Start().

Vyhledávání podle jména a tagu:

C#:

```
public GameObject ruka;
public GameObject nohaNepritele;
public GameObject hrac;


void Start(){
ruka = GameObject.Find("Ruka"); //hledá objekt s názvem Ruka
nohaNepritele = GameObject.Find("Nepritel/Noha");
//hledá objekt s názvem Noha, která je dítě objektu s názvem Nepritel
Hrac = GameObject.FindWithTag("Hrac");
//hledá objekt s tagem jménem Hrac
}
```

JS:

```
public ruka : GameObject;
public nohaNepritele : GameObject;
public hrac : GameObject;


function Start(){
ruka = GameObject.Find("Ruka"); //hledá objekt s názvem Ruka
nohaNepritele = GameObject.Find("Nepritel/Noha");
//hledá objekt s názvem Noha, která je dítě objektu s názvem Nepritel
Hrac = GameObject.FindWithTag("Hrac");
//hledá objekt s tagem jménem Hrac
}
```

Jak je výše uvedeno, příkaz pro hledání podle jména je:

```
GameObject.Find("název_objektu");
```

příkaz pro hledání podle tagu je:

```
GameObject.FindWithTag("název_tagu");
```

Vyhledávání podle konkrétního jména/tagu má nevýhodu v tom, že požadovaný objekt musí mít unikátní jméno/tag. Pokud skript nalezne objekt splňující podmínky, ukončí vyhledávání a jestli jsou ve scéně další objekty splňující podmínky, skript už je nevyhledává. Vyhledávaný objekt musí být v době vyhledávání aktivní, jinak objekt nebude nalezen a proměnná zůstane null.

Vyhledávání child objektu:

C#:

```
public GameObject nohaHrace;
public GameObject nepritel;
public GameObject nohaNepritele;


void Start(){
 nohaHrace = this.transform.Find(„noha“).gameObject;
 //Hledá objekt s názvem „noha“, který je child objektem objektu ke kterému
 //je přiřazený script
 nepritel = GameObject.Find(„nepritel1“);
 //Hledá objekt s názvem „nepritel1“
 nohaNepritele = nepritel.tranform.Find(„noha“).gameObject;
 //Vyhledá objekt jménem „noha“, který je child objektem objektu „nepritel“
}
```

JS:

```
public nohaHrace : GameObject;
public nepritel : GameObject;
public nohaNepritele : GameObject;

function Start(){
 nohaHrace = this.transform.Find(„noha“).gameObject;
 //Hledá objekt s názvem „noha“, který je child objektem objektu ke kterému
 //je přiřazený script
 nepritel = GameObject.Find(„nepritel1“);
 //Hledá objekt s názvem „nepritel1“
 nohaNepritele = nepritel.tranform.Find(„noha“).gameObject;
 //Vyhledá objekt jménem „noha“, který je child objektem objektu „nepritel“
}
```

Příkaz pro vyhledávání child objektu je v obou jazycích:

```
„Parent_objekt“.transform.Find(„název_child_objektu“).gameObject;
```

Pokud je výchozí objekt ten, ve kterém je přiřazený skript, jako rodičovský objekt se použije **this.** Vyhledávání pomocí child objektu je přesnější v tom, že neprohledává všechny objekty ve scéně, ale pouze child objekty určitého objektu.

Přiřazení proměnné komponent

K přiřazení proměnné komponenty je nutné mít nejdříve přiřazenou hodnotu GameObject, ze které je možné přistupovat k jednotlivým komponentám.

C#:

```
public Animator animatorHrace;
public GameObject nepritel;
public Animator animatorNepritele;


void Start(){
animatorHrace = this.getComponent<Animator>();
 //proměnná „animatorHrace“ se přiřadí z objektu, ke kterému je script
 //přiřazen
 nepritel = GameObject.Find(„nepritel1“);
 //Hledá objekt s názvem „nepritel1“
 animatorNepritele = nepritel.getComponent<animator>();
 //proměnná „animatorNepritele“ se přiřadí z objektu „nepritel“
}
```

JS:

```
public animatorHrace : Animator;
public nepritel : GameObject;
public animatorNepritele : Animator;


function Start(){
animatorHrace = this.getComponent(Animator) as Animator;
 //proměnná „animatorHrace“ se přiřadí z objektu, ke kterému je script
 //přiřazen
 nepritel = GameObject.Find(„nepritel1“);
 //Hledá objekt s názvem „nepritel1“
 animatorNepritele = nepritel.getComponent(Animator) as Animator;
 //proměnná „animatorNepritele“ se přiřadí z objektu „nepritel“
}
```

Příkaz pro přiřazení komponenty je

Pro C#:

```
vychoziObjekt.getComponent<nazevKomponenty>();
```

Pro JS:

```
vychoziObjekt.getComponent(nazevKomponenty) as nazevKomponenty;
```

Název komponenty může být kterákoliv komponenta v Unity, včetně názvu skriptu. Pokud je vyžadováno, aby přirazená hodnota komponenty byla v objektu, ve kterém je skript přiřazen, použije se jako výchozí objekt **this**.

3.4.4 Kolekce

Kolekce slouží v Unity mimo jiné i pro práci s různými komponenty a objekty, kterých je ve scéně větší množství. v unity je možné použít velké množství kolekcí, ovšem níže v kapitole jsou uvedeny dva nejpoužívanější typy kolekcí, obě tyto kolekce jsou použitelné v C# i v JS.

Obě zmíněné kolekce jsou upravitelné v okně Inspector, pokud mají veřejný modifikátor přístupu.

Vestavěný Pole

Vestavěný Pole je nejzákladnější typ pole, který je již zabudovaný v syntaxi C# i v JS kódu. Je limitovaný pevně danou velikostí, ovšem pomocí příkazu je možné velikost změnit. Tento typ pole je nejrychlejší a používá se, pokud je předem znám počet předmětů uložených v poli.

Deklarace Pole

Probíhá podle syntaxe programovacího jazyka.

V C# je to:

```
public TypPole[] nazevPole;
```

V JS je to:

```
public var nazevPole : TypPole [];
```

Je nutné definovat typ pole, typ pole může být běžný Int, String, Float atd. Typ pole může být i komponenta, případně objekt. Následně se definuje název pole.

Přiřazení objektů do pole

Inicializace objektů do pole probíhá nejčastěji pomocí příkazu:

```
nazevPole = GameObject.FindGameObjectsWithTag("název tagu");
```

V příkazu je nutné definovat název již deklarovaného pole, na konci je nutné uvést tag, podle kterého bude příkaz hledat všechny objekty. Ovšem příkaz vyhledává objekty v nespecifickém pořadí, proto je často nutné pole seřadit.

Seřazení GameObjectů v poli podle jména

Pokud je potřeba seřadit objekty podle názvu (například při kombinaci více polí), je nutné na začátku skriptu importovat do skriptu Linq pomocí příkazu:

Pro C#:

```
using System.Linq;
```

Pro JS:

```
import System.Linq;
```

Následně je nutné použít příkaz na seřazení objektů v poli.

Pro C#

```
nazevPole = nazevPole.OrderBy(go=>go.name).ToArray();
```

Pro JS:

```
nazevPole = nazevPole.OrderBy(function(go) go.name).ToArray();
```

To znamená, že pokud je nutné za běhu aplikace seřadit pole, je nutné mít minimálně 2 pole, ovšem je možné seřadit pole přímo při inicializaci, kde se použije kombinace výše zmíněných příkazů:

Pro C#:

```
nazevPole = GameObject.FindGameObjectsWithTag("NázevTagu").OrderBy(go=>go.name).ToArray();
```

Pro JS:

```
nazevPole = GameObject.FindGameObjectsWithTag("NázevTagu").OrderBy(function(go)
go.name).ToArray();
```

Generický seznam

Generický seznam má dynamickou velikost a umožnuje přidávat, odebírat a měnit položky za běhu aplikace.

Pro využití kolekcí je nutné na začátek skriptu přidat:

Pro C#:

```
using System.Collections.Generic;
```

Pro JS:

```
import System.Collections.Generic;
```

Deklarace seznamu:

Pro C#:

```
List<TypSeznamu> nazevSeznamu = new List<TypSeznamu>();
```

Pro JS:

```
var nazevSeznamu : List.<TypSeznamu> = new List.<TypSeznamu>();
```

V obou případech je nutné uvést TypSeznamu (Int, String, GameObject, komponenty), následně je třeba uvést název seznamu.

Přiřazení položek do seznamu

Přidávání položek do generického listu je možné pomocí příkazu:

```
nazevSeznamu.Add(novaPolozka);
```

V tomto momentě se novaPolozka přidá na konec seznamu.

Pokud chceme přidat položku na určitou pozici v seznamu, je nutné využít příkaz:

```
nazevSeznamu[cisloPozice] = novaPolozka;
```

Hodnota cisloPozice určuje, na kterou pozici se přidá nová položka, např. pozice 0 značí přidání položky na začátek listu.

Přiřazení objektů do seznamu

V případě, že chceme přidat do seznamu hromadně položky objektů, je nutné si pomoci například for each cyklem.

Pro C#:

```
foreach (var promena in GameObject.FindGameObjectsWithTag("název tagu"))
{
        nazevSeznamu.Add(promena);
}
```

Pro JS:

```
for (var promena: GameObject in GameObject.FindGameObjectsWithTag("název tagu"))
{
        nazevSeznamu.Add(promena);
}
```

For each cyklus je v obou případech tvořen podle syntaxe daného kódu, stejně jako u pole se hledají objekty pomocí tagu objektu, v cyklu je nutné si definovat dočasnou proměnnou, která se následně po provedení každého jednoho cyklu přidá do seznamu.

Seřazení GameObjectů v seznamu podle jména

Stejně jako u předchozí varianty se k seřazení objektu podle jména používá Linq, který je nutné importovat na začátku skriptu.

Pro C#:

```
using System.Linq;
```

Pro JS:

```
import System.Linq;
```

Následně je nutné použít příkaz na seřazení GameObjectů v seznamu.

Pro C#

```
nazevSeznamu = nazevSeznamu.OrderBy(go=>go.name).ToList();
```

Pro JS:

```
nazevSeznamu = nazevSeznamu.OrderBy(function(go) go.name).ToList();
```

Jak je vidět na obou příkazech, tak příkaz na seřazení listu je podobný příkazu na seřazení pole. Jediným rozdílem je závěr příkazu, který není **ToArray()**, ale **ToList()**.

Využití pole nebo seznamu pro komponenty

V případě, že je potřeba využít výše zmíněné pole nebo generický seznam pro některou komponentu, je možné využít kombinaci příkazů v této a v předchozí kapitole.

Prvním krokem je vyhledat si objekty pomocí tagu a následně z objektu získat komponentu pomocí příkazu: **nazevObjektu.getComponent**.

Pokud je potřeba využít pole komponenty, je jedním z možných řešení vytvořit si pomocné pole GameObjectů a z tohoto pole následně pomocí for cyklu přidat položky do pole komponenty.

Příklad pro C#:

```
GameObject[] poleObjektu; //pomocné pole
nazevKomponenty [] poleKomponenty;

poleObjektu = GameObject.FindGameObjectsWithTag ("název tagu");
poleKomponenty = new nazevKomponenty [poleObjektu.Length]; //nastaví délku pole
for (int i = 0; i < poleObjektu.Length; i++){
	poleKomponenty[i] = poleObjektu[i].GetComponent<nazevKomponenty>();
}
```

Příklad pro JS:

```
var poleObjektu : GameObject[]; //pomocné pole
var poleKomponenty : nazevKomponenty [];

poleObjektu = GameObject.FindGameObjectsWithTag ("název tagu”);
poleKomponenty = new nazevKomponenty [poleObjektu.Length]; //nastaví délku pole
for (var i = 0; i < poleObjektu.Length; i++){
	poleKomponenty[i] = poleObjektu[i].GetComponent(nazevKomponenty) as nazevKomponenty;
}
```

U kolekcí je možné přidávat komponenty podobným způsobem jako objekty. Pomocí for each cyklu je možné vyhledat komponenty, následně stačí do seznamu přidat komponentu dočasné proměnné z cyklu.

Příklad pro C#:

```
foreach (var promena in GameObject.FindGameObjectsWithTag("název tagu"))
{
	nazevSeznamu.Add(promena.getComponent<nazevKomponenty>());
}
```

Příklad pro JS:

```
for (var promena: GameObject in GameObject.FindGameObjectsWithTag("název tagu")){
	nazevSeznamu.Add(promena.getComponent (nazevKomponenty) as nazevKomponenty);
}
```

4 Tvorba aplikace

Hlavní náplní této bakalářské práce bylo zpočátku vytvořit několik samostatně spustitelných aplikací jako tematické sady 2D úloh v návaznosti na aplikace Klikání myší, Tahání myší a Píšeme na klávesnici, které byly vytvořeny podle aplikací sloužících jako přílohy k učebnici Informatika pro 1. stupeň základní školy.

V průběhu práce se změnila náplň praktické části na vytvoření jedné kompletní aplikace, jejíž náplní je sloužit v hodinách informatiky pro 1. stupeň základní školy jako nástroj na naučení se určitých aspektů ovládání počítače. Do aplikace se vytvořily zbývající sady úloh a přidaly se již vytvořené sady.

Aktuální, již hotová aplikace, obsahuje celkem 8 tematických sad:

- Klikání myší
- Tahání věcí
- Kreslení čáry
- Kreslení a vybarvování
- Ovladače
- Psaní na klávesnici
- Doplňování a úprava texu
- Přehrávání zvuku

Každá z těchto 8 sad obsahuje 4–6 unikátních úloh, celkem je ve hře 42 různých úloh. Jednotlivé sady úloh se spouští pomocí hlavního menu.

Sady Klikání myší a Tahání věcí vytvořili Aleš Velek a Kateřina Márová jako samostatné aplikace, tyto aplikace byly vytvořeny jako první a od těchto dvou aplikací se odvíjel koncept dalších tematických sad. v těchto sadách byly provedeny pouze drobné úpravy.

Sada Psaní na klávesnici byla vytvořena mnou, také jako samostatně spustitelná aplikace. Aplikace byla vytvořena o jeden rok později než předchozí dvě aplikace. Tato sada byla později výrazně upravena, jedna úloha se přesunula do jiné sady a zbylé úlohy prošly výraznou změnou.

Sadu Přehrávání zvuku vytvořil také jako samostatnou aplikaci Jiří Vlasák. Tato aplikace byla vytvořena již v době psaní této bakalářské práce. Tato sada byla zpracovaná trochu jiným způsobem než ostatní sady. Proto bylo nutné tuto sadu výrazněji upravit.

Všechny tyto 4 aplikace byly vytvořeny v rámci předmětu KIN/TDSA (Tvorba didaktického software).

Zbylé 4 sady a všechny ostatní části aplikace již byly kompletně vytvořeny jako součást bakalářské práce.

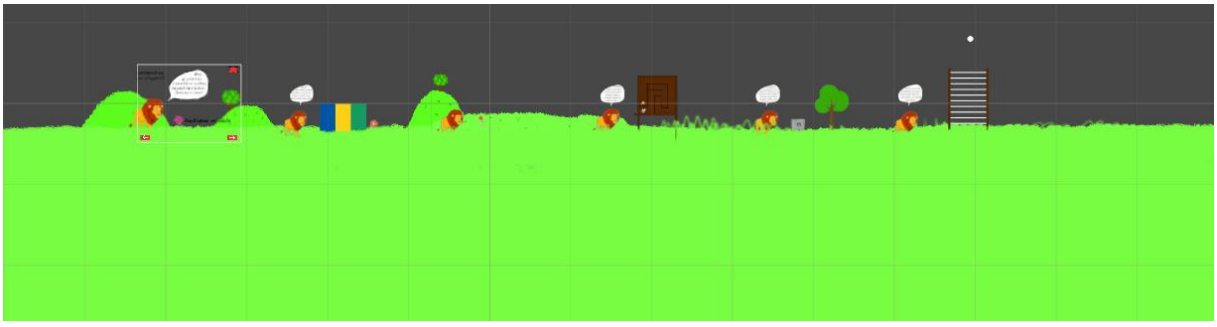
Aplikace obsahuje v celkovém počtu 9 scén, jednu scénu pro hlavní menu a zbylých 8 scén pro jednotlivé sady úloh.

4.1 Koncepce jednotlivých tematických sad

Každá sada má určité zaměření, ve kterém by se měl uživatel hry hraním zlepšovat. Každá sada obsahuje 4–6 úloh. v každé sadě je některé zvířátko, které provází hráče celou sadou. Zvířátko komunikuje s hráčem pomocí bublin, ve kterých se nachází text. Tato komunikace slouží převážně k zadání úkolů jednotlivých úloh a poskytuje zpětnou vazbu.

Pří spuštění sady kamera automaticky najede na „uvítací část“, kde se nachází zvířátko, které se představí.

Všechny sady jsou vytvořené jako jedna scéna v Unity a jednotlivé úlohy jsou poskládány vedle sebe.



Obrázek 19 – Náhled na celou sadu

4.1.1 Koncept kamery

Kamera je nastavená v perspektivním módu, a tak dává sadám „nádech“ 3D prvků. Většina objektů ve scéně je v ose Z na hodnotě 0, zatímco kamera je při spuštěné hře v ose Z na hodnotě -8.5.

Scéna obsahuje několik objektů (v závislosti na počtu úloh), které mají pouze komponentu Transform, tyto objekty se liší pouze pozicí na ose X. Na začátku hry se pomocí skriptu načtou tyto pozice a podle nich se určuje, kam se přesune kamera.

Přesun kamery zajišťují dvě tlačítka, díky kterým se kamera přesune na předchozí/další pozici, pokud není kamera na první/poslední pozici. Stisknutím tlačítka se kamera drobným obloukem přesune na požadovanou pozici.

Kamera má kromě 2 tlačítek pro přesun kamery také jedno tlačítko pro ukončení scény (přesun do hlavního menu).

Původně byla využívána GUI tlačítka (v době tvorby prvních aplikací to byla jediná možnost grafického rozhraní), ovšem později byla GUI tlačítka nahrazena běžnými tlačítky (UI komponenty).

4.1.2 Grafický koncept

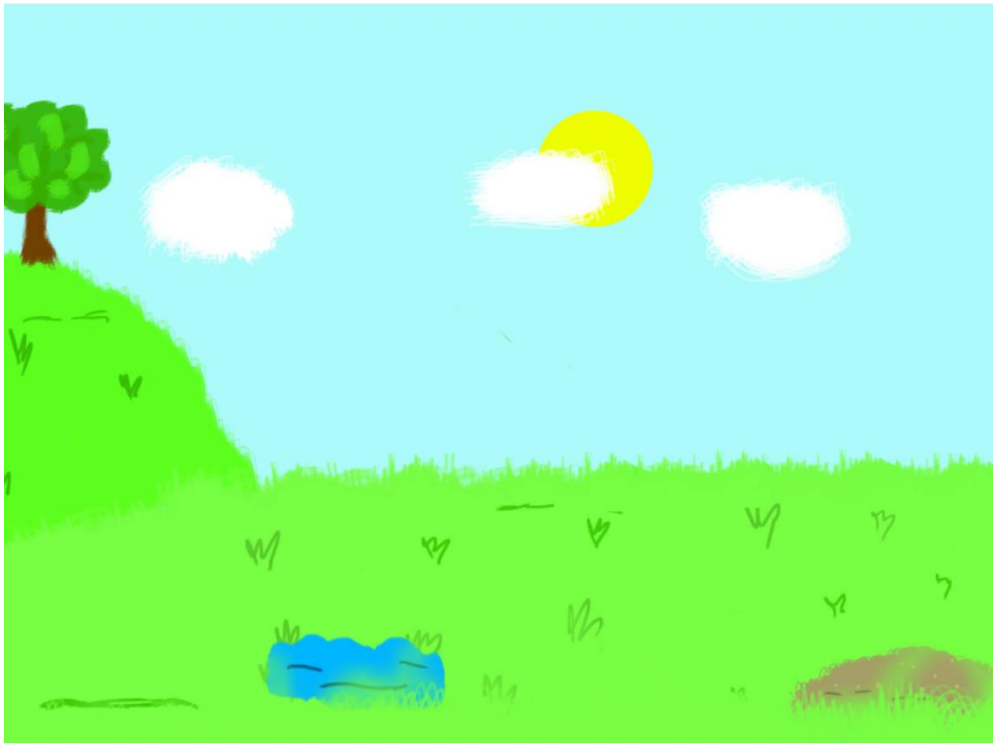
Grafická část každé sady je vytvořena vcelku jednoduchým způsobem. Na ose Z na pozici 0 je vytvořen obrázek, který slouží jako pozadí úlohy, tyto obrázky obsahují ve všech případech trávu. v některých případech obsahují kopce, stromy nebo vodu v závislosti na potřebě dané úlohy. Zbytek obrázku je průhledný. Pravá a levá strana obrázku musí navazovat na předchozí/další obrázek (pokud se nejedná o první/poslední obrázek). Každá sada úloh obsahuje jeden obrázek pro „uvítací část“ a následně jeden obrázek pro každou úlohu.

Později byly vytvořeny další obrázky jako pozadí, které byly vloženy mezi výše zmíněné obrázky úloh, aby se zvýšil rozestup mezi jednotlivými úlohami, jelikož při některém rozlišení obrazovky bylo možné vidět části následujících/předchozích úloh.

Další část grafické stránky tvoří tráva. Kromě trávy, která je na obrázku zmíněném výše, každá scéna obsahuje několik objektů s obrázkem trávy, které jsou umístěné na ose Z na pozici -2, to znamená že tráva je malý kousek před obrázky a při přesunu kamery mezi jednotlivými úlohami, je tato tráva před obrázky vidět.

Daleko za výše zmíněnými obrázky se nachází mraky a slunce. Mraky jsou tvořeny jedním obrázkem a jejich objekt je na ose Z na pozici 168. Objekt s obrázkem slunce je na ose Z na pozici 309. Slunce je oproti mrakům přibližně ve dvojnásobné vzdálenosti od kamery. Tato velká vzdálenost způsobuje, že při posunu kamerou se mraky i slunce pohybují o malý kousek.

V místě, kde nejsou žádné obrázky, se vykresluje světle modrá barva značící nebe. To je vykreslené díky nastavení na komponentě kamery.



Obrázek 20 – Náhled na grafiku

4.1.3 Uživatelské rozhraní

Ve všech tematických sadách jsou UI komponenty, které zajišťují přesun mezi jednotlivými úlohami pomocí šipek, případně tlačítko pro ukončení. Canvas je ve všech sadách nastaven v módu Screen Space – Camera (Canvas je připnutý na kameru, nezáleží na tom, jaká je pozice kamery, UI komponenty jsou na obrazovce stále ve stejné pozici). Screen mode Scaler je nastavený na hodnotu Scale With Screen Size. Tím pádem velikost komponent se mění podle aktuálně používaného rozlišení, jako výchozí rozlišení je nastavené 1024x768.

Zároveň ve většině jednotlivých úloh jsou využívány UI komponenty pro potřeby dané úlohy.

V mnou vytvořených sadách má každá úloha jeden objekt, který slouží jako složka, ve které jsou objekty s potřebnými UI komponenty. Tyto objekty (složky) jsou aktivovány/deaktivovány podle potřeby (v momentě deaktivace objektu se deaktivují všechny pod-objekty).

V sadě Klikání myší vyžaduje UI komponentu pouze jedna úloha. Ostatní úlohy již žádné UI komponenty nevyžadují, stejně jako všechny úlohy v sadě Tahání věcí.

V sadě Přehrávání zvuku je Canvas využíván v módu World Space, tím pádem je pevně daný ve scénách.

4.1.4 Zapínání a vypínání úloh

Veškeré mnou vytvořené sady obsahují jeden skript zajišťující průběh hry. Tento skript zajišťuje aktivování/deaktivování objektů, spouštění animací atd. podle aktuální pozice kamery. Tento skript je aktivní po celou dobu běhu scény.

Aktivování/deaktivování objektů slouží převážně k zapnutí určité úlohy, zobrazení potřebného Uživatelského rozhraní. Příklad využití je například v momentě, kdy se kamera přesune z druhé pozice na třetí pozici (pozice pro druhou hru), tak skript deaktivuje objekty potřebné pro první úlohu a aktivuje objekty se skriptem úlohy a objekty s UI komponenty určené pro druhou hru.

Toto řešení aktivování/deaktivování objektů je použito z důvodu optimalizace hry (méně aktivních objektů, menší nároky na počítač) a zároveň to usnadňuje tvorbu některých skriptů.

V sadách Klikání myší, Tahání věcí a Přehrávání zvuku není toto řešení použité, jelikož mají jinak řešený Canvas, případně Canvas nepoužívají, viz předchozí kapitola. Zároveň mají většinu objektů zajišťujících jednotlivé úlohy aktivní neustále. Toto řešení má nevýhodu například v sadě Přehrávání zvuku v momentě, kdy si hráč nechá v úlohách 2–5 přehrát některý zvuk a následně se přesune na jinou úlohu, se zvuk stále přehrává, i když nesouvisí s aktuální úlohou.

4.2 Zkompletování aplikace

Jak již bylo v práci zmíněno, původní plán této bakalářské práce bylo vytvořit jednotlivé sady úloh, které měly sloužit jako samostatně spustitelné aplikace, a tak bylo vytvořeno 8 odlišných projektů. v momentě, kdy došlo k rozhodnutí tyto aplikace zkompletovat do jedné aplikace, bylo nutné vytvořit jeden projekt a do tohoto projektu postupně přesunout veškeré assety.

Aby bylo možné spouštět jednotlivé sady úloh, bylo vytvořeno hlavní menu, ze kterého se jednotlivé sady spouští.

Hlavní důvody ke zkompletování aplikace byly:

- Jednodušší spuštění – pro uživatele je jednodušší spustit pouze jednu aplikaci a mezi jednotlivými sadami přecházet rychle pomocí hlavního menu, než nuceně vypínat celou aplikaci a spouštět jinou.
- Jednodušší distribuce – podobný důvod jako v předešlém bodě, ať už se jedná o distribuci pro platformu Windows nebo pro webovou platformu, je jednodušší vytvořit jednu aplikaci.
- Nastavení – v případě potřeby úpravy nastavení aplikace (rozlišení aplikace, spuštění v okně atd.), je opět jednodušší tuto změnu nastavení provést pouze jednou.
- Velikost aplikace – kdyby se porovnal součet velikostí jednotlivých aplikací, tento součet by byl značně vyšší než velikost jednotné aplikace z toho důvodu, že spousta assetů se vyskytuje v různých sadách (například obrázky).

Za nevýhodu celé aplikace by se dalo považovat to, že v případě potřeby spuštění pouze jedné aplikace se spouští celá aplikace, toto může být nevýhoda zvlášť u webové aplikace, kde se při spuštění hry stahují všechna data.

4.2.1 Přesouvání všech projektů do jednoho

Hlavním úkolem tedy bylo přesunout veškeré používané assety v projektu sady do projektu jednotné aplikace.

Unity obsahuje funkci, která umožnuje z určitých assetů vytvořit balíček a tento balíček přesunout do jiného projektu, tato funkce mi výrazně pomohla u veškerého přesouvání. Tento balíček se vytvořil kliknutím v projektovém okně na scénu pravým tlačítkem a zvolením možnosti **Export Package**, tato funkce zajistila zabalení celé scény, všech objektů ve scéně (včetně jejich nastavení a komponent) a všech assetů používaných ve scéně do jediného balíčku. Toto exportování má taky velikou výhodu v tom, že se exportují pouze používané assety (balík je vyčištěn od nepoužívaných assetů, které by pouze zbytečně zabíraly místo).

Tento balíček stačilo naimportovat v projektu pro jednu aplikaci a scéna byla prakticky hned spustitelná.

Problém tohoto importování nastal v momentě, kdy se importovaly assety, které měly stejné jméno jako již některý asset v projektu, tyto assety se neimportovaly a ve scéně se využil původní asset se stejným jménem. To sice bylo v pořádku u všeobecných věcí, které byly ve všech sadách stejné (tráva, slunce, mraky), ale problém nastal například u obrázků bublin ke komunikaci, které se jmenovaly ve většině sadách stejně („bublina1“, „bublina2“ atd.), ale měly jiný obsah, nebo u skriptů, které také měly často stejný název, ale jiný kód. Proto bylo nutné před exportováním balíčků si udělat „pořádek“ v assetech a všem nastavit unikátní název.

4.2.2 Import Klikání myší a Tahání věcí

Od těchto dvou prvotních sad úloh, které byly vytvořené Alešem Velkem a Kateřinou Márovou, mi bohužel nebyly dodány zdrojové kódy finálního projektu, ale pouze kódy některé před-finální verze. Takže verze těchto 2 sad, kterou jsem já mohl zařadit do společného projektu, nebyly totožné s finální verzí těchto 2 úloh (které jsem měl možnost vidět pouze ze spustitelné aplikace). Rozdíly byly v některých chybějících assetech (například voda pod rybičkami), případně nebyly kompletně funkční skripty 2 úloh.

Tím pádem bylo nutné 2 skripty upravit a assety znovu vytvořit. Chybějící obrázky byly znovu vytvořeny a nakresleny podle finální verze.

Zároveň tyto sady byly nastavené pro pevně dané rozlišení, proto bylo nutné v sadě provést pár drobných úprav, aby sady vypadaly stejně pro různá rozlišení.

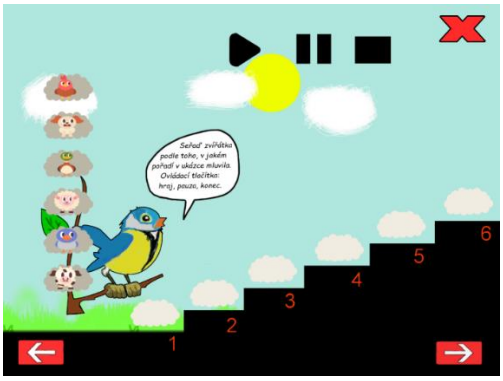
4.2.3 Import Přehrávání zvuku

Jak již bylo výše zmíněno, tato sada od Jiřího Vlasáka byla vytvořená trochu jiným způsobem, to platí hlavně o kameře, grafické stránce a celkově jinak umístěných objektech. Zároveň hra byla vytvořená pouze pro rozlišení 16:9 a bylo možné spustit hru pouze v okně. Tím pádem před importováním bylo nutné tuto hru „přepracovat“, aby tato sada vypadala podobně jako ostatní sady.

Hlavní změnou musela projít kamera, aby odpovídala konceptu ostatních sad, hlavní změnou bylo to, že původně byla kamera v módu Ortografickém a v ostatních sadách je používaná kamera v módu Perspektivním. Kvůli této změně se musely manuálně přesunout všechny objekty ve scéně, případně bylo potřeba některé objekty úplně upravit. Zároveň bylo nutné přepsat skript zajišťující funkci Drag and drop.

Také byl předělán grafický kabát aplikace, místo stejného opakujícího se pozadí, bylo nahráno pozadí z jiné sady, bylo přidáno slunce, mraky, tráva atd.

Obrázek 21 – Původní sada Přehrávání zvuku



Obrázek 22 – Upravená sada Přehrávání zvuku

4.2.4 Hlavní menu

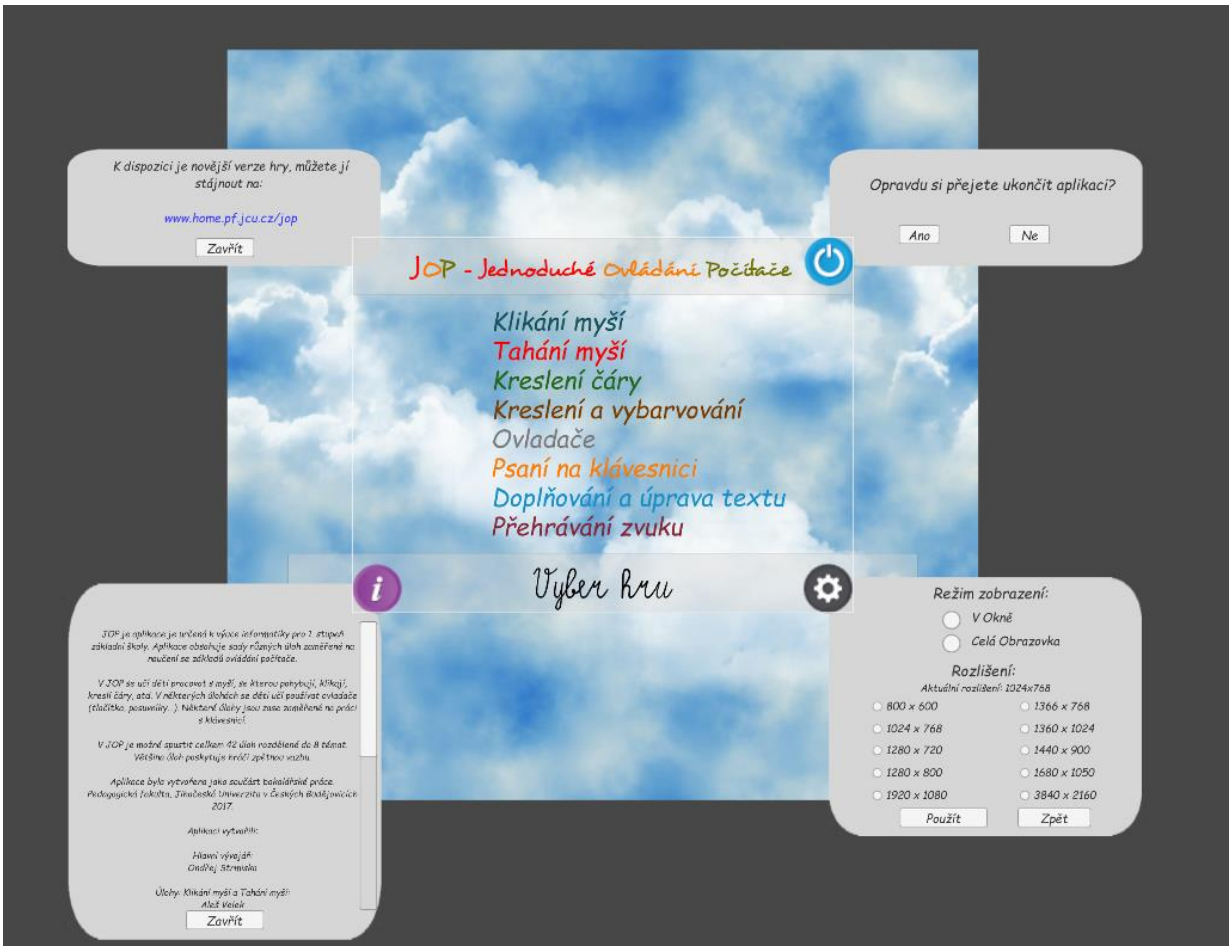
Aby bylo možné jednotlivé sady úloh spouštět, bylo potřeba vytvořit hlavní menu aplikace. Kromě spouštění jednotlivých úloh je možné z tohoto menu ukončit celou aplikaci, změnit rozlišení, případně zjistit informace o aplikaci.

Jako pozadí celého menu je nastavený opakovatelný obrázek[3] oblohy a zároveň je k obrázku přiřazen skript, že se obrázek neustále posouvá v čase doleva.

[3] Obrázek, jehož postranní hrany plynule navazují na hrany z opačné strany.

Zbytek celého hlavního menu (kromě objektu s komponentou Camera) je tvořen v Canvasu pomocí UI komponent.

Podstatnou část hlavního menu tvoří texty sloužící jako tlačítka pro spuštění jednotlivých úloh, každý tento text má jinou barvu, která se mění po najetí myší, případně kliknutím na tlačítko. Tyto texty jsou ve fontu **Comic Sans MS Italic**. v horní části menu je panel, ve kterém je tlačítko pro vypnutí aplikace (zrušení fullscreen módu), obrázek s názvem aplikace, který je ve fontu **DESYREL_, a** tlačítko, které se zobrazí v případě, že je dostupná nová aktualizace. v dolní části obrazovky je panel, ve kterém se nachází tlačítko pro zobrazení informací o aplikaci, a tlačítko pro změnu nastavení aplikace. Zároveň se zde nachází text, který se dynamicky mění v závislosti, na kterou z aktivních možností najede hráč myší, případně se zobrazuje načítání scény. Tento text je ve fontu **skolacek-ce** (dětské psací písmo).



Obrázek 23 – Hlavní menu

Tlačítka v hlavním menu

Podstatnou částí hlavního menu jsou čtyři tlačítka: Vypnutí, Aktualizace, Nastavení a Informace. Tato tlačítka jsou umístěna v rozích obrazovky. Každé z těchto tlačítek má svůj vlastní panel, který obsahuje animaci, při kliknutí na tlačítko se panel „otevře“ přesunutím panelu do středu obrazovky, každý z panelů obsahuje tlačítko pro „zavření“ panelu.

Samotné přesunutí panelu doprostřed obrazovky nedeaktivuje ostatní tlačítka, proto bylo nutné vytvořit další objekt s průhledným panelem přes celou obrazovku (který je defaultně vypnutý), tento objekt obsahuje také animaci, která způsobuje „zatmavení“ všech komponent, které jsou na objektech v hierarchii objektů níže.

Tím pádem například při spuštění tlačítka Nastavení se provede několik procesů:

1) Spustí se animace na objektu s panelem Nastavení, která přesune panel do středu obrazovky.
2) Panel Nastavení se přesune v hierarchii na poslední pozici.
3) Panel pro zatmavení (zatím deaktivovaný) se přesune v hierarchii na předposlední pozici.
4) Aktivuje se objekt s panelem pro zatmavení (na žádnou z komponent nepůjde kliknout, kromě komponent pod-objektů panelu nastavení).
5) Spustí se animace na objektu s panelem pro zatmavení (zatmaví se všechny objekty kromě panelu nastavení).

V momentě, kdy se klikne na tlačítko pro schování nastavení, se provedou tyto procesy:

1) Spustí se animace na objektu s panelem pro přesun panelu na své úvodní místo.
2) Spustí se animace na objektu s panelem pro zatmavení, která zesvětlí veškeré objekty.
3) Deaktivuje se objekt s panelem pro zatmavení (půjde kliknout na všechny komponenty).

Panel Ukončit

Panel s tlačítkem pro ukončení se nachází v pravém horním rohu obrazovky, má poměrně jednoduchý účel, obsahuje buď tlačítko pro potvrzení ukončení aplikace, nebo tlačítko pro zavření panelu ukončení.

Ovšem pokud aplikace běží na platformě WebGL, tak je toto tlačítko defaultně schované a tlačítko se zobrazí pouze v tom momentě, kdy hra poběží v režimu fullscreen. v takovém případě slouží tlačítko jako funkce ukončení režimu fullscreen.

Panel Aktualizace

Panel Aktualizace je defaultně skrytý a obsahuje pouze upozornění na novou verzi aplikace. Na stránkách aplikace je vytvořen textový soubor obsahující číslo poslední verze aplikace. Při spuštění hlavního menu se hra pomocí jednoduchého skriptu připojí k tomuto souboru. Pokud skript zjistí, že číslo poslední verze je vyšší než číslo používané verze aplikace, tak se tento panel zobrazí, ve všech ostatních případech zůstává panel skrytý (i pokud se k tomuto souboru nemůže připojit).

Panel Informace

Panel s informacemi o hře se nachází v levém dolním rohu obrazovky a na tomto panelu lze nalézt základní informace o aplikaci, panel obsahuje pouze scrollovatelný text a tlačítko pro zavření panelu.

Panel Nastavení

Tento panel se nachází v levém dolním rohu a obsahuje veškeré možnosti nastavení rozlišení. v horní části panelu si může uživatel vybrat, zda chce mít aplikaci zobrazenou v okně nebo v módu fullscreen (defaultně se uživateli spustí aplikace v módu fullscreen). Ve spodní části panelu se nachází možnosti o vybrání si konkrétního rozlišení, na výběr je celkem z 10 základních rozlišení:

- 800x600
- 1280x720
- 1360x768
- 1440x900
- 1920x1080
- 1024x768
- 1280x800
- 1360x1024
- 1680x1050
- 3684x2160

Při spuštění aplikace hra poběží v rozlišení, které má uživatel defaultně nastavené na svém počítači. Ovšem pod podmínkou, že poměr rozlišení je maximálně 4:3, jinak se nastaví rozlišení 1024x768. Tato podmínka je vytvořena, protože aplikace není optimalizována pro větší poměr stran a nemusela by se správně zobrazovat.

Aplikace ověřuje toto rozlišení pomocí příkazu:

```
void Start () {
        widthTemp = Screen.width; // Screen.width = počet pixelů na šířku
        heightTemp = Screen.height; // Screen.width = počet pixelů na výšku
        widthVybrany = Screen.width;
        heightVybrany = Screen.height;

         if (((double)heightTemp / (double)widthTemp) >= 0.75){
                        widthTemp = 1024;
                        heightTemp = 768;
                        widthVybrany = 1024;
                        heightVybrany = 768;

                }
                aktualniRozliseni.text = "Aktuální rozlišení: " + widthTemp + " x " +
heightTemp;
                nastavPuvodniHodnoty();
                Screen.SetResolution(widthTemp, heightTemp, zobrazeniFullScreen);
        }
```

Proměnná heightTemp = počet pixelů na šírku a widthTemp = počet pixelů na výšku. Pokud je podíl těchto čísel větší než 0.75 (poměr 3:4), nastaví se rozlišení 1024x768, vypíše se uživateli aktuální rozlišení a zavolá se funkce nastavPuvodniHodnoty(), která zaškrtne toggle pro zobrazení v okně, a pokud je aktuální rozlišení některé z výše uvedených, tak se zaškrtne i toggle s rozlišením.

Celý tento panel je neaktivní, pokud je aplikace spuštěna na platformě WebGL.

4.3 Skripty v projektu

Celý projekt obsahuje celkem 112 skriptů, které zajišťují chod veškeré aplikace. Veškeré skripty jsou napsány v jazyce C#. v době tvorby sady Psaní na klávesnici byl používán jazyk Javascript. Ovšem později byly veškeré skripty změněny na C# a nově vytvořené skripty byly programovány v jazyce C#.

Tato změna přišla především z toho důvodu, že valná většina komunity Unity používá právě tento programovací jazyk a v případě shánění řešení problémů (ať už na různých fórech nebo u video-tutoriálů) bylo mnohem snazší najít řešení pro C# kód než pro JS kód. Zároveň JS skripty vykazovaly v projektu různé chyby.

Mnou vytvořené C# skripty byly vytvořené v programu Visual Studio, jelikož druhý nabízený (defaultní) software MonoDevelop vykazoval spoustu chyb, nabízel horší predikci kódu a byl nestabilní.

V ostatních sadách (mnou nevytvořených) jsou také použity pouze C# skripty.

4.4 Grafika a obrázky

Naprostá většina obrázků, která je v mnou vytvořených sadách, je stažena z různých stránek na internetu, které nabízejí tyto obrázky zdarma. Obrázky byly většinou staženy ze 3 různých stránek:

- **freepik.com** – stránka nabízející desítky tisíců obrázků zdarma ke stažení pro komerční i nekomerční užití, podmínkou je uvést v aplikaci autora obrázků.
- **classroomclipart.com** – stránka nabízející přes sto tisíc obrázků pro personální a edukační účely.
- **flaticon.com** – stránka obsahující převážně ikony, ale i běžné obrázky. Veškerý obsah je zdarma, pokud se v softwaru uvede jméno autora.

Většinu těchto obrázků bylo nutné upravit v grafickém editoru, v mém případě byl použit freeware program GIMP, který postačil na veškeré základní úlohy, jako jsou otočení obrázku, úprava velikosti obrázků, odstranění pozadí či tvorbu některých vlastních obrázků.

U všech stažených obrázků je dodržené licencování obrázků a autoři obrázků jsou uvedeni v hlavním menu v panelu Informace.

4.5 Tvorba jednotlivých úloh a aspektů hry

V této kapitole bude popsán postup tvorby některých mnou vytvořených úloh, případně aspektů hry, které bylo potřeba vytvořit k funkcionalitě některých úloh.

Úlohy byly kompletně vytvořené buď podle předlohy, viz kapitola Učebnicová předloha, případně se těmito úlohami nechaly inspirovat, případně byly vytvořené kompletně samostatně.

Aplikace obsahuje celkem 42 úloh, z toho 27 úloh jsou mnou vytvořené. Z těchto 27 úloh je možné většinu z nich dokončit a spustit je znovu. Jedinými výjimkami jsou 2 kreslící úlohy (hra nerozpozná co hráč nakreslil). Všechny ostatní úlohy se mohou spustit znovu (nebo se spustí automaticky).

Většina mnou vytvořených úloh byla navržena tak, aby obsahovala určitý prvek náhody. Tento prvek náhody se většinou vyskytuje při startu úlohy a zajistí náhodné rozřazení objektů zamícháním kolekcí atd. Tento prvek náhody zajistí ve většině úloh, že v případě opětovného spuštění úlohy, hráč nebude dělat úplně to samé, ale bude mít trochu poupravené zadání. Podobným příkladem může být to, že pokud bude hra spuštěna v učebně a více žáků bude hrát stejnou úlohu, všichni žáci nebudou s největší pravděpodobností dělat úplně to samé. Tento prvek náhody je v některých hrách limitován, aby se stupňovala obtížnost úlohy.

Úlohy v sadách, které jsem netvořil já, neobsahují možnost úlohu znovu spustit a ani neobsahují žádný prvek náhody (kromě úlohy Skořápky).

4.5.1 Psaní na klávesnici – Vytvoř slovo

Popis úlohy – úloha Vytvoř slovo nutí hráče z předem vytvořené šablony necelého slova doplnit jedno písmeno do šablony, a tím vytvořit podstatné jméno prvního pádu. v této úloze bylo nejtěžší najít takové šablony, které budou obsahovat alespoň tři slova, ale žádné z těchto slov nebude natolik složité, aby ho neznalo dítě na prvním stupni základní školy, příkladem takové šablony je **du_y**, z kterého lze vytvořit podstatná jména:

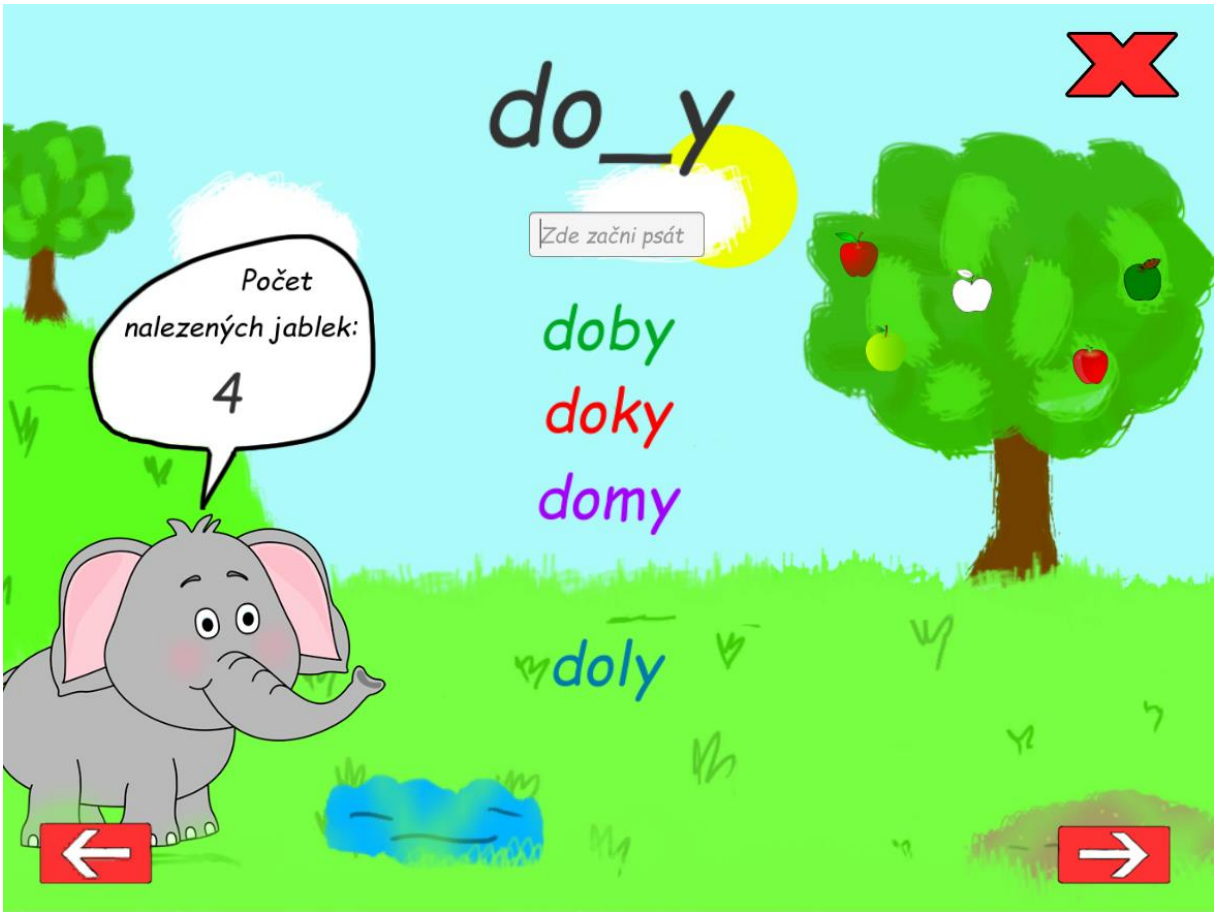
- duny
- duby
- dudy
- duhy

Cílem úlohy je vytvořit co nejvíce slov, na základě počtu vytvořených slov se odráží finální hodnocení.

Funkcionalita úlohy – úloha obsahuje několik UI komponent, jedno textové pole (pro psaní písmena), 2 tlačítka (pro pokračování na další slovo/spuštění úlohy znovu) a několik textových komponent (na zobrazení šablony, skóre a již vytvořených slov). Na stromě je celkem 16 objektů s jablky (z toho 8 prázdných) a 10 různých bublin pro komunikaci s hráčem.

Funkce Update() zajišťuje zobrazení aktuální šablony, na začátku nastaví na strom prázdná jablka. Následně funkce ověřuje, zda bylo zadáno správné písmeno. Pokud ano, tak se zobrazí slovo v textové komponentě a jedno jablko se změní na barevné. Pokud hráč zadá 2x špatné písmeno, zobrazí se tlačítko s možností nechat se poddat, kterým je možné zobrazit zbylá slova, ovšem poté za tato slova již nedostane „body“. Pokud jsou zobrazena všechna slova, deaktivuje se textové pole a bublina s pochválením. Pokračovat k dalšímu slovu je možné příslušným tlačítkem. Na konci všech slov uživatel dostane vyhodnocení, zobrazí se na stromě počet jablek podle procentuální úspěšnosti a vypíše se příslušná textová bublina.

Účel úlohy – děti hledají písmena na klávesnici a zároveň jsou nuceny tvořit vlastní slova, tím se rozvíjí jejich slovní zásoba.



Obrázek 24 – Úloha Vytvoř slovo

4.5.2 Psaní na klávesnici – psaní na čas

Popis úloh – úlohy Psaní písmen na čas a Psaní slov na čas jsou si velice podobné. Hráč si vybere obtížnost písmen (slov) a hráčův úkol je napsat co nejvíce písmen/znaků/slov za jednu minutu. Hráči se během psaní neustále objevují písmena (znaky) nebo slova, která musí hráč napsat do textového pole. Tyto znaky/slova jsou vybrány z náhodně zamíchaného pole podle toho, jakou si hráč vybral obtížnost. Po uplynutí jedné minuty se hráči ukáže jeho výsledné hodnocení podle počtu napsaných znaků/slov a jeho pořadí oproti ostatním zvířatům (šnek nejpomalejší – gepard nejrychlejší). Toto pořadí je vykreslené pomocí tabulky, ve které může hráč vidět, kolik toho stihla napsat ostatní zvířátka, tento počet je jiný pro každou obtížnost.

Po uplynutí jedné minuty má hráč možnost spustit si psaní znovu, případně zvolit jinou obtížnost.

Funkcionalita úloh – i tyto úlohy jsou tvořeny z větší části z UI komponent, kromě několik tlačítek (na vybrání obtížnosti) a jednoho textového pole, úlohy obsahují spoustu textových komponent, a to buď pro zobrazení požadovaného znaku/slova, skóre a zobrazení času, nebo hry obsahují 25 textových komponent pro tabulku.

Při vybrání obtížnosti se spustí čas, spustí se animace „běhu času“ (hořící svíčka), aktivuje se textové pole. Zároveň se pomocný generický seznam naplní z příslušného pole (podle vybrané obtížnosti), tento seznam se zamíchá.

Následně se pomocí funkce Update() odpočítává čas, vypisuje aktuální požadované slovo. Pokud je požadované slovo stejné jako obsah textového pole, přičte se skóre a slovo se ihned změní, zároveň se vymaže obsah textového pole.

Po skončení časového limitu se porovná získané skóre se skórem ostatních zvířátek. Podle skóre se zobrazí obrázek nejrychlejšího překonaného zvířete a ukáže se tabulka.

Účel úloh – je motivovat hráče k naučení se rychlejšímu psaní a zkusit v dané obtížnosti překonávat svůj nejlepší výkon.

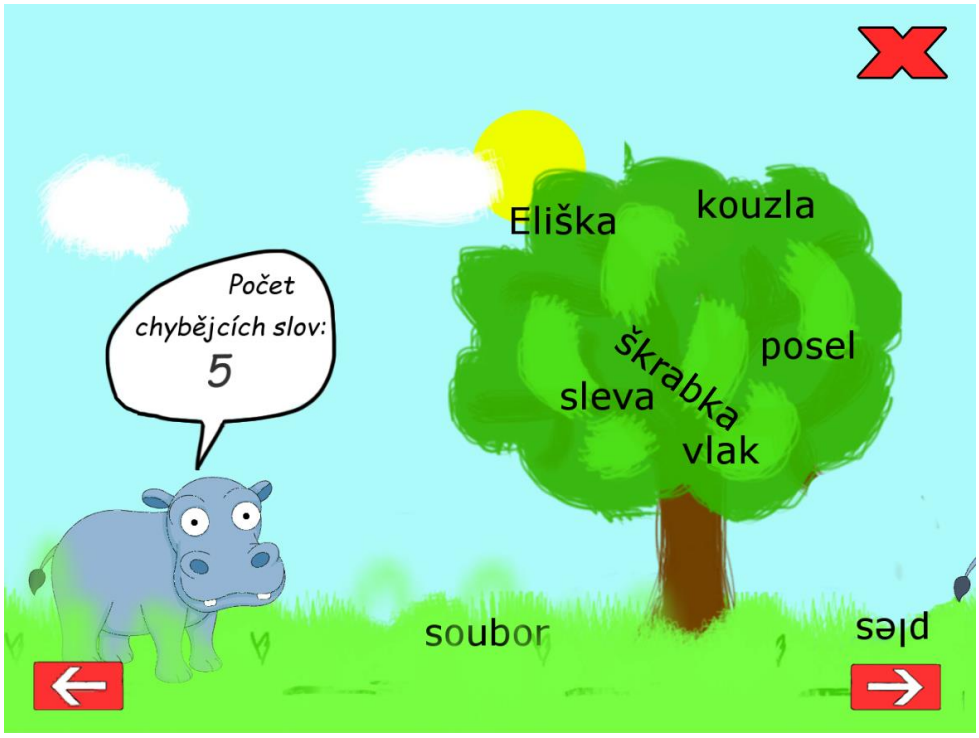
4.5.3 Doplňování textu – Změň na zvíře

Popis úlohy – v úloze je umístěno na stromě několik slov, hráč má za úkol kliknout na tato slova (slova se zobrazí v textovém poli) a následně ze slova smazat některé písmeno (případně více písmen) tak, aby ze slova vzniklo zvířátko. Pokud se to hráči povede, slova začnou padat ze stromečku dolů, ovšem slovo se může zaseknout na jiném slově. Cílem hry je dostat všechna slova na zem.

Funkcionalita úlohy – úloha obsahuje celkem 8 objektů jako slov, kde každý objekt má obrázek se svým slovem, zároveň má každý objekt komponenty Box Collider 2D a RigidBody 2D (zpočátku je pozice zamrzlá), zároveň každý z těchto obrázků má na sobě skript s několika proměnnými (id, špatné slovo, správné slovo atd.) a s funkcí OnMouseDown(), která detekuje kliknutí na obrázek (detekce je aktivní, pouze pokud obrázek ještě nespadnul). Tato detekce aktivuje funkci v hlavním skriptu hry, zobrazí textové pole a přednastaví se proměnná špatné slovo.

Textové pole obsahuje dynamickou funkci OnValueChanged(), která při změně obsahu v textovém poli zavolá funkci v hlavním skriptu, tato funkce ověřuje, zda je nový obsah stejný jako požadované slovo, objekt se slovem odmrzne a objekt začne padat směrem dolů.

Účel úlohy – naučit děti mazat z textu písmena a zároveň tvořit vlastní slova.



Obrázek 25 – Úloha Změna na zvíře s použitým gravitačních prvků

4.5.4 Ovladače – Změna barvy míče

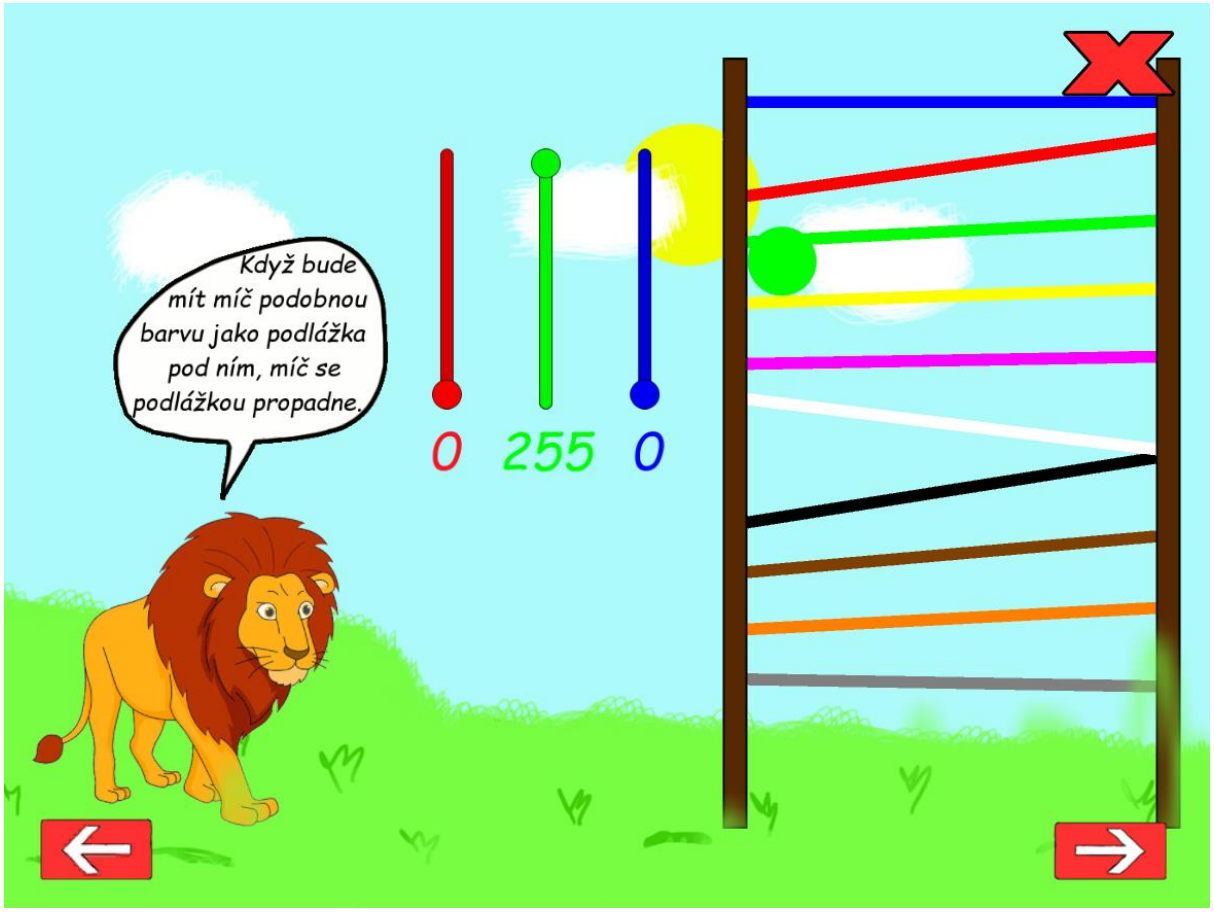
Popis úlohy – v úloze jsou na začátku celkem tři posuvníky a několik různě barevných podlážek. Když hráč pohne některým posuvníkem, objeví se hráči míč, který bude stát na nejvyšší podlážce. Princip hry je pomocí posuvníků, které reprezentují jednotlivé barvy RGB[4] modelu, nastavit barvu míče tak, aby měl míč barvu podobnou podlážce pod ním, tím se míč podlážkou propadne. Cílem hry je dostat míč na zem.

Funkcionalita úlohy – v této úloze je celkem 10 podlážek rozdělených do 4 kategorií, které mají na sobě Box Collider 2D a skript, který detekuje, zda se dostali do kolize s míčem (aktivuje se proměnná), zároveň je ve hře objekt s míčem, který obsahuje komponenty Box Collider 2D a RigidBody 2D. Úloha využívá několika bublin a v neposlední řádě úloha obsahuje 3 posuvníky, které mají nastavenou barvu podle toho, kterou část RGB modelu reprezentují. Tyto posuvníky mohou být pouze celá čísla a mít hodnotu 0-255. Posuvníky obsahují dynamickou funkci OnValueChanged(), která při posunu v posuvníku zavolá určitou funkci hlavního skriptu úlohy. Ve hře jsou ještě 3 textové komponenty, které se mění podle hodnoty posuvníku.

Podlážky jsou rozdělené do 4 kategorií (podle pořadí), stejně tak barvy podlážek jsou rozdělené do 4 kategorií. Barvy jsou rozdělené podle obtížností, aby hráč pochopil princip RGB v průběhu hry a nedostal například jako první barvu na namíchání některou těžkou barvu, se kterou si nebude umět poradit a radši hru vynechá.

Při spuštění hlavního skriptu úlohy se detekují všechny podlážky ve hře (přidají se do polí). Následně se zamíchá pořadí barev v jednotlivých polích a barvy se přiřadí podlážkám. Zároveň se podlážkám nastaví drobná náhodná rotace (aby míč nepadal pouze dolů, ale mohl se i kutálet).

[4] Model RGB neboli červená-zelená-modrá je způsob míchání barev používaný v monitorech. v tomto modelu je barva nastavená pomocí kombinace červené, zelené a modré barvy, rozsah hodnot těchto barev je 0-255.



Obrázek 26 – Úloha Změna barvy míče s použitím posuvníků pro volbu barvy

Barvy podlážek jsou rozdělené do 4 úrovní:

1. Základní barvy – červená, zelená, modrá. Jeden posuvník musí být nahoře, zbylý 2 dole.
2. Kombinace 2 barev – růžová, žlutá. Barvy jsou kombinací 2 barev (2 posuvníky musí být nahoře). Azurová barva byla vyřazena.
3. Bílá/Černá – tyto 2 barvy jsou vytvořené, pokud jsou všechny barvy na maximálních/minimálních hodnotách.
4. Složité barvy – oranžová, hnědá, šedá. Barvy, u kterých je složitější nastavit správnou hodnotu.

Jelikož hráč jen málokdy nastaví perfektní shodu barvy míče s podlážkou (například 122,122,122, není 122,122,123, i když barvy vypadají téměř stejně), je nutné detekovat barvu míče. Což není tak úplně jednoduché, jak se může na první pohled zdát. a tak hlavní skript obsahuje funkci, která pomocí mnoha podmínek zjistí, jakou barvu má aktuálně míč (tyto podmínky nemusí být perfektně nastaveny).

Ve funkci Update() hlavního skriptu úlohy je zajišťován chod hry (nastavuje barvu míče), v této funkci je for cyklus, který vyzkouší všechny jednotlivé podlážky a pokud má podlážka aktivní pomocnou proměnnou (míč se podlážky dotknul) a zároveň má míč podobnou barvu jako podlážka, tak podlážka bude ignorovat kolizi (tato kolize se aktivuje v případě nového spuštění hry), míč se podlážkou propadne.

Účel úlohy – účelem úlohy je naučit děti tomu, jak funguje míchání barev, jak mohou smíchat složitější barvy a zároveň se naučí používat posuvníky.

4.5.5 Kreslení čáry

Kreslení čáry je použito v 11 úlohách v sadách Kreslení čáry a Kreslení a vybarvování. Bohužel samotné Unity nenabízí přímo žádný nástroj na kreslení čáry, je nutné si pomoci vlastními skripty.

Většina čar je vytvořená jako nový objekt (prefab) ze skriptu dané úlohy a tento prefab obsahuje skript zajišťující tvorbu čáry.

Rovná čára

Níže vypsaný skript jménem **lajna_nova** tvoří rovnou čáru podle toho, zda je stisknuté levé tlačítko. v momentě stisknutí levého tlačítka nastaví první a poslední pozici čáry podle polohy myši, ale poslední pozice čáry se mění s každým snímkem (podle pozice myši) do té doby, dokud není levé tlačítko myši puštěno.

```
using UnityEngine;

public class lajna_nova : MonoBehaviour {
	Vector2 mousePos; // pozice myši
	startPos; // 1. pozice čáry
	endPos; // Poslední potice čáry
	public LineRenderer line; // Komponenta LineRenderer, nutné přiřadit v Inspectoru
	start; // Začala se čára tvořit?
	isPressed; // Je tlačítko spuštěné?
	public bool konec; // Je čára ukončená? Ptají se jiné scripty.

	void Update () {
		if(!konec){ // Pokud není tvorba scriptu ukončená
		    var v3 = Input.mousePosition;  // Vytvoří se pomocná proměnná komponenty
Vector3, která je nadefinována z pozice myši
		    v3.z = 8.5f; // Nastaví se vzdálenost od kamery
		    v3 = Camera.main.ScreenToWorldPoint(v3); // Pozice se upraví podle kamery
		    mousePos = v3; // mousePos se nastaví z pomocné proměnné
		    if(Input.GetMouseButtonDown(0){ // Zjišťuje, zda je levé tlačítko myši
stisknuté
		       isPressed = true;
		        if(start == false){ //Pokud ještě tvorba nezačala
				start = true;
				startPos = mousePos; // Nadefinuje se 1. pozice čáry
				line.numPositions = 1; // Nastaví se počet pozic čáry
		      line.SetPosition (0, startPos); // Nastaví se 1. pozice čáry

		    }
		    }
		    if(Input.GetMouseButtonUp(0)){ // Zjišťuje, zda je upuštěné levé tlačítko
myši
				start = false;
				isPressed = false;
				konec = true;
		    }
		    if(isPressed == true){ // Pokud je tlačítko stisknuté
				endPos = mousePos; // Nadefinuje se konečná pozice čáry
		      line.numPositions = 2; // Čára se skládá ze dvou bodů
		      line.SetPosition (1, endPos); // Nastaví se konečná pozice čáry
		    }
		}
	}
}
```

Prefab s tímto skriptem je volán ze skriptů jednotlivých úloh. Níže je uveden skript se jménem **priklad_vyuziti_lajny**. Tento skript nejprve vytvoří objekt z prefabu a zároveň k objektu přistupuje. v momentě, kdy se na prefabu v komponentě skriptu **lajna_nova** aktivuje proměnná **konec**, tak se na skriptu **priklad_vyuziti_lajny** provedou potřebné příkazy (například se zjistí, zda čára splňuje určité požadavky), tvorba se ukončí a v dalším snímku se vytvoří nový objekt s čárou, ovšem ta se začne tvořit až v momentě stisknutí levého tlačítka.

```
using UnityEngine;

public class priklad_vyuziti_lajny : MonoBehaviour {
public GameObject objekt; // pomocná proměnná pro budoucí objekt
public bool tvorim; // Je čára tvořena?
public GameObject objToSpawn; // Prefab ze kterého se bude čára tvořit
Color seda = new Color(0.77f,0.78f,0.823f); // Nastavená barva
       void Update () {
              if(tvorim == false){ //Pokud se již čára netvoří
                     tvorim = true; // čára se tvoří
                     objekt = Instantiate(objToSpawn); // Vytvoří nový objekt z prefabu
                     objekt.transform.parent = gameObject.transform; //objekt se stane
podobjektem objektu ke kterému je přiřazen tento scriptu
                     objekt.tag = "rovnaCara"; // nastaví tag objektu
                     objekt.GetComponent<lajna_nova>().line.startColor = seda; // nastaví
první barvu čáry
                     objekt.GetComponent<lajna_nova>().line.endColor = seda; // nastaví
poslední barvu čáry
                     objekt.GetComponent<lajna_nova>().line.startWidth = 0.1f; // nastaví
první šířku čáry
                     objekt.GetComponent<lajna_nova>().line.endWidth = 0.1f; // nastaví
poslední šířku čáry
              }
                  if(objekt.GetComponent<lajna_nova>().konec){ // Pokud je v objektu
aktivovaná proměnná konec
                     //Příkazy podle potřeby hry
                     tvorim = false;
              }
       }
}
```

Obrázek 27 - úloha Osy využívající kreslení čáry

Kreslená čára

Pokud je potřeba vytvořit kreslenou čáru, tak komponeta LineRenderer musí obsahovat více bodů, resp. přesně tolik bodů, v kolika místech se směr čáry mění (čára může mít tisíce bodů). Jako ukázka takto vytvořené čáry poslouží jeden obsáhlejší skript jménem **kreslena_cara**, který přistupuje k prefabu, který obsahuje komponentu LineRenderer. Tento skript funguje podobně jako předchozí 2 skripty s tím rozdílem, že zde se přistupuje přímo ke komponentě. Dalším rozdílem je, že dokud se táhne myší, tak na komponentě LineRenderer stále roste počet pozic a nové pozice se nastaví podle pozice myši.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class kreslena_cara : MonoBehaviour {
	public Vector2 mousePos; // Pozice myši
	public bool isPressed; // Je stisnuto levé tlačítko?
	public bool tvorim; // Tvořím lajnu?
	public GameObject objToSpawn; // Prefab ze kterého se bude čára tvořit
	public GameObject objekt; // objekt k vytvoření
	public List<Vector2> pointsList; // seznam pozic, ve kterých čára byla
	void Start () {
			pointsList = new List<Vector2>();} // inicializace seznamu
	void Update () {
	var v3 = Input.mousePosition;  // Vytvoří se pomocná proměnná komponenty Vector3,
která je nadefinována z pozice myši
		v3.z = 8.5f; // Nastaví se vzdálenost od kamery
		v3 = Camera.main.ScreenToWorldPoint(v3); // Pozice se upraví podle kamery
		mousePos = v3; // mousePos se nastaví z pomocné proměnné
		if(Input.GetMouseButtonDown(0)) { //pokud je levé tlačítko myši stisknuté
			isPressed = true;
		}if(Input.GetMouseButtonUp(0)){ // Zjišťuje, zda je levé tlačítko myši
stisknuté
			isPressed = false;
			tvorim = false; }
		if(isPressed == true){
		if (tvorim == false){ //Pokud se již čára netvoří
		tvorim = true; // čára se tvoří
			objekt = Instantiate(objToSpawn); // Vytvoří nový objekt z prefabu
			objekt.transform.parent = gameObject.transform; //objekt se stane
podobjektem objektu ke kterému je přiřazen tento scriptu
			objekt.tag = "kreslenaCara"; // nastaví tag objektu
			objekt.GetComponent<LineRenderer>().startColor = Color.blue; //
nastaví první barvu čáry
			objekt.GetComponent<LineRenderer>().endColor = Color.blue; // nastaví
poslední barvu čáry
			pointsList.RemoveRange(0,pointsList.Count); // Vymaže veškeré hodnoty
v seznamu
		}if (!pointsList.Contains (mousePos)){ // Pokud seznam neobsahuje pozici
mousepos
			pointsList.Add (mousePos); // Přidá se do seznamu pozice mousePos
			mousePos;
			objekt.GetComponent<LineRenderer>().numPositions = pointsList.Count;
// Nastaví se počet bodů v čáře, podle počtu hodnot v seznamu
			objekt.GetComponent<LineRenderer>().SetPosition (pointsList.Count - 1,
mousePos); }}}} //nastaví se poslední bod čáry podle aktuální pozice myši
```

Rovná čára s colliderem

Kvůli úloze Libovolné kreslení, která obsahuje nástroj Guma, bylo nutné zjistit, jak přidat čáře collider, bohužel Unity ani v tomto případě nenabízí žádnou vhodnou cestu pro vytvoření collideru k LineRendereru, proto bylo nutné si opět vypomoci skriptem. v tomto případě jsem využil již vytvořenou funkci někoho jiného[6], tuto funkci jsem přizpůsobil svým potřebám.

```
private void addColliderToLine()
	{
		col = gameObject.AddComponent<BoxCollider> (); // Přidá k objektu komponentu Box Collider
		float lineLength = Vector3.Distance (startPos, endPos); // Zjistí délku čáry
		col.size = new Vector3 (lineLength, tloustka, 1f); // Nastaví velikost Collider
		Vector3 midPoint = (startPos + endPos)/2; // Nastaví střed Collideru
		col.center = new Vector3(0,0,0);
		col.transform.position = midPoint; // Nastaví pozici Collideru
		// Následující řádky počítají úhel mez prní a poslední pozicí
		float angle = (Mathf.Abs (startPos.y - endPos.y) / Mathf.Abs (startPos.x - endPos.x));
		if((startPos.y<endPos.y && startPos.x>endPos.x) || (endPos.y<startPos.y && endPos.x>startPos.x))
		{
			angle*=-1;
		}
		angle = Mathf.Rad2Deg * Mathf.Atan (angle);
		if(!float.IsNaN(angle)){
			col.transform.Rotate (0, 0, angle);
		}
	}
```

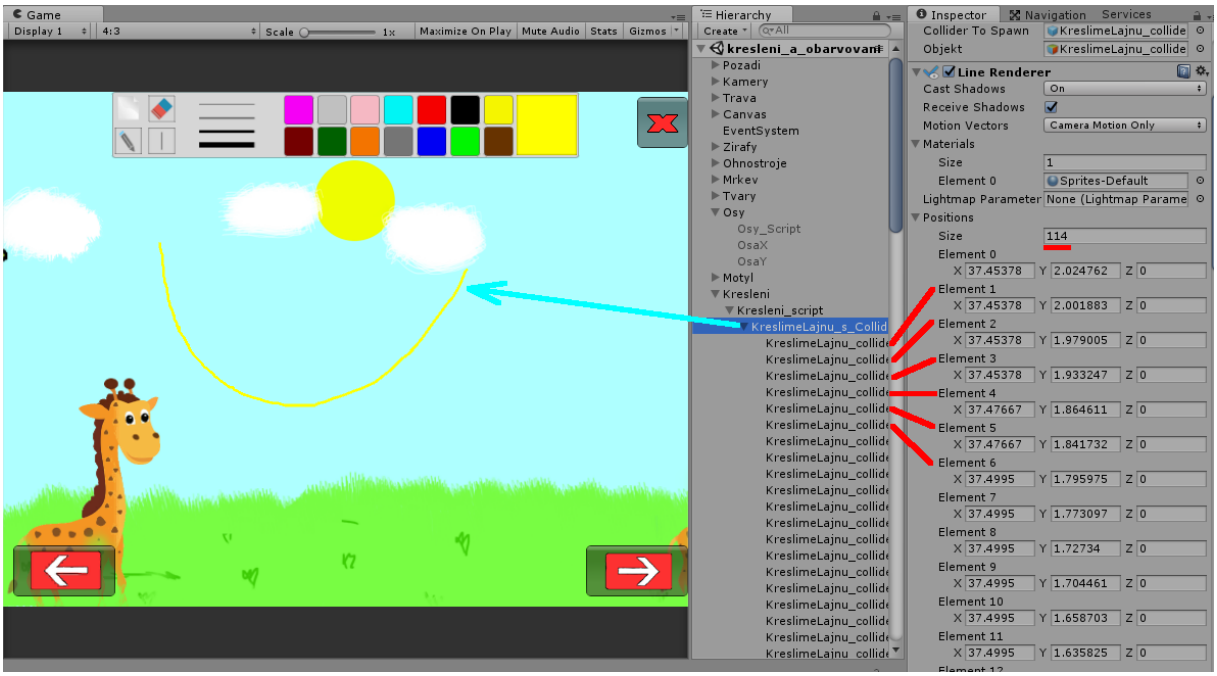
Tuto funkci je možné zavolat například ve skriptu **lajna_nova** v momentě upuštění levého tlačítka myši, funkce se zavolá jednoduše pomocí příkazu:

```
addColliderToLine();
```

Kreslená čára s colliderem

Stejně jako v předchozím případě bylo nutné přidat kreslené čáře collider kvůli nástroji Guma v úloze Libovolné kreslení. Bohužel jsem již na internetu nenalezl žádný návod na vytvoření takového collideru zdarma. Proto byla využita znalost předchozího kódu.

Celá čára s colliderem je tedy vyřešená pomocí tvorby objektu z prefabu (ten tvoří čáru), který tvoří další objekty z prefabu, kde každý jednotlivý objekt reprezentoval 2D Box Collider mezi jednotlivými body pozic komponenty LineRendereru. Každý z Prefabů vyžaduje skript. v praxi to vypadá tak, že při nakreslení jedné čáry o 100 pozicích se vytvoří jeden objekt obsahující komponentu LineRenderer a 99 objektů obsahujících komponentu BoxCollider2D, viz Obrázek 28 – Kreslená čára s colliderem.



Obrázek 28 – Kreslená čára s colliderem

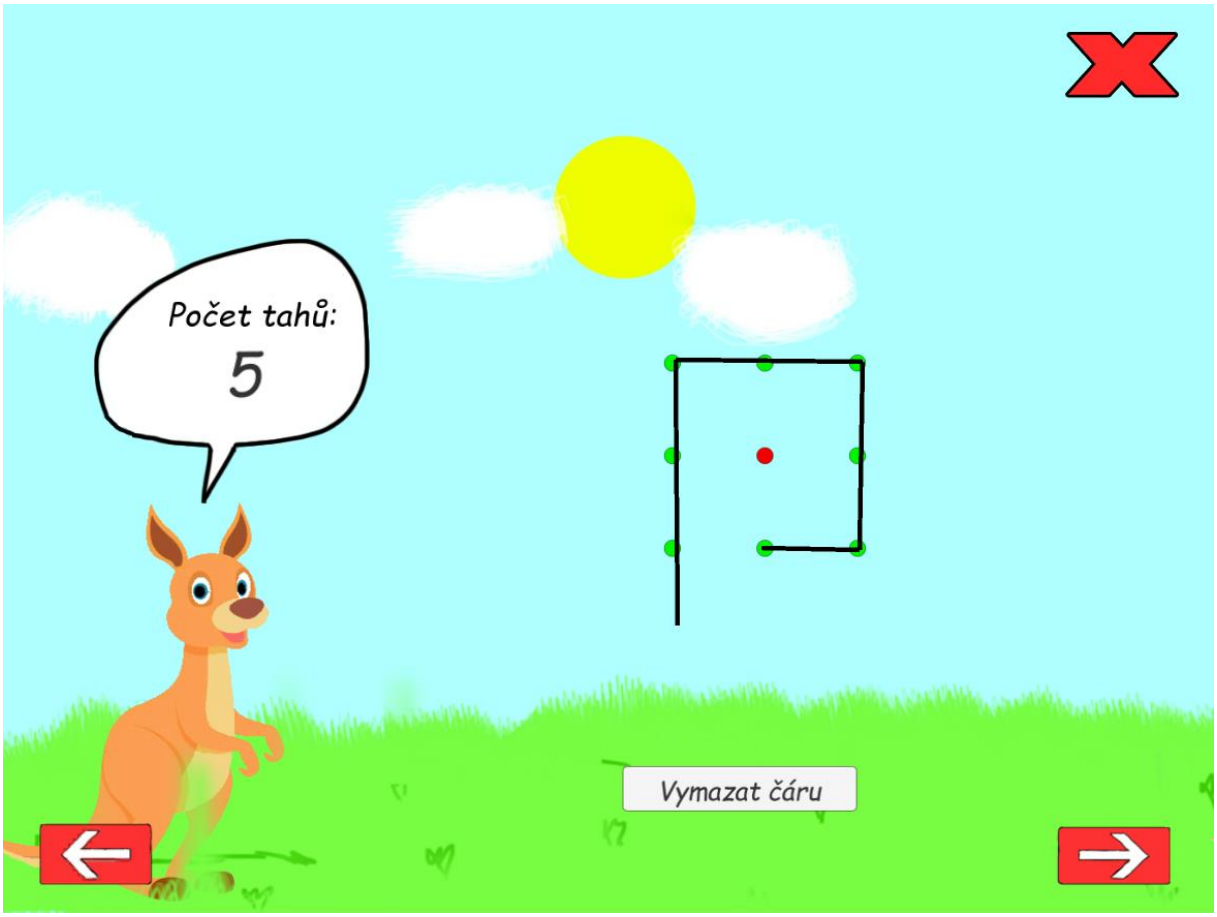
Ve skriptu úlohy je tedy nutné přidat funkci, ve které se přistupuje ke skriptu prefabu s komponentou LineRenderer a z tohoto objektu se vytvoří nový objekt, který již obsahuje komponentu BoxCollider2D. u tohoto druhého prefabu je nutné nastavit startovní a konečnou pozici collideru a následně zavolat funkci vytvoření collideru. Tím pádem se ze skriptu úlohy vytvoří objekt se skriptem, ke kterému se přistupuje, aby se vytvořil další objekt se skriptem, který můžeme upravovat. Pro ukázku:

```
if(pointsList.Count > 1){ // Pokud čára obsahuje aspoň 2 pozice
        objekt.GetComponent<KreslimeLajnu_s_colliderem>().objekt2                    =
Instantiate(objekt.GetComponent<KreslimeLajnu_s_colliderem>().colliderToSpawn); //Z objektu
čáry vytvoří nový objekt
        objekt.GetComponent<KreslimeLajnu_s_colliderem>().objekt2.transform.parent          =
objekt.transform; // nastaví objekt2 jako dítě objektu objekt
        objekt.GetComponent<KreslimeLajnu_s_colliderem>().objek2t.GetComponent<kreslime_coll
ider>().startPos = pointsList[pointsList.Count - 2]; //Nstaví se startovní pozice
        objekt.GetComponent<KreslimeLajnu_s_colliderem>().objekt2.GetComponent<kreslime_coll
ider>().endPos = pointsList[pointsList.Count -1]; //Nstaví se konečná pozice
        objekt.GetComponent<KreslimeLajnu_s_colliderem>().objekt2.GetComponent<kreslime_coll
ider>().addColliderToLine(); //Vytvoří se Collider
}
```

Z toho důvodu, že byla potřeba vymyslet toto poměrně komplikované řešení jenom kvůli jednomu nástroji v jedné úloze, byla vytvořena úloha s názvem Gravitace, aby toto řešení kreslené čáry s colliderem našlo další více viditelné využití.

4.5.6 Kreslení čáry – Protínání

Popis úlohy – tato úloha je inspirována známou úlohou [7], ve které má hráč za úkol propojit 9 bodů pomocí co nejmenšího počtu čar. Úloha je splněná v momentě, kdy se hráči podaří spojit body ve 4 tazích. Po celou dobu úloha komunikuje s hráčem pomocí bublin. Nejprve se očekává, že každý přijde na to, jak spojit úlohu v 5 tazích, v momentě, kdy to hráč takto spojí, je vybídnut ke spojení ve 4 tazích. Pokud se mu první pokus nevyvede, hráč dostane možnost si zobrazit nápovědu, pokud se mu nepodaří spojit body ani napodruhé, hráč dostane možnost zobrazit si řešení úlohy.



Obrázek 29 – Úloha Protínání s kreslením čáry tahem

Funkcionalita úlohy – v úloze je 9 bodů, které mají skript. Tento skript obsahuje pouze jednu bool proměnnou, která značí, zda je bod čarou přeškrtnutý.

Všechnu ostatní funkcionalitu zajišťuje hlavní skript, který například tvoří čáru. Skript ve skutečnosti nevytváří jednu čáru, ale několik objektů s komponentou Line Renderer, kde čára začíná na pozici, ve které předchozí čára skončila.

V úloze je několik scénářů, pokud hráč nespojil všech 9 bodů v 5 tazích, tvoří neomezený počet čar, pokud hráč spojil 9 bodů v 5 tazích, hráč může vytvořit pouze 4 tahy, v případě prvního neúspěchu se hráči zobrazí nápověda, která se schová, když hráč začne kreslit. v případě druhého neúspěchu se zobrazí obrázek s řešením. Pokud se hráči podaří spojit všechny body ve 4 tazích úloha končí.

Nejtěžší na této hře bylo vymyslet způsob ověření, zda čára protíná body. Řešení bylo vytvořeno na základě stránky [8]. Ve skriptu se hrou byla vytvořena funkce, která je zavolána vždy po dokončení čáry, v této funkci jsou matematické vzorce, které ověřují, zda bod leží na přímce:

```
public void spocitej(){
	ax = pozice[pocetLajn-1].x; //pozice bodu A, na ose X
	ay = pozice[pocetLajn-1].y; //pozice bodu A, na ose Y
	bx = pozice[pocetLajn].x;   //pozice bodu B, na ose X
	by = pozice[pocetLajn].y;   //pozice bodu B, na ose Y
	sx = bx - ax; //směrový vektor na ose X
	sy = by - ay; //směrový vektor na ose Y
	slope = (ay-by) / (ax-bx); // Úhel přímky
	YIntersect = -slope * ax + ay; // Y Inteselect je vypočítán pomocí úhlu a pozice bodu A
	for(int i = 0; i < body.Length; i++){ //for cyklus projede všech 9 bodů
		xx = body[i].transform.position.x; // pozice bodu na ose X
		xy = body[i].transform.position.y; // pozice bodu na ose Y
		lStrana = xy; // dosadí se levá strana rovnice
		pStrana = slope * xx + YIntersect; // doplní se pravá strana rovnice
		float rozdil = lStrana - pStrana; //porovnání rovnice, ideálně by měl být rozdíl 0
		//Další příkazy
	}
}
```

Tato funkce má nevýhodu v tom, že je vytvořená tak, že střed bodu musí ležet přesně na prostředku čáry. Ovšem pro uživatele je téměř nemožné nakreslit čáru, která na desetitisíciny protíná střed bodu. Proto je ve funkci nastavená „tolerance nepřesnosti“ a proměnná rozdílu nemusí být přímo 0. Ovšem tato tolerance se velmi těžko určuje z toho důvodu, že tato tolerance nemůže být pevně daná, čím menší je směrový vektor na ose X, tím větší je tento rozdíl. Proto je ve funkci spousta podmínkových příkazů, které tuto toleranci mění.

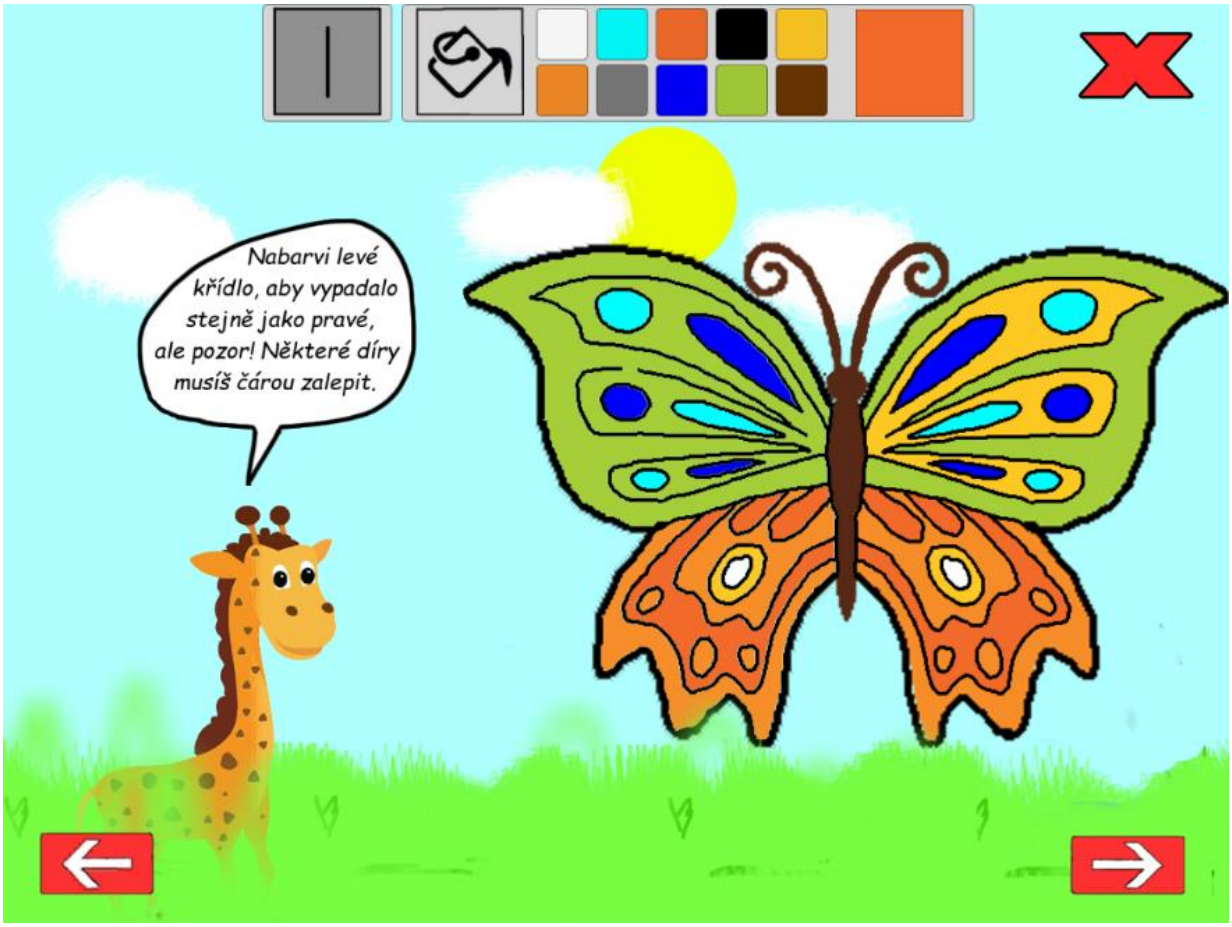
Zároveň bylo nutné ověřit, zda bod neleží pouze na přímce určené čarou, ale zda leží přímo na úsečce.

Úloha obsahuje skryté řešení, kdy je možné propojit všechny body ve 3 tazích, ovšem toto propojení je možné pouze při vyšších rozlišeních a větším poměru stran aplikace.

Účel úlohy – tato logická úloha je nejenom intelektuální výzvou, ale zároveň se hráč může naučit dalšímu typu kreslení lajny (lajna tahem).

4.5.7 Kreslení a obarvování – Motýl

Popis úlohy – v úloze je zobrazené obarvené pravé křídlo motýla a neobarvené levé křídlo, cílem hráče je pomocí nástroje Kreslit čáru dokreslit mezery v křídle a následně obarvit celé levé křídlo po vzoru pravého křídla. Pokud se nezaplní mezery, tak se mohou obarvit křídla spojená mezerou.



Obrázek 30 – Úloha Motýl

Funkcionalita úlohy – jelikož Unity neobsahuje žádné kreslící panely, bylo nutné vše vytvořit pomocí Skriptu. Celé levé křídlo se skládá z 19 různých obrázků, kde 18 obrázků jsou bílé části křídla s komponentou Polygon Collider 2D a jeden obrázek vyznačuje černou hranici mezi těmito obrázky.

Na všech 18 objektech s obrázky je skript, který má několik proměnných (objekt s hlavním skriptem, název potřebné barvy, potvrzení správné barvy, zda je zalepená díra a s jakým objektem je díra vytvořená). u objektů, které nemají žádnou díru, je zaškrtnuté, že díra je zalepená. Pokud má objekt zůstat bílý, tak se potvrdí správná barva automaticky.

Zároveň tento skript obsahuje funkci, že pokud je na objekt kliknuto a je vybrán nástroj Kbelík (kontroluje se pomocí hlavního skriptu), tak se změní barva objektu podle vybrané barvy a otestuje se správnost barvy. Pokud u tohoto objektu není zalepená díra, obarví i spojenecký objekt.

Ve stejném čase je aktivní hlavní skript celé úlohy. v tomto skriptu se rozhoduje, zda je aktivní nástroj Kreslit čáru (pomocí kterého se zalepují díry), případně je aktivní nástroj Kbelík, který obarvuje objekty vybranou barvou. Nástroj Kreslit čáru vytváří rovné čáry, a pokud hráč nakreslí čáru na požadované místo (místo díry), tak se tím zalepí díra. Jinak čára zmizí.

Účel úlohy – naučit děti funkci Plechovka, která se vyskytuje ve většině grafických editorů.

4.6 Výstup aplikace

Hra byla vytvářena a optimalizována jako spustitelná aplikace na platformě Windows a WebGL (webové rozhraní).

Jelikož je hra vytvořená personální edicí softwaru Unity, tak se při startu hry zobrazí uživateli logo Unity během načítací obrazovky, k logu bylo přidáno pozadí, které je vytvořeno z hlavního menu.

4.6.1 Windows

Při výstupu do platformy Windows je hra v portable verzi[5], vytvoří se .exe soubor, ke kterému se vytvoří složka se stejným názvem obsahující soubory ke hře. v tomto případě je při nasazování aplikace na cílové počítače nutné zkopírovat .exe soubor i složku. Celková velikost souboru a obsahu složky je přibližně 875 MB.

Kompilace programu – aby byla aplikace spustitelná pouze z jednoho souboru, bylo nutné stáhnout aplikaci Enigma Virtual Box. Následně pomocí návodu [9] byl vytvořen jediný .exe soubor, ze kterého je možné spustit aplikaci bez nutnosti jiné složky. Velikost tohoto .exe souboru je necelých 900 MB

Nevýhoda tohoto řešení je, že **není možné změnit název .exe souboru**, jelikož hra stále přistupuje ke složce, akorát tato složka je zabudovaná v souboru a není možné změnit název složky. Tedy v momentě, kdy se změní název souboru, soubor nenajde příslušnou složku a aplikace se nespustí. Další nevýhodou je nutnost opakovat kompilaci kdykoliv vyjde nová verze aplikace.

Výhoda řešení je v bezpečnosti souborů, běžný uživatel nemůže omylem smazat příslušnou složku, případně soubory v ní, čímž by se stala aplikace nespustitelná, zároveň nemá možnost k jednotlivým souborům přistupovat a zjišťovat z nich informace o hře (existují programy, které umožní zobrazit některé části programu jako jsou obrázky, skripty atd.), pokud neprolomí kódování kompilace.

Ovšem hlavní výhodou této kompilace je v jednoduchosti distribuce. Pro běžného uživatele je jednodušší kopírovat pouze jeden soubor než k souboru kopírovat celou složku, která musí mít stejný název.

[5] Portable – hra je spustitelná bez jakékoliv nutnosti instalace aplikace nebo zápisu do registrů, hra může být spuštěna například ze serveru, ovšem je rychlejší spustit aplikaci z lokálního disku.

4.6.2 WebGL

Ze hry je možné udělat výstup do platformy WebGL, která je spustitelná ze všech běžně používaných prohlížečů (mimo Internet Explorer), výstup z takové aplikace je jeden html soubor, jeden soubor obrázku (pozadí načítací obrazovky), jedna složka obsahující grafiku okolo aplikace a skript zajištující spuštění. v poslední řadě výstup obsahuje složku, která obsahuje soubory aplikace. Všechny tyto soubory jsou potřebné ke správnému spuštění. Celý tento výstup zabírá přibližně 120 MB.

Při otevření stránky s hrou se začne automaticky stahovat celý obsah hry do zařízení (tento obsah se vymaže při zavření stránky). Celý progres stahování je viděn v ukazateli načítání, který se plní podle počtu stažených dat. Při běžném připojení k internetu se hra stáhne a spustí do 30 vteřin.

Tuto velikost je možné redukovat pomocí komprese souborů, ovšem při použití komprese souborů aplikace vypadá při spuštění jako nefunkční, jelikož ukazatel načítání se nemění, jelikož jsou všechny soubory kompresovány do jednoho souboru a ukazatel načítání nerozpozná, z kolika procent je soubor stažen a tím pádem je ukazatel po celou dobu načítání na začátku.

Pokud je stránka otevřena v nepodporovaném prohlížeči, stránka předem varuje, že Unity WebGL není podporováno v prohlížeči.

4.7 Testování hry

Celá hra byla testována studenty oboru Učitelství pro 1. stupeň základních škol v rámci předmětu Didaktika informačních technologií pro 1. stupeň ZŠ. Studenti měli za úkol vyzkoušet si jednotlivé úlohy v aplikaci z pohledu žáků 3. tříd základních škol a do textového dokumentu sepsat úlohy, které je zaujali a které jim naopak připadali nedokonalé, nesrozumitelné atd. Případně co by na úlohách změnili.

Aplikaci tímto způsobem otestovalo celkem 31 studentů a vyjádřili se k většině úloh. Někteří se i vyjádřili k celkovému dojmu z aplikace, který byl většinou kladný. Spousty názorů byly navzájem protichůdné, některým se aplikace po grafické stránce líbila, některým připadala grafická stránka příliš kýčovitá atd.

Nejčastěji bylo na aplikaci chváleno:

- Celkové zpracování aplikace.
- Didaktická stránka – většina úloh je didakticky kvalitní a většina úloh je přiměřeně obtížná pro žáka 3. třídy.
- Zábavnost – většina úloh je pro žáky zábavná a mají tedy motivaci vyzkoušet si další úlohy.
- Mezipředmětové propojení – některé úlohy pracují i se znalostmi z jiných předmětů.
- Různorodost aktivit – většina úloh je unikátní.
- Různorodost sad – každá sada se zaměřuje na jiné téma a každá sada má své „zvířátko“, které hráče po sadě provází.

Nejčastěji zmiňované zápory aplikace:

- Chyby v textech – v některých bublinách byly gramatické či stylistické chyby, případně byly použity nespisovné výrazy.
- Nesjednocený design – nesjednocený font a nesjednocené obrázky. Například Klára Štáhlichová napsala: „*[…] Stejně tak se mi nelíbí nesjednocený design všech obrázků. Působí to na mě tak, že jsou obrázky (postavičky, zvířátka atd.) postahovány různě z internetu, více by se mi líbilo mít vše v jednom stylu.*“
- Neadekvátní obtížnost některých úloh – některé úlohy jsou pro žáka 3. třídy příliš obtížné.
- Neobjevující se kontrola v některých úlohách.
- Kýč – Barbora Šupková: „*Žákům se snažíme předkládat reálné věci, snažíme se je ochránit od kýče a předkládat skutečnost tak, jak ji mohou vidět. Z tohoto hlediska by měl být dodržován skutečný vzhled jednotlivých postav a také proporce…*“

Nejčastěji chválené úlohy:

- Stavění věže – úloha klade důraz na pečlivost a trpělivost dítěte. Úloha je přiměřeně obtížná a žáci se učí jemné motorice myši.
- Barvení míče – originální úloha, ve které se žáci učí, jak funguje míchání jednotlivých barev.

Nejčastěji kritizované úlohy:

- Sluneční paprsky a déšť – úloha je poměrně zmatená a neobsahuje žádnou zpětnou vazbu.
- Skořápky – úloha se soustředí více na postřeh než na klikání myší.

4.7.1 Úpravy po testování

Podle zpětných reakcích byla spousta úloh upravená, většinou text v bublinách, obtížnost některých úloh, případně prohozené pořadí úloh. Většina oprav byla podle uvedených návrhů. Dvě úlohy prošly kompletní rekonstrukcí.

Sluneční déšť a paprsky

Aplikace nyní vyhodnocuje, zda je aspoň jedna čára tažená ze slunce a jedna z mraku přeškrtnutá čarou taženou z nebe.

K úloze je nyní použit vzorec:

$$U_a = \frac{(x_4 - x_3).(Y_1 - Y_3) - (y_4 - y_3).(x_1 - x_3)}{(y_4 - y_3).(x_2 - x_1) - (x_4 - x_3).(y_2 - y_1)}$$

$$U_b = \frac{(x_2 - x_1).(Y_1 - Y_3) - (y_2 - y_1).(x_1 - x_3)}{(y_4 - y_3).(x_2 - x_1) - (x_4 - x_3).(y_2 - y_1)}$$

Kde platí, že x a y jsou vektorové souřadnice úseček.

```
x1 = x-souřadnice počátečního bodu 1. přímky
y1 = y-souřadnice počátečního bodu 1. přímky
x2 = x-souřadnice konečného bodu 1. přímky
y2 = y-souřadnice konečného bodu 1. přímky
x3 = x-souřadnice počátečního bodu 2. přímky
y3 = y-souřadnice počátečního bodu 2. přímky
x4 = x-souřadnice konečného bodu 2. přímky
y4 = y-souřadnice konečného bodu 2. přímky
```

Čáry se protínají, pokud:

$0 \leq U_a \leq 1$

$0 \leq U_b \leq 1$

Tento vzorec byl převeden do programovacího jazyka C# a kdykoliv se dokončí čára, která je tažená z oblohy, tak je zavolána funkce zjistiZdaSeProtina(), která využívá výše zmíněný vzorec a vykonává potřebné funkce.

```
public void zjistiZdaSeProtina() {
       //Nadefinují se vektory právě vytvořené čáry
       Vector2 p1 = new Vector2(novaCara.startPos.x, novaCara.startPos.y);
       Vector2 p2 = new Vector2(novaCara.endPos.x, novaCara.endPos.y);
      for (int i = 0; i < cary.Count; i++) { //For cyklus projede již vytvořené čáry
                //Nadefinují se vektory čáry, se kterou se ověřuje protínání
       Vector2 p3 = new Vector2(cary[i].startPos.x, cary[i].startPos.y);
                Vector2 p4 = new Vector2(cary[i].endPos.x, cary[i].endPos.y);
                //Nadefinuje se jmenovatel, ten je u Ua i Ub stejný
                float jmenovatel = (p4.y - p3.y) * (p2.x - p1.x) - (p4.x - p3.x) * (p2.y -
p1.y); //Nadefinuje Ua a Ub, podle vzorečku
                float Ua = ((p4.x - p3.x) * (p1.y - p3.y) - (p4.y - p3.y) * (p1.x - p3.x))
/ jmenovatel;
                float Ub = ((p2.x - p1.x) * (p1.y - p3.y) - (p2.y - p1.y) * (p1.x - p3.x))
/ jmenovatel;
                //Ověří se, zda se čáry protínají

                if (Ua >= 0 && Ua <= 1 && Ub >= 0 && Ub <= 1)
                {
                   //Zde je vypsán kód, který se provede pokud se čáry protínají
                }
        }
     }
```

Hledání písmen na klávesnici

Přidán vizuální efekt a s ním úkol pro hráče, ve kterém musí hráči zvětšit oheň a tím opéct kukuřici, hráči již nehledají písmenka do nekonečna, ale pouze do doby, než se dostatečně zvětší oheň.

Tento oheň se zvětšuje v momentě, kdy hráči zmáčknou požadované písmenko, pokud zmáčknou špatné písmeno, oheň se zmenší. Celkem je nutné zmáčknout devětkrát správné písmeno, aby se úloha dokončila. Každé špatné kliknutí navýší potřebný počet správných kliknutí o dvě kliknutí

5 Závěr

Bakalářská práce se zabývala vytvořením Interaktivní didaktické aplikace v nástroji Unity 3D pro výuku základů ovládání počítače. V teoretické části byla analyzována učebnicová předloha a již vytvořené aplikace k této učebnici. Zároveň byly popsány základní prvky enginu Unity 3D.

V praktické části byla vytvořena aplikace, ve které je možné spustit až 42 různých úloh, které jsou rozděleny do 8 tematických sad jako je Klikání myší, Psaní na klávesnici, Kreslení čáry atd. Aplikaci je možné spustit na počítačích s operačním systémem Windows nebo ve všech běžně používaných prohlížečích. K aplikaci byly vytvořeny webové stránky, ze kterých je možné aplikaci spustit nebo stáhnout. Tyto stránky jsou umístěné na školním serveru.

Celá aplikace byla otestována studenty oboru Učitelství pro 1. stupeň základních škol, kteří aplikaci většinou pozitivně ohodnotili a napsali poznámky k jednotlivým úlohám, na základě těchto poznámek byly některé úlohy upraveny či případně kompletně předělány.

Cíle práce považuji za splněné, ovšem trochu mě mrzí, že se mi nepodařilo nastavit texty v aplikaci, aby nemusely být fixně dané (všechny texty v bublinách jsou momentálně obrázky, které se aktivují/deaktivují podle potřeby, texty v sadě Doplňování a úprava textu také není možné jednoduše zaměnit). Tím pádem v aktuální verzi není možné aplikaci přeložit do více jazyků, aplikace by musela projít výraznou rekonstrukcí.

Engine Unity 3D bych mohl doporučit ke tvorbě podobných didaktických aplikací pro práci s počítačem, ale i třeba k mobilním aplikacím nebo k aplikacím pro platformy umožňující přehrát virtuální realitu.

5.1 Publikování hry

Celá aplikace je dostupná na stránkách fakulty školy http://home.pf.jcu.cz/jop/. K aplikaci byly vytvořeny webové stránky, ze kterých je možné aplikaci spustit nebo stáhnout. Zároveň je možné na webových stránkách najít některé informace o hře, případně obrázky ze hry. Nakonec jsou na stránce formuláře pomocí kterých mě mohou uživatelé e-mailem kontaktovat.

Nakladatel učebnice Informatika pro 1. stupeň základní školy, kterým je Computer Press, přidal na své stránky do souborů ke stažení odkaz na webové stránky aplikace. http://knihy.cpress.cz/informatika-pro-1-stupen-zakladni-skoly.html

6 Literatura a zdroje

[1] The True Purpose of Microsoft Solitaire, Minesweeper, and FreeCell. *Mental_floss* [online]. 2015 [cit. 2017-04-11]. Dostupné z: http://mentalfloss.com/uk/technology/32106/the-true-purpose-of-solitaire-minesweeper-hearts-and-freecell

[2] VANÍČEK, Jiří. *Informatika pro 1. stupeň základní školy: informační a komunikační technologie*. v Brně: Computer Press, 2012. ISBN 9788025137499.

[3] *Imagine* [online]. [cit. 2017-02-27]. Dostupné z: http://imagine.input.sk/popis.html

[4] Unity 3D - plaforms [online]. [cit. 2017-02-27]. Dostupné z: https://unity3d.com/unity/multiplatform

[5] *Unity 3D - Dokumentace* [online]. [cit. 2017-04-11]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[6] The AppGuruz [online]. 2017 [cit. 2017-02-24]. Dostupné z: http://www.theappguruz.com/blog/add-collider-to-line-renderer-unity

[7] Wikipedia. *Wikipedia* [online]. 2017 [cit. 2017-02-24]. Dostupné z: https://en.wikipedia.org/wiki/Thinking_outside_the_box

[8] *Stackoverflow* [online]. 2014 [cit. 2017-02-25]. Dostupné z: http://stackoverflow.com/questions/907390/how-can-i-tell-if-a-point-belongs-to-a-certain-line

[9] *Enigmaprotector Forum* [online]. 2011 [cit. 2017-02-25]. Dostupné z: http://enigmaprotector.com/forum/viewtopic.php?f=20&t=569

7 Seznam obrázků