Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Interaktivní aplikace pro rozvoj kreativity

Karel Zavřel

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Karel Zavřel

Informatika

Název práce

Interaktivní aplikace pro rozvoj kreativity

Název anglicky

An interactive application for a development of creativity

Cíle práce

Cílem rešeršní části diplomové práce je prostudování teoretických principů z oblasti 3D modelování, animace a objektově orientovaného programování v jazyce C#. Dílčím cílem je i analýza již existujících aplikací podobného charakteru.

Cílem praktické části diplomové práce je vytvoření několika desítek 3D modelů částí stavebnice použitých při tvorbě aplikace pomocí zvoleného modelovacího nástroje. Jejich následné importování do příslušného herního enginu a vytvoření aplikace pro mobilní zařízení se systémem Android. Ta bude uživateli umožňovat tvorbu důmyslnějších modelů z těchto částí s následnou možností uložení a vykreslení v obrazové formě. Aplikace bude taktéž sloužit jako podpora rozvíjení kreativity uživatelů.

Metodika

Prvním krokem bude prostudování základních technik 3D modelování, animace objektů a práce s vybraným herním enginem, ve kterém bude tvořeno samotné jádro aplikace prostřednictvím programovacího jazyka C#. Taktéž zde bude provedena analýza již existujících obdobných aplikací.

Dalším krokem bude navržení výsledné funkčnosti a vzhledu grafického uživatelského rozhraní vytvářené aplikace.

Dále budou, dle podkladů a zvolených technik modelování, vytvořeny 3D modely jednotlivých objektů představujících části stavebnice s následným importem do vybraného herního enginu, kde za pomoci programovacího jazyku C# bude naimplementována logika aplikace. Implementace bude průběžně testována, zda jsou splněny požadované funkčnosti.

Posledním krokem bude sestavení aplikace pro mobilní zařízení s operačním systémem Android a její následné otestování s reálnými uživateli při plnění několika definovaných úkolů. Dále odstranění objevených nedostatků z hotové aplikace, v reakci na provedené testování.

Doporučený rozsah práce

60 – 80 stran

Klíčová slova

Návrh aplikace, rozvoj kreativity, OS Android, 3D grafika, modelování, modelovací nástroj, herní engine, objektově orientované programování, C#

Doporučené zdroje informací

DERAKHSHANI, Dariush. Maya: průvodce 3D grafikou. 1. vyd. Praha: Grada, 2006, 428 s., 8 s. barev. obr. příl. Průvodce (Grada). ISBN 80-247-1253-9.

LAMMERS, Jim a Lee GOODING. Maya 4: kompletní průvodce. Praha: SoftPress, c2002, 542 s. ISBN 80-86497-30-5.

RIDDELL, Danny. MAYA 5 pro Windows a Macintosh: názorný průvodce. Brno: Computer Press, 2004, 412 s., 8 s. barev. obr. příl. ISBN 80-722-6956-9.

ROUDENSKÝ, Petr a Mokhtar M. KHORSHID. Programujeme hry v jazyce C#. Brno: Computer Press, 2011, 248 s. ISBN 978-80-251-3355-2.

WATKINS, Adam. Creating games with Unity and Maya: how to develop fun and marketable 3D games. Burlington, MA: Focal Press, c2011, 528 s. ISBN 978-0-240-81881-8.

ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. Moderní počítačová grafika. 2., přeprac. a rozš. vyd. Praha: Computer Press, 2004, 609 s., 16 s. barev. obr. příl. ISBN 80-251-0454-0.

Předběžný termín obhajoby

2017/18 LS – PEF

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 11. 1. 2018

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 1. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 25. 03. 2018

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Interaktivní aplikace pro rozvoj kreativity" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 29.3.2018 _____

Poděkování

Rád bych touto cestou poděkoval vedoucí své diplomové práce paní Ing. Daně Vynikarové, Ph.D. za cenné rady a připomínky, jež mi byly velmi nápomocné při zpracování této práce. Také bych rád poděkoval všem zúčastněným osobám, jež byly nápomocny při testování výsledné aplikace, za jejich vstřícný přístup a projevenou ochotu.

Interaktivní aplikace pro rozvoj kreativity

Abstrakt

Diplomová práce se zaměřuje na problematiku vývoje interaktivních aplikací a její cíl představuje návrh a implementaci aplikace, jenž by napomohla při rozvoji dětské kreativity, a také je seznámila s tím, jak lze moderní technologie využívat tvůrčím způsobem.

Teoretická část práce zahrnuje nastudování způsobů reprezentace dat v rámci počítačové grafiky a různých metod tvorby jejich obsahu s primárním zaměřením na třídimenzionální prostor. Dále také prostudování principů objektově orientovaného programování, jež je využito při implementaci výsledné aplikační logiky, spolu s analýzou vhodných softwarových nástrojů.

Praktická část práce se zabývá návrhem grafického uživatelského rozhraní, implementací aplikační logiky a kompletací veškerých komponent multimediálního obsahu do výsledné interaktivní aplikace, která je v závěru práce otestována s reálnými uživateli.

Výsledná aplikace v podobě editoru dětské stavebnice, jejíž grafická podoba je vyobrazena v příloze, umožňuje přetvořit myšlenku do virtuální podoby, a prostřednictvím poskytované funkcionality vykreslit návod pro následné uskutečnění v reálném světě.

Klíčová slova: návrh aplikace, rozvoj kreativity, OS Android, 3D grafika, modelování, modelovací nástroj, herní engine, objektově orientované programování, C#

An interactive application for a development of creativity

Abstract

The thesis is focused on an issue of development of interactive applications and its goal represents design and implementation of an interactive application which would assist with a development of children´s creativity and which also acquaints them how they can use a modern technology by a creative way.

The theoretical part of the thesis includes a study of data´s representation methods used within computer graphics and a study of various methods used to create a content of the data with a primary aiming on the three-dimensional graphics space. Furthermore also a study of principles of object-oriented programming which is used to implement resulting application´s logic and analysis of chosen software tools.

The practical part of the thesis deals with making design of graphical user interface, implementation of an application´s logic and assembly of all components of multimedia content to the resulting interactive application which is tested with real users in the last part of the thesis.

The resulting application as an editor of children´s building kit, which graphical form is shown in the annexe of the thesis, enables transfer of an idea into a virtual form and enables to draw manual of its construction by an included function. Afterwards the manual can be used for creation of the model in the real world.

Keywords: application design, development of creativity, OS Android, 3D graphics, modelling, graphics modelling tool, game engine, object-oriented programming, C#

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam použitých zkratek

CAD	Computer-Aided Design
CDR	Corel Draw
CGI	Computer-Generated Imagery
CSG	Constructive Solid Geometry
CV	Control Vertices
DOF	Degree Of Freedom
EPS, PS	PostScript
FBX	Filmbox
FPS	First-Person Shooter
GIF	Graphics Interchange Format
GIMP	GNU Image Manipulation Program
GNU license	General public license
GUI	Graphical User Interface
HD	High Definition
IDE	Integrated Development Environment
IK	Inverse Kinematics
JPEG či JPG	Joint Photographic Experts Group
MEL	Maya Embedded Language
NURBS	Non-Uniform Rational Basis Spline
OOP	Object-oriented programming
Open-source software	software s pro veřejnost volně dostupným zdrojovým kódem
PDF	Portable Document Format

PEGI	Pan European Game Information
PNG	Portable Graphics Network
PPI	Pixels Per Inch, pixely na palec
RGB	Red, Green, Blue
RPG	Role-Playing Game
SVG	Scalable Vector Graphics
TIFF	Tag Image File Format

1 Úvod

Dosavadní průběh druhé dekády 21. století se nese ve znamení úspěšného rozvoje na poli mobilních zařízení, konkrétně chytrých telefonů, tzv. smartphonů, jejichž počátky lze datovat již čtvrt století zpátky.

Od té doby se mnohé změnilo, přístroje, v závislosti na neustále se zvyšujících požadavcích obyvatelstva, prošly značným rozvojem. Už nepředstavují jen pouhá zařízení umožňující lidem vzájemnou vzdálenou komunikaci zprostředkovávanou pomocí hovorů a textových zpráv, nýbrž plnohodnotná zařízení, která se svým repertoárem poskytovaných funkčností rovnají i stolním počítačům, a která je leckdy v mnohých ohledech a díky své mobilitě i předčí.

Mobilní zařízení se prostřednictvím poskytovaných aplikací, od jednoduchých kalendářů, zápisníků a plánovačů úkolů, přes multimediální aplikace až po sofistikovanější aplikace umožňující vzdálené řízení inteligentních domů, staly součástí každodenního života většiny obyvatel, které mnohdy ovlivňují už od raného dětství. A s vizí budoucího technologického pokroku se tento jejich vliv bude nejspíše ještě zvyšovat.

Právě nejmladší generace jsou tímto ovlivňovány nejvíce, kdy do styku s moderními technologiemi přicházejí ve stále nižším věku a zpravidla pouze za účelem hraní her. Motivací pro toto téma diplomové práce je tak napomoci rozvoji dětské tvořivosti prostřednictvím aplikace, jenž by podnítila jejich zájem zkoušet přemýšlet nad novými věcmi a nebrat technologie pouze z pohledu prostředku pro bezcílné ukrácení volného času, ale také jako nástroj, jehož prostřednictvím lze utvářet nové věci a převádět myšlenky a nápady ve skutečnost.

Teoretická část diplomové práce, vycházející ze studia tištěných a internetových zdrojů, se věnuje podstatným skutečnostem ohledně počítačové grafiky a vývoje interaktivních aplikací. Zprvu jsou zde popsány způsoby reprezentace dat v počítačové grafice, některé techniky jejich tvorby v rámci třídimenzionálního virtuálního prostoru a zanalyzovány dva softwarové nástroje, jež lze k této činnosti využít. Následně se zde nachází popis vývoje samotné interaktivní aplikace s využitím objektově orientovaného programování a analýza dvou nástrojů sloužících pro jejich vývoj. Jako poslední jsou zde popsány nalezené existující řešení dané problematiky.

Praktická část práce se věnuje popisu postupu tvorby aplikace s využitím nastudovaných technik zmíněných v rámci literární rešerše. Prvotní část se zaměřuje na

návrh aplikace s ohledem na požadované funkčnosti a vzhled. Dále je popsána tvorba obsahu užitím vybraných softwarových nástrojů a jejich kompletace do výsledné interaktivní aplikace, jež je v poslední části otestována za dobrovolné účasti uživatelů reprezentujících definovanou cílovou skupinu.

2 Cíl práce a metodika

V této kapitole je definován cíl diplomové práce, rešeršní i praktické části. Jsou zde definovány i jednotlivé dílčí cíle a metodika zpracování celé práce.

2.1 Cíl práce

První část, rešeršní část práce se zabývá definováním teoretických východisek, ze kterých následně vychází zpracování druhé části, tj. praktické části diplomové práce. Hlavním cílem rešeršní části práce tak je definování těchto teoretických východisek.

S ohledem na téma zpracovávané práce lze do dílčích cílů rešeršní části zařadit prostudování teoretický principů z oblasti 3D grafického modelování objektů a vlastnosti objektově orientovaného programování v programovacím jazyce C#. Dále prostudování potenciálně vhodných 3D grafických modelovacích nástrojů a herních enginů, které lze při vývoji samotné aplikace v rámci praktické části práce využít. Posledním dílčím cílem rešeršní části je analýza již existujících aplikací podobného charakteru.

Hlavním cílem praktické části diplomové práce je naimplementovat pomocí vybraných nástrojů, grafického modelovacího nástroje a herního enginu, aplikaci běžící na mobilních zařízeních s operačním systémem Android. Aplikace samotná představuje editor dětské stavebnice, a jejím účelem je napomoci rozvoji kreativity dnešní mládeže. Mezi dílčí cíle praktické části náleží uskutečnění výběru vhodných nástrojů, jejichž vlastnosti jsou analyzovány v rámci rešeršní části práce. Navrhnout uživatelské rozhraní výsledné aplikace a aplikaci samotnou. A v poslední řadě otestovat funkční aplikaci pomocí testu s uživateli.

2.2 Metodika

Zde je definována metodika práce vedoucí k výše vytyčeným cílům diplomové práce. V rešeršní části práce je využita především metoda analýzy, za pomoci které jsou zkoumány potenciální vhodné nástroje využitelné při implementaci výsledné mobilní aplikace. Dále je využita metoda komparace, kdy jsou srovnány vlastnosti těchto nástrojů, na jejímž základě jsou následně vybrány vhodné nástroje pro samotnou tvorbu aplikace.

Prvotním úkolem je prostudování základních modelovacích technik využívaných při vytváření 3D grafických objektů, a následně způsoby jejich animování. V závislosti na tomto kroku jsou vybrány dva 3D grafické modelovací nástroje a je provedena analýza

jejich vlastností a dostupných funkcí. Dalším krokem je provedení obdobné analýzy dvou vybraných nástrojů, tzv. herních enginů. A rozebrání základních vlastností objektově orientovaného programování, které je využito při výsledné tvorbě. Posledním krokem v rámci rešeršní části diplomové práce je zanalyzování dostupných existujících řešení dané problematiky, tedy obdobných již existujících aplikací.

V praktické části práce je využito metody komparace, kdy na základě teoretických východisek definovaných v rešeršní části jsou vybrány vhodnější z nástrojů pro vytvoření 3D grafických modelů a implementaci výsledné aplikační logiky. Prvním krokem samotné tvorby aplikace je návrh grafického uživatelského rozhraní, vhodné rozmístění jednotlivých ovládacích prvků, a s tím související návrh požadované funkčnosti hotové aplikace. Následuje tvorba 3D grafických modelů využitých v aplikaci, která vychází z prostudovaných technik modelování a animování. Dalším krokem je sestavení všech komponent aplikace do výsledného celku prostřednictvím zvoleného nástroje a implementace požadované funkčnosti užitím objektově orientovaného programovacího jazyku. Veškerá implementovaná funkčnost prochází zpětným testováním správnosti. Dokončená aplikace pro mobilní zařízení s operačním systémem Android je následně otestována za dobrovolné pomoci uživatelů, kteří budou mít jasně definovaný seznam úkolů, jež musí splnit.

Posledním krokem je vyhodnocení zpětných vazeb z uživatelského testování na výslednou aplikaci, na jehož základě jsou opraveny případné nalezené nedostatky. Také jsou zde zhodnoceny dosažené výsledky.

3 Teoretická východiska

V této kapitole jsou definována všechna teoretická východiska sloužící jako podklad pro vypracování praktické části diplomové práce. Vychází se z cíle rešeršní části práce a jeho dílčích cílů, jež byly definovány v rámci kapitoly 2.1 Cíl práce.

První část kapitoly je věnována grafickým modelovacím nástrojům. Jsou zde popsány tři základní modelovací techniky 3D objektů, a následně možné způsoby, jak lze tyto objekty animovat. Poté je zde provedena analýza dvou vybraných 3D grafických modelovacích nástrojů. Vybranými nástroji jsou Maya 2016 od společnosti Autodesk a Blender 2.79 od společnosti Blender Foundation. Tyto nástroje byly zvoleny záměrně z důvodu široké komunity uživatelů, kteří je využívají, ať už ve svém profesním životě, či jako volnočasovou aktivitu. Oba nástroje disponují obsáhlou a přehledně zpracovanou dokumentací, dostupnou v obou případech online na oficiálních internetových stránkách jednotlivých společností. Tím jsou k dispozici relevantní zdroje pro čerpání informací nezbytných pro provedení analýzy vlastností a poskytovaných funkcionalit obou zvolených modelovacích nástrojů a následně výběr vhodnějšího pro samotnou tvorbu aplikace v rámci praktické části práce.

Druhá část kapitoly se zabývá počítačovým softwarem zvaným herní engine a objektově orientovaným programováním. Jsou zde popsány principy a základní techniky objektového programování. Následně je zde provedena analýza dvou zvolených herních enginů, na jejíž základě je vhodnější z nich vybrán pro tvorbu aplikace. Vybranými herními enginy jsou Unity 5.6.5 od společnosti Unity Technologies a Unreal Engine 4 vyvinutý společností Epic Games.

V poslední části kapitoly je provedena analýza existujících řešení problematiky, které se týká tato diplomová práce. Jedná se tak o rozbor tří existujících aplikací se zaměřením na 3D reprezentaci dětské stavebnice s možností editace.

3.1 Počítačová grafika a grafické modelovací nástroje

Tato kapitola se zabývá grafickými modelovacími nástroji používanými při tvorbě 3D grafických modelů různých objektů a možnými způsoby a technikami, jak tyto modely vytvářet a následně animovat.

V první řadě je důležité definovat, co to je grafický modelovací nástroj. Jedná se o softwarové vybavení umožňující vytváření tzv. computer-generated imagery (CGI),

neboli počítačem generovaných obrázků, snímků, prostřednictvím technik 3D modelování objektů, jejich animace a vykreslování výsledných scén. Tyto nástroje se řadí mezi tzv. vektorový software, jinými slovy k reprezentaci obsahu využívají vektorovou grafiku. [1, s.32]

Principům reprezentace obrazových informací v počítačové grafice je věnována následující podkapitola, která má za cíl popsat vlastnosti těchto obrazových reprezentací.

3.1.1 Způsoby reprezentace obrazových informací

V základu je způsob reprezentace obrazu v počítačové grafice rozdělen do dvou skupin, na bitmapovou či rastrovou grafiku a vektorovou grafiku, viz níže:

3.1.1.1 Rastrová grafika

První způsob reprezentace obrazu, tedy rastrová (bitmapová) grafika, využívá tzv. rastru, neboli do mřížky uspořádaných obrazových bodů, pixelů z anglického výrazu picture element, které představujícího nejmenší prvek obrazu, a které mají své pevné souřadnice vůči mřížce a nesou určitou barevnou hodnotu. [1, s. 40, 41] Mezi hlavní faktory ovlivňující výslednou velikost souborů s rastrovými obrázky patří rozměr obrázku, rozlišení a barevná hloubka obrazu. Rozlišení udává míru detailu obrázku a je definováno jako počet pixelů na jednotku délky, například na palec (PPI, pixels per inch). [2], [3] V závislosti na médiu, na kterém je výsledný obrázek zobrazován, je pak potřeba mít minimálně určité rozlišení, aby nedošlo ke ztrátě informace a kvality. Rozměr rastrových obrázků je tvořen mřížkou pixelů pevné velikosti. V případě rozměru rastrového obrázku v tzv. full HD (high definition), tedy rozlišení 1920 x 1080, je mřížka daného obrázku tvořena 2 073 600 pixely. [4] Tato reprezentace obrázku s pevnou mřížkou má ovšem nevýhodu, která spočívá ve špatné možnosti jeho škálování. Je -li obrázek zvětšován aniž by byl navýšen počet pixelů obrázku, tak se mřížka vzhledem k rozlišení stává viditelnější a lidskému oku se obrázek jeví jako čtverečkovaný, dochází ke ztrátě kvality. Požaduje -li se, aby byl obrázek kvalitně zobrazen i ve větším rozměru, je zapotřebí vyšší hodnota rozlišení při vytváření či snímání obrázku, tedy i vetší mřížka původního obrázku. S tím je ovšem spojena i nutnost větší velikosti souboru s obrázkem. [1, s. 41]

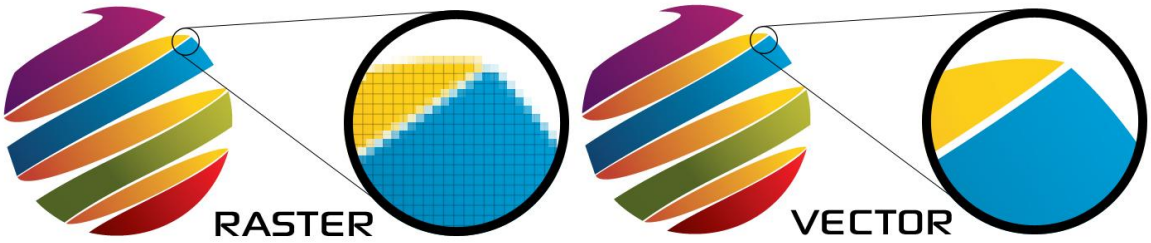
Posledním faktorem ovlivňujícím výslednou velikost obrazu je barevná hloubka, která udává počet bitů na pixel, neboli kolik bitů je použito pro popsání barvy každého pixelu. [1, s. 42] To lze provést v základě dvěma způsoby, buď přímým uvedením hodnot

jednotlivých složek zvoleného barevného modelu, například RGB, v každém pixelu obrázku, nebo s využitím barevné palety, kdy hodnoty uvedené v jednotlivých pixelech slouží čistě jen jako indexy do příslušných barevných palet jakožto převodních tabulek, které jsou součástí obrázku nebo jsou vytvářeny při jeho zobrazování. [5, s. 59-61] Stejně jako s rozlišením je problém i s barevnou hloubkou obrazu, čím vyšší je barevná hloubka, tím větší rozsah barevných odstínů může být použit pro reprezentaci obrázku, ale tím větší je výsledný soubor s obrázkem. Z tohoto důvodu se používají různé metody komprese, které mají za cíl snížit velikost souboru s obrázkem s co možná nejmenší, nejlépe žádnou, ztrátou informace. [5, s. 61]

Využití rastrové grafiky je vhodné pro obrázky s velkou škálou barevných odstínů, jako jsou například digitální fotografie nebo obrázky, kdy je snaha docílit velké realističnosti. Zde je možno prostřednictvím nastavení podobných odstínů sousedním pixelům docílit hladkých přechodů mezi barvami. [4] Běžně používanými formáty pro rastrovou grafiku jsou například GIF, PNG, TIFF, JPG. [5, s. 71-78] A mezi běžně užívané programy pro úpravu rastrových obrázků patří například Photoshop nebo GIMP.

3.1.1.2 Vektorová grafika

Druhým způsobem reprezentace grafického obrazu je vektorová grafika, která je na rozdíl od rastrové grafiky nezávislá na rozlišení. To je zapříčiněno odlišným způsobem reprezentace grafické informace, která zde není uchovávána v rámci jednotlivých pixelů napříč pevnou mřížkou, ale veškeré objekty jsou definovány prostřednictvím matematických vzorců. Vektorový obraz sestává z oblastí, tvarů a objemů, jež jsou definovány prostřednictvím geometrických vzorců a souřadnic, a které je možno pomocí různých nástrojů pro vektorovou grafiku upravovat, škálovat, přidávat jim vlastnosti jako barevný odstín. Díky této reprezentaci se změna rozměru zobrazovaného obrazu neprojeví žádným zhoršením jeho kvality, jak je graficky znázorněno na následujícím obrázku (Obrázek 1), kde levou část tvoří původně rastrový s detailním zvětšením a pravou část původně vektorový obrázek s detailním zvětšením. [1, s. 41, 42]



Obrázek 1 - Rozdíl mezi rastrovou a vektorovou grafikou
[6] a [7]

Princip je takový, že matematické rovnice jsou přeloženy do bodů, jenž jsou následně pospojovány tzv. vektorovými cestami. Může se jednat o přímky či křivky, jež tvoří veškeré objekty viditelné ve vektorové grafice. Každá vektorová cesta je určena dvojicí bodů, počátečním a koncovým bodem, které daná cesta spojuje. Cesty je možné na sebe navazovat a tvořit tak složitější obrazce, pak konečný bod jedné cesty tvoří počáteční cesty druhé. Ovšem cesta nemusí být tvořena jen těmito dvěma body, ale mohou ji ovlivňovat i další body kolem ní, jako je tomu u Beziérových křivek. [8]

Beziérova křivka je jedna z nejpoužívanější aproximačních křivek užívaných ve vektorové grafice. Hojně je ovšem užívána nejen ve 2D grafice, ale i v rámci modelování v trojrozměrném prostoru a při definici fontů písma, která je založena právě na vektorové grafice, neboť nedostatky rastrové grafiky by mohly působit rušivým dojmem při jeho škálování. Beziérovy křivky mohou být obecně až n-tého řádu, ale nejznámější jsou tzv. Beziérovy kubiky, tedy křivky zadané pomocí 4 řídících bodů. Aproximační křivky zpravidla prochází pouze počátečním a koncovým bodem, ostatní body jen ovlivňují jejich tvar prostřednictvím nastavených vah. Jedna ze základních vlastností Beziérových křivek je, že změnou jediného váhového bodu je ovlivněn celý tvar křivky, proto jsou užívány především křivky nižšího řádu, které jsou postupně navazovány za sebe. Jak již bylo popsáno dříve, koncový bod jednoho segmentu tvoří zároveň počáteční bod dalšího. [5, s. 185, 186, 190-194]

Výhody vektorové grafiky oproti rastrové nespočívají jen v možnosti změny velikosti vykreslovaného obrázku bez jakékoliv ztráty kvality, ale také v úspornější reprezentaci dat prostřednictvím matematických rovnic, a tedy i menší velikosti souboru. Také úprava vektorové grafiky je poměrně snadná, to je zajištěno prostřednictvím sad nástrojů pro práci s vektorovou grafikou. S každým prvkem obrazu lze manipulovat odděleně, samostatně, úpravou jeho kontrolních bodů. [9] Vektorová grafika však není vhodná pro reprezentaci obrazů s širokou škálou barevných odstínů jako jsou například

digitální fotografie. Vektorové obrazy se vykazují nižší rozličností barevných odstínů, v základu každý objekt ve vektorové grafice může nabývat právě jednoho barevného odstínu. To napomáhá snížení velikosti výsledného souboru, ale nižší realističnosti vykreslovaného obrazu. Pro zvýšení dojmu realističnosti je možné využít některých nástrojů pro míchání barevných odstínů, řadí se sem například funkce blend z nástroje pro zpracování vektorové grafiky zvaného Inkscape. Důsledkem je ovšem, že daný efekt je z oblasti rastrové grafiky a u výsledného obrazu se tak již nejedná o čistě vektorový obraz. [4]

Výsledné vykreslení vektorového obrazu mohou zajišťovat speciální zobrazovací zařízení, nebo běžněji užívaný postup, kdy vektorový obraz projde procesem zvaným rasterizace, neboli procesem konverze vektorových obrazů do rastrových při zachování co nejvěrnější podoby [5, s. 79], a následně může být zobrazen pomocí klasických zobrazovacích zařízení, jako jsou monitory. Jak bylo popsáno dříve, 3D grafické modelovací nástroje využívají pro zobrazovaní objektů ve scéně a jejich manipulaci právě vektorovou grafiku. S objekty je zde pracováno jako s drátěnými modely, tzv. wireframes, a do výsledného obrazu jsou převáděny po vyhotovení scény pomocí vykreslování, renderingu. [1, s. 41, 42] Rendering je zjednodušeně proces během něhož dochází k převodu 3D scény do 2D rastrového obrázku. Při tomto procesu dochází k výpočtům tvarů a pohybů objektů, vlastností jim nastaveným materiálům, osvětlení a vrhaných stínů. [1, s. 329] Daný snímek pak může být použit jako součást krátké animace či filmu, nebo jen samostatně jako například fotorealistický snímek využívaný architekty pro vizualizaci jejich návrhů.

Běžným využitím vektorové grafiky je návrh fontů písma, tvorba ilustrací a různých log společností, která je většinou potřeba zobrazovat v různých velikostech, ať už na reklamních předmětech, vizitkách nebo naopak billboardech. Dalším využitím jsou technické výkresy a mapy. Jednoduchá reprezentace objektů prostřednictvím křivek definovaných řídícími body a vektory, tedy jasná ohraničující cesta daného objektu, může být využita i například v průmyslu při strojovém vyřezávání součástek navržených právě prostřednictvím vektorové grafiky, kdy je vyřezávací stroj naváděn právě po těchto cestách. [4]

Mezi formáty vektorové grafiky patří například SVG, CDR, EPS, PS, PDF. [4] A užívanými editory pro tvorbu a úpravu grafických obrazů jsou Adobe Illustrator, Inkscape, Corel Draw.

3.1.2 Reprezentace 3D objektů

Tato kapitola se zabývá způsoby reprezentace 3D objektů v počítačové grafice. Značná část objektů v trojrozměrném prostoru má charakter tělesa reprezentujícího nějaký objekt z reálného světa. Tyto objekty mají svou hranici složenou z množiny hraničních bodů a vnitřní objem. [5, s. 237]

Reprezentaci 3D objektů lze tak rozdělit na dva způsoby, hraniční a objemovou reprezentaci. Každý ze způsobů je vhodný pro jiný účel, je tedy podstatné vědět dopředu, k jakému účelu bude vytvářený model použit a v závislosti na tom zvolit vhodnou reprezentaci. Podrobnější popis obou způsobů reprezentace je uveden níže.

3.1.2.1 Hraniční reprezentace

Hraniční reprezentace je asi častější z těchto dvou způsobů reprezentování 3D objektů a spočívá v popisu množiny hraničních bodů tělesa, neboli jeho hranice. Při tomto způsobu reprezentace není uchovávána žádná informace o vnitřku vytvářených těles, ty si tak lze představit jako duté skořápky, u kterých je důležitý pouze popis jejich obalu vzhledem k účelu použití. [5, s. 240]

V rámci hraniční reprezentace 3D grafických těles je definován tzv. manifold, což je těleso, které přesně odpovídá některému z objektů v reálném světě. U tohoto způsobu reprezentace totiž může dojít k vytvoření trojrozměrného modelu, který ovšem nejde vytvořit v reálném světě, tzv. nonmanifold. Příkladem může být těleso, jehož jedné hrany, nekonečně tenké linie, se dotýkají čtyři různé plochy. Takový objekt v reálném světě nelze vytvořit a ve skutečnosti by se zde musely nacházet minimálně dvě hrany, každá spojující dvě z těchto čtyř ploch. [5, s. 240, 241]

Základními prvky hraniční reprezentace jsou body, úsečky a části rovných ploch, také označované jako vrcholy, hrany a plošky. Tyto prvky tvořící hranici tělesa jsou zpravidla uspořádány do hierarchických struktur, ze kterých musí být patrné informace o tělese, jako jsou normály ve vrcholech sloužící pro výpočet osvětlení tělesa, ohraničení plochy, poloha bodu v prostoru, případně klasifikace hran na ostré a pomocné, které

vznikají při aproximaci zakřivených ploch prostřednictvím rovinných plošek. [5, s. 242, 243]

Nejstarší způsob hraniční reprezentace je hranová reprezentace, kdy je hranice objektu definována pouze pomocí hran a vrcholů, datová struktura tak obsahuje pouze dva seznamy, jeden pro vrcholy a jeden pro hrany. V takovém případě se jedná o tzv. wire-frame modely, neboli drátové modely, které není možno přesně interpretovat z důvodu nedostatku topologický informací. [5, s. 243]

Druhým způsobem je plošková reprezentace, která obohacuje hranovou reprezentaci o topologické údaje týkající se jednotlivých plošek tělesa. Tyto plošky mohou být tvořeny obecnými polygon, z důvodu výpočetní složitosti při zpracování je ale výhodnější využívat trojúhelníky. Přidanou topologickou informací jsou údaje, které vrcholy tvoří danou plošku. Pořadí zadání těchto vrcholů navíc určuje směr normálového vektoru plošky a tedy její viditelnost. [5, s. 244]

Nejznámějším způsobem hraniční reprezentace je tzv. winged-edge, neboli okřídlená hrana, jinak známá jako strukturovaná plošková reprezentace. Zde je každé těleso definováno pomocí tří seznamů v hierarchickém uspořádání, od seznamu vrcholů na nejnižší úrovni, přes sezname na sebe navazujících hran na střední úrovni, až po seznam ploch na nejvyšší úrovni. Jedná se o strukturu poskytující kvalitní topologická data, jelikož záznam každého prvku obsahuje ukazatele i na sousední plochy či hrany a geometrická data jsou uložena na nejnižší úrovni v seznamu vrcholů. [5, s. 244-246]

Posledním způsobem hraniční reprezentace je bodová reprezentace. Jedná se o množinu bodů nesoucích informaci o souřadnicích, barvě, normálovém vektoru, a přestavuje tak část plochy trojrozměrného tělesa. Tyto body jsou zpravidla získávány prostřednictvím digitálního snímání reálného objektu, ovšem paměťová náročnost takto vytvořeného modelu je příliš velká. [5, s. 246]

Hraniční reprezentace se využívá především při tvorbě 3D modelů pro účely herního a filmového průmysl, neboť její paměťová a výpočetní náročnost není tak velká a snadno se tyto modely vykreslují, to ovšem neplatí pro bodově reprezentované modely.

3.1.2.2 Objemová reprezentace

Druhým způsobem reprezentace těles v trojrozměrném prostoru je objemová reprezentace. Tato reprezentace uchovává informace o vnitřku daných těles nikoliv jeho

povrchu, jako je tomu u hraniční reprezentace. To má za důsledek, že lze snadno provádět detekci, zda se bod nachází uvnitř tělesa. Vzhledem k tomu je výhodné objemovou reprezentaci využívat jako pomocnou datovou strukturu, která například rozděluje prostor scény a usnadňuje detekci kolizí mezi objekty v této scéně, tedy rychlé vyhledávání. Takovou strukturou je například Octree, která je podrobněji popsána dále v textu. [10]

Základními prvky objemové reprezentace jsou voxel a buňka. Voxely představují nejmenší elementy trojrozměrného prostoru tvaru kvádru či krychle a jsou uspořádány do pravoúhlé mřížky. V celém svém objemu mají konstantní hodnotu a osm voxelů je uspořádáno do buňky, jejíž vnitřní hodnoty jsou získávány užitím interpolace prvního, či druhého řádu. To přináší méně hrubé výsledky, než interpolace na úrovni jednotlivých voxelů, kdy se zjišťují hodnoty mezi jejich středy, jakožto vzorky. [5, s. 255-257]

V objemové reprezentaci dochází k podobnému problému jako u rastrových obrázků z dvourozměrného prostoru, kdy je potřeba uchovávat velké množství dat týkajících se každého pixelu daného obrázku, i zde je nutno uchovat značný objem informací o jednotlivých voxelech umístěných v pravoúhlé mřížce. Objemová reprezentace je tak poměrně paměťově náročná, přestože je každý voxel reprezentován jako jediný bod této mřížky a hodnoty mezi nimi jsou získávány prostřednictvím interpolace hodnot ve středu sousedních voxelů.

Uplatnění nachází objemová reprezentace trojrozměrných těles především v oblasti vizualizace dat ve vědecké oblasti a v oblasti medicíny, například při simulaci kapalin nebo zobrazování dat získaných ze zařízení jako magnetická rezonance. Také je ovšem využitelná v oblasti herního průmyslu při vytváření terénu, neboli okolního prostředí v dané hře. Objemová reprezentace je zde užita s cílem vytvořit taková prostředí, která není možno definovat pomocí tzv. výškových map. Jedná se o prostředí s konkávními prvky, jako jsou například jeskyně a skalní převisy, kdy významnou roli hraje i objem pod těmito prvky. Využitelná je pro simulace případné deformace terénu po detekci kolize s objektem. [11]

V této kapitole dříve zmiňovaná reprezentace objektu prostřednictvím hodnot voxelů se nazývá buněčný model a spadá do výčtové reprezentace. Při tomto způsobu dochází k přesnému určení, vyčíslení obsazeného prostoru a zobrazování je prováděno od zadních voxelů po přední, vede to tedy k několikanásobnému překreslování, jež zvyšuje

náročnost. Výčtová reprezentace je ovšem vhodná pro využití jako pomocné struktury pro vyhledávání v podobě octree, neboli oktalového stromu. [10]

Datová struktura octree pracuje na principu dělení ohraničeného prostoru na menší podprostory, a to až na úroveň jednotlivých voxelů. Jedná se o stromovou strukturu, v níž jsou jednotlivé uzly reprezentovány jako podprostory nadřazeného uzlu, tedy prostoru. Zpravidla mají tvar krychle pro její snadnou zpracovatelnost a paměťovou nenáročnost, ale lze provést i implementaci s užitím kvádrů. [12]

Počáteční uzel stromu, neboli kořen stromu, definuje a ohraničuje celý prostor, který bude dále dělen na menší podprostory, jak budou data o modelech postupně přibývat. V případě, že se v tomto uzlu objeví nový objekt, je tento uzel rozdělen ve všech třech osách trojrozměrného prostoru na poloviny, a tím je definováno osm pravidelných potomků tvořících stejně velké podprostory daného prostoru. Toto dělení probíhá až na nejnižší definovanou úroveň. Jedná se tak o reprezentaci stromu, kde každý uzel má osm potomků, z tohoto důvodu oktalový strom. Samotné dělení může probíhat až v době potřeby, kdy jsou do daného prostoru dodána nová data objektu, tím vzniká nevyvážený strom s možností reprezentace různých úrovní detailu. [12]

Primárním užitím této datové struktury v herním průmyslu je usnadnění a urychlení detekce kolize různých objektů napříč scénou. Základní myšlenkou je neprocházet úplně celý prostor a nekontrolovat kolize jednoho objektu s každým jiným, ale prohledávat pouze příslušné větve stromu, kde by ke kolizi mohlo dojít. [13]

Druhou možností je nevyužívat octree jako datovou strukturu pro vyhledávání, ale jako strukturu pro reprezentaci tělesa, kdy se jedná o hierarchickou reprezentaci buněčného modelu, která je automaticky zjemňována dle tvaru reprezentovaného tělesa. Zde může nabývat každý z osmi potomků tří hodnoty, kdy je buď prázdný, plně obsazený, nebo částečně obsazený, jež je dále dělen na menší podprostory. Takto reprezentovaný objekt je vhodný pro provádění booleovských operací jako průnik, sjednocení a rozdíl. [11]

Posledním zmíněným způsobem objemové reprezentace je konstruktivní geometrie těles, neboli Constructive Solid Geometry (CSG). Jedná se o způsob reprezentace užívaný převážně v systémech počítačem podporovaného projektování, neboli computer-aided design (CAD). Modely jsou zde skládány prostřednictvím množinových operací aplikovaných na tzv. CSG primitiva, mezi která se řadí například kvádr, koule, jehlan, válec, toroid a kužel. [20, s. 246, 247]

Množinové operace zahrnují operaci sjednocení, průniku a rozdíl, kdy celý proces jejich aplikací na daná primitiva, s cílem vytvořit složitější objekt, nápadně připomíná skutečný konstruktérský postup, ve kterém se používá vrtání, sváření či odříznutí nechtěných částí. Celý postup konstrukce výsledného složitějšího tělesa z CSG primitiv je popsán prostřednictvím tzv. CSG stromu. Kořen tohoto stromu je tvořen již sestaveným výsledným tělesem, listy obsahují pouze CSG primitiva, vnitřní uzly reprezentují aplikované množinové operace a hrany mezi jednotlivými uzly reprezentují geometrické transformace. Nevýhodou tohoto způsobu reprezentace je jeho neurčitost, kdy například stejné těleso lze zkonstruovat různými způsoby, různým pořadím aplikace množinových hodnot. Taktéž není vhodný pro zobrazovaní, kdy je vhodnější převedení do hraniční reprezentace. [5, s. 246-248]

3.1.3 Techniky modelování objektů

V současnosti existuje několik možných způsobů, jak lze trojrozměrný počítačový model pořídit. Tato kapitola popisuje některé z nich, největší důraz klade na tři základní způsoby tzv. animačního modelování, které je poskytováno téměř všemi modelovacími nástroji. [14]

První možná metoda je získání objektu užitím algoritmu, který ho generuje, tzv. procedurální modelování. To je dále dělitelné dle různých kriterií na skupiny metod vhodných pro určité typy získávaných modelů. V základu se jedná o L-systémy, což jsou algoritmy sloužící ke generování rostlin. Fraktální geometrie využívající se pro generování okolní krajiny, hor, kamenů. Poslední skupinou jsou systémy částic sloužící k simulaci ohně či dýmu a generování hejn ptáků a explozí. [5, s. 265, 266]

Modelování založené na obrazech, image based modeling, je druhou možnou metodou, která umožňuje získání trojrozměrného modelu. Tato metoda využívá rekonstrukce modelu z poskytnutých snímků z fotoaparátu, načež má značný vliv rozloha a členitost snímaného objektu. Nebo lze využít 3D scanneru, kde ovšem tato omezení taktéž přetrvávají. [5, s. 265, 266] Výsledný model většinou vyžaduje ruční provedení finálních dokončovacích prací animátorem, jako je dodefinování míst, která nebylo možno přesně naskenovat, či zacelení vzniklých děr. Jedná se však o metodu zajišťující co nejrealističtější převedení objektu na počítačový model. [14]

Další možnou metodou užívanou pro získání trojrozměrného modelu je tzv. digital sculpting, neboli digitální tvarování či sochařství. Tato metoda představuje jakousi analogii k procesu, jako by byl objekt utvářen sochařem například z jílu. Využívá se zde nástrojů pro tažení a tlačení, kdy se z jednoduchých objektů takto získávají složitější finální modely. Metoda může být použita jak při objemové reprezentaci, tak i při hraniční, kdy je pracováno se sítí polygonů místo voxelů. Hraniční reprezentace představuje výhodu, že lze s modelem pracovat na několika různých úrovní detailu, kdy se v závislosti na tom některé části polygonové sítě zjemňují větším počtem polygonů. Změny na jedné úrovni detailu se přepisují i do ostatních úrovní. Metoda má hlavní přínos především v usnadnění práce při modelování organických objektů, kdy by bylo značně obtížné zobrazit všechny detaily ruční úpravo polygonové sítě. [14]

V další části kapitoly jsou popsány tři metody vytváření trojrozměrných počítačových modelů, které jsou využívány při ručním modelováním. Patří sem modelování pomocí tzv. NURBS, polygonální modelování a modelování prostřednictvím dělení povrchů.

3.1.3.1 Modelování pomocí NURBS

Při této metodě je využíváno matematického aparátu NURBS (non-uniform rational basis spline) ke generování povrchů a křivek, nepravidelných racionálních B-křivek, které definují finální tvar objektu. Výsledkem je hladší zaoblená geometrie vytvářených těles, než může poskytnout polygonální modelování. [1, s. 114, 115]

Geometrie těles je zde definována prostřednictvím křivek, například Beziérových křivek, které byly blíže popsány v rámci kapitoly 3.1.1 Způsoby reprezentace obrazových informací v části o vektorové grafice. Tvar těchto křivek definují řídící vrcholy, tzv. control vertices (CV), které křivka aproximuje. Pouze dva z těchto vrcholů leží přímo na křivce a určují její počátek a konec, zatímco ostatní řídící body ovlivňují její tvar. NURBS povrchy jsou naopak definovány křivkami zvanými izoparmy, jejichž tvar taktéž definují příslušné řídící vrcholy, a tyto křivky určují segmenty a sekce sledující povrchové zakřivení tělesa. Segmenty jsou definovány horizontálními izoparmami a sekce vertikálními. Čím více segmentů model obsahuje, tím náročnějším se stává pro zpracování, ale tím vyšší úroveň detailu může představovat. [1, s. 114, 115]

Tvorba modelu touto metodou většinou započíná úpravou definovaných NURBS primitiv, nebo ručním vytvářením křivek, na než jsou následně aplikovány různé operace užitím funkcí poskytovaných daným modelovacím nástrojem. Mezi tyto funkce, kdy jsou názvy brány z nástroje Maya a je tedy možné, že v některých jiných nástrojích se nepatrně liší, patří například Revolve, která vytváří točený povrch tím, že označenou křivku nechá rotovat kolem rotačního bodu, je tedy vhodná pro osově symetrické objekty jako kulaté nohy nábytku, kuželky, sklenice a láhve. Dalším často užívanou funkcí je vyztužení, Loft, která mezi dvěma a více vybranými křivkami vytvoří povrch, jehož zakřivení je definováno každou touto vybranou křivkou. Poměrně užitečnou funkcí je i vytlačení, Extrude, kdy jsou definovány dvě křivky, jedna udávající profil objektu a druhá definující směr vytlačení. Tato funkce je vhodná pro tvorbu pružin. Většina modelovacích nástrojů obsahuje větší množství funkcí, jako ještě například zkosení, rovinné povrchy, omezené povrchy a jejich kombinace umožňující animátorovi přesně definovat všechny rozličné tvary potřebné pro sestavení finálního tělesa. To tak může sestávat z více záplat spojených, sešitých do jediného povrchu, nebo mohlo vzniknout povrchovými úpravami NURBS primitiva. [1, s. 117-127]

Nevýhodou této metody je náročnější výpočetní složitost, neboť křivky jsou reprezentovány matematickými vzorci. Metoda je tedy využívána při tvorbě, kdy není potřeba vykreslování v reálném čase, ale probíhá formou předzpracování. Jejím prostřednictvím vznikají především modely organických objektů. Také je využívána v automobilovém průmyslu pro reprezentaci a vizualizaci velmi detailně propracovaných návrhů nových vozidel, nebo při tvorbě osově symetrický objektů. [1, s. 114, 115, 119, 120]

3.1.3.2 Polygonální modelování

Další značně využívanou metodu při tvorbě trojrozměrných počítačových modelů představuje polygonální modelování. Tělesa jsou zde reprezentována prostřednictvím hraniční reprezentace, kdy tuto hranici tvoří tzv. mesh, síť polygonů. Tato síť sestává z vrcholů, vertices, určujících její geometrické souřadnice, vrcholy jsou spojeny hranami, edges, a ohraničují příslušné plošky, faces, jednotlivých polygonů. Zpravidla jsou využívány trojúhelníky nebo čtyřúhelníky pro jejich snadnost zpracování. [1, s. 113]

Polygonální modelování se využívá nejčastěji pro tvorbu modelů, jež tvoří část multimediálního obsahu aplikací, počítačových her, a to především z důvodu snadnosti vykreslení polygonálních modelů. V počítačových hrách je nutné vykreslování v reálném čase, důraz se tedy klade na co možná nejjednodušší síť polygonů, jež tvoří finální model. Takto vytvořené modely, které sestávají i z jediného kusu povrchu, jsou výhodné i pro animační tvorbu, kdy při jejich deformaci nedochází ke vzdalování švů, jako tomu může být u modelů vzniklých z většího množství NURBS povrchů. [1, s. 113, 114]

Samotná tvorba modelu prostřednictvím této metody může být založena na třech způsobech. Jedním je vytvoření finálního modelu z poskytnutých geometrických primitiv, úpravou jejich geometrie, vytlačováním nových stěn. Druhým způsobem je ruční kreslení polygonu prostřednictvím poskytovaného nástroje, kdy si animátor může ručně vykreslit polygony prostřednictvím nadefinování jejich vrcholů. Posledním způsobem je tvoření polygonové sítě za použití NURBS křivek, kdy se definuje pomocí nástroje počet polygonů dle požadované úrovně detailu. [1, s. 147-151]

Nevýhodou modelů vzniklých užitím polygonálního modelování je jejich nedokonalé zakřivení, neboť oblé tvary jsou pouze aproximovány rovinnými polygony. To lze omezit definováním vyšší úrovně detailu, tedy hustší polygonové sítě, ručně nebo použitím nástroje smooth, ovšem při větším přiblížení bude hranatost opět patrná. Navíc rostoucí počet polygonů tvořících síť modelu způsobuje i růst složitost jeho zobrazení, je tedy nutné zvážit jaká míra detailu je pro výslednou scénu skutečně zapotřebí. [1, s. 112, 113]

3.1.3.3 Metoda dělení povrchů

Metoda dělení povrchů představuje třetí z velmi používaných metod při vytváření trojrozměrných modelů. Jedná se o metodu, která spojuje vlastnosti obou předchozích metod. [1, s. 116]

Animátor s tvorbou nového modelu začíná na jednodušších polygonových modelech, které následně převádí do dělení povrchů. Zde se mu dostává možnosti modelování tělesa na několika úrovních detailu s možností opětovného vrácení se k prvotní úrovni, a provedení tak rychlé změny, která se aplikuje napříč všemi úrovněmi až po tu nejdetailnější. Lze zde také využít definování NURBS, které slouží pro vyhlazení polygonové geometrie. [1, s. 186-191]

Nevýhodou u této metody modelování je její výpočetní náročnost, která zpravidla přesahuje náročnost modelů vytvářených pomocí NURBS. Zpravidla se tedy hotové modely převádějí zpět do polygonální reprezentace, pro jejich rychlejší zpracování. [1, s. 116]

3.1.4 Techniky animování modelů

Po skončení fáze modelování všech požadovaných objektů spolu s přiřazením příslušných materiálů a textur, které objektům mohou dodat další úroveň detailu, přichází na řadu zpravidla animace. Animace vnáší do výsledné scény dynamiku. V zásadě se může jednat o dynamické snímání celé scény, nebo jen přípravu jednoho modelu pro vývoj hry, kterému je vytvořena sada animací reprezentujících jeho jednotlivé pohyby, od chůze, přes skok, až po jednoduché pohupování se na místě. Stejně jako u procesu modelování trojrozměrných objektů, i v animaci existuje několik rozdílných přístupů využívaných pro specifické úkoly. [5, s. 483]

Animaci lze v základu rozdělit do dvou kategorií, na nízkoúrovňovou, která se týká animace objektu po spojité dráze, kam patří metody jako klíčování a animační křivky, a na vysokoúrovňovou pracující i s detekcí kolize, kam se řadí přímá a inverzní kinematika, dynamika. [5, s. 483-487]

3.1.4.1 Klíčování

Metoda animování zvaná klíčování neboli keyframing je jedna z nejčastěji používaných metod při tvorbě animací objektů. Poprvé byla použita v dílnách Walta Disneye, kdy hlavní animátor zakreslil klíčové snímky animovaného filmu a všechny mezisnímky byly dokresleny rukama pomocných animátorů. Od té doby se začalo pracovat na odstranění ruční malby mezisnímků, která pouze prodlužovala celý proces. [5, s. 484]

Metoda se začala aplikovat i v rámci trojrozměrných animací, kdy animátor nadefinuje tzv. klíčové snímky scény a algoritmus dopočítá zbytek prostřednictvím interpolace hodnot z klíčových snímků. V rámci definování klíčových snímků je možné zachytit nejen geometrické změny objektů a mezi objekty, ale také i úpravu atributů jakožto barvy těles, textury, průhlednost. Jedná se tedy o metodu, kdy je zadána počáteční a koncová pozice a vše mezitím je dopočteno počítačem. [5, s. 484]

Právě možnost využití algoritmu pro dopočítání změn hodnot u objektů, jež jsou animovány, je velmi důležitá. Animaci lze totiž definovat jako rychle promítanou sekvenci

statických obrázků, která pro lidské oko budí dojem plynulého pohybu. Tato frekvence však musí být alespoň 30 a více snímků za vteřinu, z toho vyplývá, že ruční kreslení či nastavování hodnot atributů by bylo nejen velmi zdlouhavé, ale i velmi náročné vzhledem k tomu, o jak nepatrné změny se mezi sousedními snímky jedná. V rámci většiny grafických modelovacích nástrojů mohou být vytvořené klíčové snímky manipulovány. Mohou být vytvářeny jejich kopie, či přesunuty na pozdější či dřívější snímky, což má za následek zpomalení či zrychlení průběhu animace. [15]

K prostému klíčování může být přidána animační křivka sloužící pro reprezentaci pohybu tělesa po dráze. Při takto vytvářené animaci se opět určují klíčové snímky, počáteční a koncový, zbylé jsou dopočteny. Nejprve je důležité definovat samotnou dráhu, tedy tvar křivky, změny rychlosti a orientaci objektu. V jednotlivých snímcích je poloha tělesa určena interpolací poloh z klíčových snímků. Orientaci tělesa definuje jeho lokální souřadnicový systém, jak se průběžně natáčí při pohybu po dané křivce a rychlost tělesa je ovlivněna parametrizací křivky, na jejíž základě je měněna hodnota tečného vektoru. [5, s. 484-487]

3.1.4.2 Přímá kinematika

Metoda přímé kinematiky se řadí mezi tzv. vysokoúrovňové metody počítačové animace. Jako první je potřeba definovat pojem kinematika, jedná se o část fyziky, která se zabývá studiem pohybu nezávisle na silách, jež ho způsobují. Vzájemným působením těchto sil a objektů se zabývá dynamika. [5, s. 487]

V rámci kinematiky je pozornost věnována především animaci pohybů lidského těla, organických modelů. Zde je definován termín segmentová struktura, která reprezentuje strukturu složenou z posloupnosti pevných částí, jež se ve svých spojeních mohou otáčet. Tato posloupnost by měla být na jednom konci pevně upevněna a druhý mít volný, zvaný koncový efektor. Každou takovouto strukturu charakterizují stupně volnosti, neboli Degree of freedom (DOF), což jsou hodnoty reprezentující informace o každém segmentu dané soustavy. Patří sem tři hodnoty určující pozici segmentu v prostoru a tři hodnoty definující jeho natočení. [5, s. 487-489] Složitější struktura může reprezentovat celý skelet humanoidního modelu. Jedná se tak o reprezentaci kostry, jejímž animováním, zpravidla formou otáčení, ovlivňujeme přidruženou geometrii tělesa. Kostra tedy sestává z několika segmentů, kostí, jež jsou spojeny klouby, a umožňují tak pohyb jednotlivých

jejích částí, při čemž platí hierarchické uspořádání zajišťující, že daná kost se otáčí kolem rodičovských rotačních bodů. Ovšem pohyb humanoidního modelu není jediným vhodným využitím kostry, také ji lze využít pro reprezentaci pohybu pláště či stromu větru. [1, s. 268, 269]

Metodu přímé kinematiky si lze představit jako ručně prováděnou úpravu loutky, kdy animátor postupuje od rodičovského segmentu k potomkovi. Jinými slovy, je -li například cílem položit nějaký objekt na stůl, je zapotřebí rozhýbat segmenty ruky. Nejprve se provede pohyb v kloubu reprezentující rameno paže, následně v lokti a až nakonec pohyb zápěstím, případně prsty. Podřízené segmenty se tak otáčejí v jejich vrchních kloubech. [1, s. 269]

Nevýhodu tohoto přístupu představuje nutnost zdlouhavého animování, kdy animátor musí postupně nastavovat kosti do požadovaných pozic napříč různými časovými rámci, z tohoto důvodu je častěji využívána inverzní kinematika, jenž tyto nedostatky eliminuje. [1, s. 269]

3.1.4.3 Inverzní kinematika

Zatímco u přímé kinematiky animátor nastavuje jednotlivé kosti neboli segmenty kostry, a tím získává výsledný pohyb, tak u inverzní kinematiky je tomu naopak. Animátor definuje koncovou pozici, tedy daný pohyb a mechanismus inverzní kinematiky dopočítá pozice jednotlivých segmentů soustavy, jedná se tedy o obrácený přístup, kdy se postupuje od listů segmentové struktury. [16]

Inverzní kinematika využívá tzv. manipulátorů inverzní kinematiky, IK handles. Tyto manipulátory propojují základnu, konec daného segmentového systému s kořenovým kloubem pro daný segment. V rámci této metody animování ovlivňuje kosti a klouby pouze pohyb jim přidruženého manipulátoru. V závislosti na jeho pohybu definuje prvek zvaný řešitel, IK solver, natočení všech ostatních kloubů obsažených v dané segmentové soustavě. Při využití výše modelovaného příkladu položení objektu na stůl je v rámci inverzní kinematiky postup následující. Animátor přesune přidružený manipulátor do finální pozice na stole, přičemž řešitel dopočítá natočení zbylých kloubů, zápěstí, loket, nakonec rameno. [1, s. 270]

Při této metodě animování mohou nastávat situace, kdy je nejednoznačný způsob nastavení zbylých kloubů, který řeší IK řešitel. Jinými slovy, může existovat více druhů

nastavení pro dosažení konečného stavu. Zabránění neurčitosti spočívá v nastavení různých omezení, mezi které může patřit například nastavení možnosti ohybu jednotlivých kloubů jen v určitých směrech a úhlech. [16] Případně při vytváření manipulátoru trochu natočit kloub tím směrem, kterým se má při následné manipulaci ohýbat, a poté definovat počáteční a koncový bod IK řetězce. Tímto způsobem je zameteno případnému nastavení manipulátoru, aby se například koleno humanoidního modelu prohýbalo opačným směrem. [1, s. 292, 293]

Při využití obou těchto zmiňovaných technik pro animování polygonálního modelu, jehož síť je připojena k dané kostře, může docházet k deformaci geometrie při ohýbání kloubů. Toto je řešeno v závislosti na použité metodě ohybu. Pokud se jedná o pevný ohyb, tak je deformaci v podobě vzájemného překrývání jednotlivých polygonů sítě zabráněno užitím flexoru, neboli mřížky sloužící pro vyhlazení geometrie v místě ohybu. Je -li užit hladký ohyb, může docházet k deformaci v podobě zužování polygonové sítě v místě ohybu. Tento problém je naopak řešen nastavením plášťových vah, přesněji definováním, na jaký úsek polygonové sítě má každý kloub vliv při své manipulaci, a také jakou vahou ji ovlivňuji. [1, s. 286-291]

3.1.4.4 Snímaní pohybu

Snímání pohybu neboli motion capture představuje metodu pro získávání dat o pohybu snímaného objektu, která jsou následně převedena na trojrozměrný počítačový model. Motion capture tak představuje jeden z nejrychlejších způsobů získání co nejvěrohodnějších animací daného tělesa. Cílem je tedy získat co nejvíce dat popisujících pozici a natočení jednotlivých částí snímaného objektu v prostoru. Především kořenového objektu a koncových efektorů, kdy je případně prostřednictvím inverzní kinematiky dopočítán zbytek údajů, pro něž není dostatek nasbíraných dat, způsobený například zakrytím senzoru. [17]

Při snímání pohybu je možné data zpracovávat buď v reálném čase, nebo v rámci post produkce a snímanými objekty mohou být organické objekty, jako jsou člověk a zvířata, nebo i anorganické, pevné předměty. V současnosti je tato technologie využívaná především pro zábavní průmysl, tedy pro vývoj her a filmů. Snímáním pohybů skutečného herce získávají data pro trojrozměrné modely postav, kterým tak definují sadu animací reprezentující jejich jednotlivé druhy pohybu zahrnující například chůzi, běh, skákání,

sezení a jiné. Také je možno snímat pohyb obličejových svalů herce. Mimika a pohyby celého těla jsou však zpravidla snímány odděleně, prostřednictvím detailnějšího pohledu. [17]

V současnosti existuje několik technologií, jež mohou být pro snímání pohybu užity. Jedná se například o mechanický, akustický, magnetický, inerciální či optický způsob snímání. Každá z těchto technologií má svá omezení, například uzavřenost prostoru, v němž se snímaný herec může pohybovat, nebo rušivý vliv elektromagnetických polí vyzařovaných technikou v okolí. Jednou z nejvyužívanějších metod je metoda optického snímání, kdy je snímaný herec nebo objekt opatřen značkami, které jsou po jeho těle rozmístěny dle poloh definovaných orientačních bodů na kostře modelu. Systém pracuje na bázi infračerveného světla a poloho objektu v prostoru určuje z více kamer, které daný objekt v reálném světě snímají. [17]

3.1.5 Maya 2016

Maya představuje modelovací, animační a vykreslovací počítačový software poprvé vydaný v roce 1998 společností Alias System Corporation. V současné době je vyvíjený firmou Autodesk, přesněji od roku 2005, a je využívaný pro tvorbu ve filmovém a herním průmyslu. Za dobu svého vývoje prošel značnými změnami, kdy z původního animačního softwaru byl vyvinut mocný víceúčelový nástroj. Společnost Autodesk se prostřednictvím velké komunity profesionálních uživatelů zařadila do čela světového žebříčku v oblastech 3D návrhů, zábavy a médií (herní a filmový průmysl), a softwaru pro architekturu, plánování staveb a konstrukcí. Nástroj má podporu na platformách Windows a OS X a nové verze jsou vydávány prakticky každoročně. [18]

Jedná se o software poskytující řadu nástrojů napříč různými oblastmi využití. Pro trojrozměrné modelování se zde nachází nástroje pro polygonální modelování, modelování pomocí křivek, symetrické modelování, sculpting a další. Pro tvorbu animací jsou zde nástroje pro základní druhy animací, jako je klíčování, animace pomocí skriptu, inverzní a dopředná kinematika, geodetické provázání voxelů, využívané pro lepší spojení kostry se sítí polygonů, jež se mohou překrývat a v neposlední řadě časový editor pro efektivnější úpravu průběhu animací. V oblasti dynamiky je nový nástroj pro realističtější a detailnější simulace tekutin zvaný Bifrost, nástroj pro tvorbu materiálů reprezentujících oblečení a lepší tvorba vlasů a srsti, nástroje pro dynamiku tuhých a měkkých těles, fyziku střel.

V oblasti vykreslování jsou poskytnuty nástroje pro lepší stínování ve scéně, možnost vytváření šablon snímků pro rychlejší a efektivnější možnost znovupoužití a další nástroje. V neposlední řadě je zde možnost skriptování prostřednictvím jazyka Python nebo Maya Embedded Language (MEL), který byl vytvořen dle Unixového shellu. [19]

Veškeré způsoby animování modelů, které byly popsány v rámci kapitoly 3.1.4 Techniky animování modelů, jsou v rámci software Maya dostupné. Software v současné době poskytuje jejich optimalizované metody i nové způsoby, jež nebyly popsány. Taktéž je podporována většina ze způsobů modelování popsaných v kapitole 3.1.3 Techniky modelování objektů, co se ručně vytvářených modelů týče. V novějších verzích Mayi je také umožněno přímo nastavení projektu vytvořeného v rámci některého z herních enginů Unreal Engine a Unity 3D, kterým je věnována kapitola 3.2.3 Unity 3D a 3.2.4 Unreal Engine, a to z důvodu snazšího exportování vytvořených modelů, jakožto částí multimediálního obsahu budoucí aplikace, z Mayi do jednoho z těchto herních enginů.

Uživatelům tohoto nástroje je dostupná rozsáhlá online dokumentace spolu s připravenými návody, které je provedou napříč různými oblastmi tohoto nástroje a pomohou jim při získávání zkušeností. Jedná se tak o prvotní bod cesty stát se ze začátečníka profesionálním vývojářem. [20]

Autodesk Maya představuje placený software, kdy si uživatel může zvolit období, po které by si přál tento software využívat. Existují tři varianty, mezi které patří měsíční při ceně 185 $ neboli 6 153.58 Kč, roční s cenou 1 470 $ neboli 49 228.61 Kč a tříletá licence za 4 410 $ přepočteno na 147 685.82 Kč. [21], [22] Taktéž je poskytována 30-ti denní bezplatná zkušební verze. V poslední řadě je, díky filozofii, jenž razí firma Autodesk, že je důležité podporovat vzdělávání s cílem vychovat nové vývojáře, kteří by usilovali o neustálé inovace napříč všemi technickými oblastmi, dostupná také tříletá licence softwaru pro studenty, učitele a univerzity zcela zdarma. Studentská verze zdarma se vztahuje na veškerý vyvíjený software, ne jen na animační software Maya, a je určena pouze pro účely vzdělávání, ať už v rámci školy nebo samostatně. Nesmí být využívaná pro tvorbu obsahu za účelem komerčního využití. [23]

Kromě tohoto víceúčelového nástroje je možné využít méně obsáhlou a levnější verzi zvanou Maya LT, která představuje software vhodný především pro nezávislé vývojáře her. Tento software poskytuje uživatelům nástroje pro tvorbu trojrozměrných

modelů postav a terénů a animační nástroje spolu s možností přivedení jich k životu pomocí enginu a jednoduchého skriptování. [24]

3.1.6 Blender 2.79

Blender představuje software pro tvorbu trojrozměrného obsahu zahrnující nejen modelovaní, ale i tvorbu animací a simulací, rigging (neboli manipulaci s modely s kostrou), skládání a sledování pohybu, vykreslování scén i vývoj her. Nástroj se v závislosti na efektivních metodách tvorby animace a vykreslování uchytil i v rámci filmového průmyslu, kde je využíván pro tvorbu reklam či krátkých animovaných filmů. [25]

Nástroj byl poprvé představen v roce 1995 holandským animačním studiem NeoGeo. Od roku 1998 se o jeho vývoj a šíření starala nově založená firma Not a Number, kdy ovšem došlo k ukončení jeho vývoje. Ovšem v roce 2002 po získání finančního obnosu byl jeho vývoj opět zprovozněn a Blender je od té doby distribuován neziskovou nadací Blender Foundation [26]. V současné době je šířený jako open-source nástroj pod GNU licencí. Jedná se tak o nástroj vyvíjený především lidmi po celém světě, studenty, nadšenci, vědci, ale i profesionály z oboru. [25]

Blender poskytuje podporu pro platformy Windows, Linux a OS X a je plně ke stažení zdarma. Vzhledem k jeho šíření jako open-source softwaru mají uživatelé možnost jeho volného užívání, distribuování, zkoumání a modifikování. Přestože využívá různých komponent jako například jazyka Python šířeného pod jinou licencí, tak všechny tyto komponenty tvořící Blender jsou brány dle licence GNU verze 3. A veškerý obsah vytvořený užitím tohoto nástroje je výhradním majetkem dotyčného vývojáře, který s ním může nakládat dle svých potřeb a přání. Blender je tedy možné využívat i pro komerční účely bez omezení. [27]

Přestože je Blender zcela zdarma, uživatelé zde mají možnost za poplatek 9.90 euro měsíčně získat přístup k balíčku obsahujícímu všechny soubory s vydanými videi a návody, dále přes sto hodin kvalitního tréninku s cílem získat potřebné znalosti a dovednosti pro efektivní využívání Blenderu při své tvorbě, také možnost synchronizace nastavení přes více zařízení, 1500 textur ke stažení a funkci pro sdílení obrázků přímo z Blenderu. [25]

Blender poskytuje uživatelům funkčnosti ve všech možných oblastech. Jedná se například o různé způsoby modelování trojrozměrných objektů, prostřednictvím polygonálního modelování, modelování pomocí křivek, sculpting a další. Další oblastí je vykreslování, kde je využíváno renderovacího enginu Cycle, podpory různých osvětlovacích modelů a materiálů, dále tvorba animací zahrnující nástroje pro inverzní i dopřednou kinematiku, synchronizaci se zvukem, animaci po křivce, klíčování a další. Také možnost editace videí a tvorby hry prostřednictvím herního enginu Blender, který je vestavěn. Zde je umožněno vytvoření herní logiky pomocí skriptů v jazyce Python, zavedení umělé inteligence, dostupná podpora pro dynamické osvětlení a stínování, a v neposlední řadě také fyzika tuhého tělesa. Dále možnost kompozice celé scény v rámci Blenderu a rozsáhlé možnosti přizpůsobení jednotlivých ovládacích prvků Blenderu. [28]

Obdobně jako v předchozím případě se softwarem Autodesk Maya, i Blender podporuje všechny způsoby tvorby animace, které byly zmíněny v předchozím textu. Stejně tak i většinu metod pro vytváření trojrozměrného obsahu.

Nové verze tohoto nástroje jsou vydávány po dvou až čtyřech měsících, vývoj probíhá tedy poměrně rychle, s cílem srovnat Blender s komerčními nástroji, jako je například Autodesk Maya. Verze 2.26 byla první vydanou verzí Blenderu jako open-source softwaru pod GNU licencí v roce 2002. Aktuální verze nese označení 2.79 a byla nasazena do provozu v polovině září loňského roku. Nyní je již připravována verze 2.8. [29]

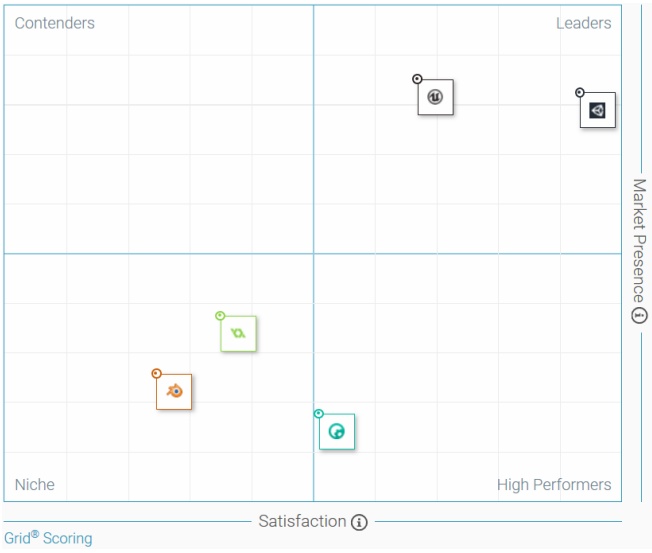
3.2 Herní engine a OOP

Zatímco předchozí kapitola 3.1 Počítačová grafika a grafické modelovací nástroje byla věnována problematice reprezentace dat v počítačové grafice, různým metodám vytváření a animování trojrozměrných modelů a analýze dvou vybraných modelovacích nástrojů, tedy částečné tvorbě multimediálního obsahu aplikace, tak tato kapitola se zabývá tvorbou samotného jádra aplikace. Během této fáze vývoje jsou všechny součásti tvořící multimediální obsah aplikace sestaveny užitím softwaru zvaného herní engine, do několika komplexnějších scén tvořících jednotlivé úrovně finální aplikace.

V první části této kapitoly jsou definovány pojmy herní engine a multimediální obsah interaktivní aplikace. Je zde popsáno, jaké moduly může tento softwarový nástroj vývojářům nabídnout, se zaměřením na fyzikální engine poskytující mimo jiné funkce pro detekci kolizí mezi objekty. V další části jsou popsány principy objektově orientovaného

programovaní, jenž je využito při psaní skriptů zajišťujících aplikační logiku. V poslední řadě je zde provedena obdobná analýza z předcházející kapitoly, tentokráte se zaměřením na dva vybrané herní enginy, Unity 3D a Unreal Engine. Ty byly vybrány na základě prozkoumaných statistických analýz provedených společností G2 Crowd, jenž jsou zveřejněny na stránkách této společnosti. [30]

Společnost G2 Crowd se zabývá hodnocením produktů na základě recenzí získaných z různých online zdrojů, ze sociálních sítí a od uživatelů registrovaných na stránkách této společnosti. Tyto nasbíraná data jsou následně zpracována algoritmem a je sestaven graf ve formě mřížky, který znázorňuje míru spokojenosti uživatelů a podíl na trhu. Spokojenost uživatelů je založena na uživatelských recenzí a na podíl na trhu mají vliv údaje nasbírané z deseti různých zdrojů, jako jsou velikost prodejce a sociální dopad [30]. Dle vypočtených údajů se do grafu na konkrétní pozice zaznamenají příslušné produkty. Ty jsou takto rozřazeny do čtyř kategorií, které tvoří leaders neboli produkty na vedoucích pozicích, contenders tzv. uchazeči o vedoucí pozici, jež nemají tak značné zastoupení mezi uživateli, dále high performers představující uživatelsky kladně hodnocené produkty s nižším tržním podílem a poslední skupinu zvanou niche neboli produkty s malým podílem na trhu bez značného zastoupení mezi uživateli [30]. Toto rozřazení ukazuje (Obrázek 2) týkající se přímo vyhodnocení herních enginů pro rok 2018. Z grafu je patrné, že vedoucí pozici (Leaders) mezi dostupnými herními enginy zaujímají právě výše zmíněné Unity 3D a Unreal Engine, jedná se tak o produkty, které mají značný podíl na trhu a jsou kladně hodnoceny uživateli.



Obrázek 2 - Graf pro hodnocení herních enginů
[30]

U vybraných herních enginů tak existuje velká komunita vývojářů, kteří je využívají při tvorbě svých interaktivních aplikací, což má jistě vliv na společnosti vydávající tyto produkty. Zvyšuje se tak možnost kvalitněji zpracovaných dokumentací daných nástrojů a četnost aktualizací na nové verze poskytující vylepšené funkčnosti.

3.2.1 Herní engine a jeho komponenty

Multimediálním obsahem aplikace jsou myšleny všechny komponenty, ze kterých se výsledná aplikace skládá. Pod tímto pojmem se skrývají veškeré textury vytvořené pomocí bitmapových či vektorových grafických nástrojů, jako jsou Photoshop, GIMP či Inkscape a jiný software, které jsou použity pro tvorbu grafického uživatelského rozhraní nebo jsou mapovány přímo na některé trojrozměrné modely. Dále se pod tímto termínem mohou skrývat trojrozměrné modely tvořící hlavní část aplikací pracujících s 3D grafikou, importované z modelovacích nástrojů v příslušném datovém formátu. Taktéž veškeré audio soubory a v poslední řadě i soubory se skripty napsanými v některém z programovacích jazyků, jež jsou podporovány příslušnými herními enginy. Skripty zprostředkovávají veškerou herní logiku, propojují objekty ve scéně a zajišťují jejich vzájemnou interakci a především provazují všechny úrovně do jediného funkčního celku. [31, s. 4]

Herní engine je definován jako software poskytující vývojové prostředí pro rychlý a snadný vývoj her pro různé typy koncových zařízení, počítač, konzoly či mobilní zařízení. Jedná se o tzv. jádro interaktivní aplikace a lze sis jej představit jako jakýsi prostředek pro sestavení výsledné aplikace z částí tvořících její multimediální obsah. Základní myšlenkou je usnadnění vývoje her prostřednictvím znovuvyužití nebo přizpůsobení herního enginu dané hře, jedná se tak znovuvyužitelnou komponentu, sloužící k definování rámce celé vytvářené aplikace. Existuje ovšem i možnost napsání vlastního enginu, přímo šitého na míru vytvářené hře, to ale značně komplikuje její vývoj a zvyšuje i časovou náročnost projektu. [32, s. 5]

Toto vývojové prostředí může ve velmi omezeném provedení reprezentovat pouhý software pro vykreslování trojrozměrných objektů v reálném čase. U herního enginu se ale zpravidla jedná o robustní systém poskytují řadu funkcionalit, modulů, mezi než se řadí například ono zmiňované vykreslování 2D a 3D grafiky v reálném čase, umělá inteligence, modul pro správu souborů, zvukových souborů, spouštění a správa animací jednotlivých objektů. Dalšími funkcemi jsou skriptování, záležitosti okolo sítí, zpracování vstupů

a generování výstupů, tvorba a správa grafického uživatelského rozhraní a především engine pro fyziku a graf scény pro reprezentaci rozlehlých světů, který je popsán dále v textu. [32, s. 6]

Fyzikální engine patří k nejdůležitějším modulům poskytovaným v rámci robustních herních enginů. Jedná se o software simulující některé fyzikální jevy z oblastí, jako jsou dynamika tekutin, dynamika měkkých těles a dynamika tuhých těles, kdy je zkoumáno především vzájemné působení objektů a sil způsobujících jejich pohyb. Tyto enginy se v základu rozdělují do dvou skupin, kdy jedna skupina slouží pro přesnější vědecké výpočty a simulace zkoumaných jevů, kdy není kladen přílišný důraz na rychlé zpracování a je zde možnost složitějších výpočtů pro vyšší přesnost získaných výsledků. Druhá skupina představuje fyzikální enginy využívané v herním průmyslu, kdy je třeba rychlých výpočtů v reálném čase i za cenu snížení přesnosti, dochází tak k pouhé aproximaci skutečných jevů. [33]

Jednu z nejdůležitějších částí fyzikálního enginu tvoří mechanismus detekce kolizí, který se vztahuje na oblasti vědeckých simulací, tvorbu videoher i robotiku. Při výpočtu kolizí různých objektů ve scéně je potřeba nejen kolizi objevit, ale také na ni řádně reagovat, kdy se může jednat například o způsobení poškození daného objektu, simulaci výbuchu či vzájemné odražení kolizních objektů. Důležitý krok ale představuje samotná detekce kolize, kdy se jedná o výpočetně náročný mechanismus. Opět zde záleží především na účelu použití, kdy pro vědecké simulace je potřeba kvalitních a přesných výpočtů, naopak při vykreslování v reálném čase v rámci prezentace hry je důraz kladen na rychlost, a je tedy dovoleno určitých kompromisů v přesnosti výpočtů. [34]

Scény v aplikacích mohou být značně rozsáhlé, a obsahovat tak nespočet různých objektů, které by mohly vzájemně kolidovat. Zde by bylo provádění detekce kolizí každého objektu s každým velmi neefektivní, a proto se využívají různé způsoby optimalizace detekcí. Příkladem může být využití ohraničujících objektů, tzv. bounding boxů. Jedná se o jakési obálky, s méně složitou povrchovou geometrií, ohraničující jednotlivé složitější objekty ve scéně. Obalovými předměty mohou být například koule, jež mají malé nároky na výpočetní výkon, naopak nedovolují moc přesné ohraničení daného objektu. Dalšími obalovými objekty mohou být kvádry, které disponují schopností přesnějšího obalení objektu ve scéně, mají ale i větší nároky na výpočetní zpracování. Optimalizace detekce kolize následně spočívá v prvotní kontrole průniku těchto obalových

objektů a teprve při detekci tohoto průniku se přechází na podrobnější zkoumání vzájemného průniku jednotlivých povrchových geometrií. Taktéž je možné použití více úrovní těchto obálek, kdy u složitějších objektů uzavírají různé obálky jen jeho komponenty, a nakonec jedna obálka kompletně celý objekt. Následkem je provádění několika méně náročných detekcí, než se přejde na detekci sítě polygonů tvořící povrch daného objektu. [34]

Detekci kolizí lze také zoptimalizovat rozdělením prostoru samotné scény, například využitím datové struktury octree, jenž byla podrobněji popsána v rámci kapitoly 3.1.2 Reprezentace 3D objektů v části věnované objemové reprezentaci. Při této metodě je nejprve zúžen samotný prostor, ve kterém může ke kolizi dojít, a teprve až poté prováděna kontrola vzájemných kolizí mezi objekty obsaženými v tomto podprostoru, i s případným užitím obalových objektů pro dosažení lepších výsledků zefektivnění této metody. [5, s. 410-412]

Při průběhu aplikace jsou v podstatě cyklicky prováděny tři činnosti: aktualizace virtuálního světa, jeho prezentace a zpracování vstupů od uživatele. Všechny tyto činnosti má na starosti aplikační logika, která v závislosti na stavu objektů ve scéně a vstupů od uživatele provádí příslušné definované akce. Ať už se jedná o pasivní prvky s pevně definovanými vlastnostmi, aktivní prvky obohacené o umělou inteligenci a tedy schopnost rozhodování, či statické prvky tvořící virtuální svět. Všechny provedené změny musí být poté prezentovány uživateli s cílem jeho zanoření do děje. Pro navození realističtějšího dojmu je při prezentaci využíváno nejen vizuálních prvků scény, ale i například zvukových. Problém nastává při přílišné rozloze virtuálního světa, kdy by bylo neefektivní jeho prezentování jakožto jediného celku se všemi příslušnými prvky. Tento problém řeší modul tzv. grafu scény, jenž bývá součástí robustnějších herních enginů. [32, s. 7, 14-22]

Scéna představuje množinu objektů spolu s definovanými informacemi nezbytnými pro její správné vykreslení, jako jsou zobrazované objekty reprezentované jejich povrchovou geometrií, případně textury a materiály, které jsou jim přidruženy. Dále obsahuje nezobrazované prvky představující světla a kamery, prvky určující její logickou strukturu a hierarchicky definované transformace. Graf scény představuje strukturu umožňující logické uspořádání objektů a jejich efektivní transformování. Jedná se přitom o n-ární strom, kde každý uzel má právě jednoho předchůdce. Listy tohoto stromu obsahují informace o geometriích jednotlivých modelů, o použitých texturách či barevných

vlastnostech, vnitřní uzly reprezentují jednotlivé transformace aplikované na dané objekty uložené v příslušných větvích stromu, kdy se konečná transformace získá jejich složením od kořene stromu. Tento strom je v rámci vykreslování scény procházen zleva doprava. [5, s. 397-401]

3.2.2 Principy objektově orientovaného programování

Objektově orientované programování (OOP) představuje nový způsob myšlení při tvorbě počítačových programů, jenž je naprosto odlišný od dřívějších programovacích paradigmat, chápáno ve smyslu způsobu myšlení. OOP se netýká pouze samotné implementace, tedy psaní kódu, jedná se i o jiný způsob myšlení již v rámci návrhu daného programu. Hlavní důraz je zde kladen na možnost opětovného využití již objeveného, vytvořeného a hlavně řádně otestovaného, což značně urychlí vývoj nového programu. [35]

Myšlenka OOP spočívá ve vytváření abstraktního modelu reálného či smyšleného světa, jedná se tak o jakousi simulaci skutečnosti prostřednictvím definovaných objektů, které spolu vzájemně komunikují. Výhoda tohoto přístupu spočívá v lepší přehlednosti a čitelnosti výsledného kódu pro člověka, programátora. To má za následek snazší a intuitivnější orientaci v kódu programu, a s tím spojený i efektivnější vývoj kvalitního softwaru. [36, s. 54, 55]

Základními komponentami tohoto abstraktního modelu skutečnosti jsou objekty, které spolu mohou vzájemně komunikovat. Každý z těchto objektů má svůj vnitřní stav, který je reprezentován hodnotami jeho atributů, jež popisují jednotlivé vlastnosti daného objektu. Například v případě objektu psa to mohou být atributy popisující jeho jméno, věk či rasu. Objekty mohou mít také definované chování prostřednictvím metod reprezentujících, co daný objekt umí provádět. Zde je možné si chování představit jako schopnost psa štěkat, tedy metodu štěkej. [35] Všechny tyto objekty mohou vzájemně komunikovat, kdy k této komunikaci dochází prostřednictvím zpráv vysílaných jedním objektem k druhému. Těmto zprávám se také říká volání metod a právě jednotlivé metody daných objektů reprezentují schopnost objektu na danou zprávu reagovat. [36, s. 57, 58]

Dalším důležitým prvkem OOP jsou třídy, které reprezentují jakousi šablonu na objekty stejného typu. Třída definuje předpis rozhraní, které budou mít všechny objekty jejího typu, neboli názvy a datové typy jednotlivých atributů a metody. Takové objekty

jsou nazývány instancemi dané třídy a představují už konkrétní objekt daného typu, například konkrétního tříletého psa jmenujícího se Buddy jakožto instanci třídy Pes. [36, s. 55, 56]

Objektově orientované programování je založeno na zapouzdření, dědičnosti a polymorfismu.

3.2.2.1 Zapouzdření

Princip zapouzdření (encapsulation) umožňuje uchovávat data o stavu, tedy atributy, a metody, které s nimi pracují, uvnitř objektu tak, aby nebylo možné k nim zvenčí libovolně přistupovat. Dochází tak ke skrývání vnitřní implementace daného objektu, se kterým je možno komunikovat prostřednictvím poskytnutého rozhraní. Význam zapouzdření spočívá nejen v ukrytí vnitřní implementace, takže nikdo neví, jak objekt daný úkol vnitřně řeší, ale především zabezpečuje konzistenci programu, neboť nikdo nemůže zvenčí daný objekt upravit nedovoleným způsobem, který by mohl zapříčinit vznik neočekávaných chyb. Poskytnuté rozhraní si lze představit jako sadu metod, které jsou viditelné zvenčí, a skrze které je možné s daným objektem komunikovat. Součástí veřejného rozhraní bývají zpravidla metody označované jako gettery a settery sloužící k získání či případnému nastavení hodnoty privátních, neveřejných atributů daného objektu. Také sem mohou patřit další metody, nezbytně nutné pro správnou komunikaci s objektem, které tvůrce daného objektu poskytne. [36, s. 284-286]

Díky zapouzdření je možné si výsledný program představit jako spolupracující jednotky, které si mezi sebou předávají zprávy prostřednictvím svých veřejných rozhraní. Navíc je umožněno změnit vnitřní implementaci jedné této jednotky, aniž by se ohrozila funkčnost zbytku programu za předpokladu, že nebylo změněno rozhraní, které ostatní objekty již využívají ke komunikaci s daným objektem. [36, s. 285, 286]

3.2.2.2 Dědičnost

Druhým pilířem objektově orientovaného programování je dědičnost. Jedná se o hierarchickou vazbu typu předek - potomek mezi dvěma třídami. Předek neboli rodičovský typ deklaruje obecné vlastnosti daného datové typu. Potomek neboli dceřiný typ dědí veškeré nesoukromé vlastnosti a chování a také vazby rodičovského typu a zpravidla definuje ještě své vlastní, neboť dědičnost se využívá právě ke specifikaci

speciálního typu daného nadtypu. Dědičnost tak přináší výhodu v podobě možnosti rozšíření funkčnosti některé třídy bez nutnosti redundantního kódu, popřípadě provedení jednotné změny, kdy se změna v rodičovské třídě projeví zároveň ve všech potomcích. [37]

V objektově orientovaných programovacích jazycích, jako jsou C# a Java, je ovšem povoleno dědění pouze z jediného nadtypu, z jediné třídy. Vícenásobná dědičnost není podporována z důvodu udržení přehledného a jednoduchého kódu programu. Ale je zde povolena vícenásobná implementace rozhraní, neboli také dědičnost rozhraní. [37] Rozhraní (interface) definuje veřejné rozhraní pro komunikaci s objektem, který jej implementuje. Jedná se tak o s předpis definující, které metody musí daná třída implementovat, pokud dědí z tohoto rozhraní. Zjednodušeně řečeno se jedná o speciální typ třídy, která neobsahuje atributy ale pouze hlavičky metod, všechny veřejné. Tím, že se rozhraní objektu deklaruje způsobem jako nová třída, je zajištěno, že může být použito jako datový typ objektu. [38]

V OOP jsou podporovány tři druhy dědění. Prvním je tzv. přirozené dědění založené právě na myšlence existence určité množiny instancí s podobnými vlastnostmi. Toto dědění je využito pro definování speciálních případů určitého datového typu, jak bylo popsáno dříve v textu. Druhým typem je dědí typu, přesněji rozhraní. Tento způsob je využíván nejčastěji a váže se na dědění z tříd i interfaců. Dovoluje instanci potomka nastoupit tam, kde je očekávána instance předka, zastoupit jej, neboť má stejné rozhraní jako jeho předek zpravidla obohacené o vlastní metody. Posledním typem dědičnosti je dědění implementace, které je vázáno na dědění tříd. Jedná se o skutečnost, že potomek zdědí veškerou implementaci svého předka, je ale nezbytné dodržení rozhraní, aby bylo možné tyto metody využít. Potomek tak automaticky dědí všechny veřejné metody svého předka, které může využívat tak, jak jsou naimplementovány, nebo je může překrýt svou vlastní implementací. [36, s. 176-178]

Jako předek bývá často užívána tzv. abstraktní třída. Jedná se o třídu, od níž nemohou být vytvářeny žádné instance. Takovéto třídy mohou obsahovat definice atributů i metod a abstraktní metody, kdy se jedná o pouhou deklaraci dané metody bez její implementace. Takovéto metody musí implementovat až daný potomek, nebo se taktéž stává abstraktní třídou. Abstraktní třída tedy představuje jakousi kombinaci vlastností konkrétní třídy objektů a interfacu. [36, s. 186, 187]

V rámci dědičnosti v OOP není možná vícenásobná dědičnost, jak bylo popsáno dříve, ovšem při dědění vzniká hierarchický graf, strom jednotlivých tříd, kdy každá má právě jednoho potomka. To znamená, že dědí -li třída od jiné, tak zdědí její rozhraní a implementaci, pokud ovšem tato třída rovněž představuje potomka něčí třídy, tak její potomek dědí i tyto vlastnosti. [36, s. 156]

3.2.2.3 Polymorfismus

Třetí pilíř objektově orientovaného programování tvoří tzv. polymorfismus. Polymorfismus neboli mnohotvárnost přestavuje jakousi schopnost objektu vydávat se v různých situacích za instance různých datových typů, tedy různých tříd. Objekt následně reaguje na zaslanou zprávu dle toho, čí skutečnou instancí je, které mateřské třídy, a ne dle toho, za čí instanci se vydává, za jakého předka. Toto lze uvést na příkladu, kdy existuje obecná třída zvíře, která deklaruje společné vlastnosti, jež zdědí její potomci, může se jednat o metodu "vydej zvuk". Jednotlivými potomky mohou být třídy pes, kočka a další zvířata, které zdědí deklarovanou metodu od předka, ale její implementaci řeší každá samostatně až na své úrovni hierarchie. Pokud poté existuje instance třídy pes ovšem datového typu zvíře na níž je zavolána metoda "vydej zvuk", je provedena právě metoda potomka. Zjednodušeně, polymorfismus zajišťuje možnost deklarace metod, jež jsou společné různým třídám, ovšem s různou reakcí typickou pro danou třídu, daného potomka. Tedy definování metod se stejnou hlavičkou ale jiným chováním, jinou implementací. [36, s. 128, 129]

3.2.2.4 Zásady objektově orientovaného programování

Kromě těchto uvedených vlastností objektově orientovaného programování je důležité mít při vývoji programů na mysli i zásady správného programovaní s využitím OOP. Existuje devět základních zásad, které jsou stručně zmíněny v následujícím textu a mezi něž patří: [39, s. 39]

- **Programování proti rozhraní**

Programování proti rozhraní, nikoliv proti vnitřní implementaci, kdy platí, že by měl být výsledný kód vytvářen pomocí instancí datového typu nesvázaného s konkrétní implementací. Jedná se tedy o využívání dědičnosti od abstraktních tříd či interfaců, s cílem snazšího zavádění pozdějších úprav. [39, s. 40-44]

- **Důsledné skrývání implementace**

Jedná se o ukrytí vnitřní implementace pro okolní objekty, které tak s daným objektem mohou komunikovat pouze přes poskytnuté rozhraní. Objekt by měl skrýt to, jak vnitřně funguje a poskytnout jen důležité informace pro práci s ním. Důvodem je vyšší bezpečnost před neoprávněnou změnou, a také souvisí s další zásadou. [39, s. 44-47]

- **Zapouzdření a odpoutání kódu**

Zapouzdření a odpoutání kódu umožňuje provádění změn, aniž by došlo k výskytu fatálních chyb, neboť ostatní objekty s daným objektem pracují pouze přes poskytnuté rozhraní a nevidí jeho vnitřní implementaci, která tak může být nezávisle měněna. [39, s. 47, 48]

- **Přednost skládání před dědičností**

Dědičnost by měla být využívána jen v případech, kdy se jedná o potomky, které jsou speciálními případy svého nadtypu, neboť narušuje princip zapouzdření. [39, s. 48-50]

- **Dodržování vysoké soudržnosti**

Dodržování vysoké soudržnosti, neboli koheze. Jedná se o zásadu definující, že jeden objekt by se měl zaměřit pouze na jednu činnost. Programy by měly sestávat z vyššího počtu menších tříd, které každá plní jen konkrétní podobné úkoly, stejně tak metody. [39, s. 50, 51]

- **Návrh řízený odpovědnostmi**

S předchozí zásadou souvisí i návrh řízený odpovědnostmi (responsibility-driven design). Dodržování této zásady taktéž usnadňuje budoucí úpravy programu. Již při návrhu jsou rozmyšleny oblasti, za které budou ta či ona třída odpovědny, při následných změnách pak není nutné procházet kompletně celý kód programu a hledat, kde všude je měněná funkčnost využívána a ovlivňována. [39, s. 52]

- **Udržování minimální vzájemné provázanosti**

Další důležitou zásadu při vytváření programu s užitím OOP představuje udržování minimální vzájemné provázanosti, tzv. coupling. Jedná se o snižování počtu závislostí jedné třídy na jiné, kterou tato daná třída využívá ke své správné činnosti. [39, s. 52]

- **Vyhnutí se duplicitnímu kódu**

Využívání duplicitního kódu vede často ke značným chybám vzniklých úpravou na několika různých místech. [39, s. 53, 54]

- **Podřízení návrhu architektury programu snaze o maximální efektivitu**

Jedná se o situaci, kdy je věnována přílišná pozornost dopředné snaze o optimalizaci kódu, který ještě není ani schopen provozu. [39, s. 54, 55]

Tyto zásady by měl každý vývojář v objektově orientovaném programování dodržovat, aby vytvářel efektivní, správně strukturovaný a přehledný kód s co nejmenším počtem chyb.

3.2.3 Unity 3D

Herní engine Unity 3D vydávaný společností Unity Technologies spatřil světlo světa v polovině roku 2005 na celosvětové konferenci vývojářů pořádané společností Apple, tehdy s podporou pouze pro koncová zařízení s OS X. Od té doby se leccos změnilo a v současné době se jedná o jeden z nejúspěšnějších softwarových nástrojů pro vývoj interaktivních aplikací na různé platformy, konkrétně se jedná o 28 platforem zahrnujících osobní počítače, herní konzole, mobilní zařízení, zařízení pro virtuální realitu, televizní zařízení a web. [40]

Unity 3D se řadí na přední místa v oboru herních enginů a vývoje počítačových her. Tento nástroj lze využít pro tvorbu 2D a 3D aplikací i aplikací pro virtuální realitu. Studie z roku 2016 prezentovaná na oficiálních stránkách Unity dokládá, že 34 procent z 1000 nejlepších her na mobilní zařízení je vytvořeno právě prostřednictvím Unity, které se drží na předních pozicích i v oblasti virtuální reality. [41]

Nové verze softwaru jsou vydávány přibližně po jednom až dvou letech. V roce 2011 byla vydána verze 3.0, tu rok poté nahradila verze 4.0, která vydržela na trhu s průběžnými aktualizacemi až do verze 4.6, kdy ji v roce 2015 nahradila verze 5.0. Dva roky poté byla opět verze 5.6 nahrazena nově číslovanou verzí 2017.1, kdy v současné době je připravována verze 2018.1 a toto nové číslování zůstává zavedeno s ohledem na plánované každoroční vydávání nových verzí softwaru. [42]

Každá vyvíjená aplikace potřebuje implementovat herní logiku napsanou prostřednictvím programovacího jazyka. Aktuální verze Unity podporuje skriptování v jazyku C#, což je objektově orientovaný programovací jazyk podobný Javě a C++, a UnityScript, jazyk navržený speciálně pro Unity, který je obdobou Javascriptu. Starší verze podporovaly skriptování pomocí třetího jazyka zvaného Boo s podobnou syntaxí Pythonu, ten byl ovšem s příchodem verze 5.0 odebrán. Všechny skripty jsou tvořeny

prostřednictvím integrovaného vývojového prostředí (IDE) zvaného MonoDevelop, jež je dodáváno s Unity. [43]

V současné době je tento herní engine distribuován ve třech variantách: Personal, Plus a Pro. Personal je určeno především pro studenty, začátečníky v oboru a nadšence, kteří se chtějí věnovat vývoji her. Tato varianta je zcela zdarma, uživatelé tak nemusejí platit žádné licenční poplatky, a přesto je garantováno, že veškerý obsah, který prostřednictvím tohoto nástroje vyvinou, je plně v jejich vlastnictví. Personal verze poskytuje začínajícím vývojářům přístup ke všem funkcím základního herního enginu, zajišťuje nepřetržité aktualizace softwaru a umožňuje vývoj pro všechny publikované platformy. Získávání nových zkušeností při vývoji je podpořeno také značným množstvím dostupných výukových materiálů s ukázkami kódů. Těmto zájemcům o vstup do světa vývojářů může být nápomocna i široká komunita uživatelů Unity, kteří sdílejí své nápady na často řešené problémy prostřednictvím fóra, a kteří jsou většinou ochotni pomoci a poradit nově začínajícím. Přičemž lze také navštívit Assets Store, kde je možno získat zdarma či za poplatek multimediální obsah do svých prvních vyvíjených her. [44]

Kromě zmíněných vlastností nepřetržité aktualizace, podpory všech platforem a přístupu ke všem funkcím základního enginu zahrnuje verze Personal i reklamy, možnost provádět nákupy v aplikaci, analytické jádro a podporu vývoje pro aplikace umožňující více hráčů (multiplayer), až pro 20 souběžných uživatelů. [44]

Platí ovšem, že verze Personal smí být využívána pro komerční účely pouze pokud daná firma či jedinec nevykazuje roční hrubý příjem vyšší než 100 tis. $ (USD, United States dollar) nebo nebyly jinak získány finanční zdroje převyšující tuto částku. V takovém případě je dle smluvních podmínek uživatel nucen přejít na verzi Plus nebo Pro. [44]

Verze Plus je zpoplatněna částkou 35 $ za měsíc a je doporučována vývojářům, jenž to s tvorbou interaktivních aplikací myslí vážně a svou tvorbu chtějí zveřejňovat. Uživatelům poskytuje veškeré funkčnosti jako verze Personal obohacené o možnost nastavení vlastní úvodní obrazovky, profesionální editor na návrh grafického uživatelského rozhraní, možnost 50 souběžných uživatelů pro multiplayerové aplikace, lepší správu reklam pro vyšší zisky a zpracování zpráv o chybách v reálném čase. I u verze Plus existuje podmínka pro komerční užívání, která představuje obdobu podmínky u Personal, zde ovšem do částky převyšující 200 tis. $. [45]

Poslední možností je verze Pro, která je určena pro profesionální vývojáře, kteří ocení možnosti pokročilého přizpůsobení softwaru a vyžadují plně flexibilní nástroj pro svou tvorbu. Jedná se o verzi zpoplatněnou částkou 125 $ za měsíc. Unity Pro poskytuje vývojářům veškeré funkčnosti jako Unity Plus, navíc navyšuje počet souběžných uživatelů při multiplayerové aplikaci na 200. Dále poskytuje měsíční a čtvrtletní vydání zaměřené na kvalitu a stabilitu, lepší podporu a umožňuje přístup ke zdrojovým kódům. Poslední poskytovanou výhodou je, že již neklade žádná omezení na výši výdělků při komerčním užívání softwaru. [46]

3.2.4 Unreal Engine

Unreal Engine vyvíjený firmou Epic Games patří v současné době spolu s Unity 3D mezi přední software užívaný pro tvorbu her a simulací. První verze byla představena v roce 1998, kdy byla využita pro vývoj stejnojmenné hry Unreal, od té doby prošel software značnými změnami a v současné době je aktuální Unreal Engine 4.

Verze Unreal Engine 1 byla původně navržena pro hry typu FPS (First-person shooter) neboli střílečky z pohledu první osoby, využití ovšem nalezla i v rámci her typu RPG (Role-playing game), tedy her u nichž hráč hraje v roli fiktivní postavy dle určitých pravidel. Už v rámci této verze došlo ke sjednocení hardwarového a softwarového vykreslování, mechanismu detekce kolizí, osvětlování v reálném čase, barevného osvětlení, filtrování textur a práci se zvukovými soubory pod tento jeden herní engine. [47] Verze Unreal Engine 2 byla vydána 4 roky poté a optimalizovala veškeré funkčnosti verze 1. Verze 3 následně přinesla změnu ve výpočtech osvětlení, které nyní byly prováděny pro každý pixel, nikoliv vrchol jako tomu bylo u předchozích verzí a výhodu plně programovatelného shaderu [48, s. 58-65]. V roce 2012 byla následně vydána verze 4, která byla původně zdarma poskytnuta pouze školám, od roku 2015 je zdarma dostupná všem.

V současnosti představuje Unreal Engine 4 software pro vývoj her, mobilních aplikací a aplikací pro virtuální realitu, tvorbu podnikových aplikací, filmů a simulací, tedy vhodný nástroj pro využití všude, kde je třeba práce s technologií v reálném čase. V rámci tohoto nástroje je velký důraz kladen především na vykreslování v reálném čase, poskytnutí možnosti vytvářet fotorealistické snímky a pracovat s velkými detaily i při rychlosti zobrazování devadesát snímků za vteřinu v oblasti virtuální reality. Vykreslovací

engine ovšem není jedinou poskytovanou funkčností, Unreal Engine představuje kompletní sadu nástrojů bez nutnosti dokupovaní doplňků. Zahrnuje v sobě tedy nejen vykreslovací nástroje, ale i fyzikální engine pro simulace fyzikálních jevů, nástroje pro práci se zvukovými soubory, nástroj pro návrh grafického uživatelského rozhraní, animační systém pro kinematiku, správu souborů, 2D grafiku, nástroj pro síťové aplikace a multiplayer, nástroj pro práci s rozlehlými terény, pro logování chyb, skriptování a další, podrobněji popsané v online dokumentaci. [49]

Jedná se o engine, jež je napsán čistě pomocí programovacího jazyku C++ a na rozdíl od Unity je zde poskytnut přístup ke zdrojovým kódům zdarma všem členům komunity pro potřeby studia, ladění, upravování či rozšiřování funkčnosti enginu prostřednictvím služby GitHub, kde je ke stažení. [50] Ohledně tvorby herní logiky pomocí skriptů napsaných programovacími jazyky, tak do vydání verze 4 bylo možné užívat UnrealScript, jazyk podobný Javě, který byl vyvinut speciálně pro Unreal Engine a vizuální skriptovací systém Kismet. S příchodem čtvrté verze došlo k vypuštění jazyka UnrealScript a zůstalo tak pouze C++ a byl vyvinut nový vizuální skriptovací systém zvaný Blueprint, který umožňuje vytváření herních prvků bez nutnosti zásahů do kódu. Systém je založený na uzlech reprezentujících objektově orientované prvky, jako jsou třídy, a vazbách mezi nimi představující funkce. Je tak umožněna vyšší úroveň spolupráce mezi programátory a návrháři, kdy jedni mohou definovat počátek a zbylí doladit specifikace. [51]

Unreal Engine je možné využívat na operačních systémech Windows, Linux a OS X a v současné době je poskytnuta podpora pro vývoj na přibližně 15 platforem zahrnujících i platformy pro mobilní zařízení, virtuální realitu a herní konzole. [52]

Pro začínající vývojáře jsou k dispozici online dostupné návody, široká komunita vývojářů často se zapojujících do komunikace prostřednictvím fóra, kde je možné nalézt užitečné informace při získávání nových zkušeností při vývoji s tímto softwarem. Stejně jako u Unity 3D, i zde je poskytnut obchod s multimediálním obsahem. [49]

Ohledně licenčních podmínek, Unreal Engine je distribuován od roku 2015 zdarma, pod bezplatnou licencí EULA, s možností využití všech poskytovaných funkčností. Pouze v případě užití pro komerční účely, kdy vývojář prodává své projekty vyvinuté prostřednictvím tohoto nástroje, si společnost Epic Games účtuje 5 % poplatky z hrubého

výnosu nad 3 tis. $ za každý produkt za kalendářní čtvrtletí dle podmínek výše zmíněné licence. [52]

3.3 Existující aplikace

V rámci této kapitoly jsou stručně charakterizovány tři existující aplikace na obdobnou problematiku, jenž je tématem této diplomové práce. Při vytváření aplikace v rámci praktické části práce bude snaha dosáhnout obdobných kvalit s vyvarováním se nalezených nedostatků a snahou o poskytnutí intuitivnějšího a přehlednějšího grafického uživatelského rozhraní, jež by zapříčinilo větší ochotu uživatelů tuto aplikaci využívat.

3.3.1 Draw Bricks

Draw Bricks představuje jednu z existujících aplikací na obdobnou problematiku, jenž je tématem této diplomové práce. Jedná se o aplikaci určenou pro mobilní zařízení s operačním systémem Android, která je volně dostupná v rámci online distribuční služby zvané Google Play Store. [53]

Myšlenka aplikace spočívá v poskytnutí trojrozměrného pracovního prostoru uživateli, kde může popustit uzdu své fantazie a z předpřipravených dílků stavebnice sestavit libovolné složitější modely. Jedná se tak o editor stavebnice na způsob reálné stavebnice lego, který uživateli umožňuje stavbu připravených modelů v galerii aplikace, jak je znázorněno v části c) obrázku (Obrázek 3) nebo stavbu vlastních modelů prostřednictvím připravených kostiček v menu v obrázku část b). Uživatel si zde může vybírat z různých druhů kostiček, případně i speciálně otexturované pro lepší vizuální efekt. Samozřejmostí je také možnost změny barvy dané kostičky, její rotace, přesunutí a smazání. Poslední funkčností kromě možnosti sdílení obrázku výsledného modelu s jinými uživateli, je ruční ovládání vloženého charakteru s možností procházení sestavenou scénou dle obrázku část a).

Obrázek 3 - Draw Bricks rozhraní a scény
[53]

Tato aplikace je distribuována pro mobilní zařízení jako smartphone či tablet s operačním systémem Android verze 4.0 a vyšší a jedná se již o verzi aplikace 12.0, je tedy zřejmé, že autor neustále vyvíjí snahu o její inovování. Autorem je webový návrhář a vývojář mobilní aplikací Bruno Sousa, který se podílel například na tvorbě aplikace Icon Paint a mezi jím vyvinuté aplikace patří například Nutrihelper, My Picture Puzzle a PokeInfo [54]. Hodnocením aplikace spadá do evropského standardu PEGI 3, její obsah tedy vyhovuje pro všechny věkové skupiny a neohrožuje mravní výchovu mládeže [55]. Dle počtu uživatelských stažení a na základě převážně kladných recenzí lze říci, že se daná aplikace ve své kategorii řadí mezi úspěšnější.

3.3.2 VirtualBlock

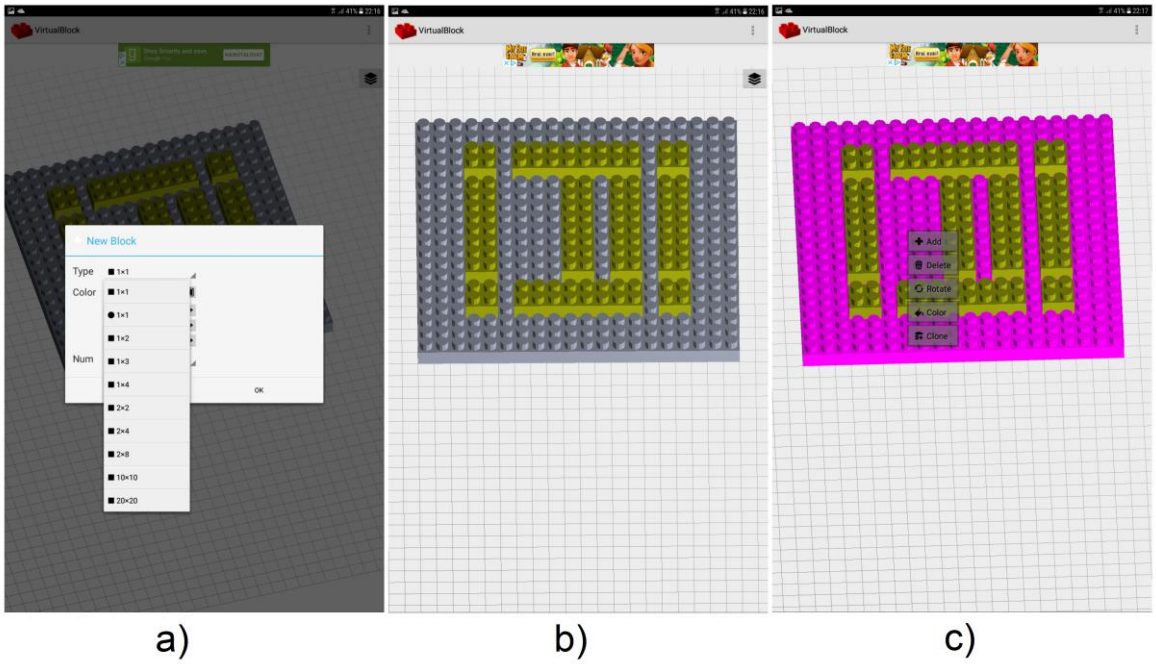
VirtualBlock reprezentuje další existující aplikaci pro mobilní zařízení distribuovanou, stejně jako předchozí zmíněnou, prostřednictvím online distribuční služby Google Play Store. [56]

Aplikace VirtualBlock je také zaměřena na koncepci editoru stavebnice lego s cílem podpořit kreativitu uživatelů tak, že mají k dispozici sadu definovaných kostiček,

se kterou mohou stavět libovolné modely. Funkčnost zahrnuje správu kostiček, kdy si uživatelé mohou vkládat nové kostičky do scény, následně s nimi manipulovat, přemisťovat, rotovat, měnit barvu a mazat je. Poslední funkčností je možnost zachycení snímku, který se ukládá do vnitřní paměti zařízení.

Aplikace obsahuje čtyři módy pro spuštění: demo, blank, local a network. Všechny tyto módy poskytují uživateli deset druhů kostiček rozlišených tvarem a rozměrem. Uživatel má možnost definovat si seznam kostiček, kdy z těchto deseti druhů jeden vybere, nastaví počet kusů a finální barvu, tím se mu přidá tento zadaný počet kostiček do seznamu a může se na ně rychleji odvolat, nemusí je opětovně vytvářet jednu po druhé. Takto je možné definovat i různé počty různě obarvených, ale typově stejných kostiček. Jediný mód demo má tento seznam již předpřipravený, v ostatních módech je to od začátku plně v kompetenci uživatele.

Uživatelské rozhraní pro výběr kostičky reprezentuje část a) obrázku (Obrázek 4), zde se jedná o mód local. Část b) představuje scénu editoru s neoznačenou kostičkou, část c) ukazuje rozbalené rozhraní pro manipulaci s kostičkami.



Obrázek 4 - VirtualBlock rozhraní
[56]

Aplikace je distribuována pro mobilní zařízení s operačním systémem Android verze 2.3 a vyšší. Verze aplikace je 2.7, ovšem poslední aktualizací prošla z kraje roku 2017, vývoj byl tedy zpomalen, ne -li pozastaven. VirtualBlock vyvinula společnost NULL

Product, mezi jejíž distribuované aplikace patří Sticky Camera a ImageReader. Hodnocením aplikace spadá taktéž do standardu PEGI 3, jako tomu je u Draw Bricks. S ohledem na počet uživatelských stažení a především na spíše záporně formulované recenze uživatelů se dá konstatovat, že aplikace VirtualBloack se řadí spíše mezi méně úspěšné aplikace ve své kategorii. [56] Způsobeno je to nejspíše špatně navrženým rozhraním, a s tím spojenou horší ovladatelností.

3.3.3 LEGO Digital Designer

Posledním charakterizovaným programem, který se zabývá touto problematikou, je program LEGO Digital Designer od společnosti LEGO Group. V tomto případě se nejedná o aplikaci pro mobilní zařízení, ale editor stavebnice navržený pro stolní počítače s operačním systémem Windows nebo pro MAC s OS X. [57]

Původním záměrem bylo nabídnout uživatelům možnost navrhnut si vlastní model, který mohl být následně uveden do výroby a dodán danému zájemci ke koupi v jejich vlastním LEGO balení, v tzv. programu DESIGN byME. Tento program měl nebývalý úspěch a každoročně se do něj zapojilo na několik milionů lidí, jež si využitím nástroje LEGO Digital Designer navrhovali pestrou škálu vlastních modelů, ovšem v důsledku snahy o dodržování standardů kvality byla tato služba na začátku roku 2012 pozastavena. Přesto byl nástroj dále vyvíjen a uživatelé jej mohou stále využívat pro navrhování nových modelů, které mohou umisťovat do svých veřejných galerií na stránkách LEGO Digital Designer. [58]

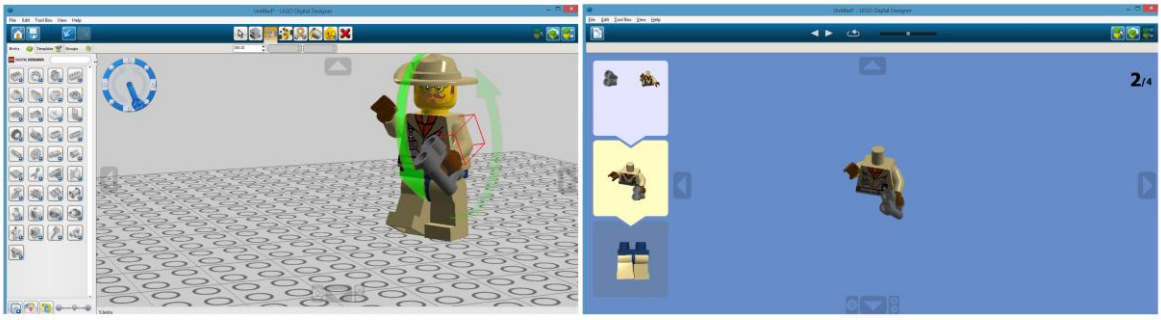
Jedná se o vysoce propracovaným editor stavebnice LEGO umožňující uživatelům volbu mezi třemi režimy chodu, které si vybírá hned na úvodní obrazovce. Prvním je LEGO Digital Designer, kdy má uživatel k dispozici pestrou škálu připravených kostiček rozličných druhů rozdělených do 37 kategorií. Tyto kostičky může libovolně umísťovat do pracovního prostoru a vytvářet tak složitější modely nejrůznějších tvarů. V tomto případě má ovšem pevně definované barvy, což ale není příliš omezující problém, neboť každá kostička, nejedná -li se o příliš specifický tvar, je připravena průměrně v 10 až 20 různých barevných odstínech, které mohou být i po umístění do pracovního prostoru vzájemně nahrazovány. [59, s. 2]

Druhý režim je tzv. LEGO MINDSTORMS, který je zaměřený na práci s kostičkami pro tvorbu robotů. Zde jsou opět pevně definovány barvy jednotlivých kostek,

jejichž počet je oproti předchozímu režimu značně zredukovaný, jak počet kostiček dle tvaru, tak i počet možných barev. [59, s. 2]

Poslední režim je LEGO Digital Designer EXTENDED, který poskytuje možnost změny barvy každé kostičky, kdy si uživatel může vybrat ze 78 definovaných odstínů včetně průhledných. Nevýhodou je, že u specificky vypadajících kostiček, které jsou za normálních okolností tvořeny více barvami, dochází k jejich reprezentaci užitím jediné barvy. Tento režim ovšem obsahuje i kostičky, které se v předchozích nevyskytují. [59, s. 2]

Každý tento režim poskytuje uživateli stejnou funkčnost v rámci vytváření modelů. Jedná se samozřejmě o správu kostiček, přidávání, přesouvání, mazání, rotace, změna barvy či prohození s jinak barevně definovanou stejného typu. Dále se zde nachází nástroje pro výběr dle různých kritérií, výběr dle barvy, dle tvaru, dle barvy a tvaru společně, výběr po jedné či více kostičkách a nakonec výběr všech provázaných kostiček. Poslední funkcí sloužící pro nastavení kostiček ve scéně je tzv. hinge tool, neboli nástroj pro úpravu pohyblivých prvků znázorněný v části a) obrázku (Obrázek 5). Často využívanou funkcí programu je i možnost vykreslení podrobného návodu na sestavení daného modelu, jež reprezentuje část b) obrázku, tím je tak zajištěna možnost stavby navrženého modelu i v reálném světě prostřednictvím skutečné stavebnice. [59, s. 5-10]



Obrázek 5 - LEGO Digital Designer rozhraní, návod
[57]

4 Vlastní práce

Tato kapitola reprezentuje praktickou část diplomové práce. Zabývá se popisem jednotlivých kroků napříč celým vývojovým cyklem v rámci něhož je vytvářena výsledná interaktivní aplikace reprezentující virtuální editor dětské stavebnice.

Kapitola je rozdělena do pěti částí, kdy první část je věnována definování cílové skupiny uživatelů, požadavků na vytvářenou aplikace a volbě vhodných nástrojů využitých pro její vývoj. Volba těchto nástrojů probíhá vzájemným porovnání nástrojů na základě vlastností popsaných v rámci provedených analýz v teoretické části a s ohledem na kladené požadavky na výslednou aplikaci.

Druhá část je věnována návrhu grafického uživatelského rozhraní použitého v jednotlivých scénách.

Další dvě části se zabývají tvorbou prvků multimediálního obsahu aplikace. V prvním případě se jedná o tvorbu textur použitých na ovládací prvky rozhraní či na objekty ve scéně, dále trojrozměrné modely částí stavebnice a dalších objektů. Druhý případ reprezentuje vývoj aplikační logiky pomocí její implementace v rámci skriptů založených na objektově orientovaném programovacím paradigmatu.

Poslední část je věnována otestování výsledné aplikace za dobrovolné účasti zástupců definované cílové skupiny uživatelů, spolu s vyhodnocením případných nálezů.

4.1 Cílová skupina, požadavky na aplikaci a výběr nástrojů

Tato kapitola je zaměřena na definování a zjištění podstatných informací, které je potřeba znát při každé tvorbě nového softwaru, ať už se jedná o informační systémy pro firmy či školy, nebo o aplikaci určenou veřejnosti, která primárně slouží pro zábavné vyplnění volného času.

Prvotní úkol představuje definovat účel, pro nějž je aplikace vyvíjena, s tím také souvisí definování cílové skupinu uživatelů, kteří budou danou aplikaci či software využívat. Této problematice je věnována první část této kapitoly.

Ve druhé části jsou specifikovány požadavky na vyvíjenou aplikaci, které zahrnují požadavky na funkcionalitu, jenž si zpravidla určují zadavatelé projektu, a požadavky na licenci, výkon, zátěž a další, které ovlivňují způsob nakládání s aplikací, a také mají vliv na minimální potřebnou konfiguraci koncových zařízení, na nichž bude aplikace provozována.

Poslední část této kapitole je následně věnována výběru vhodných nástrojů, jenž jsou použity při samotné tvorbě interaktivní aplikace. Nástroje jsou vybrány metodou komparace neboli porovnání na základě provedených analýz v rámci teoretické části diplomové práce, a také s ohledem na definované požadavky.

4.1.1 Cílová skupina

S ohledem na tématiku aplikace, která reprezentuje editor dětské stavebnice v trojrozměrném virtuálním prostoru, je jejím zamýšleným účelem poskytnout prostředí, ve kterém by mohli uživatelé přetvářet své nápady a myšlenky do trojrozměrných modelů sestavených z částí této stavebnice. Aplikace jim tak umožňuje poddat se své fantazii, objevit a rozvíjet svého tvořivého ducha, být kreativní.

Cílovou skupinu uživatelů této aplikace tvoří primárně děti předškolního věku a žáci prvního stupně základní školy. U těchto skupin dětí je potřeba podporovat jejich tvořivost a napomáhat rozvoji jejich skrytých talentů, jakožto potenciálních malých umělců. Využití by ovšem mohla najít bez rozdílu u každého, kdo má zájem zkoušet něco navrhovat. Především rodiče malých dětí bývají často zapojováni do takovýchto aktivit, kdy jsou otcové žádáni o vymýšlení nových modelů různých vozidel či staveb pro své nejmenší a jejich následné realizaci prostřednictvím skutečných stavebnic.

Pro všechny takovéto uživatele tvoří tato interaktivní aplikace vhodný prostředek, jak své nápady realizovat téměř v hmatatelné podobě, od vizuálního modelu je to už jen krůček ke skutečné realizaci.

4.1.2 Funkční a obecné požadavky

V této části kapitoly jsou specifikovány požadavky na výslednou interaktivní aplikaci. Obecně platí, že jeden požadavek by měl zachycovat pouze jedno měřitelné očekávání zadavatele na vytvářený software. Vhodnou pomůckou je například, že každý požadavek by měl být tzv. SMART. Mělo by být specifikováno (Specific), co přesně se od výsledného softwaru očekává. Dále by měla existovat možnost, jak zkontrolovat, že toho bylo dosaženo, měřitelnost požadavku (Measurable). Žádný požadavek by neměl být kladen jako nedosažitelný cíl (Achievable). Měl by být relevantní (Relevant) a mělo by být definováno jeho časové ohraničení, kdy je třeba ho realizovat (Time-bound). Obdobný způsob kategorizace požadavků je například také FURPS (Functionality - funkčnost,

Usability - použitelnost, Reliability - spolehlivost, Performance - výkonnost, Supportability - podpora, udržovatelnost).

Požadavky lze poté rozdělit do dvou druhů, obecných a funkčních.

4.1.2.1 Obecné požadavky

Obecné požadavky se netýkají přímo poskytovaných funkcionalit, ale spíše kladou různá omezení na vytvářený software. Může se jednat o požadavky na prioritní vlastnosti výsledné aplikace, jež mohou zahrnovat například požadavek na nízkou paměťovou náročnost, přenositelnost. Dále požadavky na to, jakým způsobem probíhá řešení dané problematiky, kam spadají například předepsané standardy a jaký programovací jazyk má být pro vývoj využit. A nakonec tzv. vnější požadavky, mezi které patří například doba řešení, podmínky dodání, cenová omezení a testy. [60, s. 11, 12 (originální s. 98, 99)]

O.1: Návrh grafického uživatelského rozhraní pro děti

Při návrhu grafického uživatelského rozhraní je potřeba dbát na přehlednost a snadnost porozumění, neboť cílovou skupinu uživatelů tvoří převážně děti předškolního věku a děti z prvního stupně základní školy, jak bylo definováno v předchozí kapitole. U těchto uživatelů platí předpoklad, že se jedná o slabší čtenáře či dokonce nemusí umět číst vůbec, proto je zásadní sdělit nejdůležitější informace prostřednictvím, v dnešní době zaběhlých, grafických ikon, doplněných textovými popisky, je -li to nutné. Ovládání aplikace by tak mělo být založeno na obdobném ovládání podobných existujících aplikací. Interní nápovědu dotyčným malým uživatelům případně vysvětlí starší člen rodiny.

Taktéž je nezbytné interaktivní aplikaci ztvárnit tak, aby reprezentovala příjemné grafické prostředí, jenž upoutá dětskou pozornost.

O.2: Využití objektově orientovaného programování

S ohledem na popsanou skutečnost v rámci kapitoly 3.2.2 Principy objektově orientovaného programování, že v současné době je nejvíce využíváno objektově orientovaného programování, jakožto programovacího paradigmatu, pro tvorbu přehledných, strukturovaných a efektivních programů, je i v rámci této diplomové práce vyžadováno, aby byla aplikační logika implementována právě prostřednictvím objektově orientovaného programování. Oba potenciální softwary vybrané pro tvorbu aplikace, herní enginy, jež byly analyzovány v rámci teoretické části v kapitolách 3.2.3 Unity 3D

a 3.2.4 Unreal Engine, podporují skriptování prostřednictvím objektově orientovaného programovacího jazyku C#.

O.3: Volně šiřitelná aplikace

Distribuce této interaktivní aplikace probíhá volnou, neplacenou formou. Bylo by možné a nejefektivnější využít v budoucnu online distribuční služby Google Play Store, jež byla zmíněna již v kapitole 3.3 Existující aplikace, a která slouží primárně pro distribuci aplikací pro mobilní zařízení s operačním systémem Android, placených i dostupných zdarma.

O.4: Možnost rozšíření aplikace

Na výslednou interaktivní aplikaci je taktéž kladen požadavek, že je možné ji s nevelkými zásahy do zdrojového kódu rozšířit o nové prvky. Ty reprezentují například nové kostičky v kategorii či celé nové kategorie.

O.5: Podporovaná platforma

Aplikace je navržena pro mobilní zařízení, smartphone či tablet, s operačním systémem Android minimální verze 4.1, a neměla by být příliš výpočetní a paměťově náročná pro zajištění hladkého běhu i na méně výkonných zařízeních.

O.6: Testování s uživateli

Výsledná interaktivní aplikace je otestována s uživateli, jenž test podstoupí dobrovolně a anonymně. Během testu je plněno několik přesně definovaných úkolů, jejichž cílem je určit nejasné či nepřehledné rozhraní aplikace.

Testování probíhá na dvou mobilních zařízení, zástupců smartphonů a tabletů, s konfigurací odpovídající doporučené konfiguraci pro čistý běh aplikace.

4.1.2.2 Funkční požadavky

Funkční požadavky se zabývají přímo konkrétními funkcionalitami výsledné interaktivní aplikace, specifikují tedy přání zadavatele, co musí daný software poskytovat koncovému uživateli. [60, s. 6, 7 (originální s. 93, 94)]

F.1: Uložení modelu do paměti zařízení

Jeden z nejdůležitějších funkčních požadavků reprezentuje možnost uložení modelu do paměti zařízení, aby bylo možné se k jeho úpravě později vrátit. Případně je tím umožněno přesunutí modelu z jednoho zařízení do druhého, kde může být následně načten a upraven, za předpokladu, že druhé zařízení taktéž disponuje touto aplikací. Navíc by

nehrozila ztráta dat při vypnutí aplikace, neboť to by bylo spojeno s nutností vždy začínat od "nuly", což vede k velmi nepraktické a neefektivní výsledné aplikaci, jež se jistě v dnešním světě nemůže uchytit.

F.2: Načtení modelu z paměti zařízení

Druhým funkčním požadavkem, úzce souvisejícím z předchozím, je možnost načtení uloženého modelu z paměti zařízení, aby bylo možné provést případné úpravy či si model jen prohlédnout. Může se jednat i o načtení modelu, který byl původně vytvořen na jiném koncovém zařízení a do tohoto pouze přesunut, ovšem k jehož tvorbě byla ale nutně využita tato interaktivní aplikace.

F.3: Správa kostiček

S ohledem na skutečnost, že daná aplikace reprezentuje editor dětské stavebnice, musí být uživateli poskytnuta možnost správy jednotlivých částí této stavebnice, konkrétně různých kostiček, aby vůbec mohl sestavit či upravit nějaký model. Tento funkční požadavek lze rozdělit do několika dílčích požadavků:

F.3.1: Vložení nové kostičky

Jedná se o možnost přidání nové kostičky do prostoru scény poté, co byla uživatelem zvolena z poskytnutého seznamu, jež je reprezentován posuvným menu. Kostička musí být následně přemístěna pohybem do nějakého místa ve scéně, kdy tento pohyb je možné ovládat současně ve všech třech osách trojdimenzionálního prostoru. Taktéž je během tohoto přesunu uživateli umožněno kostičkou otáčet kolem její osy po devadesáti stupňových úhlech. Kostička musí být přidána do vyhrazeného pracovního prostoru, aniž by se protínala s jinou již existující a umístěnou ve scéně. Pokud je porušena některá z těchto dvou podmínek, kostička nebude do scény zařazena.

F.3.2: Výběr existující kostičky

Po zapnutí možnosti "Výběr kostiček" je umožněno uživateli vybírat pouhým dotykem mezi existujícími kostičkami umístěnými ve scéně. Tato funkcionalita je stěžejní pro pozdější možnosti změny barev jednotlivých kostiček, či jejich mazání a přesouvání. Doporučení, po skončení práce s výběrem je vhodné zrušit volbu "Výběr kostiček", aby nevznikal rušivý efekt při otáčení modelu, kdy okamžitě po dotyku s některou z kostiček, je uživateli nabídnuto její přemístění či smazání.

F.3.3: Přesun existující kostičky

Předpokladem pro tuto funkcionalitu je proběhnutý výběr již existující kostičky, kterou uživatel zamýšlí přemístit. Následný pohyb ve scéně je totožný jako při vkládání nové kostičky, taktéž i pravidla pro úspěšné zpětné vložení této kostičky, kdy se nesmí protínat s jinou již existující a zároveň musí být umístěna ve vyhrazeném prostoru.

F.3.4: Smazání libovolné existující kostičky

Taktéž pro tuto funkcionalitu je nezbytný výběr jedné kostičky z již existujících, kdy je uživateli zobrazena nabídka s možností přesunu či smazání. Při zvolení smazání bude tato kostička nevratně odstraněna ze scény.

F.3.5: Smazání poslední přidané kostičky

Jedná se funkci, ke které není nutno specifikovat prostřednictvím ručního výběru, kterou kostičku si uživatel přeje smazat. Po zvolení této možnosti bude automaticky odstraněna poslední řádně přidaná kostička ve scéně. Úkon lze opakovat až do vymazání všech existujících kostiček ve scéně ve zpětném pořadí, než v jakém byly do scény přidávány.

F.3.6: Změna barvy kostičky

Uživateli je umožněno změnit barvu kostičky dvěma způsoby, kdy jeden představuje nastavení "globální" barvy, jíž budou nabývat všechny nově vkládané kostičky do scény. To se automaticky nastaví po výběru barvy, v případě kdy není vybrána žádná z již existujících kostiček ve scéně. Pokud však došlo k výběru jedné existující kostičky a následně byla provedena změna barvy, jedná se tak o druhý způsob, kdy se změní pouze barva této vybrané kostičky a ostatní nově vkládané budou nabývat předchozí nastavené barvy.

F.4: Správa pohybu po scéně

Nezbytnou funkcionalitou je taktéž pohyb ve scéně, aby bylo uživateli umožněno prohlížet vytvářený model ze všech možných úhlů. I tento požadavek může být rozdělen na dílčí požadavky:

F.4.1: Otáčení kamery

V případě této interaktivní aplikace nedochází k otáčení samotného modelu, ale rotaci kamery okolo něj. Uživateli je umožněno, prostřednictvím interakce s displejem mobilního zařízení jedním prstem, otáčet kamerou v požadovaném směru s cílem dosažení lepšího úhlu pohledu.

F.4.2: Přiblížení kamery

Interakce s displejem mobilního zařízení dvěma prsty zajišťuje možnost přibližování a oddalování kamery k danému modelu s cílem poskytnutí přesnějšího pohledu na jeho část, například pro zajištění lepší pozice pro výběr kostičky. Přibližování i oddalování je možné provádět, dokud není dosaženo nastavených hranic.

F.4.3: Přesun kamery do stran

Jedná se o funkcionalitu zajišťující posun rotačního bodu kamery pro získání detailnějšího pohledu na některou část modelu. Tento posun uživatel způsobí prostřednictvím pohybu tří prstů po displeji. Taktéž jsou zde nastaveny hranice, za které není možné rotační bod přemístit.

F.4.4: Návrat kamery do výchozí pozice

Funkcionalita, jenž usnadňuje návrat do původního pohledu na sestavovaný model, iniciovaná dotykem čtyř prstů na displej. To zajistí postupný návrat rotačního bodu kamery do výchozí pozice, a také zpětné natočení kamery samotné.

F.5: Zobrazení detailu kostičky a půdorysu scény

Tato funkcionalita uživateli zpřehledňuje správu kostiček. Aplikace obsahuje volitelně zobrazitelné okno s náhledem z jiné kamery. Zde je při pohybu kostičky ve scéně zobrazen půdorys dané scény, aby byla uživateli umožněna lepší orientace ve všech osách trojdimenzionálního prostoru. V případě že uživatel zrovna neprovádí přesun nějaké kostičky, je v tomto okně zobrazen pohled z jiné kamery zajišťující detailní pohled na kostičku, jejíž ovládací prvek je nastaven na středu posuvného menu, aby si uživatel mohl prohlédnout přesný tvar kostičky před jejím samotným vložením.

F.6: Vykreslení návodu

Další neméně důležitá požadovaná funkcionalita reprezentuje možnost vykreslení sestaveného modelu ve formě návodu na jeho stavbu, jenž se uloží ve formě sekvence navazujících obrázků do paměti zařízení. To uživateli dává možnost sestavení jeho originální stavby prostřednictvím skutečné stavebnice.

F.7: Možnost uživatelského nastavení

Poslední požadovaná funkcionalita představuje možnost nastavení přezdívky uživatele a změnu vzhledu avatara, neboli maskota aplikace dle uživatelských preferencí.

4.1.3 Výběr vhodných nástrojů pro tvorbu

Tato kapitola se věnuje výběru vhodných softwarových nástrojů, jež budou následně využity pro tvorbu výsledné interaktivní aplikace, jakožto editoru dětské stavebnice. Výběr samotný je zde proveden metodou komparace vlastností dvou zvolených grafických modelovacích nástrojů a dvou herních enginů s ohledem také na požadavky na aplikaci, jenž byly v předchozí kapitole definovány. Z tohoto důvodu lze tuto kapitolu rozdělit na dvě části:

4.1.3.1 Výběr grafického modelovacího nástroje

První část se zaměřuje na výběr jednoho ze dvou vybraných grafických modelovacích nástrojů, kdy vhodnější z nich bude využit pro následnou tvorbu části multimediálního obsahu navrhované interaktivní aplikace, konkrétně pro tvorbu trojrozměrných modelů jednotlivých částí stavebnice, kostiček.

Vybrány byly komerční nástroj Maya od společnosti Autodesk a open-source software Blender od nadace Blender Foundation, které byly v rámci teoretické části v kapitolách 3.1.5 Maya 2016 a 3.1.6 Blender 2.79 zanalyzovány. Výstup těchto dvou kapitol zde slouží jako podklad pro provedení vzájemné komparace těchto nástrojů a zvolení vhodnějšího z nich pro nastávající tvorbu aplikace.

Průběh samotného srovnání reprezentuje tabulka (Tabulka 1), do níž byla zvolena kritéria vlastností, jež by mohly být využity v rámci tvorby trojrozměrných modelů pro aplikaci reprezentující grafický editor dětské stavebnice. Jedná se zde o často využívané a pro tuto tvorbu vhodné modelovací metody, způsoby animace a dále obecné vlastnosti vztahující se ke spolupráci těchto nástrojů se softwarem pro vývoj počítačových her. Je tak brán ohled i na proces exportování trojrozměrných modelů a jejich přesun do herních enginů. Další důležité vlastnosti představuje dostupnost online dokumentace softwaru a návodů pro práci a případná široká komunita uživatelů. V poslední řadě tvoří důležitá kritéria také podpora pro operační systém Windows, na kterém běží zařízení sloužící k vývoji, dále dostupnost těchto grafických nástrojů a možnost případného komerčního užití vytvořeného obsahu.

Grafické modelovací nástroje			
Název		Maya 2016	Blender 2.79
Animace	**Klíčování**	✓	✓
	Inverzní kinematika	✓	✓
Dostupnost online dokumentace		✓	✓
Dostupnost zdarma		30 -ti denní trial verze či tříletá studentská verze	✓
Export přes FBX	**Podpora FBX**	✓	✓
	Nutnost ručně exportovat a importovat	✗	✓
Komerční využití vytvořených modelů		✗	✓
Modelování	**Polygonální modelování**	✓	✓
	Modelování pomocí NURBS	✓	✓
	Dělení povrchů	✓	✓
Návody na oficiálních stránkách zdarma		✓	✓
Podpora pro Windows		✓	✓
Široká komunita uživatelů		✓	✓
Unity 3D	**Přímé napojení projektu**	✓	✗
	Odeslání objektu přímo do projektu	✓	✗
Unreal Engine	**Přímé napojení projektu**	✓	✗
	Odeslání objektu přímo do projektu	✓	✗
Zvolený software		✓	✗

Tabulka 1 - Komparace dvou grafických modelovacích nástrojů

Z tabulky (Tabulka 1) vyplývá, že jako vhodnější nástroj je zvolen komerční software Maya od společnosti Autodesk i přesto, že studentská verze softwaru s tříletou licencí neumožňuje komerční využití vytvořených modelů. Oba nástroje splňují požadavky pro tvorbu interaktivní aplikace odpovídající rozsahu a tématu této diplomové práce a za předpokladu nekomerčního šíření vyvíjené aplikace se jako vhodnější modelovací nástroj jeví Maya se svou vlastností přímého napojení na oba vybrané herní enginy. Díky tomu je zajištěn následný rychlejší a snazší přenos vytvářeného trojrozměrného obsahu do příslušného enginu, ať už bude v následující části zvolen kterýkoliv z nich.

4.1.3.2 Výběr herního enginu

Tato část je věnována výběru vhodnějšího ze dvou zvolených herních enginů pro následné sestavení vytvářené aplikace, konkrétně Unity 5.6.5 od společnosti Unity Technologies a Unreal Engine 4 od společnosti Epic Games. Výběr je taktéž proveden metodou komparace vlastností popsaných v rámci vypracovaných analýz těchto softwarů

v teoretické části práce v kapitolách 3.2.3 Unity 3D a 3.2.4 Unreal Engine. Další kritérium pro výběr tvoří požadavky na výslednou aplikace definované v předchozí kapitole.

Vybraný herní engine bude následně využit při finální tvorbě aplikace reprezentující editor dětské stavebnice. V tomto zvoleném herním enginu budou vytvořeny jednotlivé scény, do nichž budou umístěny všechny prvky multimediálního obsahu, jako jsou textury, modely. Bude zde nastaveno osvětlení a vytvořeno grafické uživatelské rozhraní. V poslední řadě zde budou přidány skripty zajišťující aplikační logiku a správné fungování interaktivní aplikace.

Průběh srovnání dvou zvolených softwarů pro vývoj počítačových her a mobilních aplikací reprezentuje tabulka (Tabulka 2), v níž jsou zvoleny důležité vlastnosti softwaru pro tvorbu aplikace s ohledem na definované požadavky jakožto kritéria srovnání. Z požadavků se jedná především o důraz na využití objektově orientovaného programovacího jazyku pro implementaci aplikační logiky, s cílem dosažení přehledného a efektivního zdrojového kódu. Dále jsou vybrány poskytované vlastnosti softwarů, jenž jsou potřebné pro úspěšný vývoj dané interaktivní aplikace. Sem náleží editor pro tvorbu grafického uživatelského rozhraní, fyzikální engine, vykreslovací engine umožňující tvorbu 3D aplikací, podpora běhu na zařízení s operačním systémem Windows, které je využito v rámci vývoje, možnost skriptování za použití objektově orientovaného programovacího paradigmatu a podpora pro vývoj aplikací pro platformu Android. K obecným vlastnostem zvoleným jako kritéria pro srovnání náleží především dostupnost vývojového softwaru zdarma, časté vydávání nových verzí rozšiřujících a optimalizujících poskytované funkčnosti, široká komunita uživatelů, dostupná a kvalitně zpracovaná dokumentace jako podpora při vývoji, možnost komerčního využití vytvořeného produktu a pozice na trhu herních enginů, která poskytuje jistou záruku, že se jedná o kvalitní software pro vývoj aplikací, a jehož vydavatelé usilují o neustálé inovace.

Herní enginy			
Název		Unity 5.6.5	Unreal Engine 4
Časté vydávání nových verzí		✓	✗
Dostupnost zdarma		✓	✓
Editor grafického uživatelského rozhraní		✓	✓
Fyzikální engine		✓	✓
Komerční užití výsledného produktu		(Personal) až do výše zisku 100 tis. $ za rok	nutno platit 5% ze zisku nad 3 tis. $ za každý produkt za rok
Podpora pro Windows		✓	✓
Podrobná dokumentace		✓	✓
Skriptování - OOP	C#	✓	✗
	C++	✗	✓
Široká komunita uživatelů		✓	✓
Tvorba 3D grafických aplikací		✓	✓
Tvorba aplikací pro Android		✓	✓
Vedoucí pozice na trhu		✓	2. v pořadí
Vykreslovací engine		✓	✓
Zvolený software		✓	✗

Tabulka 2 - Komparace dvou herních enginů

Z tabulky (Tabulka 2) vyplývá, že pro následnou tvorbu výsledné aplikace je zvolen software Unity od společnosti Unity Technologies, který disponuje podporou pro skriptování prostřednictvím čistě objektového programovacího jazyku C#. Také se jedná, dle zanalyzovaných statistických údajů zmíněných v teoretické části práce, o držitele vedoucího postavení na trhu s herními enginy, a tedy kvalitní software pro vývoj využívaný i profesionálními firmami zabývajícími se vývojem počítačových her. Výrobce taktéž vydává časté nové verze, téměř každoročně, je tedy zajištěno rozšiřování a optimalizace poskytnutých funkčností. Navíc i verze zdarma umožňuje případné komerční využití výsledného produktu až do značné částky za předpokladu, že by poskytnutý obsah vytvořený komerčním nástrojem Maya byl v budoucnu předělán například prostřednictvím open-source softwaru Blender. S tímto využitím ovšem není v rámci diplomové práce prozatím počítáno.

4.2 Návrh grafického uživatelského rozhraní

Kapitola je věnována jedné z nejdůležitějších částí vývoje interaktivní aplikace, přesněji návrhu grafického uživatelského rozhraní (GUI, Graphical User Interface), jež je základním prostředkem umožňujícím interakci mezi uživatelem a aplikací.

Grafické uživatelské prostředí reprezentuje ovládací prvek této aplikace, a jeho návrh pak představuje část procesu, kdy se vývojář musí umět vžít do role koncového uživatele jakožto reprezentanta cílové skupiny uživatelů, pro které je daná aplikace navrhována. Musí být schopen uvažovat v rovině člověka nezainteresovaného do vývoje, tedy člověka, který vidí až finální produkt, a který musí být schopen tomuto produktu snadno porozumět a orientovat se v něm. V případě, kdy je definovaná cílová skupina uživatelů tvořena převážně malými dětmi, je toto vcítění se a následný návrh ještě o něco obtížnějším procesem, neboť upoutat a především si udržet dětskou pozornost bývá velmi náročné. Navíc je pravděpodobné, že většina z těchto uživatelů prozatím neovládá čtení, to vede k nutnosti vyjádřit důležité informace spíše grafickou formou než čistým textem. Může se jednat například o text doplněný ikonkami, či různá barevná odlišení jednotlivých prvků.

Správně navržené grafické prostředí by mělo být nejen přehledné a snadno pochopitelné, ale také zapamatovatelné, aby nehrozilo, že po delší době nevyužívání dané aplikace se musí uživatel opět učit, jak s ní zacházet. Všechny takové nedostatky mohou vést ke snížení prožitku z využívání takové aplikace a podněcují uživatele, aby se poohlédl po jiné, lépe zpracované.

V rámci této diplomové práce je pro návrh grafického uživatelského rozhraní využit online nástroj zvaný Fluid, který lze využít právě pro tvorbu tzv. wireframů neboli předběžných nákresů či skic ovládacího rozhraní aplikace, které reprezentují rozmístění jednotlivých ovládacích prvků zobrazených na displeji koncového mobilního zařízení, na němž je aplikace spuštěna. Software Fluid představuje nástroj poskytující možnost současné spolupráce několika lidí na vytvářeném prototypu, umožňuje tvorbu návrhů pro mobilní zařízení s operačním systémem Android, iOS, Windows, i pro stolní počítače a zařízení jako jsou chytré hodinky. Vzhledem ke skutečnosti, že projekt je uložený v cloudu, je k němu umožněn přístup z libovolného místa online a je poskytnuto předpřipravených prvků, ze kterých snadnou manipulací vznikají finální díla, s možností importu vlastních obrázků. [61] V neposlední řadě nástroj umožňuje přidat do vytvářených

prototypů i možnost interakce prostřednictvím klikání myši či dotykem na mobilních zařízení, což jen pozvedne výsledný dojem a lépe ukáže plánovanou funkcionalitu. [62]

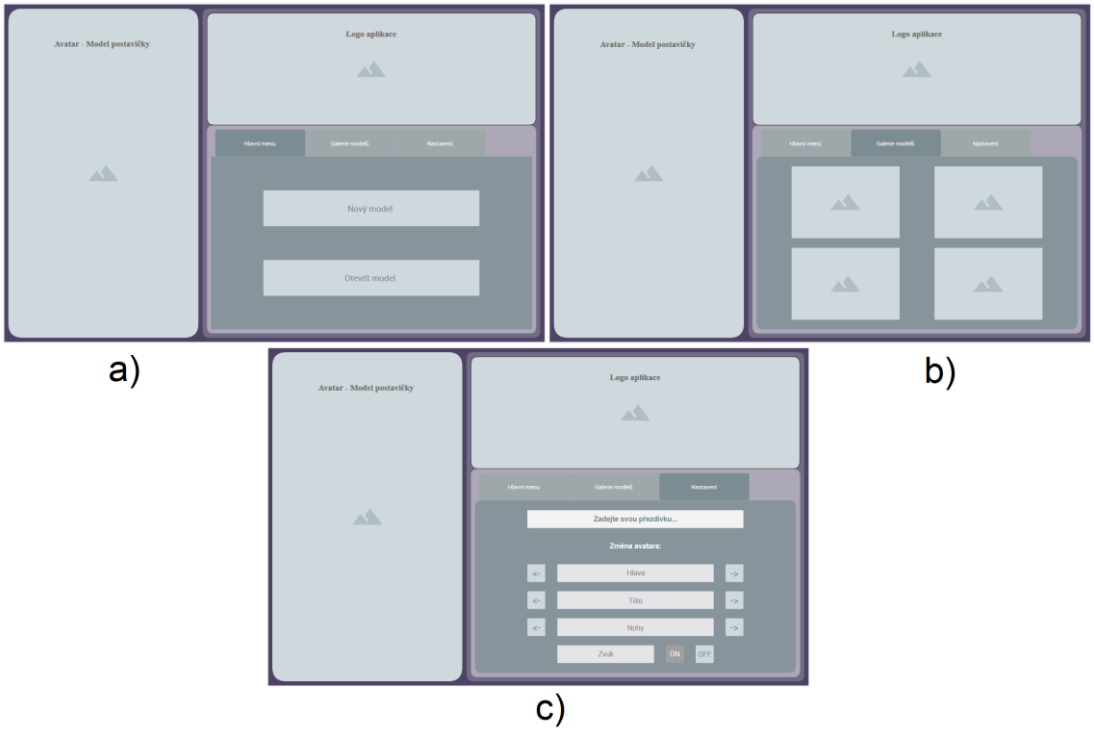
Nástroj Fluid je využit v této diplomové práci pro tvorbu prototypů grafického uživatelského rozhraní reprezentující jednotlivá zobrazení napříč oběma plánovanými scénami výsledné aplikace. Jedná se tak o ztvárnění všech obrazovek v rámci scény s hlavním menu aplikace a následně i obrazovku s rozhraním užitým ve scéně představující samotný editor. Obě tyto scény jsou podrobněji popsány v rámci následujících podkapitol.

4.2.1 Hlavní menu aplikace

V této kapitole je vytvořen návrh grafického uživatelského rozhraní pro scénu s hlavním menu výsledné aplikace, které sestává z pěti obrazovek. Jedná se o prototypy zabývající se čistě rozvržením struktury rozhraní, rozmístěním prvků, jejich popisy jsou pouze ilustrační a není zde nijak řešena grafická stránka, která je podrobněji popsána a vyobrazena v rámci kapitoly 4.3 Tvorba multimediálního obsahu.

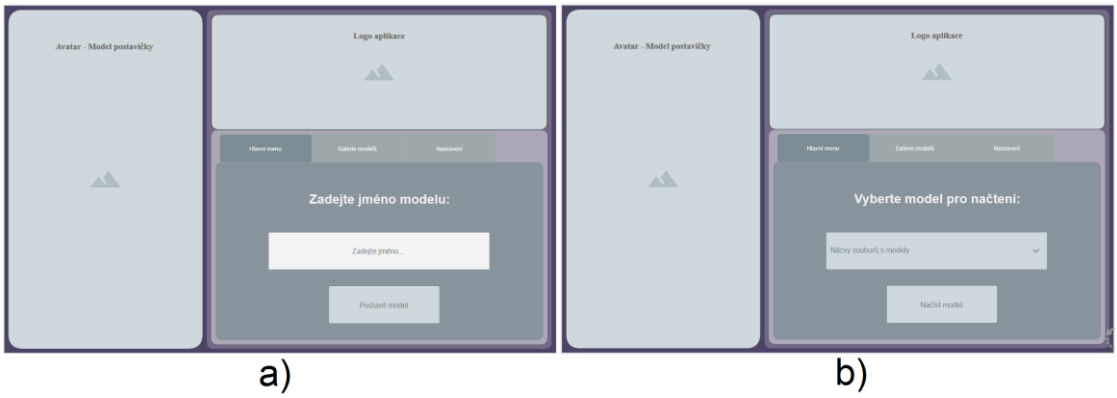
Každá obrazovka scény s hlavním menu aplikace sestává ze tří částí, které jsou graficky rozeznatelné na následujících obrázcích. Jedna část reprezentuje obrázek s logem aplikace, druhá část má pouze orientační význam a vyhrazuje prázdné místo, ve kterém je viditelný trojrozměrný model maskota aplikace umístěný v prostoru scény. Poslední část tvoří stěžejní prostor hlavního menu, který osahuje všechny jednotlivé panely jimiž je menu tvořeno.

Obrázek (Obrázek 6) reprezentuje vzhled tří stěžejních položek hlavního menu, konkrétně se jedná o přední stranu (hlavní menu), galerii poskytnutých modelů a panel nastavení aplikace. Galerie poskytnutých modelů je zobrazena v části b) obrázku a je tvořena obrázky daných modelů sloužících jako tlačítka pro jejich načtení. Zde se jedná o modely uložené v rámci aplikace, které nejsou vytvořeny samotným uživatelem, naopak jsou mu poskytnuty pro bližší prozkoumání a předvedení, co vše lze s aplikací vytvořit. Panel s možností změny nastavení je znázorněn na obrázku v části c). V rámci nastavení je uživateli umožněno definování přezdívky a změna vzhledu maskota aplikace. Přední část menu je poté vyobrazena v části a) obrázku a tvoří ji dvě tlačítka sloužící pro otevření zbylých dvou panelů vyobrazených na obrázku (Obrázek 7).



Obrázek 6 - Návrh GUI, hlavní menu

Obrázek (Obrázek 7) reprezentuje prototypy uživatelského rozhraní s panely pro vytvoření nového modelu a načtení existujícího modelu. V části a) obrázku je znázorněn panel sloužící pro vytvoření nového modelu s následným načtením scény čistého editoru. Panel je tvořen vstupním polem, do kterého je uživatel vyzván zadat název modelu, který zároveň slouží jako budoucí název souboru pro uložení vytvářeného modelu. Část b) představuje panel pro načtení existujícího modelu uloženého v paměti zařízení a obsahuje drop-down menu (rozbalovací nabídku), která je vyplněna všemi soubory s vytvořenými modely uloženými v paměti zařízení.



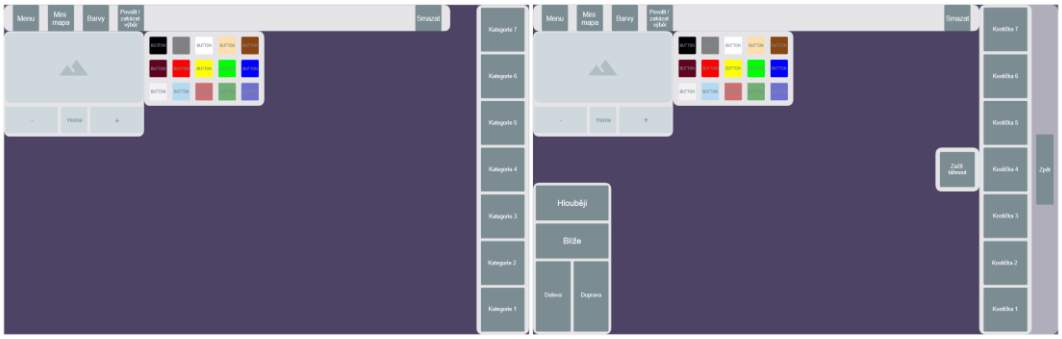
Obrázek 7 - Návrh GUI, nový model, načíst model

Tímto je zastoupena veškerá navrhovaná funkcionalita hlavního menu, konkrétní grafická stránka je řešena v rámci kapitoly o tvorbě multimediálního obsahu aplikace.

4.2.2 Editor, pracovní plocha

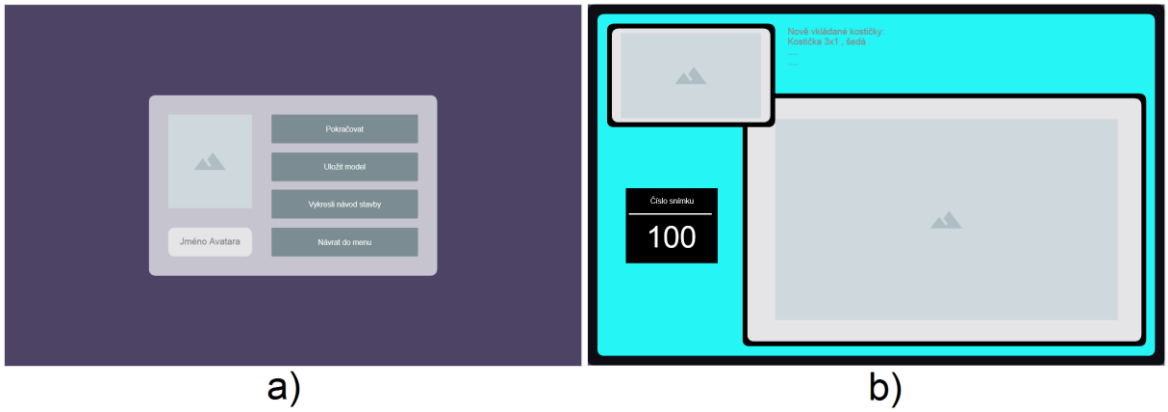
Tato kapitola se zaměřuje na návrh grafického uživatelského rozhraní pro scénu představující samotný editor. Jsou zde zobrazeny čtyři návrhy reprezentující jednotlivá rozložení ovládacích prvků v rámci této scény. Prototypy taktéž slouží pouze pro orientační rozmístění jednotlivých ovládacích prvků, jejichž grafický vzhled je tvořen v následující kapitole.

Obrázek (Obrázek 8) znázorňuje základní uživatelské rozhraní v rámci editoru. Část a) obrázku promítá základní vzhled, kdy barevná paleta a panel reprezentující mini mapu, která zajišťuje pohled shora, lze volitelně nastavovat jako viditelné či skryté. Horní lišta je viditelná vždy a obsahuje ovládací prvky pro zobrazení těchto dvou menších panelů, dále pro pozastavení aplikace, smazání poslední přidané kostičky a umožnění výběru jednotlivých kostiček, jež byly již umístěny do prostoru scény. Panel vpravo v části a) obrázku reprezentuje posuvný pás kategorií, do nichž jsou připravené kostičky, jakožto stavební prvky, roztříděny. V části b) obrázku je vyobrazena situace, kdy je vkládána nová kostička do pracovního prostoru. Posuvný pás kategorií je schován a místo něj je zobrazen posuvný pás kostiček spadajících do této kategorie. Po výběru jedné z těchto poskytovaných kostek se zobrazí samostatné tlačítko, jež slouží k počátku přemístění nově vkládané kostičky. Jakmile započne uživatel na tomto tlačítku táhnout prstem, kostička je automaticky posouvána v daném směru. Taktéž se nově objevil panel vlevo, který zajišťuje možnost otáčení vkládané kostičky okolo její osy a posun této kostky ve třetí ose trojdimenzionálního prostoru.



Obrázek 8 - Návrh GUI, editor

Obrázek (Obrázek 9) tvoří zbylé dvě možné obrazovky v rámci scény editoru. Část a) obrázku znázorňuje pozastavení aplikace a zobrazení menu, kde je uživateli umožněno uložit vytvářený model do paměti mobilního zařízení, přejít zpět do hlavního menu interaktivní aplikace, pokračovat ve stavbě modelu a v poslední řadě nechat si vykreslit podrobný návod, jak byl daný model sestaven. Kdy je tento návod uložen taktéž v paměti zařízení jako sekvence navazujících snímků. Část b) obrázku definuje vzhled právě jednoho snímku z vykreslovaného návodu, který obsahuje pro uživatele důležité informace, aby mohl například provést rekonstrukci stavby vytvořeného modelu pomocí skutečné stavebnice. Snímek obsahuje dva obrázky z různých pohledů na model, podrobný popis aktuálně vkládaných kostek a číslo snímku kvůli návaznosti.



Obrázek 9 - Návrh GUI, editor - menu, návod

Tímto je návrh grafického uživatelského rozhraní pro obě scény interaktivní aplikace kompletní a navržené prototypy slouží jako podklad pro následnou práci v herním enginu, kde musí být toto rozhraní stvořeno do finální podoby spolu s implementovanou funkčností.

4.3 Tvorba multimediálního obsahu aplikace

Tato kapitola popisuje postup tvorby multimediálního obsahu v podobě trojrozměrných modelů a textur využitých ve výsledné interaktivní aplikaci. Nejsou zde zahrnuty skripty řešící aplikační logiku, přestože je lze taktéž zařadit mezi prvky multimediálního obsahu, ty jsou popsány v následující kapitole 4.4 Implementace aplikační logiky.

Kapitole je rozčleněna do dvou částí, kdy první část se týká tvorby pozadí použitého v rámci scény s hlavním menu a druhá část se zabývá tvorbou jednotlivých trojrozměrných modelů, jež jsou využity v obou scénách aplikace.

4.3.1 Tvorba pozadí

Zde je popsána tvorba obrázků pro pozadí pomocí nástroje pro tvorbu a editaci vektorové grafiky zvaného Inkscape. Obrázky jsou tedy vytvořeny pomocí křivek zadaných v tomto nástroji a jsou uloženy ve formátu SVG (Scalable Vector Graphics). Následně jsou převedeny do rastrové podoby, což umožní jejich přiřazení jednotlivým prvkům v rámci scény vytvářené v herním enginu Unity a následné vykreslení na koncovém mobilním zařízení, na němž je aplikace spuštěna.

Přesněji se jedná o tvorbu obrázků reprezentujících nebe a krajinu, které doplňují celkový vzhled aplikace. Snahou je docílit animovaného vzhledu, který by pozitivně zapůsobil na představitele cílové skupiny uživatelů.

Postup tvorby spočívá v základním nastavení plátna na rozměry dle používaných v rámci výsledné aplikace, jedná se tak o rozměr 1920 x 1200 pixelů. Při tvorbě nebe je na pozadí využito barevného přechodu, získaného použitím nástroje Výplň a obrys, kde je zvolena volba lineárního přechodu. Zde jsou nastaveny počáteční a koncová barva, například bílá či bleděmodrá a sytější modř. Následně je pomocí ovládacího prvku nastaven směr přechodu. V tomto případě se již nejedná o čistě vektorovou grafiku, neboť ta umožňuje vybarvit objekt pouze užitím jediného barevného odstínu nastaveného v celé ploše objektu. Tento barevný přechod se již řadí mezi rastrové efekty. Následně jsou do obrázku přidány další vrstvy, ve kterých budou umístěny jednotlivé mraky, které se tak mohou vzájemně překrývat.

Tvorba jednotlivých mraků spočívá v prvotním definování jejich obrysu pomocí křivek prostřednictvím nástrojem pro kresbu Beziérových křivek. Poté co je obrys nakreslený, je opětovně využito barevného lineárního přechodu pro jejich obarvení. Stejný princip je použit i na jejich obrys, kdy jej lze použít částečně v jiném směru, aby mraky ve výsledku na pozadí trochu vynikly. Lze také vytvořit v jejich objemu další neúplné křivky, které výsledku dodají trochu plastický tvar, případně mrak lehce zprůhlednit či rozostřit. Takto jsou vytvořeny tři druhy mraků, které jsou vhodně rozmístěny na pozadí v rámci několika vrstev s cílem dosažení tzv. bezešvého obrázku, kdy je možné jej v případě potřeby opakovaně promítat za sebou, aniž by byl vidět rušivý přechod mezi jednotlivými obrazy.

Obdobným způsobem je vytvořena i krajina, která sestává z jednotlivých nakreslených kopců s porostem, jež se v několika vrstvách nad sebou částečně překrývají a vytváří dojem vzdáleného horizontu.

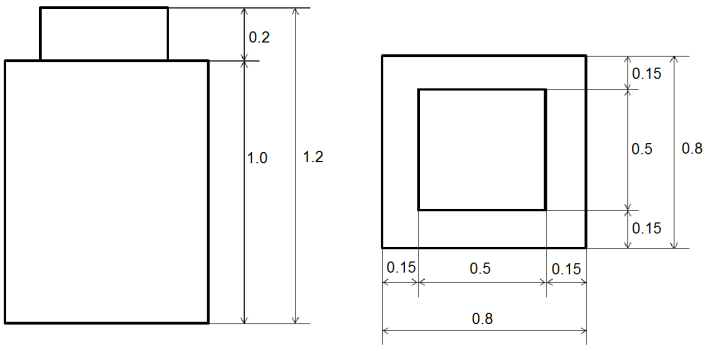
Posledním krokem tvorby je exportování výsledného obrazu do rastrového formátu, například PNG (Portable Network Graphics).

4.3.2 Tvorba trojrozměrných modelů

Pro tvorbu trojrozměrných modelů je využit grafický modelovací nástroj Maya, který byl v rámci kapitoly 4.1.3.1 Výběr grafického modelovacího nástroje vybrán jako vhodnější ze dvou navrhovaných nástrojů. Za pomoci tohoto nástroje jsou vymodelovány jednotlivé prvky stavebnice, maskot aplikace, blízké okolní prostředí a tlačítka, kdy všechny tyto modely jsou reprezentovány pouze hraniční reprezentací.

V první řadě je nástroj použit pro tvorbu trojrozměrných modelů jednotlivých částí stavebnice, tedy kostiček. Základní rozměry jsou vyobrazeny na obrázku (Obrázek 10) a násobí se s počtem požadovaných výstupků na vršku kostičky. V případě tedy, že je požadována kostička se šesti výrůstky ve dvou řadách po třech, rozměr 3 x 2, tak její velikost by byla 2.4 x 1.6 mřížkových jednotek v rámci nástroje Maya. Každou kostičku reprezentuje zastřešující objekt tzv. group s potomky, jež jsou tělo kostičky a každý jednotlivý výstupek. Kostička na obrázku je tedy tvořena:

- Brick_1x1 - groupovací objekt,
    - BrickBody_1x1 - tělo kostičky,
    - BrickCleat_1_1x1 - výrůstek kostičky.



Obrázek 10 - Nákres základní kostičky

Pro účely aplikace je vytvořeno 63 různých modelů kostiček lišících se nejen velikostí co do počtu výstupků, ale i tvary, prozatím existují vysoké, nízké a šikmé.

Kostičky jsou navrženy tak, aby sestávaly z co možná nejmenšího počtu polygonů, aby jejich zpracování nebylo výpočetně náročné a aplikace běžela v rozumných mezích i na méně výkonných koncových zařízeních.

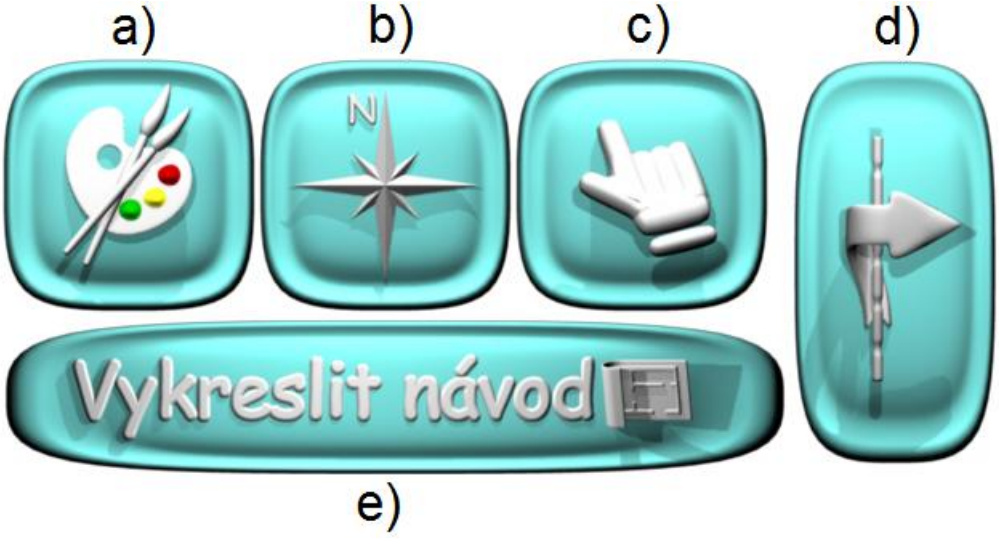
Při jejich tvorbě je využíváno především logických operací představujících sjednocení a rozdíl. Do vytvořeného těla kostičky požadovaných rozměrů je pomocí funkce rozdíl vytvořen příslušný počet děr pro výrůstky jiných kostiček, které by mohly být umístěny pod touto. Následně jsou rozmístěny všechny výrůstky a pomocí nástroje pro sjednocení prvků pod rodičovským objektem jsou spojeny do výsledného celku. Rodičovský objekt zde představuje prázdný objekt bez geometrie zastřešující všechny části příslušné kostičky.

Takto vytvořenému modelu je přiřazen materiál reprezentující lesklý plast. Software Maya k tomu využívá materiál typu Phong E, který je vhodný pro lehce lesklé povrchy, jako je například plast a kůže. V rámci vlastností daného materiálu se nastaví příslušná barva, průhlednost, odrazivost světla či další parametry. S ohledem na zvolený typ materiálu je dále také možné nastavit vlastnosti se právě samotné odrazivosti světla, zde lze určit hrubost povrchu, zrcadlení a velikost a jas odlesku. Pro zvolenou reprezentaci plastové stavebnice jsou nastaveny parametry tak, aby byl vytvořen hladký plast s trochou lesku. Textury zde nejsou na kostičky nanášeny.

Další vytvořený model pomocí metody polygonálního modelování představuje statický model maskota aplikace sestávající z několika samostatných částí, u nichž je uživateli v rámci výsledné aplikace umožněno měnit jejich materiály s texturami. Tím se docílí několika možných vzhledů daného modelu.

V rámci tohoto modelovacího nástroje jsou také vytvořeny textury pro ovládací prvky grafického uživatelského rozhraní, aby budily dojem plastičnosti, jak je vyobrazeno na obrázku (Obrázek 11). Jedná se o vymodelované objekty sloužící jako ikony pro netextové ovládací prvky či grafické doplnění textových, kdy tento text je vytvořený v softwaru Maya funkcí Create Text ve formátu Comic Sans MS, kdy je zvolen typ Bevel neboli zkosený a styl konvexní směrem ven. Tyto texty a modely ikon jsou umístěny do rámečku vzniklého metodou dělení povrchů pro jemnější zaoblení a následně vykresleny pomocí Maya Software a uloženy ve formátu PNG. Ve vrchní části obrázku je zobrazen vzhled tlačítek pro a) paletu barev, b) panel s mini mapou, c) funkci výběru, d) otočení kostičky vpravo. Část e) obrázku reprezentuje vzhled tlačítka z menu jakožto textového

ovládacího prvku doplněného o grafickou ikonku pro snazší pochopení pro uživatele bez schopnosti čtení.



Obrázek 11 - Textury pro tlačítka

Posledním trojrozměrným modelem je část krajiny zobrazená v rámci scény s hlavním menu, do které je umístěn statický model maskota, a která je následně doplněna o vzdálené textury zbytku krajiny a nebe vytvořených v rámci předchozí kapitoly.

4.4 Implementace aplikační logiky

Tato kapitola se věnuje již samotnému sestavení výsledné aplikace ze všech doposud vytvořených komponent multimediálního obsahu, prostřednictvím užití vybraného herního enginu Unity 5.6.5 od společnosti Unity Technologies.

Jedná se o proces tvorby, kdy jsou vytvořeny dvě základní scény, které tvoří logické uspořádání interaktivní aplikace. Do těchto scén jsou zařazeny všechny potřebné modely a textury, je zde nastaveno osvětlení a různé kamery promítající výsledný obraz. Dále jsou zde převedeny prototypy grafického uživatelského rozhraní z kapitoly 4.2 Návrh grafického uživatelského rozhraní do výsledné podoby, prostřednictvím nástroje pro tvorbu rozhraní poskytovaného v rámci vývojového softwaru Unity. Tato podoba rozhraní je oproti vytvořeným prototypům obohacena o detailní grafické ztvárnění jednotlivých ovládacích prvků.

Poslední fázi tohoto procesu výroby zastupuje vytvoření zbylé části multimediálního obsahu interaktivní aplikace, kterou znázorňují skripty, jež implementují aplikační logiku a jejich umístění do obou scén spolu s provázáním s jednotlivými prvky

v těchto scénách. Všechny skripty jsou napsány pomocí objektově orientovaného programovacího jazyku C# v rámci poskytovaného integrovaného vývojového prostředí (IDE) zvaného MonoDevelop. To představuje snadno ovladatelný textový editor obohacený o funkčnosti zajištující vývojáři možnost ladění kódu, a také poskytuje nápovědu při tvorbě implementace. [43] Pro zobrazování chybových zpráv ohlašujících problémy se syntaxí kódu či informující například o nevyužívaných proměnných slouží konzole, jež představuje jedno z volitelných oken v rámci rozhraní softwaru Unity. Zde mohou být vývojářem zobrazeny i hlášky, které mu usnadňují ladění vytvářeného kódu. [63]

Kapitola samotná je rozdělena do tří částí, kdy první z těchto částí popisuje základní koncepci tvorby skriptů v rámci vývojového softwaru Unity a zbylé dvě části jsou věnovány popisu jednotlivých vytvořených scén interaktivní aplikace. V rámci nich je uveden seznam přidružených skriptů s jejich základním popisem užití a s následně uvedenými částmi kódu reprezentujícími důležité metody implementující požadované funkcionality aplikace.

4.4.1 Skripty v Unity

Každý skript, soubor, představuje právě jednu třídu objektů. Může se jednat o pomocné třídy reprezentující datové struktury či statickou třídu pro předávání důležitých hodnot, kdy tyto skripty nejsou přímo vkládány do scény, ale jejichž instance jsou využívány v rámci jiných skriptů, nejedná -li se o statickou třídu, od které nelze instanci vytvořit. Druhou možnost představují třídy jakožto potomci třídy MonoBehaviour, ty jsou přímo vkládány do scény a mohou být vázány na konkrétní objekty. Zde se ručně nevytvářejí jejich instance pomocí konstruktorů, ale jsou vytvořeny právě připojením skriptu k objektu ve scéně. To představuje jakousi komponentu s možností ovládání a ovlivňování připojeného herního objektu, kdy vytvoření instance má v režii editor. [43]

Třída MonoBehaviour slouží jako základní třída poskytující funkce pro kontrolu a interakci s herními objekty ve scéně. Kromě této vnitřně implementované třídy jsou poskytnuty ještě další dvě, konkrétně Transform a RigidBody. Třída Transform se v základu využívá k přístupu k informacím o poloze a natočení objektů, tedy jejich souřadnicím, a také poskytuje metody pro manipulaci s nimi. Poslední důležitou a často nezbytně využívanou třídou je RigidBody, která poskytuje možnost využívání funkcí

fyzikálního enginu, které zajišťují simulaci fyzikálních jevů jako působení síly na objekt, tažení, gravitaci či detekci kolizí. [64]

U druhého typu tříd, jakožto potomků MonoBehaviour existují tři důležité metody, jejichž prostřednictvím je skript prováděn. Jedná se o metodu Start, jenž je volána po spuštění aplikace, a je tedy využívána pro inicializaci používaných proměnných v rámci skriptu. Druhou metodou je Update, která se volá v každém snímku neboli framu běžící aplikace. Zde bývá uveden kód, který se má provádět a slouží k manipulaci s herními objekty. Často se jedná o implementovanou reakci na nějaký vstup od uživatele například v podobě stisku klávesy. Třetí poskytovanou metodou je Awake, která je volána pouze jednou za životnost skriptu předtím, než jsou volány jednotlivé metody Start, a využívá se především pro inicializaci proměnných ještě před spuštěním kódu. To umožňuje následné předávání referencí mezi skripty navzájem, například v rámci metody Start, bez vzniku závažných chyb. Právě zavedení konstruktorů v těchto třídách, aby bylo možné vytvořit jejich instanci a data předávat skrze ní, by mělo za následek pouze chybové chování, proto jsou využívány pomocné třídy, které nedědí chování z třídy MonoBehaviour nebo metoda Awake, která danou inicializaci provede jednou na počátku a nehrozí pak vzájemné překrytí v rámci postupného volání metod Start.

Kromě těchto tří metod jsou zde také další, které zpravidla fungují jako reakce na nastalou událost, například OnCollisionEnter, OnCollisionExit, OnMouseOver a spousta dalších, které jsou volány v situacích, kdy došlo k těmto specifikovaným událostem.

4.4.2 Hlavní menu

Tato kapitola se týká samotné tvorby prvotní scény, která je zobrazena uživateli po zapnutí aplikace. Jedná se o scénu reprezentující hlavní menu interaktivní aplikace, které uživateli poskytuje několik funkčností, mezi něž náleží změna osobního nastavení, načtení modelu z galerie modelů a spuštění editoru pro započetí stavění nového modelu.

Scéna obsahuje několik komponent, jež ji charakterizují. V první řadě se jedná o statický trojrozměrný model maskota aplikace, reprezentujícího postavičku ze stavebnice, zasazeného do okolního prostředí, jež je v základu taktéž tvořeno trojrozměrným modelem doplněným o vzdálené pozadí. Dále je zde nastaveno osvětlení pomocí směrového světla a reflektoru nastavených na požadované hodnoty tak, aby scéna působila příjemným dojmem. Posledním prvkem umístěným ve scéně je prázdný objekt,

který slouží jako jednotné místo pro umístění či přichycení všech skriptů potřebných pro správný chod této scény.

Účel tohoto prázdného objektu spočívá v přehlednější reprezentaci skriptů ve scéně, které tak nejsou rozházeny napříč celou scénou, kdy by každý samostatný skript byl přichycen k jiného hernímu objektu, ale všechny jsou obsaženy v jediném místě pro jejich snadnější správu. Ve výsledné aplikaci není tento prázdný objekt vykreslen.

Poslední a nejdůležitější část této scény tvoří samotné grafické uživatelské rozhraní sestávající z několika samostatných panelů s ovládacími prvky. V horní části, která zůstává viditelná po celou dobu interakce s menu, se nachází logo aplikace. Ve spodní části jsou naopak zobrazovány a skrývány jednotlivé panely, které znázorňují nabídku volitelného uživatelského nastavení, galerii modelů, rozhraní pro vytvoření nového modelu, či rozhraní pro načtení modelu uloženého v paměti mobilního zařízení. Posledním součástí menu je samostatný panel sloužící pro zobrazování zpráv uživateli. Do tohoto panelu jsou v případě potřeby zobrazovány zprávy, jako například informace, že zadané jméno souboru již existuje a zda si uživatel přeje soubor přepsat. Tento panel má časové nastavení a v případě, že se zobrazí, tak je spuštěn dvou vteřinový odpočet, kdy je poté následně opět zprůhledněn.

Správná funkčnost této scény spočívá ve spolupráci čtyř skriptů, dvou statických tříd ConfigFile a InnerStoringScript, třídy jedináčka neboli singletonu ModelSaveLoadScript, a jednoho potomka třídy MonoBehaviour.

Z tohoto výčtu vyplývá, že jediný poslední zmíněný skript lze přímo umístit do prostoru scény a připnout k připravenému prázdnému objektu. Skript poté zajišťuje správný chod menu, které si lze představit jako přepínání záložek s kartami. Po stisku jednoho ze tří viditelných tlačítek dochází ke zviditelnění příslušného panelu představujícího danou část menu a ke zprůhlednění zbylých.

K jednotlivým tlačítkům umístěným v rozhraní lze jednoduše, pomocí poskytovaného editoru pro tvorbu rozhraní v rámci herního enginu Unity, přidružit veřejnou metodu, která je implementována v tomto skriptu, SwitchingPanelsByTabs. Každé tlačítko totiž obsahuje skript, který zajišťuje činnost tzv. odposlouchávače událostí, a který zajišťuje relevantní akci, jež je provedena v závislosti na zachycení této události. Primárně se jedná o událost stisku tohoto tlačítka, lze ovšem využít i jiné, například najetí kurzoru na tlačítko nebo jeho opuštění.

Veřejná metoda, jež je volána při stisku daného tlačítka, představuje již konkrétní implementaci napsanou vývojářem této interaktivní aplikace. V rámci hlavního menu se jedná o nastavení viditelnosti jednotlivých panelů a změnu barvy příslušné záložky. Samotná inicializace těchto panelů proběhla v rámci metody Start s využitím funkce pro nalezení herního objektu ve scéně dle jména. Pro ukázku, panel pro zobrazení hlášky uživateli se nazývá LogLabelPanel, chce -li ho následně vývojář využít v rámci skriptu, napíše kód (Zdrojový kód 1) do metody Start.

```
GameObject logLabelPanel = GameObject.Find("LogLabelPanel");
```

Zdrojový kód 1 - Inicializace herního objektu

GameObject představuje datový typ herního objektu, funkce Find slouží k nalezení herního objektu dle předaného textového řetězce, který v tomto kódu reprezentuje jméno hledaného panelu. Tímto způsobem je možné inicializovat soukromé proměnné. Veřejné proměnné umožňují ruční přetažení objektu v rámci editoru Unity do příslušného políčka vyhrazeného skriptem, mají ovšem za následek, že jsou viditelné a měnitelné i mimo rámec tohoto objektu, není zde tedy zajištěno zapouzdření popsané v kapitole 3.2.2.1 Zapouzdření.

ConfigFile skript zajišťuje uložení uživatelských preferencí, je tím tak reprezentován konfigurační soubor, kam se uloží informace jako nastavená přezdívka či indexy částí, ze kterých je sestaven model maskota aplikace.

InnerStoringScript reprezentuje statickou třídu pro předávání informací napříč scénami. Jedná se o třídu s několika proměnnými, které určují, zda je možné otáčet kamerou ve scéně, zda dochází k načítání modelu z paměti zařízení či stavbě nového a dalších.

ModelSaveLoadScript představuje statickou třídu pro uchování informací týkajících se ukládání modelu do paměti či jeho načítání. Taktéž obsahuje funkce pro práci s těmito soubory, pro zápis a čtení.

Kód (Zdrojový kód 2) znázorňuje jednu z implementovaných metod zajišťujících spuštění scény editoru v režimu načítání modelu z paměti zařízení, která je součástí skriptu SwitchnigPanelsByTabs, a která je volána po stisku tlačítka Načíst model.

```
public void clickOnLoadModelButton(){
  string filesName = names [openModelDropdown.value];
  modelSaveLoadScriptInstance.setFileName (filesName);
  if (modelSaveLoadScriptInstance.fileExists ()) {
    InnerStoringScript.setIsLoading (true);
```

```
    setStaticVariables ();
    SceneManager.LoadScene ("EditorScene");
  } else {
      setLogLabel ("Daný soubor se nepodařilo otevřít. Soubor neexistuje
nebo je porušený!", false);
  }
}
```
Zdrojový kód 2 - Metoda pro spuštění editoru v režimu načítání modelu

Metoda vezme z rozbalovací nabídky vybranou hodnotu reprezentující jméno souboru s modelem a toto jméno uloží do proměnné v rámci statické třídy. Dále zkontroluje, zda takovýto soubor existuje, pokud ne, tak vypíše uživateli hlášku o jeho nenalezení, jinak pokud existuje, nastaví příznak načítání modelu a spustí scénu editoru.

4.4.3 Editor

Scéna editoru je po grafické stránce na první pohled prázdná. Obsahuje osvětlovací prvky, sadu kamer pro různé účely a řadu prázdných objektů, kdy ani jedna z těchto částí není ve výsledku vykreslována. Navíc neobsahuje žádné pozadí, neboť to by v dané aplikaci působilo pouze rušivým dojmem. Pro práci s editorem, kdy uživatel umisťuje malé modely kostiček do pracovního prostoru je efektivnější využít jednobarevné pozadí, aby nedocházelo ke splývání kostiček se zátiším.

Prázdné objekty zde plní několik funkcí. Jeden reprezentuje opět objekt pro všechny skripty ve scéně, další nadřazený prvek kamery, se kterým je manipulováno k dosažení efektu otáčení kamery okolo vytvářeného modelu. Poslední slouží pro definování pozice pro detailní pohled na kostičku zobrazitelný v okně mini mapy ještě před samotným vložením kostičky do scény.

Stěžejní část zde tvoří grafické uživatelské rozhraní vycházející z prototypu navrženému v rámci kapitoly 4.2.2 Editor, pracovní plocha, kde se nachází také jeho podrobnější popis.

V rámci této scény jsou využity skripty reprezentující statickou třídu jako u hlavního menu InnerStoringScript, dále třída dle návrhového vzoru zvaného jediináček ModelSaveLoadScript, dvě pomocné třídy definující datovou strukturu octree MyOctree a MyOctreeNode. A šest tříd jakožto potomků třídy MonoBehaviour, BrickSelectionScript, CreateObjectScript, DataSaveLoadScript, DetailOfPartRotationScript, ScrollRectScript a TouchCameraControlScript.

Datová struktura octree je v rámci aplikace využita pro efektivnější ukládání jednotlivých částí vytvářeného modelu, pro jejich méně výpočetně náročné zpracování při vkládání nových kostiček do scény, kdy nemusí být prováděna kontrola kolize vkládané kostičky se všemi již existujícími v celé scéně, ale pouze s těmi ve stejném vyhrazeném podprostoru. Jak bylo popsáno v kapitole 3.1.2.2 Objemová reprezentace, struktura octree představuje hierarchický stromový graf, kde každý uzel, který není koncovým, má osm potomků a právě jednoho předka, až na kořen stromu. Octree definuje základní pracovní prostor ve tvaru krychle, do které je uživateli umožněno vkládat nové objekty.

Princip funkčnosti zajišťuje, že je -li do tohoto vymezeného prostoru vložen objekt, dojde k jeho rozdělení na osm stejných podprostorů, dělí se na poloviny v každé ose. Poté se provede kontrola, zda se objekt protíná s více než jedním podprostorem daného prostoru. Pokud se protíná s více podprostory, tak je objekt ponechán na úrovni nadřazeného uzlu stromu, pokud se protíná právě s jedním a nebylo dosaženo maximálního dělení, tak se postoupí do tohoto podprostoru a opětovně se provede celý proces dělení na menší podprostory.

Třída MyOctree je implementovaná dle návrhové vzoru jedináček, kdy je zajištěno, že při každém vytváření nové instance této třídy, je vrácena jedna a ta samá, jinými slovy od dané třídy je možno vytvořit právě jednu instanci, která je přiřazeno do všech referencí datového typu MyOctree napříč aplikací. Vždy se tedy pracuje s jedinou datovou strukturou, aby bylo zajištěno ukládání nových kostiček do téhož stromu.

Třída MyOctreeNode poté reprezentuje jednotlivé uzly daného stromu. Každý tento uzel udržuje informace o do něj přiřazených herních objektech v upravené podobě na tzv. vnitřní objekty OctreeNode, které představují dvojice hodnot, herní objekt a jeho obalový objekt, v podobě seznamu. Dále udržuje odkaz na svého předchůdce a seznam odkazů na osm potomků. Seznam objektů každého uzlu obsahuje pouze takové herní objekty, které jsou plně obsaženy v rámci jeho vyhraněného prostoru, jejichž obalový objekt se neprotíná s jiným prostorem.

Strom je vytvářen až v případě, kdy je nezbytné zanoření s daným objektem do nižších podprostorů. To způsobuje, že se může jednat o nevyvážený strom, kdy některý z potomků kořenu stromu nemusí být dále dělen, neboť neobsahuje objekty, které by protínaly jeho sousední prvky. Naopak při odstraňování prvků ze scény, a tedy i z tohoto

stromu, nedochází ke smazání prázdného podprostoru z důvodu ušetření režie potřebné k jeho případnému opětovnému vytvoření při vkládání nového objektu.

Kód (Zdrojový kód 3) ukazuje implementaci metody sloužící pro vložení nového objektu s jeho obalovým objektem do stromu. Metodě je předán vkládaný objekt a jeho obalový objekt reprezentovaný objektem typu Bounds. Samotná metoda využívá dvou dalších pomocných metod, z nichž je zde uvedena implementace pouze jedné, kdy ta následně využívá ještě dalších dvou pomocných metod. První podmínka plní funkci ukončení postupného zanořování ve stromu, kdy je pomocí metody intersectsWithObjectsInNode zjištěno, zda se vkládaný objekt neprotíná již s některým uloženým v rámci tohoto prostoru. Pokud ano, metoda vrátí hodnotu false a objekt nemůže být vložen z důvodu nalezené kolize již s jiným existujícím. V případě, že se neprotíná s žádným jiným objektem v prostoru a prostor nelze dále dělit, je tento objekt uložen v rámci něj do seznamu a úspěšně přidán do scény. Pokud lze prostor dále dělit, je zavolána pomocná metoda subAddBrickToOctree, která zjistí s kolika potomky se objekt protíná, pokud se jedná o jediného potomka, je celý cyklus proveden opětovně na úrovni tohoto potomka. V případě více potomků se ověří, zda se kostička neprotíná s jinou již existující v rámci celého podstromu tohoto uzlu, kdy pokud se neprotíná, tak je úspěšně přidána a uložena v seznamu tohoto nadřazeného prvku. Pokud se protíná, tak ji nelze přidat do scény.

```
public bool addBrickToOctree(GameObject brick, Bounds bounds){
  if (intersectsWithObjectsInNode (bounds)) {
    return false;
  } else {
    if (sizeOfBounds > minSize) {
      return subAddBrickToOctree (brick, bounds);
    } else {
      objectsInNode.Add(new OctreeObjects(brick, bounds));
      return true;
    }
  }
}
private bool subAddBrickToOctree(GameObject brick, Bounds bounds){
  getIntersectedChildren (bounds);
  if (countOfIntersectedChildren == 1) {
    return      children[indexOfIntersectedSpace].addBrickToOctree(brick,
bounds);
  } else {
    if (!checkIntersectsWithObjectsInEachOfChildren (bounds)) {
      return false;
    } else {
```

```
      objectsInNode.Add (new OctreeObjects (brick, bounds));
      return true;
    }
  }
}
```
Zdrojový kód 3 - Metody pro přidání nového objektu do octree

Další zajímavější metodu z tohoto skriptu představuje kontrola kolize objektu s paprskem (Zdrojový kód 4), která slouží k nalezení kostičky při zapnuté funkci výběru. Pokud se paprsek protíná s daným ohraničením vyhrazeného prostoru, tak je užitím pomocné checkCollisionsWithObjectsInNode zjištěno, zda se paprsek protíná s nějakou kostičkou. Pokud ano, je nastavena do proměnné v rámci statické třídy, spolu s jejím obalovým objektem, kde se na ní může dotázat jiný script. Pokud k protnutí nedošlo, tak se zjistí, zda daný prostor má potomky v podobě podprostoru, kdy by se celý proces opakoval pro každý tento podprostor, pouze pokud v jeho sourozencích nebyla prozatím nalezena shoda.

```
public bool checkCollisions(Ray ray){
  bool condition = false;
  if (boundsOfNode.IntersectRay (ray)) {
    if (checkCollisionsWithObjectsInNode (ray)) {
      return true;
    } else {
      if (childrenAreSet) {
        for (int i = 0; i < children.Length; i++) {
          if (!condition) {
            condition = children [i].checkCollisions (ray);
          }
        }
      } else {
        return false;
      }
    }
  }
  return condition;
}
private bool checkCollisionsWithObjectsInNode(Ray ray){
  for (int i = 0; i < objectsInNode.Count; i++) {
    if (objectsInNode [i].getBoundsOfThisBrick ().IntersectRay (ray)) {
      InnerStoringScript.setSelectedBrick
(objectsInNode [i].getBrickInModel ());
      InnerStoringScript.setSelectedBrickBounds
(objectsInNode [i].getBoundsOfThisBrick ());
      return true;
    }
  }
```

```
  return false;
}
```
Zdrojový kód 4 - Kontrola kolize s paprskem

Za správnou funkčnost celé scény editoru je zodpovědná poměrně rozsáhlá implementace, kdy není možné ji celou uvádět zde. Z tohoto důvodu následující text využívá především slovního popisu implementace s případnými ukázkami zajímavějších metod. Prozatím bylo v textu již zmíněno, co umožňují skripty zastupující datovou strukturu octree, samozřejmě ve výsledku obohacené o další metody zajišťující předávání informací uložených v atributech a promazávání a vyhledávání prvků napříč stromem. Dále byla popsána funkčnost statické třídy, která slouží převážně pro krátkodobé uchování dat, k nimž je potřeba přistupovat z více skriptů. A především uchovává i seznam postupně přidaných kostiček do scény, který slouží ke správné funkci pro uložení objektu do souboru a pro mazání kostiček v opačném pořadí, než byly přidávány.

Skript TouchCameraControlScript obsahuje metody týkající se pouze ovládání hlavní kamery ve scéně, kdy je zpracováván vstup od uživatele v podobě dotyku prsty s displejem mobilního zařízení. Pohyb jednoho prstu má za následek rotaci kamery okolo modelu, pohyb dvou prstů od sebe a k sobě způsobuje přibližování a oddalování. Přesun rotačního bodu kamery do stran je reprezentován pohybem tří prstů a dotykem čtyř prstů je navráceno výchozí nastavení kamery.

Chod posuvného pásu s kategoriemi a jednotlivými kostičkami má na starost skript zvaný ScrollRectScript. Zde jsou atributy v podobě jednorozměrných polí, které uchovávají názvy jednotlivých podkategorií a počty jejich prvků, kdy je následně automaticky zajišťováno načítání příslušných tlačítek. Také je využíváno vnitřního skriptu ScrollRect, který umožňuje posun panelu vertikálním směrem prostřednictvím jeho tažení. V závislosti na počátku a následném konci tohoto tažení jsou volány veřejné metody z prvního zmiňovaného skriptu, které zajistí správný přepočet vzdáleností jednotlivých tlačítek od středu panelu, kdy je následně nejbližší tlačítko vycentrováno do tohoto středu.

S touto funkčností také souvisí skript zvaný DetailOfPartRotationScript, který právě v závislosti na skončení tažení posuvného pásu zjistí z názvu tlačítka, jež je vycentrováno do středu pásu, název kostičky, kterou následně promítne do okna mini mapy, je -li viditelné. Tím je uživateli poskytnuta možnost detailního pohledu na vkládaný objekt ještě před samotným vložením. V režii tohoto skriptu je také skrývání a zobrazování

této mini mapy a přepínání mezi promítnutím detailu objekty, či pohledu shora na vytvářený model. Tento pohled je poskytnut při přesunu kostičky v rámci scény, aby měl uživatel možnost vidět i pohyb v rámci třetí osy trojdimenzionálního pracovního prostoru.

Skript BrickSelectionScript implementuje funkčnost pro vysílání paprsku v místě dotyku prstu s displejem, je -li umožněn výběr kostiček. Následně je provedena detekce tohoto paprsku napříč celým stromem octree a případně nastavena reference na danou kostičku. Jednou z dalších metod je zde metoda pro změnu materiálu (Zdrojový kód 5). Zde je převzata reference na vybranou kostičku, z předaného tlačítka reprezentujícího jednu konkrétní barvu je zjištěn název materiálu. Pokud je reference prázdná, znamená to, že není vybrána žádná kostička, což má za důsledek nastavení globálního materiálu, se kterým jsou vkládány nové kostičky, na tento vybraný. Pokud však byla vybrána kostička, je zavolána metoda changeBrickColor, jež změní barvu všem potomkům v hierarchii dané kostičky, přesněji jejímu tělu a špuntíkům. Prvotní reference ukazuje na zastřešující objekt, kdy by se změna jeho materiálu neprojevila na kostičce samotné, proto je nutné přistoupit přímo k jednotlivým prvkům a ty změnit.

```
private void changeMaterial(Button button){
  brickForMaterialChange = InnerStoringScript.getSelectedBrick ();
  string buttonName = button.name;
  string materialName = buttonName.Substring (0, buttonName.Length - 6);
  if (brickForMaterialChange != null) {
    changeBrickColor (materialName);
  } else {
    InnerStoringScript.setUsedMaterial
(Resources.Load ("Models/Materials/" + materialName) as Material);
    setLogLabel ("Barva nově vkládaných kostiček změněna na " +
InnerStoringScript.getUsedMaterial ().name + ".");
  }
}
private void changeBrickColor(string materialName){
  Transform[] allChildren =
brickForMaterialChange.GetComponentsInChildren<Transform>();
  for (int i = 1; i < allChildren.Length; i++) {
    allChildren [i].gameObject.GetComponent<MeshRenderer> ().material =
Resources.Load ("Models/Materials/" + materialName) as Material;
  }
}
```

Zdrojový kód 5 - Změna materiálu

Poslední dva skripty v této scéně zvané CreateObjectScript a DataSaveLoadScript jsou nejrozsáhlejšími v rámci celé aplikace, neboť implementují základní a nejdůležitější

funkcionality. První z nich obsahuje metody pro vytvoření kostičky prostřednictvím vytvoření instance z prefab objektu dané kostičky uloženého v rámci multimediálního obsahu aplikace ve složce Resources. Zde je třeba nastavit její základní posunutí v osách x a z, v závislosti na jejím rozměru. Má -li kostička lichý počet špuntíků v daném směru, je třeba jí posunout o 0.4 bodu vzdálenosti, tedy o polovinu základní velikosti, oproti kostičce se sudým počtem, aby bylo zajištěno jejich vzájemné dosednutí. Zároveň je jí vytvořen obalový objekt.

Tímto je ovšem vyřešeno pouze vytvoření její instance, kostička ještě musí být umístěna do scény. To zajišťuje ovládací prvek, jenž je zviditelněn při vytvoření instance kostičky. Zde uživatel započne tažení a plynulým pohybem umístí kostičku do požadovaného místa. Skript tak obsahuje další metody snímající pozici prstu v souřadnicích okna, displeje, a ty následně přepočítávají na souřadnice v prostoru. Pohyb v třetí ose je zajištěn prostřednictvím poskytnutých tlačítek, zároveň je možné i kostku otáčet kolem její osy po devadesáti stupňových úhlech. Jakmile je kostička uvolněna, provede se korekce její pozice, aby zapadla do pevně definované mřížky dané rozměry základní kostičky. Po této korekci pozice přichází na řadu umístění kostičky do datové struktury octree, a s tím související kontrola, zda se kostička neprolíná s jinou již existující. V případě že došlo ke kolizi, tak kostička není přidána a její instance je zničena. V opačném případě je umístěna ve scéně a přiřazena do stromu a listu v rámci statické třídy pro potřeby uložení objektu. Skript tak obsahuje na dvacet metod zajišťujících tyto funkcionality spolu s výpisem hlášek o chybě či úspěšném vložení pro uživatele.

Poslední skript DataSaveLoadScript obsahuje metody sloužící ke zpětné transformaci dat po načtení souboru s modelem tak, aby mohly být vytvořeny jednotlivé kostičky. Dále iniciuje uložení dat o nějž se stará skript ModelSaveLoadScript i s nutnou transformací do textové reprezentace pro uložení. Modely jsou totiž uloženy v paměti zařízení ve formě textových souborů, kde každý řádek obsahuje informace o jednotlivých kostičkách v pevně definovaném tvaru. O každé kostičce jsou ukládány čtyři základní informace spojené znakem #, konkrétně název daného herního objektu, jež ji reprezentuje, její pozice, natočení a název přiřazeného materiálu. Výsledný záznam jedné kostičky vypadá následovně:

Brick_3x1(Clone)#(6.0, 1.0, 0.4)#(0.0, 0.0, 0.0, 1.0)#GreyColor (Instance).

Kompletní implementace metody pro uložení je znázorněna v kódu (Zdrojový kód 6), kdy je nejprve vybrán ze statické třídy seznam všech kostiček ve scéně. Pokud tento seznam obsahuje více než jen rodičovský objekt, pod kterým jsou sdružovány všechny kostičky modelu, tak je vytvořen objekt sloužící pro zápis dat do souborů na příslušné adrese se jménem zadaným při načítání scény. Z každé kostičky se následně načtou potřebné informace a uloží do formátu jednoho řádku souboru, který se zapíše do daného souboru. To probíhá, dokud není celý seznam kostiček prošlý, poté je model kompletně uložen.

```
public bool saveData(){
  return saveDataToTxtFile();
}
private bool saveDataToTxtFile(){
  allBricks = InnerStoringScript.getAllBricksInModel();
  if (allBricks.Count > 1) {
    StreamWriter sw =
new StreamWriter (pathToSaveFiles + fileName, false);
    for (int i = 1; i < allBricks.Count; i++) {
      string line = allBricks [i].name + "#"
        + allBricks [i].transform.position.ToString () + "#"
        + allBricks [i].transform.rotation.ToString () + "#"
        + allBricks [i].GetComponentsInChildren<Transform> () [1]
.gameObject.GetComponent<MeshRenderer> ().material.name;
      sw.WriteLine (line);
    }
    sw.Close ();
    return true;
  } else {
    return false;
  }
}
```

Zdrojový kód 6 - Uložení modelu do souboru

V rámci skriptu DataSaveLoadScript funkce načtení modelu je následně pomocí několika metod tento načtený vstup ze souboru přetransformován do požadovaných tvarů a datových typů. Názvy neobsahují závorky, pozice je reprezentována typem Vector3 a natočení datovým typem Quaternion. Na základě těchto hodnot a takto postupně zpracovaného celého souboru lze vytvořit jednu kostičku po druhé a opětovně sestavit původní model.

Další zajišťovanou funkčností je chod menu editoru a vykreslení návodu na stavbu vytvořeného modelu. Vykreslení návodu je implementované prostřednictvím 16 metod, které se starají o správné zobrazení potřebných panelů, správu kostiček modelu, vykreslení

snímku kamery do textury a její následné uložení ve formátu rastrového obrázku PNG do paměti zařízení. Na počátku funkce je celý postavený model nastavený jako neviditelný, poté je procházen seznam kostiček, jimiž je tvořen, a po čtyřech kostičkách postupně zviditelňován. Vždy jsou promítnuty záběry ze dvou kamer do připravených obrázků v panelu, přidán text upřesňující, které kostičky byly přidány a následně uložen snímek z hlavní kamery. Poté jsou tyto čtyři kostičky zprůhledněny a pokračuje se s další sadou, dokud není model sestaven.

Jedná se o stručný popise funkčnosti s ukázkou několika implementací konkrétních metod, kdy celý zdrojový kód není možné v jeho rozsahu připojit k práci.

4.5 Testování s uživateli

Tato kapitola se zabývá testováním výsledné interaktivní aplikace představující virtuální editor dětské stavebnice s reálnými uživateli. Jedná se o testování za dobrovolné účasti čtyř participantů spadajících do definované cílové skupiny uživatelů z kapitoly 4.1.1 Cílová skupina, kdy každý z nich potvrdil, že se již setkal s mobilním zařízením a s jeho ovládáním má alespoň částečné zkušenosti.

Kapitola samotná je rozdělena do dvou částí, kdy v první části jsou popsány potřebné informace k samotnému testování a definovány jednotlivé úkoly. V druhé části jsou zpracovány výsledky testování, kdy se hodnotí závažnost případných nalezených nedostatků a navrhuje se jejich způsob opravy.

4.5.1 Příprava a průběh testování

Aplikace je testována na dvou mobilních zařízení lišících se především velikostí, kdy se jedná o reprezentanty třídy smartphonů a tabletů, konkrétně smartphone Samsung Galaxy S6 Edge a tablet Samsung Galaxy Tab A6. Podrobná konfigurace obou zařízení je:

- Samsung Galaxy S6 Edge
    - Operační systém - Android 5.1 (Lollipop)
    - Paměť RAM - 3072 MB
    - Rozlišení displeje - 2560 x 1440
    - Rozměr displeje - 5.1 palce
    - Procesor - 8 jader, 2.1 GHz

- Samsung Galaxy Tab A6
    - Operační systém - Android 7.0
    - Paměť RAM - 2048 MB
    - Rozlišení displeje - 1920 x 1200
    - Rozměr displeje - 10.1 palce
    - Procesor - 8 jader, 1.6 GHz

Testování probíhá formou plnění zadaných úkolů, které by měli obsáhnout co nejširší oblast funkčnosti výsledné aplikace, v ideálním případě všechny poskytované funkce. Dva participanti plní zadané úkoly na jednom typu zařízení, zbylí dva na druhém, aby bylo dosaženo výsledků pro možné srovnání, například zda je větší displej pro danou aplikaci vhodnější.

Participanti jsou před samotným testováním seznámeni s jeho průběhem a jsou ujištěni, že se jedná čistě o testování daného produktu s cílem nalézt případné nedostatky. Proto není třeba být nervózní z toho, zda se jim podaří dané úkoly splnit. Dále jsou informováni o následném průběhu, kdy jim vzhledem k jejich věku bude vždy přečten zadaný úkol, který mají splnit. Při tom jsou také požádáni, zda by mohli uvažovat nahlas, sdělovat své myšlenky, říkat, co právě dělají a co hledají, aby bylo možné vytváření průběžných poznámek, které jsou v rámci následující kapitoly sjednoceny a vyhodnoceny, a které slouží pro odhalení případných nedostatků aplikace od jednoduchých kosmetických vad až po závažné problémy mařící plnění úkolů.

Seznam obsahuje deset úkolů s přesným zněním:

1. Zadej svou přezdívku a ulož nastavení.
2. Změň vzhled maskota aplikace a ulož nastavení.
3. Spusť scénu s editorem v režimu stavění nového modelu.
4. Zobraz panel s mini mapou.
5. Umísti tři různě tvarované kostičky do prostoru scény.
6. Vyber jednu z těchto kostiček a změň její barvu.
7. Vyber jinou kostičku a smaž jí.
8. Ulož model a vrať se do hlavního menu.
9. Načti svůj uložený model.
10. Přidej do tohoto modelu novou kostičku a vykresli návod na jeho stavbu.

Po ukončení testování a splnění všech úkolů je jim následně dán prostor pro jejich osobní poznatky a subjektivní ohodnocení dané aplikace. Co by na dané aplikaci navrhli za změny.

4.5.2 Vyhodnocení výsledků

Testování probíhalo formou plnění deseti zadaných úkolů čtyřmi participanty zastupujícími cílovou skupinu. Všichni participanti plnili úkoly dobrovolně a nezávisle jeden na druhém, kdy dva k plnění využili mobilního zařízení typu tablet a dva chytrého telefonu, s cílem získat data pro vzájemné srovnání.

V průběhu testování byly pořizovány poznámky popisující situace, kdy se některý z participantů zdržel během plnění úkolu. Po skončení testování byla tato nasbíraná data shrnuta a zanalyzována. Podrobný popis některých případů je uveden dále v textu a sestává z čísla daného úkolu, nastalého problému, ohodnocení jeho závažnosti a případného návrhu, jak nalezený nedostatek opravit.

Závažnost definuje hodnota od 1 do 4, kdy hodnota 1 udává tzv. kosmetickou vadu představující méně významný nedostatek, který nijak závažně neovlivňuje průběh plnění úkolu. Hodnota 2 označuje méně závažný problém, na který zpravidla nenarazí všichni uživatelé, a který sice trochu prodlouží plnění zadaného úkolu, ale nezpůsobí daný úkol neřešitelným. Hodnotou 3 je označen problém, jenž je často opakován většinou participantů a má podstatný vliv na dobu trvání plnění zadaného úkolu. Tato chyba by měla být v brzké době odstraněna. Poslední hodnota 4 označuje problém takového rozsahu, který přímo znemožňuje splnění zadaného úkolu, a který musí být neprodleně opraven.

1- Participant na první pohled neví, kde provést zadání přezdívky, po chvilce zkoumání zkouší otevřít záložku Nastavení. Zadává přezdívku, ukládá volbu a úkol je splněn. (závažnost: 1)
2- Úkol je splněn bez obtíží všemi participanty, neboť v rámci plnění předchozího úkolu viděli příslušné položky v záložce Nastavení.
3- Úkol splněn všemi participanty.
4- U některých participantů se vyskytl problém neznalosti směrové růžice, přesto byl úkol splněn všemi. (závažnost: 2)
    a. Navrhnout jiný vzhled ikonky pro mini mapu, malý plánek či obdobu označení využívaného službou online map.

5- Problém s viditelností malých kostiček při přesunu v prostoru scény, které jsou zcela zakryty prstem na displeji. (závažnost: 2-3 s ohledem na požadovanou přesnost umístění)
    a. Možnost úpravy pozice při přemisťování kostičky v prostoru scény prostřednictvím přičtení konstantní hodnoty.
6- Participant se pokouší vybrat kostičku bez povolení funkce Výběr. Způsob provedení se snaží nalézt v nápovědě, ovšem ta je spíše popisné charakteru, kdy neznalost čtení znemožňuje nalezení postupu. Po chvilce zkoušení tlačítek nalézá funkci Výběr a vybírá jednu kostičku ve scéně. Poté otevírá paletu barev ovšem bez povšimnutí, že vybraná kostička byla opuštěna. Opět zkouší vybrat kostičku a úspěšně mění její barvu. (závažnost: 2)
    a. Bylo by vhodné upravit nápovědu do srozumitelnější podoby i pro děti jakožto cílovou skupinu, lépe popsat ovládání aplikace.
7- S ohledem na předchozí úkol není problém s výběrem jiné kostičky ve scéně. Zde někteří participanti zvolili metodu přetažení kostičky na ikonku koše, což má za následek pouze její přemístění. Nebo po výběru stiskli tlačítko koše v horní liště, které neodstraní vybranou, ale poslední vloženou kostičku. Přesto byl úkol nejdéle na třetí pokus proveden úspěšně. (závažnost: 3)
    a. Navrhnout změnu ikony pro odstranění poslední kostičky ze scény, aby nebyla totožná s ikonou pro odstranění vybrané kostičky.
8- Úkol proveden bez obtíží všemi participanty.
9- Úkol proveden bez obtíží všemi participanty.
10- S ohledem na splnění předchozích úkolů byla kostička přidána bez obtíží a po krátké chvilce hledání byl vytištěn návod stavby. Zde se participanti snažili najít příslušný návod přímo v okně aplikace. (závažnost: 1)
    a. Umožnit načtení návodu z okna aplikace.

Všichni zúčastnění participanti zvládli splnit všechny úkoly bez znatelnějších obtíží. Prostřednictvím provedeného testování a zanalyzování nasbíraných dat bylo zjištěno, že aplikace obsahuje malé množství nedostatků v podobě kosmetických vad, například v podobě nejasných ikonek, a méně závažných problémů, které nijak neznemožňují její reálné využití.

5 Výsledky a diskuse

Definovaným cílem této diplomové práce s tématem interaktivní aplikace pro rozvoj kreativity je navrhnout a následně implementovat aplikaci, jež by sloužila jako nástroj pro rozvoj tvůrčí osobnosti dětí předškolního věku a žáků prvního stupně základních škol. Aplikaci, která by jim poskytla prostor a možnosti, jak proměnit jejich vizi v něco skutečného, nejprve formou virtuálního modelu, a poté s možností užití stejného postupu tvorby i v reálné podobě.

S ohledem na definovaný cíl práce a v závislosti na prostudovaných metodách tvorby multimediálního obsahu aplikací a technikách jeho propojení v ucelený výstup, byla v rámci práce vytvořena interaktivní aplikace reprezentující editor dětské stavebnice. Aplikace představuje alternativu ke skutečné stavebnici, kterou je možné využívat nezávisle na okolním prostředí, neboť je implementovaná pro mobilní zařízení s podporou, v současnosti nejrozšířenějšího, operačního systému Android. Uživatelé tak mohou přetvářet své spontánní nápady do virtuální podoby téměř okamžitě i na cestách či o přestávkách ve škole beze strachu, že by je zapomněli dříve, než by se dostali do prostředí domova ke skutečné stavebnici.

Aplikace je navržena takovým způsobem, aby přinesla nové možnosti interakce, které nejsou využity v rámci nalezených existujících řešení. Poskytuje uživatelům intuitivnější způsob ovládání typický pro práci v editorech, než jaký implementují zkoumané existující aplikace VirtualBlock a Draw Bricks, kdy umisťování kostiček do prostoru probíhá čistě formou drag-and-drop, neboli tažením a puštěním s možností současného zobrazení scény ze dvou úhlů pohledu. Druhý pohled zajišťuje kamera snímající půdorys scény promítnutá do volitelně zobrazitelného okna mini mapy s možností značného přiblížení pro zvýšení přesnosti umístění. Jednou z nejdůležitějších funkcionalit je poté možnost vykreslení postupného návodu, jak byl model tvořen. Prostřednictvím toho je možné převést tuto virtuální podobu jakkoliv složitého modelu do reálné formy sestavené ze skutečné stavebnice. V porovnání se zkoumanými aplikacemi představuje tato implementovaná funkčnost kresby návodu jistou výhodu, neboť žádná z uvedených mobilních aplikací tím nedisponuje. Pouze program LEGO Digital Designer má tuto funkcionalitu taktéž k dispozici, ovšem ten své uživatele omezuje v mobilitě, neboť se jedná o program pro platformy Windows a OS X na stolních počítačích, nikoliv o aplikaci pro mobilní zařízení.

Jednou z posledních předností aplikace je její poměrně snadná rozšiřitelnost o nové modely částí stavebnice s nutností nepatrného zásahu do zdrojového kódu, nejedná -li se o specificky tvarované části. Momentálně aplikace disponuje 63 částmi uspořádanými do deseti kategorií, které lze snadno rozšířit o nové kostičky větších rozměrů. Přidání nové kategorie sebou nese nutnost většího zásahu do kódu, ovšem pouze v rámci dvou skriptů. Také uživatelské rozhraní reprezentující panely pro volbu kostiček působí méně chaotickým dojmem, kdy je zde každá kostička zastoupena jediným ovládacím prvkem a její výsledná barva může být upravena i po umístění do prostoru scény.

A v rámci provedeného testování výsledné interaktivní aplikace za dobrovolné účasti čtyř uživatelů zastupujících definovanou cílovou skupinu nebyl shledán žádný závažný problém či nedostatek, který by přímo znemožňoval její reálné využití.

6 Závěr

Cílem této diplomové práce bylo navrhnout a následně vytvořit interaktivní aplikaci, jež by napomohla dětem při rozvoji jejich tvůrčího nadání a poskytla jim možnost převedení jejich myšlenek do téměř hmotné podoby. Ukázat jim, že moderní technologie nejsou jen prostředkem pro konzumaci již vytvořeného obsahu, ale že je lze využít i tvůrčím a inovativním způsobem pro realizaci vlastních nápadů.

V rámci práce byla navržena a implementována aplikace reprezentující editor dětské stavebnice, kdy si uživatel může prostřednictvím definovaných trojrozměrných modelů částí stavebnice sestavit komplexnější model. V současné době je na internetu dostupných několik již existujících aplikací zabývajících se touto problematikou, se kterými byla vytvořená aplikace podrobněji porovnána v rámci předchozí kapitoly, ovšem jen málokterá z nich poskytuje uživateli všechny potřebné funkce doplněné o přívětivě vyhlížející a snadno pochopitelné grafické rozhraní, které by koncové uživatele ještě více podněcovalo k práci s danou aplikací.

Interaktivní aplikace vytvořená v rámci této diplomové práce byla navržena tak, aby oproti existujícím řešením přinesla nový způsob uživatelské interakce, poskytla funkčnosti, jež nejsou samozřejmostí u všech dostupných řešení, umožnila poměrně snadné rozšíření portfolia modelů reprezentujících jednotlivé části stavebnice, a přesto aby fungovala na mobilních zařízeních, kdy ji tak lze využívat i mimo domov, na cestách či o přestávkách ve škole.

Jako možná inovace do budoucna by se mohla jevit funkcionalita, kdy by mohl být vytvořený model uložen ve formátu používaném pro trojrozměrné modely, konkrétně OBJ, a následně sdílen s ostatními uživateli prostřednictvím webové služby Sketchfab. Druhou potenciální inovací by mohla být úprava grafického uživatelského rozhraní. Jednalo by se o vytvoření panelu pro přímé otevření a procházení vykresleného návodu uvnitř aplikace. Poslední možnou budoucí novinkou by mohlo být využití modelu maskota jako animované nápovědy v rámci scény reprezentující editor, pracovní prostor. Kdyby by mohlo být využito textových, případně lépe působících zvukových, rad udílených po časových intervalech bez interakce uživatele s aplikací, či při podržení na specifickém prvku.

7 Seznam použitých zdrojů

[1] DERAKHSHANI, Dariush. *Maya: průvodce 3D grafikou*. Praha: Grada. Průvodce (Grada) , 2006. 428 s., 8 s. barevné obrázkové přílohy. ISBN 80-247-1253-9.

[2] What is Raster Graphics? - Definition from Techopedia. *Techopedia.* [online]. [cit. 2018-01-28]. Dostupné z: https://www.techopedia.com/definition/9098/raster-graphics.

[3] What´s the difference between vector & raster graphics?. *Logaster*. [online], 2014-12-25. [cit. 2018-01-28]. Dostupné z: https://www.logaster.com/blog/vector-and-raster-graphics/.

[4] Raster (Bitmap) vs Vector. *Vector conversion.* [online], 2016. [cit. 2018-01-28]. Dostupné z: https://vector-conversions.com/vectorizing/raster_vs_vector.html.

[5] ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. *Moderní počítačová grafika*. 2., přeprac. a rozš. vyd. Praha: Computer Press, 2004. 609 s., 16 s. barevné obrázkové přílohy. ISBN 80-251-0454-0.

[6] What´s the difference between vector & raster graphics?. *Logaster*. [online]. [cit. 2018-01-28]. Dostupné z: https://www.printcnx.com/wp-content/uploads/raster.jpg.

[7] What´s the difference between vector & raster graphics?. *Logaster*. [online]. [cit. 2018-01-28]. Dostupné z: https://www.printcnx.com/wp-content/uploads/vector.jpg

[8] Tutorials - What Are Vector Graphics?. *Vectr*. [online]. [cit. 2018-01-28]. Dostupné z: https://vectr.com/tutorials/what-are-vector-graphics/.

[9] ICT kompetence: studijní podpora pro oblast počítačové grafiky, digitálního zvuku, fotografie a videa. *ICT kompetence: Studijní web Katedry technické a informační výchovy Univerzity Palackého v Olomouci*. [online]. [cit. 2018-01-28]. Dostupné z: http://www.kteiv.upol.cz/frvs/ict-kubricky/?page=pocitacova-grafika/vektorova-grafika.

[10] DANNHOFEROVÁ, Jana. Počítačová grafika: Modelování těles. *Mendelova univerzita v Brně: Elektronické studijní materiály*. [online]. [cit. 2018-02-05]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/283/Knihovna%20k%20projektu/Po%C4%8D%C3%ADta%C4%8Dov%C3%A1%20grafika/Slajdy%20z%20p%C5%99edn%C3%A1%C5%A1ek/Modelovani_teles.pdf.

[11] STRACHOTA, Pavel. Modelování pevných těles. *Pavel Strachota - domovská stránka: České vysoké učení technické, Fakulta jaderná a fyzikálně inženýrská.* [online], 2015-03-12. [cit. 2018-02-05]. Dostupné z: http://saint-paul.fjfi.cvut.cz/base/sites/default/files/POGR/POGR2/05.modelovani_teles.pdf.

[12] JAMES, Mike. Quadtrees and Octrees. *I programmer*. [online], 2015-06-30 [cit. 2018-02-06]. Dostupné z: http://www.i-programmer.info/programming/theory/1679-quadtrees-and-octrees.html.

[13] NEVALA, Eric. Introduction to Octrees: General and Gameplay Programming. *Gamedev.net.* [online], 2014-01-20. [cit. 2018-02-06]. Dostupné z: https://www.gamedev.net/articles/programming/general-and-gameplay-programming/introduction-to-octrees-r3529/.

[14] SLICK, Justin. Seven Common Modeling Techniques for Film and Games: An Introduction to 3D Modeling Techniques. *Lifewire.* [online], 2017-10-17. [cit. 2018-02-15]. Dostupné z: https://www.lifewire.com/common-modeling-techniques-for-film-1953.

[15] Keyframe Animation. *Maya: Autodesk Knowledge Network.* [online], 2016-05-11. [cit. 2018-02-15]. Dostupné z: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-66ED4510-CC1B-4E11-918B-B7DC447E38A7-htm.html.

[16] Inverse Kinematics (IK). *3DS MAX: Autodesk Knowledge Network.* [online], 2017-06-15. [cit. 2018-02-15]. Dostupné z: https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/3DSMax/files/GUID-516E301F-E911-429F-9337-9FA7FAD49BB6-htm.html.

[17] The fascination for motion capture: Xsens 3D motion tracking. *Xsens.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.xsens.com/fascination-motion-capture/.

[18] Autodesk Student Community: About Autodesk Education. *Autodesk.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.autodesk.com/education/about-autodesk-education.

[19] What´s New In Maya 2018: Features. *Autodesk.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.autodesk.com/products/maya/features.

[20] Overview: Knowledge Overview. *Maya: Autodesk Knowledge Network.* [online]. [cit. 2018-02-15]. Dostupné z: https://knowledge.autodesk.com/support/maya.

[21] Předplatné Maya: Koupit software Maya 2018. *Autodesk: Maya.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.autodesk.cz/products/maya/subscribe?referrer=%2Fproducts%2Fmaya%2Fsubscribe&plc=MAYA&term=1-YEAR&support=ADVANCED&quantity=1.

[22] Maya Subscription: Buy Maya 2018 Software. *Autodesk: Maya.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.autodesk.com/products/maya/subscribe?geoNavigationPreferredSite=US&plc=MAYA&term=1-YEAR&support=ADVANCED&quantity=1.

[23] Free Software for Students & Educators. *Autodesk: Education / Maya.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.autodesk.com/education/free-software/maya.

[24] Maya LT: Game Development Software. *Autodesk: Maya LT.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.autodesk.com/products/maya-lt/overview.

[25] Blender.org: Home of the Blender project - Free and Open 3D Creation Software. *Blender.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.blender.org/.

[26] History: blender.org. *Blender.* [online], 2013-07. [cit. 2018-02-15]. Dostupné z: https://www.blender.org/foundation/history/.

[27] License: blender.org. *Blender.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.blender.org/about/license/.

[28] Features: blender.org. *Blender.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.blender.org/features/.

[29] Release Notes: blender.org. *Blender.* [online]. [cit. 2018-02-15]. Dostupné z: https://www.blender.org/download/releases/.

[30] Best Game Engine Software in 2018: Overview. *G2 Crowd.* [online]. [cit. 2018-02-18]. Dostupné z: https://www.g2crowd.com/categories/game-engine.

[31] WATKINS, Adam. *Creating games with Unity and Maya: how to develop fun and marketable 3D games.* Burlington, MA: Focal Press, 2011. 528 s. ISBN 978-0-240-81881-8.

[32] Počítačové hry: Herní engine. *Západočeská univerzita v Plzni.* [online]. [cit. 2018-02-18]. Dostupné z: https://courseware.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=21322.

[33] Physics engine. *Wikipedia: The Free Encyclopedia*. [online]. [cit. 2018-02-18]. Dostupné z: https://en.wikipedia.org/wiki/Physics_engine.

[34] KANBER, Burak. How Physics Engines Work: Introduction, Motivation, and Goals. *Build New Games: Open Web echniques for cutting-edge game development.* [online], 2012-11-08. [cit. 2018-02-18]. Dostupné z: http://buildnewgames.com/gamephysics/.

[35] ČÁPKA, David. Úvod do objektově orientovaného programování v C#: 1. díl. *ITnetwork.cz.* [online]. [cit. 2018-02-18]. Dostupné z: https://www.itnetwork.cz/csharp/oop/c-sharp-tutorial-uvod-do-objektove-orientovaneho-programovani.

[36] PECINOVSKÝ, Rudolf. *Java 7: učebnice objektové architektury pro začátečníky*. Praha: Grada. Knihovna programátora (Grada) , 2012. 496 s. ISBN 978-80-247-3665-5.

[37] ČÁPKA, David. Dědičnost a polymorfismus: 7. díl. *ITnetwork.cz.* [online]. [cit. 2018-02-18]. Dostupné z: https://www.itnetwork.cz/csharp/oop/c-sharp-tutorial-dedicnost-a-polymorfismus.

[38] ČÁPKA, David. Rozhraní (interface) v Javě: 17. díl. *ITnetwork.cz.* [online]. [cit. 2018-02-18]. Dostupné z: https://www.itnetwork.cz/java/oop/java-tutorial-interface-rozhrani.

[39] PECINOVSKÝ, Rudolf. *Návrhové vzory: 33 vzorových postupů pro objektové programování.* Brno: Computer Press, 2013. 528 s. ISBN 978-80-251-1582-4.

[40] Multiplatform: Publish your game to over 25 platforms. *Unity.* [online]. [cit. 2018-02-22]. Dostupné z: https://unity3d.com/unity/features/multiplatform?_ga=2.196370799.276161505.1521024617-278052959.1507751300.

[41] Fast Facts: The leading global game industry software. *Unity.* [online]. [cit. 2018-02-22]. Dostupné z: https://unity3d.com/public-relations.

[42] Manual: Documentation versions. *Unity: Documentation.* [online]. [cit. 2018-02-22]. Dostupné z: https://docs.unity3d.com/Manual/ManualVersions.html.

[43] Manual: Creating and Using Scripts. *Unity: Documentation.* [online]. [cit. 2018-02-22]. Dostupné z: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html.

[44] Unity Personal: Everything you need to get started. *Unity: Store.* [online]. [cit. 2018-02-22]. Dostupné z: https://store.unity.com/products/unity-personal.

[45] Unity Plus: The proven platform for achieving success. *Unity: Store.* [online]. [cit. 2018-02-22]. Dostupné z: https://store.unity.com/products/unity-plus.

[46] Unity Pro: Unity´s fully featured plan. *Unity: Store.* [online]. [cit. 2018-02-22]. Dostupné z: https://store.unity.com/products/unity-pro.

[47] LILLY, Paul. Doom to Dunia: A Visual History of 3D Game Engines - Page 4. *DailyRadar: Maximum PC.* [online]. 2009-07-21 [cit. 2018-02-25]. Dostupné z: https://web.archive.org/web/20090724065520/http://www.maximumpc.com/article/features/3d_game_engines?page=0%2C3.

[48] Epic´s Unreal Engine 3. *Maximum PC.* [online], 2004-08, s. 107. [cit. 2018-02-25]. Dostupné z: https://archive.org/details/Maximum_PC_Autumn_2004.

[49] Engine Features. *Unreal Engine.* [online]. [cit. 2018-02-25]. Dostupné z: https://docs.unrealengine.com/en-us/Engine.

[50] About Unreal Engine 4. *Unreal Engine.* [online]. [cit. 2018-02-25]. Dostupné z: https://www.unrealengine.com/en-US/features.

[51] Blueprints Visual Scripting. *Unreal Engine.* [online]. [cit. 2018-02-25]. Dostupné z: https://docs.unrealengine.com/en-us/Engine/Blueprints.

[52] Unreal Engine Frequently Asked Questions. *Unreal Engine.* [online]. [cit. 2018-02-25]. Dostupné z: https://www.unrealengine.com/en-US/faq.

[53] SOUSA, Bruno. Draw Bricks: Aplikace pro Android ve službě Google Play. *Google Play.* [online], 2018-02-17. [cit. 2018-03-03]. Dostupné z: https://play.google.com/store/apps/details?id=com.brunosousa.drawbricks.

[54] Android Apps: Portfolio. *Bruno Sousa: Web developer & mobile development.* [online]. [cit. 2018-03-03]. Dostupné z: http://brunoportfolio.com/android_apps.php.

[55] Hodnocení obsahu aplikací a her na Google Play: Nápověda Google Play. *Google: Nápověda Google Play.* [online]. [cit. 2018-03-03]. Dostupné z: https://support.google.com/googleplay/answer/6209544?visit_id=0-636565532577984345-237078870&p=appgame_ratings&rd=1.

[56] NULL PRODUCT. VirtualBlock - Block Builder: Aplikace pro Android ve službě Google Play. *Google Play.* [online], 2017-02-08. [cit. 2018-03-03]. Dostupné

z: https://play.google.com/store/apps/details?id=com.nullproduct.virtualblock&hl=cs.

[57] Download - LDD LEGO.com: LEGO Digital Designer. *LEGO.* [online]. [cit. 2018-03-03]. Dostupné z: https://www.lego.com/en-us/ldd/download.

[58] Designbyme - LDD LEGO.com: LEGO Digital Designer. *LEGO.* [online]. [cit. 2018-03-03]. Dostupné z: https://www.lego.com/en-us/ldd/designbyme.

[59] LEGO Digital Designer 4.3: User Manual. *LEGO: Support.* [online]. [cit. 2018-03-03]. Dostupné z: https://www.lhttps://www.lego.com/r/www/r/ldd/-/media/franchises/ldd2017/support/ldd%204%203%20en-manual.pdf?l.r2=1264289309ego.com/en-us/ldd/designbyme.

[60] Main Page - DCSwiki: Katedra informatiky. *Katedra informatiky: Vysoká škola báňská – Technická univerzita Ostrava / Fakulta elektrotechniky a informatiky*. [online]. [cit. 2018-03-05]. Dostupné z: http://wiki.cs.vsb.cz/images/5/58/Inz3.pdf.

[61] FluidUI.com: Create Web and Mobile Prototypes in Minutes. *FluidUI.* [online]. [cit. 2018-03-08]. Dostupné z: https://www.fluidui.com/.

[62] Fluid UI Features. *FluidUI.* [online]. [cit. 2018-03-08]. Dostupné z: https://www.fluidui.com/features.

[63] Manual: Console Window. *Unity: Documentation.* [online]. [cit. 2018-03-09]. Dostupné z: https://docs.unity3d.com/Manual/Console.html.

[64] Manual: Important Classes. *Unity: Documentation.* [online]. [cit. 2018-03-09]. Dostupné z: https://docs.unity3d.com/Manual/ScriptingImportantClasses.html.

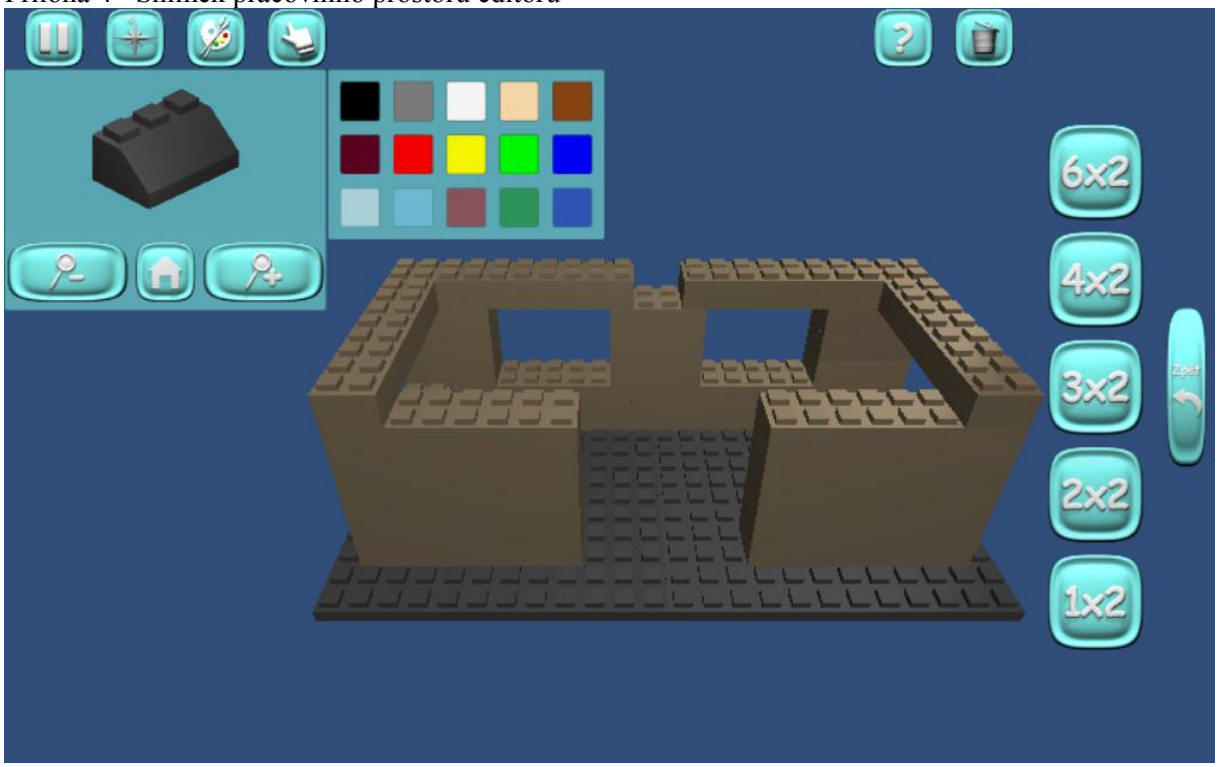
8 Přílohy

Příloha 1 - Snímek hlavní nabídky

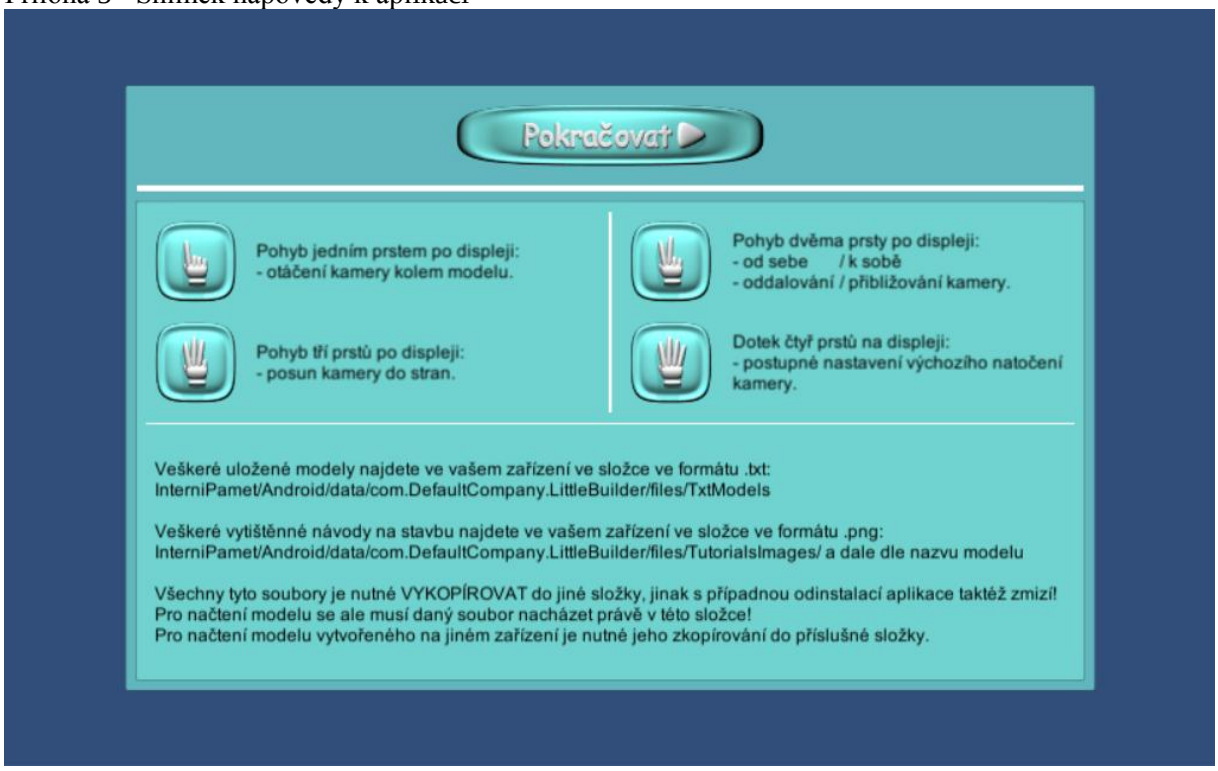
Příloha 2 - Snímek položky Nastavení

Příloha 3 - Snímek pokládání nové kostičky

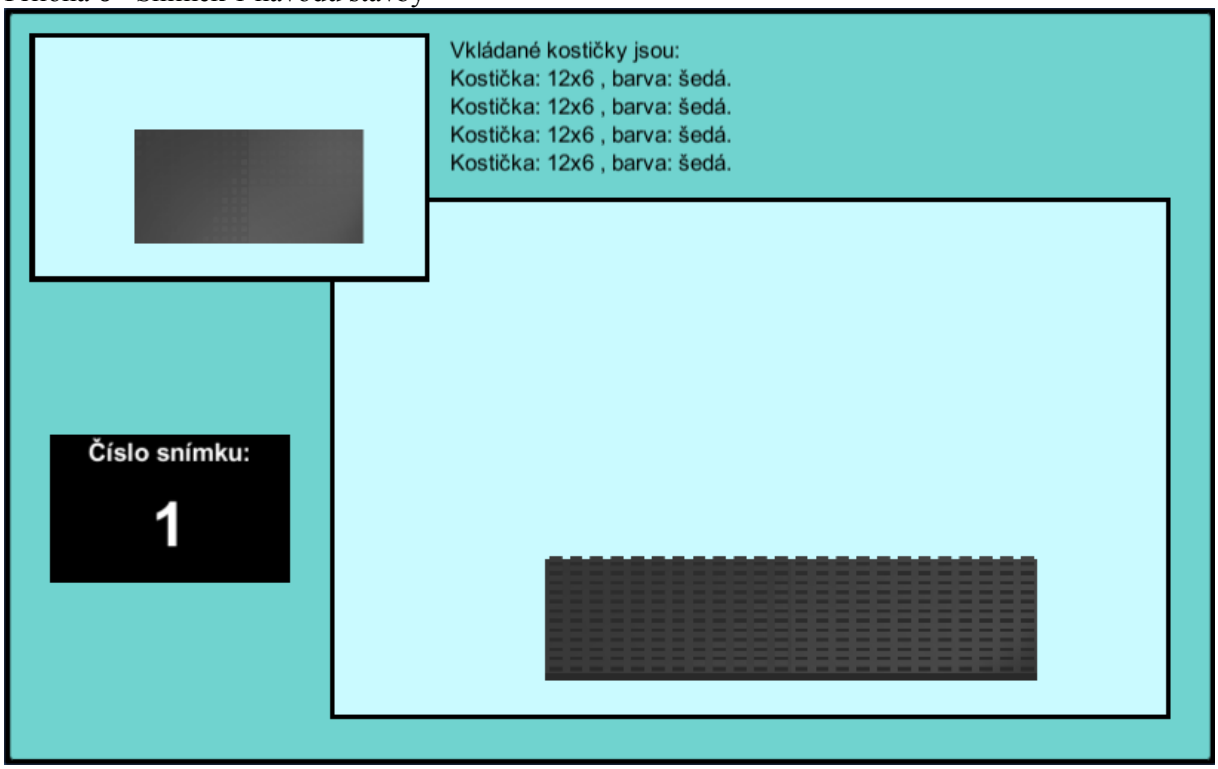
Příloha 4 - Snímek pracovního prostoru editoru



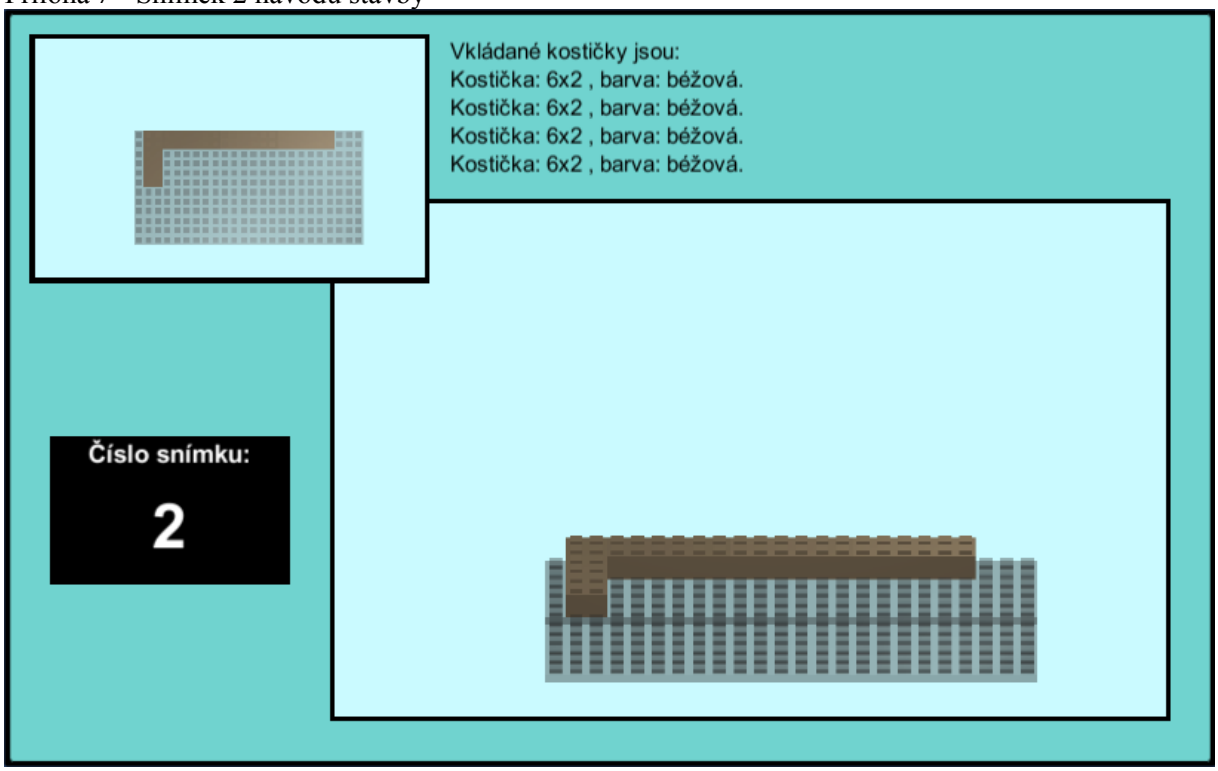
Příloha 5 - Snímek nápovědy k aplikaci



Příloha 6 - Snímek 1 návodu stavby



Příloha 7 - Snímek 2 návodu stavby



Příloha 8 - Snímek 3 návodu stavby