UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Softvérová podpora vyučovania matematiky Hejného metódou – prostredie Stavby z kociek

Bakalárska práca

2020 **Monika Vlčková**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Softvérová podpora vyučovania matematiky Hejného metódou – prostredie Stavby z kociek

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Peter Borovanský, PhD.
Konzultant:	RNDr. Dagmar Môťovská, PhD.

Bratislava, 2020 **Monika Vlčková**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Monika Vlčková
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Softvérová podpora vyučovania matematiky Hejného metódou - prostredie Stavby z kociek
Educational software for Hejny's method of mathematics teaching environment Cube Buildings

Anotácia: Aplikácie sa opierajú o didaktickú kvalitu vyučovania matematiky Hejného metódou, zadania úloh budú vyberané z učebníc matematiky Hejný, M., Jirotková, D., Slezáková, J., Bomerová, E., Michnová, J.: MATEMATIKA 1.-5., učebnice pro základní školy, Fraus, 2007-2011. Samotné matematické prostredia z týchto učebníc poskytujú gradáciu, flexibilitu a počítajú s interaktivitou, čo sa týka stvárnenia učebnej látky, úloh na riešenie, aj stratégií riešenia. Tieto vlastnosti prostredí treba využiť a preniesť do navrhovaného softvéru. Navrhovaný softvér ponúkne jednotlivým žiakom dostatočné množstvo úloh na jednotlivých úrovniach, podľa ich individuálnych potrieb, čím bude prínosom pre vyučovanie Hejného metódou. Zároveň treba zabezpečiť technickú kvalitu softvéru, kvalitu grafiky, používateľský komfort, prehľadnosť, spoľahlivosť a rýchlosť.

Cieľ: Cieľom práce je vytvoriť mobilnú aplikáciu (pre tablet) na tému zvoleného prostredia Hejného matematiky (HM). Aplikácia pre prvý stupeň ZŠ musí spĺňať zásady tvorby didaktického softvéru. Aplikácia musí byť testovaná na skupine žiakov, a následne upravená podľa zistených potrieb a event. nedostatkov. Zvolené prostredie HM pokrýva viacero typovo odlišných gradujúcich úloh/úrovní zodpovedajúcich konceptom, ktoré žiaci na danej úrovni objavujú. Aplikácia precvičuje každú úlohu/úroveň na sade predvolených a generovaných zadaní. Až po jej zvládnutí môže žiak pokročiť do ďalšej úrovne. Žiak má možnosť vytvoriť vlastné zadanie v rámci každej úlohy/úrovne. Pri návrhu nového zadania (ako aj pri jeho riešení) aplikácia indikuje počet existujúcich/zostávajúcich riešení daného zadania. Generátor zadaní musí generovať zadania s rozumným počtom existujúcich riešení. Aplikácia si ukladá výsledky práce žiaka, ponúka možnosť priebežnej kontroly a prehľad hodnotenia úspešnosti. Prvé testovanie s deťmi v triede sa predpokladá v apríli, druhé testovanie v triede sa predpokladá v júni.

Vedúci: RNDr. Peter Borovanský, PhD.
Konzultant: RNDr. Dagmar Môťovská, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Dátum zadania: 02.10.2019

Dátum schválenia: 14.10.2019 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie

Čestne prehlasujem, že som bakalársku prácu s názvom „Softvérová podpora vyučovania matematiky Hejného metódou – prostredie Stavby z kociek“ vypracovala samostatne pod vedením vedúceho bakalárskej práce, s použitím uvedených zdrojov.

V Bratislave, 31.05.2020

_____

Monika Vlčková

Poďakovanie

Veľmi rada by som sa poďakovala môjmu školiteľovi RNDr. Petrovi Borovanskému, PhD. za umožnenie spracovania tak zaujímavej a peknej témy a tiež za všetky rady a odbornú pomoc, ktoré mi poskytol pri písaní bakalárskej práce. Rovnako tak by som sa chcela poďakovať konzultantke RNDr. Dagmar Môťovskej, PhD. za možnosť nahliadnuť do jej triedy a za jej pomoc lepšie porozumieť Hejného metóde. Taktiež ďakujem Bc. Martine Bodišovej, vďaka ktorej som mala možnosť aplikáciu otestovať na žiakoch, čo dodalo práci zmysel. V neposlednom rade by som sa chcela poďakovať svojej rodine a blízkym, ktorí ma celý čas podporovali.

Monika Vlčková

ABSTRAKT

VLČKOVÁ, Monika: Softvérová podpora vyučovania matematiky Hejného metódou – prostredie Stavby z kociek (Bakalárska práca) – Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. – Školiteľ: RNDr. Peter Borovanský, PhD.: FMFI UK, 2020, 42 strán

Cieľom práce je vytvorenie mobilnej aplikácie generujúcej postupne zložitejšie úlohy rešpektujúc princípy Hejného metóde v prostredí Stavby z kociek. Aplikácia má slúžiť na rozšírenie učebných materiálov pre žiakov prvého stupňa základnej školy. Je vyvinutá v prostredí Unity, na základe zásad tvorby edukačného softvéru a pravidiel Hejného metódy. Aplikácia bola testovaná žiakmi základnej školy.

Kľúčové slová: matematika, Hejného metóda, Stavby z kociek, mobilná aplikácia, Android

ABSTRACT

VLČKOVÁ, Monika: Educational software for Hejny's method of mathematics teaching – environment Cube Buildings (Bachelor thesis) – Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics. – Supervisor: RNDr. Peter Borovanský, PhD.: FMFI UK, 2020, 42 pages

The aim of this thesis is to design and implement a mobile application creating sequences of more-and-more complex tasks based on the Cube Buildings environment respecting the Hejny's method principles. The application is used to enrich the first-grade teaching materials of elementary school. It is developed in the Unity game engine respecting principles of educational software and Hejny's method. The application was tested on elementary school pupils.

Keywords: mathematics, Hejny's method, Cube Buildings, mobile application, Android

Obsah

Zoznam obrázkov 9
Úvod 10
1. Východiská 11
1.1. Edukačný softvér 11
1.2. Hejného metóda vyučovania matematiky 12
1.3. Stavby z kociek 15
1.4. Podobné existujúce riešenia 15
1.4.1. Materiálne didaktické pomôcky 15
1.4.2. Úlohy z matematiky pre deti na základných školách 16
1.5. Predchádzajúce bakalárske práce 17
1.5.1. Bilandia 17
1.5.2. Parkety 18
1.6. Použité technológie 19
1.6.1. Unity 19
1.6.2. C# 19
1.6.3. Rider 19
2. Návrh 21
2.1. Úvodná obrazovka 21
2.2. Rozdelenie do úrovní 22
2.2.1. Prvá úroveň 23
2.2.2. Druhá úroveň 24
2.2.3. Tretia úroveň 25
2.2.4. Štvrtá úroveň 26
2.3. Obrazovka dokončenej úrovne 26
2.4. Editor vlastných úloh 27
3. Implementácia 29
3.1. Štruktúra aplikácie 29
3.2. Ovládanie 30
3.3. Implementácia úrovní 33
3.4. Generátor a kontrola úloh 34
3.5. Editor 35
4. Testovanie 36
4.1. Priebeh testovania 36
4.2. Výsledky 37

Záver 38
Zdroje 39
Prílohy 40

Zoznam obrázkov

Obrázok 1: Príklad úlohy v prostredí Stavby z kociek z učebnice pre 1. roč. ZŠ, Fraus.....15
Obrázok 2: Úlohy z matematiky pre deti na základných školách 16
Obrázok 3: Bilandia 17
Obrázok 4: Parkety 18
Obrázok 5: Úvodná obrazovka – zamknuté úrovne.....22
Obrázok 6: Úvodná obrazovka – odomknuté úrovne 22
Obrázok 7: Zobrazený panel s nulovacími tlačidlami 22
Obrázok 8: Horná lišta úloh druhej úrovne 23
Obrázok 9: Oznámenie – správne vyriešená úloha.....23
Obrázok 10: Oznámenie – nesprávne vyriešená úloha 23
Obrázok 11: Prvá úroveň – vygenerovaná úloha.....24
Obrázok 12: Prvá úroveň – označená správna odpoveď.....24
Obrázok 13: Druhá úroveň – prvý typ úlohy 25
Obrázok 14: Druhá úroveň – vyriešený druhý typ úlohy.....25
Obrázok 15: Ikona stavby kociek 25
Obrázok 16: Ikona búrania kociek.....25
Obrázok 17: Tretia úroveň – vyriešený prvý typ úlohy.....26
Obrázok 18: Tretia úroveň – druhý typ úlohy 26
Obrázok 19: Štvrtá úroveň – prvý typ úlohy 26
Obrázok 20: Štvrtá úroveň – vyriešený druhý typ úlohy 26
Obrázok 21: Obrazovka dokončenej tretej úrovne 27
Obrázok 22: Editor na tvorbu úlohy prvej úrovne.....28
Obrázok 23: Editor na tvorbu úlohy druhej úrovne.....28
Obrázok 24: Editor na tvorbu úlohy tretej úrovne.....28
Obrázok 25: Editor na tvorbu úlohy štvrtej úrovne 28
Obrázok 26: Oznámenie o nesprávne vytvorenej úlohe 28
Obrázok 27: Navigácia medzi scénami.....29
Obrázok 28: BuildCube.cs 31
Obrázok 29: CameraMove.cs 32
Obrázok 30: funkcia Awake() v skripte Data.cs.....34

Úvod

Technológie sa stali súčasťou v mnohých oblastiach našich životov a rovnako tak aj súčasťou života detí. V dnešnej dobe batoľa s mobilom v ruke nie je žiadnou výnimkou. Keď príde dieťa do školy, dokáže ovládať mobil a počítač s absolútnou ľahkosťou a niekedy aj lepšie ako samotný učiteľ. Toto sa dá využiť a výučbu obohatiť o používanie edukačných softvérov. Je veľmi dôležité, aby boli deti vedené k rozumnému využívaniu týchto výdobytkov modernej doby a používali ich aj na vzdelávanie a nie len na zábavu. Preto je potrebné, aby existovali kvalitné výukové aplikácie, ktoré by sa mohli využívať aj ako učebné pomôcky na školách. Výučba na školách sa modernizuje – využívajú sa počítače, interaktívne tabule, tablety, preto by bolo jednoduché takéto aplikácie do výučby zaviesť. Naša aplikácia by mala byť jednou z nich.

Aby bola aplikácia vhodná a dobre využiteľná, musí sa opierať o určité zásady edukačného softvéru a rovnako tak spracúvať učebnú tému. Naša aplikácia vychádza z prostredia Hejného metódy – Stavby z kociek. Hejného metóda je už sama o sebe veľmi hravá a pre deti veľmi zaujímavá. Naša aplikácia by teda mala byť pre deti aspoň rovnako zábavná a náučná.

Využívanie aplikácie počas vyučovania je pre deti veľmi zaujímavé obohatenie klasickej hodiny. No jej používanie, nemá len výhody, ale aj nevýhody. Pre deti je najprirodzenejšie učiť sa z reálnych skúseností a interakciou s druhými ľuďmi, zatiaľ čo moderné technológie nie sú pre človeka prirodzené. Pri využívaní technológií vo výučbe sa deti nedokážu tak sústrediť, prichádzajú o interakciu s ostatnými a potláča to v nich kreativitu. Preto by aplikácia mala slúžiť na obohatenie a nie sa stať náhradou klasickej výučby.

V našej práci sme sa v prvej kapitole venovali edukačnému softvéru, Hejného metóde a predchádzajúcim podobným systémom, z ktorých naša aplikácia vychádza. V druhej kapitole sme sa venovali návrhu a v tretej implementácii štyroch úrovní a editora vlastných úloh. Zavedenie aplikácie do praxe a jej testovanie na žiakoch sme opísali v poslednej kapitole.

1. Východiská

V nasledujúcej kapitole opíšeme základné pojmy týkajúce sa našej práce. Vysvetlíme, čo je edukačný softvér a aké základné podmienky by mal dobrý softvér spĺňať. Podrobnejšie opíšeme Hejného metódu a jej 12 princípov, keďže je veľmi dôležité jej dobre porozumieť. Priblížime si prostredie Stavby z kociek. Tiež sa pozrieme na podobné existujúce riešenia našej práce a predchádzajúce bakalárske práce, ktoré taktiež spracovávali jedno z prostredí Hejného metódy. A nakoniec opíšeme technológie, ktoré sme sa rozhodli použiť pri riešení tejto práce.

1.1. Edukačný softvér

Edukačný softvér je všeobecný výraz pre každý program, ktorý sa využíva v oblasti vzdelávania. Do edukačných softvérov zaraďujeme nielen programy vytvorené a určené špecificky pre vyučovanie, ale aj programy, ktoré neboli vytvorené pre tento účel, ale sú využívané pri učení a učení sa. Edukačným softvérom sa teda môže stať aj taký program, ktorý na to vôbec nebol určený pri jeho tvorbe. Aby bol softvér vhodný na používanie vo vyučovacom procese, mal by podľa [1]:

- mať primerané používateľské prostredie, ktoré je primerané veku používateľa a účelu používania,
- produktívne využívať možnosti vizualizácie, používať vizuálne manipulovateľné objekty, vizualizáciu údajov, vizualizáciu stavu práce ...,
- byť interaktívny, teda reagovať na naše požiadavky a riadenie, spolupracovať s používateľom,
- byť otvorený a nie zamknutý voči novým aktivitám, inej grafike, vlastným zadaniam ...,
- podporovať náš didaktický zámer (poskytovať spätnú väzbu, ...),
- podporovať rozmanitosť a atraktívnosť aktivít,
- byť koncentrovaný na danú tému a účel (nezaťažovať používateľa a učiteľa inými problémami a nerozptyľovať tak ich pozornosť),
- poskytovať rastúce úrovne náročnosti a podporovať individuálny prístup žiaka.

1.2. Hejného metóda vyučovania matematiky

Hejného metóda začala vznikať vďaka Vítovi Hejnému a jeho nespokojnosti so spôsobom, akým sa žiaci učia [2]. Nepáčilo sa mu, že namiesto pochopenia učiva sa radšej učia vzorce naspamäť. Preto začal vyvíjať novú metódu výučby matematiky. Experimentoval s neštandardnými typmi úloh a tie skúšal na žiakoch a svojom synovi. Myšlienky tejto metódy boli kvôli politickej situácii publikované až neskôr, v roku 1987 synom Milanom Hejným a jeho spolupracovníkmi, ktorí sa tejto téme začali venovať.

Táto metóda je založená na inom princípe ako bežná výučba matematiky. Zameriava sa na tvorbu mentálnych matematických schém, riešením rozličných na to vhodných úloh a komunikáciou ich riešenia so spolužiakmi. Vychádza z 12-tich základných princípov pre učenie sa samostatne a s radosťou. Tieto základné princípy sú: [3]

- Budovanie schém: dieťa vie aj to, čo sme ho neučili
  V ľudskom živote je úplne prirodzené používať mentálne schémy. Podľa nich človek rozmýšľa, rozhoduje sa a vie riešiť problémy, ktoré by bez nich nedokázal. Vysvetliť, čo je to schéma, sa dá na jednoduchom príklade. Keď sa niekoho opýtate, koľko krát na ceste do práce odbočí doľava, tak pravdepodobne nebude vedieť odpovedať hneď, ale keď si tú cestu v mysli predstaví, dokáže to relatívne rýchlo spočítať a povie správnu odpoveď. Rovnako je to s matematickými schémami. V Hejného metóde sa používa viacero schém, s ktorými deti pravdepodobne už prišli do kontaktu v skoršom detstve.

- Práca v prostrediach: učíme sa opakovanou návštevou
  Hejného metóda využíva na učenie približne 25 rôznych prostredí, ktoré sú tvorené na seba nadväzujúcimi úlohami. Úlohy v prostrediach sú pútavé a žiaci dokážu vyriešiť mnoho úloh bez toho, aby mali dojem, že pracujú a sú nútení úlohy riešiť. Často je to pre nich zábava a majú skôr pocit, že sa hrajú. Tým, že v prostrediach pracujú opakovane, stávajú sa im veľmi dobre známymi a nadobúdajú v nich istotu. Vďaka tomu strácajú strach z matematiky.

- Prelínanie tém: matematické zákonitosti neizolujeme
  Keďže jednotlivé matematické javy spolu veľa krát súvisia, neizolujú sa, ale precvičujú sa vo viacerých prostrediach a každé z prostredí v sebe zahŕňa viacero javov. Tým pádom sa s javmi žiaci stretávajú opakovane a v rôznych formách, čím

sa zvýšia šance, že im porozumie každé dieťa. Taktiež si potom môže každé vybrať, akým spôsobom mu najviac vyhovuje riešiť jednotlivé úlohy.

- Rozvoj osobnosti: podporujeme samostatné uvažovanie detí
  Pri riešení úloh nehodnotí správnosť riešenia úlohy učiteľ, ale nechá to na žiakov, aby si to vydiskutovali medzi sebou. Často si pri vysvetľovaní svojho riešenia ostatným sami uvedomia chybu v postupe, pokiaľ nebol správny. Vďaka tomu sa učia samostatne myslieť, diskutovať, vypočuť si názor druhých a rešpektovať sa. Keďže v škole trávia deti veľa času, tak je veľmi dôležité, aby tu boli aj vychovávané a nielen vzdelávané.

- Skutočná motivácia: keď „neviem“ a „chcem vedieť“
  V tradičnej výučbe matematiky deti nebývajú motivované k učeniu sa kvôli vedomostiam, ale skôr kvôli tomu, aby mali dobré známky alebo zo strachu zo zlej známky. Hejného metóda sa snaží nevyvíjať na deti tento tlak. Deti sú od narodenia zvedavé a chcú sa učiť, len to musí byť takým spôsobom, aby ich to zaujímalo a bavilo. Pokiaľ sa zameriava vyučujúci na chyby, deti to prestane baviť. V tejto metóde sa úspech chváli a viaže sa k nemu príjemná a radostná emócia. V prípade neúspechu sa len bez negatívnych emócií zanalyzuje, prečo vznikol.

- Reálne skúsenosti: staviame na vlastných zážitkoch dieťaťa
  Nie nadarmo sa hovorí, že jedenkrát zažiť je viac, ako tisíckrát počuť. Deti sa učia reálnymi skúsenosťami už od útleho veku. To sa dá využiť aj pri výučbe matematiky. Preto je vhodné, aby si žiaci preriešili úlohy a sami na princíp riešenia prišli. Pokiaľ by sa im riešenie prezradilo, uložilo by sa im to len do krátkodobej pamäte a o týždeň by to pravdepodobne už nevedeli. Keď na to prídu sami budú si to pamätať dlhodobo. Preto Hejného metóda necháva deti, aby samé prišli na riešenie úloh.

- Radosť z matematiky: výrazne pomáha pri ďalšej výučbe
  Deti, ktoré sú motivované vnútorne, dokážu premýšľať a rozhodovať sa samostatnejšie ako tie, ktoré sú motivovanejšie zvonka. Vnútorná motivácia sa vytvára pri vlastnom objavovaní. Úlohy v Hejného metóde sú tvorené tak, aby boli dostatočne jednoduché na to, aby ich deti vedeli samé vyriešiť a zároveň dostatočne

zložité, aby sa chvíľu potrápili, kým na riešenie prídu. Vďaka tomu majú radosť, že dokázali úlohu vyriešiť a baví ich v riešení ďalších pokračovať.

- Vlastný poznatok: má väčšiu váhu ako ten prevzatý
  Žiak matematiku objavuje. Pokúša sa riešiť úlohy vlastnou cestou a vďaka tomu získava skúsenosti. O svojich teóriách sa rozpráva so spolužiakmi a overuje si ich na ďalších úlohách. To má za následok, že rozumie tomu, čo robí.

- Rola učiteľa: sprievodca a moderátor diskusií
  Bežne sme zvyknutí na to, že učiteľ vykladá látku, vysvetľuje, čo sa má ako riešiť. V Hejného metóde je to inak. Učiteľ zadáva úlohy, pozoruje, ako sa jednotlivým žiakom darí pri ich riešení a dáva pozor, aby sa nikto nenudil a každý niečo robil. Nehodnotí, či žiaci úlohu vyriešili správne, ale vyzýva deti k tomu, aby oni sami diskusiou prišli na to, či je daná úloha vyriešená správne alebo nie.

- Práca s chybou: predchádzame u detí zbytočnému strachu
  Robiť chyby je pre človeka úplne prirodzené a o to viac, ak sa učí nové veci. Preto netreba brať chybu ako niečo zlé, ale ako niečo, z čoho sa môžeme poučiť. Pokiaľ si deti nájdu chybu, či už sami alebo za pomoci spolužiakov a pochopia, prečo ju urobili, poučia sa a nabudúce ju už nespravia.

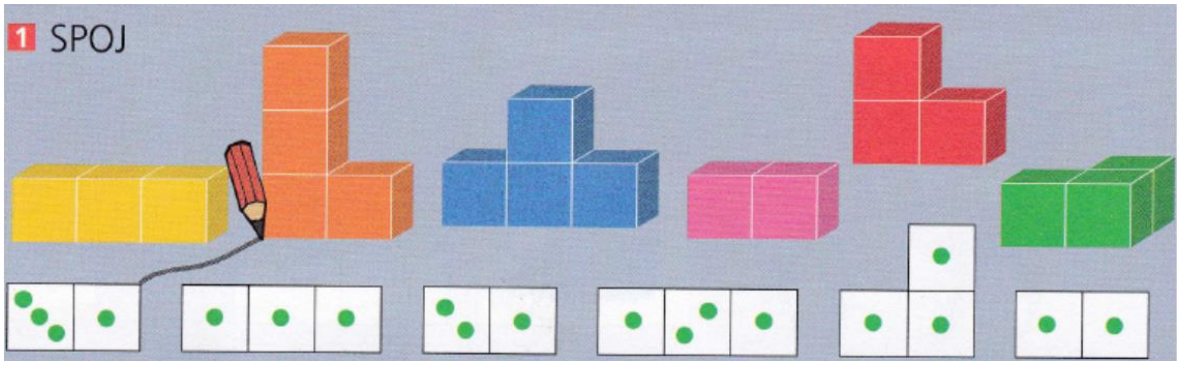
- Primerané výzvy: pre každé dieťa zvlášť podľa jeho úrovne
  Keďže jednotlivé úlohy sú rôznych obťažností od jednoduchých po zložité, každé dieťa dokáže vyriešiť aspoň nejaké z nich a zároveň šikovnejšie deti môžu riešiť aj zložitejšie zadania. Vďaka tomu žiaci nemajú z matematiky strach a šikovnejší sa nenudia. Pre každého sa nájde primeraná úloha, ktorá je preňho výzvou.

- Podpora spolupráce: poznatky sa rodia vďaka diskusii
  Každé dieťa má na výber, či chce pracovať samostatne alebo radšej vo dvojici alebo v skupine s kamarátmi. Spolupráca je vítaná a nie je považovaná za podvádzanie. Pokiaľ sa aj rozhodnú pracovať samostatne, po vyriešení úlohy sa radi zapoja do diskusie a podelia sa o svoj spôsob riešenia.

1.3. Stavby z kociek

Jedným z mnohých prostredí Hejného metódy je geometrické prostredie stavby z kociek, ktoré podporuje priestorovú predstavivosť a tiež zasahuje do aritmetiky [4]. V tomto prostredí sa žiaci zoznamujú s vlastnosťami kociek a telies. Tiež sa zdokonaľujú v počítaní a priestorovej predstavivosti. Stavbou z kociek sa rozumie len taký objekt, ktorý pozostáva z kociek rovnakej veľkosti a jednotlivé kocky sa navzájom dotýkajú vždy celou stenou.

Toto prostredie pozostáva z rozličných typov úloh. Najskôr sa deti učia správne vyjadrovať, stavajú si, porovnávajú navzájom postavené stavby. Učia sa, že stavba má podlažia. Postupne sa pridávajú termíny pôdorys a plán. Priúčajú sa podľa plánu postaviť stavbu a naopak zapísať plán postavenej stavby. Postupne sa dostanú aj k termínom ako bokorys, nárys, objem a povrch stavby, ktorým vďaka prostrediu jednoduchšie porozumejú. Na obrázku 1 môžeme vidieť jeden typ úlohy z učebnice pre 1. ročník ZŠ od vydavateľstva Fraus.



Obrázok 1: Príklad úlohy v prostredí Stavby z kociek z učebnice pre 1. roč. ZŠ, Fraus [4]

1.4. Podobné existujúce riešenia

V nasledujúcej časti opíšeme existujúce riešenia, ktoré by mohli konkurovať našej práci. Pozrieme sa na klasické materiálne pomôcky, ktoré sa využívajú v školách, ale aj na softvér, v ktorom je možné precvičovať si úlohy v prostredí stavby z kociek.

1.4.1. Materiálne didaktické pomôcky

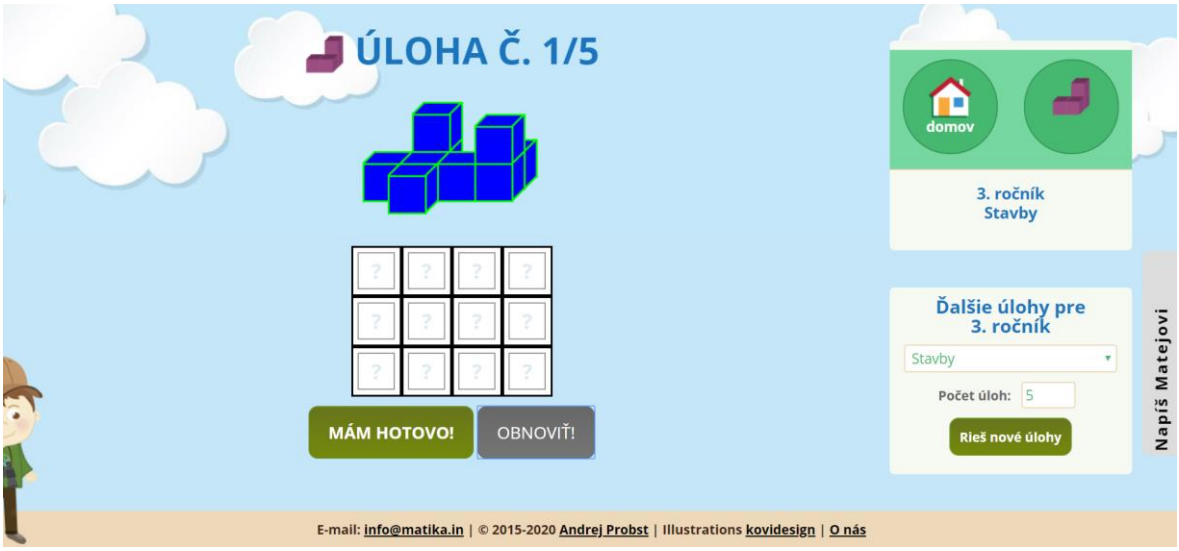
Na riešenie úloh v prostredí stavby z kociek je vhodné využiť kocky, ktoré môžu byť plastové alebo drevené. Existujú kocky, ktoré sú určené špeciálne na učenie sa Hejného metódou. Tieto má škola väčšinou zakúpené a pani učiteľka ich nosí na hodiny. A takmer

v každej domácnosti sa nájdu kocky, s ktorými sa deti hrali už odmalička. Tieto kocky si môžu deti na hodinu priniesť samé alebo ich aspoň využívať doma. Deti môžu riešiť úlohy z pracovného zošita alebo si tvoriť aj vlastné a zapisovať si plániky na papier alebo si navzájom diktovať, ako má stavba vyzerať.

1.4.2. Úlohy z matematiky pre deti na základných školách

Jediné softvérové riešenie spracúvajúce prostredie stavby z kociek, ktoré sa nám podarilo nájsť, sa nachádza na stránke matika.in [5]. Na tejto stránke sa nachádzajú úlohy pre žiakov na základnej škole v prostrediach Hejného metódy. Prostredia a úlohy sú rozdelené podľa ročníkov, takže si každý vie vybrať to, čo potrebuje.

Stavby z kociek sa na stránke nachádzali len pre tretí ročník, ale neboli spracované veľmi kvalitne. Po vybraní si daného prostredia sa spustila úloha, ktorá pozostávala zo stavby a obdĺžnikového plánu, kam bolo treba zapísať počty podlaží. Stavba bola statická, nedalo sa s ňou otáčať. Do plániku sa zapisovalo dosť komplikovane. Najskôr bolo treba prejsť kurzorom myši nad štvorček a potom zapísať číslo. Tiež bolo na prekážku, že bolo treba vyplniť všetky štvorce, aj tie, kde mala byť nula. Taktiež úlohy vôbec negradovali, všetky boli rovnakej obtiažnosti a len jedného typu. Tieto úlohy určite nemôžu byť plnohodnotnou náhradou pracovných zošitov. Na obrázku 2 je možné vidieť, ako vyzerá úloha z tejto stránky.



Obrázok 2: Úlohy z matematiky pre deti na základných školách

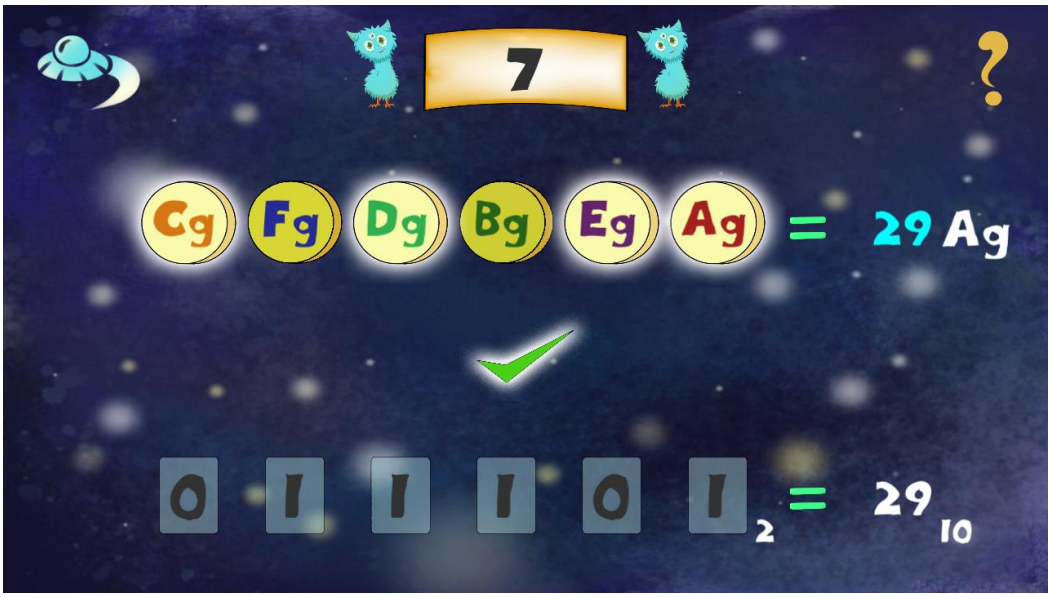
1.5. Predchádzajúce bakalárske práce

V nasledujúcej časti sa pozrieme na predchádzajúce bakalárske práce spracúvajúce jedno z prostredí Hejného metódy vyučovania matematiky. Opíšeme, ako sú spracované, čo sa nám na nich páči a naopak, čo by sme urobili inak.

1.5.1. Bilandia

Autorka bakalárskej práce [6] z roku 2019, ktorá spracúva prostredie Bilandia, je Júlia Gablíková. Aplikácia, ktorú vytvorila, spracúva dvojkovú sústavu veľmi peknou a hravou formou. Pozostáva zo štyroch úrovní, v ktorých sa deti pomocou grošov učia dvojkovej sústave. Táto mena pozostáva len z mincí takej hodnoty ktorá je vždy dvojnásobkom predchádzajúcej a označuje sa písmenami od začiatku abecedy a písmenom g. Teda Ag má hodnotu 1, Bg hodnotu 2, Cg hodnotu 4 a tak ďalej.

Deti sa najskôr zoznamujú s prevodom grošov vyšších hodnôt do počtu grošov najnižšej hodnoty a naopak, potom sa učia sčitovať a odčitovať a výsledok zapísať tak, aby bola každá hodnota použitá maximálne raz. V tretej úrovni sa oboznamujú s tým, ako prepísať groše do jednotiek a núl. V poslednej úrovni majú za úlohu sčitovať a odčitovať už v dvojkovej sústave. Gradácia je urobená veľmi dobre, posledná úroveň by mohla potrápiť aj stredoškolákov, ktorí majú s dvojkovou sústavou problém. Graficky je aplikácia tiež spracovaná pekne, groše sú odlíšené aj farebne, čo žiakom zjednodušuje orientáciu. Na obrázku 3 je zobrazená jedna z úloh v tretej úrovni Bilandie.

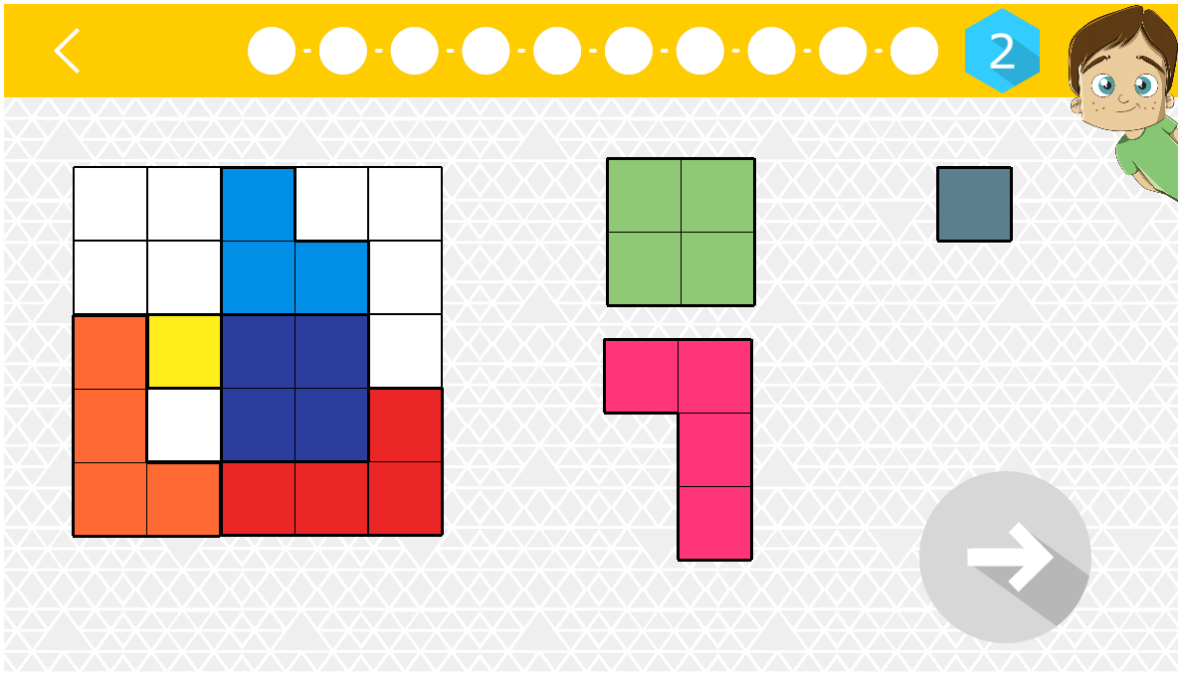


Obrázok 3: Bilandia

1.5.2. Parkety

Autorka bakalárskej práce [7] z roku 2018, ktorá spracúva prostredie Parkety, je Andrea Spišáková. Aplikácia slúži na precvičovanie si pokrývania obdĺžnikovej siete rôznymi parketami, ktoré sú tvorené rôznym počtom stenami k sebe priliehajúcich štvorcov. Aplikácia pozostáva zo štyroch úrovní. V každej úrovni je pridané nové pravidlo. V prvej úrovni je k dispozícii presne toľko parkiet, aby pokryli sieť, v druhej úrovni sa nachádzajú aj parkety naviac. V tretej úrovni je sieť už s časti pokrytá a treba nájsť všetky riešenia. A v poslednej úrovni sa môže stať, že úloha nemá riešenie. Náročnosť v jednotlivých úrovniach narastá po každej prejdenej úlohe, čo zabezpečí, že sa užívateľ nezačne nudiť.

Graficky je aplikácia spracovaná pekne a jednoducho. Sprievodná hovoriaca postavička chlapca jasne a rýchlo vysvetlí, čo treba robiť. Tento spôsob vysvetlenia je pre deti najvhodnejší, keďže nemusia nič čítať, a tak je aplikácia vyhovujúca aj pre menšie deti. Úlohy sa ovládajú veľmi jednoducho potiahnutím parkety na miesto a pustením. Jediné, čo by sa dalo vytknúť, sú animácie postavičky medzi jednotlivými úlohami a páčky kontroly všetkých riešení v tretej a štvrtej úrovni a rovnako tak v editore vlastnej úlohy, ktoré boli veľmi pomalé a pôsobili rušivo. Obrázok 4 zobrazuje úlohu v prvej úrovni.



Obrázok 4: Parkety

1.6. Použité technológie

V nasledujúcej časti sa pozrieme na technológie, ktoré sme si vybrali na vývoj našej aplikácie. Keďže sa jedná o mobilnú aplikáciu s 3D prvkami, rozhodli sme sa pre platformu Unity. Na kompilovanie skriptov Unity používa jazyk C#, takže je najefektívnejšie v tomto jazyku aj programovať. Vhodnou voľbou bol pre nás editor Rider, pretože máme dobré skúsenosti s editormi od JetBrains.

1.6.1. Unity

Unity je jedna z najobľúbenejších platforiem na vývoj hier [8]. Širokou škálou funkcionality, cross-platformovými vlastnosťami a svojou jednoduchosťou si získal vysokú popularitu nielen medzi začínajúcimi, ale aj medzi profesionálnymi vývojármi. Unity umožňuje tvorbu ako 2D, tak aj 3D hier, ktoré vďaka komponentom možno vyvíjať a škálovať rýchlejšie a efektívnejšie. V základe prichádza Unity s vlastnou C# skriptovacou API a vstavanou integritou s Visual Studiom. Keďže je Unity na trhu už od roku 2005, disponuje vysokým množstvom stiahnuteľných assetov, plug-inov a iných nástrojov. Bohatou dokumentáciou, množstvom videí, návodov a intuitívnym vývojovým prostredím výrazne šetrí čas potrebný pre zorientovanie sa. Zároveň možnosťou vyvíjať a distribuovať hru bez značných obmedzení v rámci licenciácie je Unity vhodnou vstupnou bránou do sveta vývoja hier.

1.6.2. C#

C# je silne typový objektovo orientovaný programovací jazyk vyvinutý spoločnosťou Microsoft [9]. Je to open source, je moderný, výkonný a pomocou neho možno vytvárať programy pre Windows, Linux, MacOS ale aj mobilné platformy, napr. Android. Jednou z výhod je podobnosť syntaxe s Javou a porovnateľný výkon s C++. Jazyk je veľmi obľubený aj vďaka veľkej základni užívateľov operačného systému Windows.

1.6.3. Rider

Rider je na trhu pomerne nový C# skriptovací editor od JetBrains [10]. Široké spektrum funkcionalít z neho robí plnohodnotné integrované vývojové prostredie (Integrated

Development Environment - IDE). Svojou svižnosťou, možnosťou výberu ako prednastaveného IDE pre Unity a jeho cross-platformovosťou sa Rider radí medzi absolútnych favoritov u Unity vývojárov.

2. Návrh

V nasledujúcej kapitole opíšeme grafický návrh jednotlivých obrazoviek a taktiež fungovanie našej aplikácie. Snažili sme sa vytvoriť čo najjednoduchšie prostredie, ktoré by bolo intuitívne ovládateľné a tiež graficky zaujímavé pre vekovú kategóriu, pre ktorú je táto aplikácia určená, teda pre deti prvého stupňa. Preto sme sa rozhodli, že stavby budú reprezentované ako hrady a celá aplikácia sa bude niesť v rytierskom duchu. Nepoužili sme žiaden text. Úlohy nemajú návod na riešenie. Stavili sme na minimalistický dizajn a na to, že deti sú zvedavé, skúsia všetko postláčať a veľmi rýchlo prídu na to, ako sa má aplikácia ovládať.

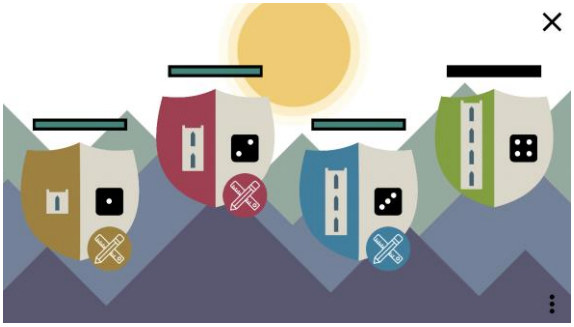
2.1. Úvodná obrazovka

Po spustení aplikácie sa zobrazí úvodná obrazovka, na ktorej sa nachádzajú štyri štíty označené štvorčekom s počtom bodiek zodpovedajúcim číslu úrovne. Tieto štíty majú tri stavy. Na začiatku je celý čierny s bielym zámkom a po kliknutí naň sa nič nestane. Samozrejme, prvá úroveň tento stav nemá, aby sa hra mohla začať hrať. Ak je úroveň odomknutá, je na štíte zobrazený obrázok - stavba s poschodiami zodpovedajúcimi číslu danej úrovne a štvorček s rovnakým počtom bodiek. Tretí stav nastane, keď je úroveň úspešne prejdená. Vtedy sa zobrazí v pravej dolnej časti okrúhle tlačidlo, na ktorom sa nachádza ceruzka a pravítko. Kliknutím na toto tlačidlo sa dostaneme do editoru úlohy danej úrovne, kde si užívateľ môže vytvoriť vlastnú úlohu daného typu. Nad každým štítom sa nachádza obdĺžnik, ktorý znázorňuje zelenou výplňou napredovanie v danej úrovni.

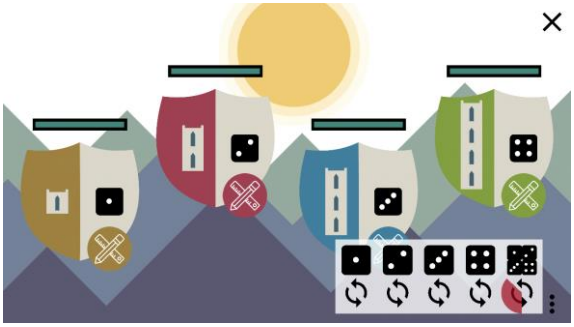
Na tejto obrazovke sa nachádzajú ešte ďalšie dve tlačidlá. V pravom hornom rohu je to X, ktoré slúži na vypnutie aplikácie a v pravom dolnom rohu tri bodky. Po kliknutí na toto tlačidlo sa zobrazí panel, na ktorom sa nachádza päť tlačidiel v tvare dvoch šípok. Nad každým tlačidlom je označenie úrovne, ktorá sa po dvojsekundovom stlačení tlačidla vynuluje. Aby bolo jasné, že stlačenie musí byť dlhé, cez tlačidlo sa vykresľuje v smere hodinových ručičiek červený kruh. Prvé štyri tlačidlá len danej úrovni nastavia napredovanie na nulu a všetko, čo bolo odomknuté, tak aj ostane. Piate tlačidlo slúži na obnovenie východiskového stavu aplikácie. Po opätovnom stlačení tlačidla s tromi bodkami sa panel skryje.

Nižšie na obrázku 5 môžeme vidieť východiskový stav úvodnej obrazovky, na obrázku 6 odomknuté všetky štyri úrovne a na obrázku 7 zobrazený panel s nulovacími tlačidlami.

Obrázok 5: Úvodná obrazovka – zamknuté úrovne



Obrázok 6: Úvodná obrazovka – odomknuté úrovne



Obrázok 7: Zobrazený panel s nulovacími tlačidlami

2.2. Rozdelenie do úrovní

Ako sme už v predchádzajúcej kapitole spomínali, aplikácia je rozdelená do štyroch úrovní. Jednotlivé úrovne majú veľmi podobnú štruktúru. Obrazovku by sme mohli pomyselne rozdeliť na tri časti. Hornú lištu, ľavú a pravú polovicu.

V hornej lište sa nachádza v ľavom rohu šípka smerujúca doľava. Po kliknutí na toto tlačidlo sa prepne obrazovka naspäť na úvodnú. V pravom rohu je fajka, ktorá slúži na odsúhlasenie vyriešenej úlohy. Medzi týmito tlačidlami je ukazovateľ napredovania. Ak je úroveň vyššia ako jedna, na ľavej strane ukazovateľa sa nachádza štvorček s počtom bodiek označujúcich predchádzajúcu úroveň. Týmto tlačidlom sa môžeme prepnúť do tejto úrovne. Podobne, po prejdení celej úrovne, sa na pravej strane objaví tlačidlo v tvare štvorčeka s počtom bodiek nasledujúcej úrovne, ktorým sa do nej môžeme dostať. Tiež sa vedľa tohto tlačidla objaví ďalšie, označené ceruzkou a pravítkom, ktorým môžeme prejsť na obrazovku editora úlohy danej úrovne. V niektorých úlohách, v ktorých to je potrené, sa naľavo od fajky

nachádza ešte okrúhla šípka, ktorá slúži na vynulovanie úlohy. Na obrázku 8 je znázornená horná lišta druhej úrovne.

Obrázok 8: Horná lišta úloh druhej úrovne

Ďalšie, čo majú všetky úrovne spoločné, je oznámenie o správnosti, respektíve nesprávnosti vyriešenej úlohy. Toto oznámenie sa zobrazí na pár sekúnd po stlačení tlačidla fajka. Pokiaľ je riešenie správne, zobrazí sa veselý rytier s nápisom Super (obrázok 9), ak bolo naopak nesprávne, rytier je smutný a pod ním je nápis Nesprávne (obrázok 10).

Obrázok 9: Oznámenie – správne vyriešená úloha

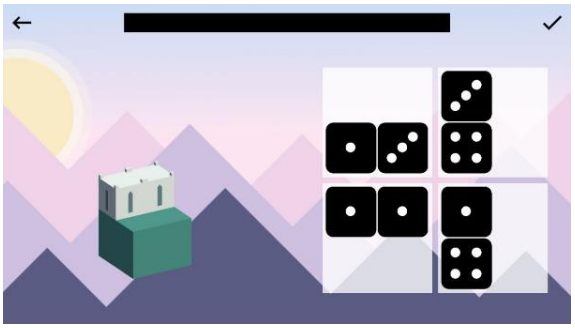
Obrázok 10: Oznámenie – nesprávne vyriešená úloha

2.2.1. Prvá úroveň

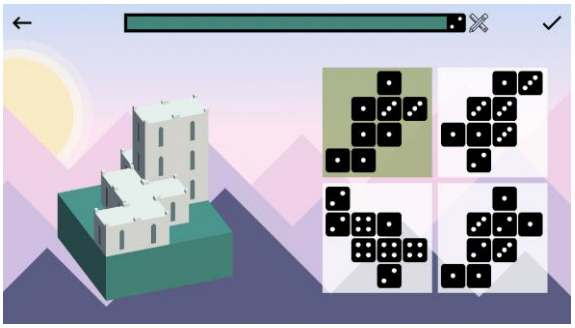
Úlohou v prvej úrovni je nájsť správny plán stavby. V zadaní sa nachádza stavba a štyri plány. Zo začiatku je veľmi jednoduché úlohami prejsť, lebo stavby pozostávajú len z pár kociek a sú veľkosti 2x2. Postupne sa zvyšuje veľkosť pôdorysu aj celkový počet kociek.

Na ľavej strane obrazovky sa nachádza hrad (stavba z kociek) na podložke. Celá stavba sa dá pohybom prsta po pravej časti obrazovky otáčať ľubovoľným smerom. V pravej časti sú štyri štvorčeky umiestnené do štvorca. V každom štvorčeku je jeden plán stavby. Práve jeden z týchto plánov je plán vľavo postaveného hradu. Po kliknutí na ktorýkoľvek plán sa podsvieti jeho pozadie na zeleno. Po opätovnom kliknutí sa podsvietenie zmení na pôvodné biele. Ak už je zakliknutý jeden plán a klikne sa na ďalší, zruší sa vyznačenie prvého a podsvieti sa novo vybraný plán. Pre kontrolu správnosti treba kliknúť na fajku. Ak

bol výber správny, hráč dostane oboznámenie a načíta sa ďalšia úloha. Na obrázkoch 11 a 12 môžeme vidieť, ako vyzerá prvá úroveň a gradáciu náročnosti jej úloh.



Obrázok 11: Prvá úroveň – vygenerovaná úloha



Obrázok 12: Prvá úroveň – označená správna odpoveď

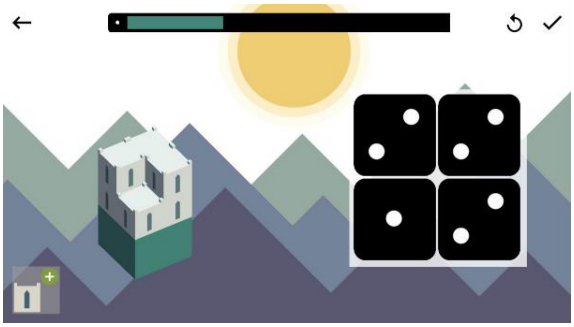
2.2.2. Druhá úroveň

Táto úroveň má veľmi podobné zadanie úloh ako predchádzajúca. Namiesto vyberania z predložených plánov, je úlohou plán vytvoriť. Prvá úroveň sa dá veľmi ľahko prejsť aj tipovaním, keďže plány sú len štyri a pri nesprávne vyriešenej úlohe, ju dieťa dostane znova, aby sa dokázalo učiť na vlastných chybách. Takže po troch nesprávnych výberoch ostane posledný správny plán. V druhej úrovni sa už presvedčíme, že dieťa naozaj rozumie tomu, čo je plán stavby. Naviac druhá úroveň obsahuje aj opačný typ úloh, a teda postaviť stavbu podľa plánu.

Aj rozloženie je veľmi podobné ako v predchádzajúcej úrovni, len v pravej časti sa miesto štyroch plánov nachádza len jeden. Generuje dva typy úloh.

Prvý typ vygeneruje stavbu a v pravej časti prázdny plán, ktorý treba správne vyplniť (obrázok 13). V prázdnom pláne sa nachádzajú štvorčeky. Po kliknutí na ktorýkoľvek z nich sa zmení na čierny s jednou bodkou. Po opätovnom klikaní sa počet bodiek zvýši vždy o jednu. Ak má štyri bodky a znova sa naň klikne alebo sa tlačidlo podrží dlhšie, hodnota sa vynuluje.

Obrázok 13: Druhá úroveň – prvý typ úlohy



Obrázok 14: Druhá úroveň – vyriešený druhý typ úlohy

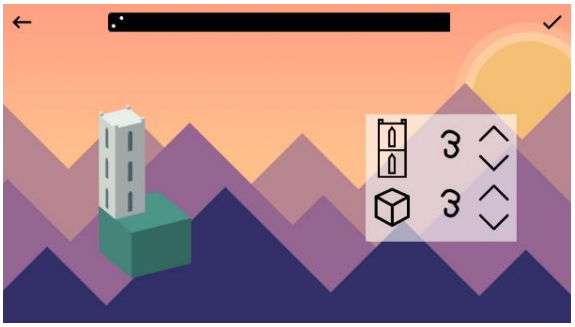
Druhý typ vygeneruje plán a úlohou je podľa neho postaviť hrad (obrázok 14). Hrad treba postaviť na podstavec v ľavej časti obrazovky. Jednotlivé kocky sa postavia po kliknutí na podstavec alebo ktorúkoľvek stenu už postavenej kocky. Kocky sa nedajú postaviť tak, aby vyčnievali cez podstavec a ani tak, aby boli vo vzduchu, teda pod kockou musí byť kocka alebo podstavec. V prípade potreby búrania jednotlivých kociek sa v ľavom dolnom rohu nachádza tlačidlo, ktorým sa dá prepínať medzi stavaním a búraním kociek. Na tlačidle je znázornená kocka hradu a v pravom rohu zelený kruh s bielym plus, ktorým prepneme na stavbu kociek (obrázok 15) a červený kruh s bielym mínus, ktorým môžeme prepnúť na búranie (obrázok 16). Celá stavba sa dá zbúrať naraz po stlačení zatočenej šípky v pravom hornom rohu obrazovky.

Obrázok 15: Ikona stavby kociek

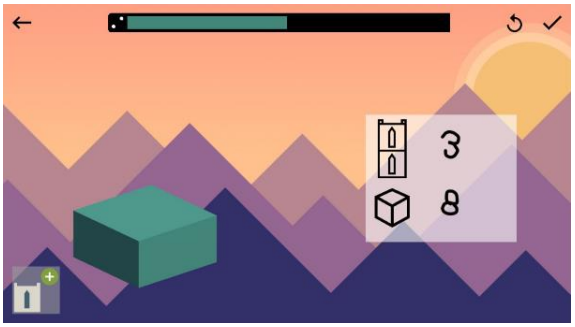
Obrázok 16: Ikona búrania kociek

2.2.3. Tretia úroveň

Tretia úroveň tiež pozostáva z dvoch typov úloh. V prvom type je vygenerovaná stavba a v pravej časti tabuľka, do ktorej treba vpísať hodnoty – do prvého riadku počet podlaží a do druhého celkový počet kociek, z ktorých stavba pozostáva (obrázok 17). Hodnota sa mení pomocou dvoch tlačidiel, šípky hore a dole, ktoré sú umiestnené v každom riadku. Šípka hore zvýši a dole zníži hodnotu. Druhý typ vygeneruje tabuľku s hodnotami a úlohou je podľa nich postaviť správnu stavbu (obrázok 18).



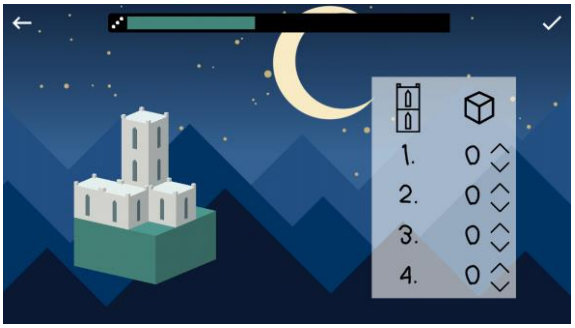
Obrázok 17: Tretia úroveň – vyriešený prvý typ úlohy



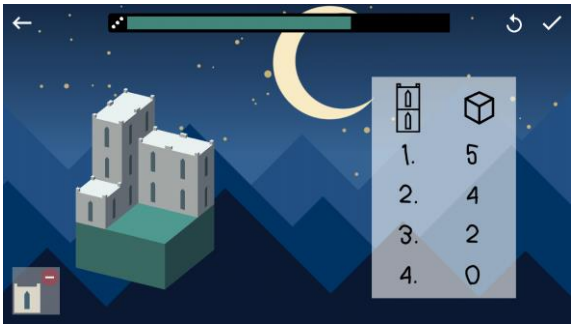
Obrázok 18: Tretia úroveň – druhý typ úlohy

2.2.4. Štvrtá úroveň

Táto úroveň je veľmi podobná ako predchádzajúca, len namiesto hodnôt počet podlaží a kociek sa v tabuľke nachádzajú počty kociek v jednotlivých podlažiach. Rovnako sa v tejto úrovni nachádzajú dva typy úloh. Jedna generuje stavbu a podľa nej treba vyplniť tabuľku (obrázok 19) a druhá, naopak, generuje tabuľku a treba postaviť stavbu (obrázok 20).



Obrázok 19: Štvrtá úroveň – prvý typ úlohy

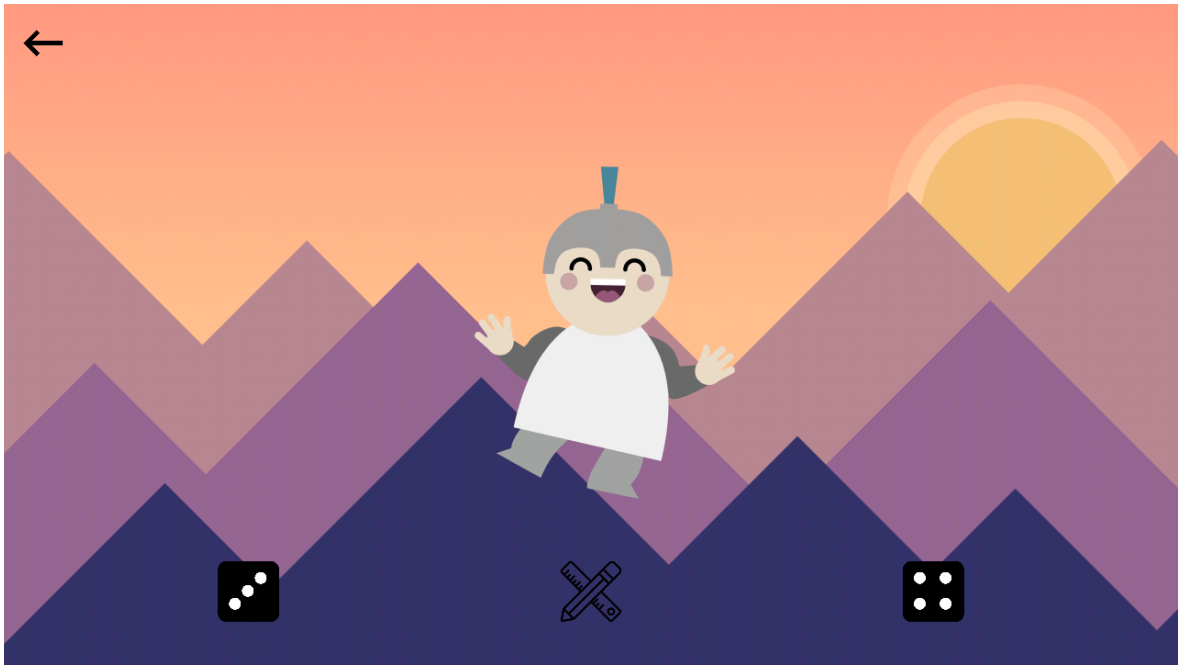


Obrázok 20: Štvrtá úroveň – vyriešený druhý typ úlohy

2.3. Obrazovka dokončenej úrovne

Po dokončení desiatej úlohy ktorejkoľvek úrovne sa odomkne nasledujúca úroveň a editor vlastných úloh. Vtedy sa na hornej lište objavia tlačidlá, ktorými sa vieme dostať do odomknutej úrovne aj editora. Keďže sú tieto tlačidlá malé a nie každý by si ich musel všimnúť, po úspešne dokončenej desiatej úlohe sa obrazovka automaticky prepne na obrazovku dokončenej úrovne (obrázok 21). Na nej sa nachádza poskakujúci tlieskajúci rytier a štyri tlačidlá. Vďaka nim má užívateľ štyri možnosti. Môže pokračovať v riešení

úloh v aktuálnej úrovni, vytvoriť si vlastnú úlohu, prejsť do ďalšej úrovne alebo sa šípkou v ľavom hornom rohu vrátiť na úvodnú obrazovku.



Obrázok 21: Obrazovka dokončenej tretej úrovne

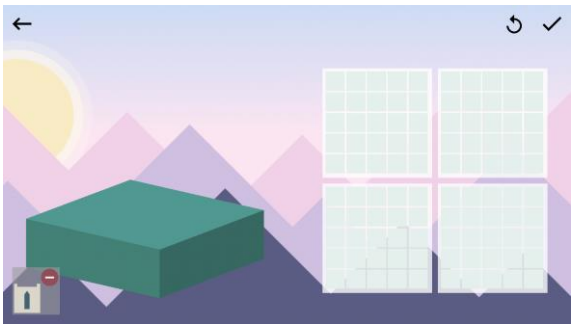
2.4. Editor vlastných úloh

Po prejdení ktorejkoľvek úrovne sa odomkne možnosť vytvorenia si vlastnej úlohy, ktorú potom žiak môže dať vyriešiť kamarátovi. Rozloženie je takmer identické ako v danej úrovni.

Pre prvú úroveň treba postaviť budovu a vyplniť všetky štyri plány, z ktorých práve jeden musí byť plánom postavenej stavby (obrázok 22). Stavba musí pozostávať aspoň z jednej kocky.

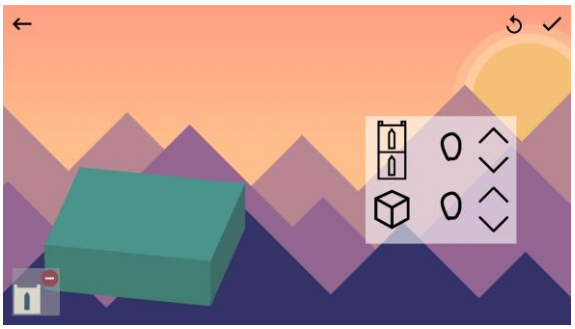
V druhej, tretej a štvrtej úrovni treba postaviť stavbu alebo vyplniť pravú časť, teda v druhej úrovni plán (obrázok 23) a v tretej a štvrtej tabuľku (obrázok 24 a 25). Aby bola úloha v týchto úrovniach správna, musí byť splnená práve jedna z podmienok. Tabuľka musí byť vyplnená správne tak, aby sa budova podľa nej dala postaviť. Počet kociek nemôže byť menší ako počet podlaží a v nižšom podlaží nemôže byť menej kociek ako vo vyššom.

Po vytvorení úlohy treba zakliknúť fajku. Pokiaľ je úloha vytvorená správne, automaticky sa spustí úloha pripravená na riešenie. Ak je chybná, zobrazí sa oznámenie (obrázok 26) a úloha sa potom môže upraviť a znovu odsúhlasiť.

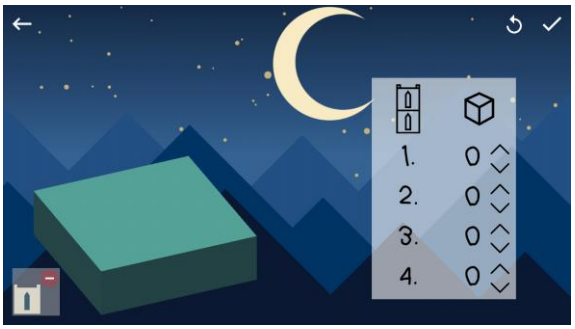


Obrázok 22: Editor na tvorbu úlohy prvej úrovne

Obrázok 23: Editor na tvorbu úlohy druhej úrovne



Obrázok 24: Editor na tvorbu úlohy tretej úrovne



Obrázok 25: Editor na tvorbu úlohy štvrtej úrovne

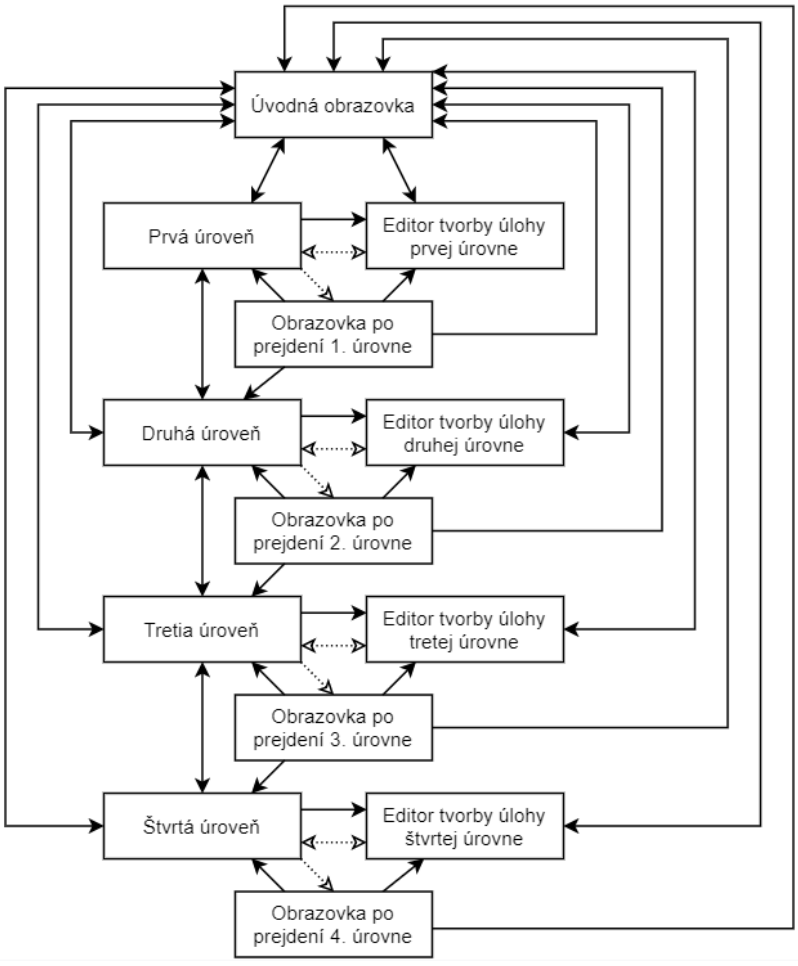
Obrázok 26: Oznámenie o nesprávne vytvorenej úlohe

3. Implementácia

V nasledujúcej kapitole si priblížime implementačnú úroveň aplikácie. Opíšeme jej štruktúru a spôsob akým je naprogramované ovládanie, ako sú implementované úrovne, ako funguje generovanie a kontrola úloh a aj editor na tvorbu úloh. Keďže je aplikácia vyvíjaná v prostredí Unity, priblížime si aj niektoré jeho funkcionality, ktoré využívame.

3.1. Štruktúra aplikácie

Aplikácia pozostáva z trinástich scén: z úvodnej obrazovky, štyroch úrovní, štyroch editorov na tvorbu vlastnej úlohy a štyroch obrazoviek po prejdení úrovne. Medzi jednotlivými scénami sa dá prepínať pomocou tlačidiel. Obrázok 27 opisuje, akým smerom sa dá medzi jednotlivými scénami pohybovať. Bodkované šípky znázorňujú prepnutie, ktoré nerobí užívateľ vedome, ale spraví to aplikácia automaticky, a to v dvoch prípadoch pri vytvorení vlastnej úlohy v editore a pri prejdení celej úrovne.



Obrázok 27: Navigácia medzi scénami

Každá scéna pozostáva z prvkov nazývaných *GameObjects*. Môžu byť rozličných typov, napríklad *Canvas, Button, Image, EventSystem,* ale aj 3D objekty ako *Cube*. Všetky tieto objekty už obsahujú predpripravené komponenty, ktoré ich špecifikujú, ale vieme im ich pridať aj my. Jedným z komponentov je aj *Script*. *Script* je trieda zdedená z *MonoBehaviour*. Môže obsahovať funkcie ako napríklad *Start(), Update()...,* vďaka ktorým vieme jednotlivým objektom pridať funkcionalitu.

3.2. Ovládanie

Aplikácia je vyvíjaná v prvom rade pre tablety s operačným systémom Android. V prípade potreby je ale dobre ovládateľná na mobilných zariadeniach a na počítačoch, keďže odchytávame dotyk na obrazovku aj myš. Na odchytávanie interakcií s aplikáciou slúži predpripravený objekt *EventSystem*. Každý objekt, s ktorým možno interagovať, obsahuje *Box Collider*. Keďže obrazovka je plocha a aplikácia obsahuje objekty definované tromi súradnicami, po kliknutí na obrazovku sa vyšle lúč (*Raycast*) kolmý na obrazovku. Tento lúč je určitej dĺžky a pokiaľ sa v jeho smere nachádza objekt, ktorý obsahuje *Box Collider,* dokážeme o tomto objekte získať informácie.

V našej aplikácii sa nachádzajú dva objekty *Ground* a *Cube,* ktoré obsahujú *Box Collider*. *Ground* je kváder, ktorý slúži ako podstavec pre stavbu a celá stavba je postavená z jednotlivých kociek *Cube*. Vďaka lúčom dokážeme stavať kocky na podstavec alebo na iné kocky a tiež ich búrať. O túto funkcionalitu sa stará trieda *BuildCube* zdedená od *MonoBehaviour*. Trieda obsahuje funkciu *Update(),* ktorá sa zavolá pri každom vykreslení obrazovky, teda niekoľko krát za sekundu. V nej, pokiaľ sa na obrazovke vykonal dotyk, vyšleme lúč z miesta dotyku. Ak narazil na objekt, zistíme súradnice dotyku. Tieto súradnice musíme prepočítať na súradnice stredu kocky, ktorej sme sa dotkli, ak chceme kocku zbúrať, alebo na súradnice stredu kocky, ktorá sa bude dotýkať steny, ktorej sme sa dotkli. Pri stavaní musíme skontrolovať, či na danom mieste môžeme kocku postaviť. Na to slúži funkcia *CanBuildCube(Vector3 position).* Ak je dané miesto vyhovujúce, vytvoríme na vypočítaných súradniciach nový objekt *Cube*.

Všetky kocky sú vytvárané ako deti prázdneho objektu *Building,* ktorý obsahuje *Script BuildCube*. Vďaka tomu vieme nimi prechádzať a zistiť, ktorej z kociek patrí vypočítaná pozícia. Toto využívame napríklad vo funkcii *DestroyCube(Vector3 position),* ktorá prejde všetkými kockami a pokiaľ narazí na takú, čo má hľadanú pozíciu, zničí ju.

Na obrázku 28 je zobrazený zjednodušený zdrojový kód triedy *BuildCube*.

```
BuildCube : MonoBehaviour{
    public GameObject newCube;
    public bool locked = false;
    public bool destroyMode = false;
    public Sprite destroyImage, buildImage;
    public Button button;
    public int SIZE = 5;
    private Vector2 TouchStartPosition, TouchEndPosition;
    private Vector3 MouseStartPosition, MouseEndPosition;

    void Update(){
        SetPositions();

        if ((!locked && Input.touchCount>0 && Input.GetTouch(0).phase==TouchPhase.Ended && !TouchMoved())
            || (!locked && Input.touchCount == 0 && Input.GetMouseButtonUp(0))&& !MouseMoved()){
            RaycastHit hit;
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            if (Physics.Raycast(ray, out hit)){
                Vector3 offset = new Vector3(0.5f, 0f, 0.5f);
                Vector3 cubePosition = hit.point + hit.normal / 2.0f;

                if (SIZE % 2 == 0) cubePosition += offset;
                cubePosition.x = (float) Math.Round(cubePosition.x, MidpointRounding.AwayFromZero);
                cubePosition.y = (float) Math.Round(cubePosition.y, MidpointRounding.AwayFromZero);
                cubePosition.z = (float) Math.Round(cubePosition.z, MidpointRounding.AwayFromZero);
                if (SIZE % 2 == 0) cubePosition -= offset;

                Vector3 touchedCubePosition = cubePosition - hit.normal;

                if (destroyMode) DestroyCube(touchedCubePosition);
                else if(CanBuildCube(cubePosition))
                    Instantiate(newCube, cubePosition, Quaternion.identity, this.transform);
            }
        }
    }

    private void DestroyCube(Vector3 cubePosition){
        foreach (Transform cube1 in transform)
        {
            if (cube1.transform.position.x.Equals(cubePosition.x) &&
                (cube1.transform.position.y >= (cubePosition.y)) &&
                cube1.transform.position.z.Equals(cubePosition.z))
            {
                GameObject.Destroy(cube1.gameObject);
            }
        }
    }

    private bool TouchMoved(){…}

    private bool MouseMoved()(){…}

    private void SetPositions()(){…}

    private bool UnderCubeIsCube(Vector3 cubePosition){…}

    private bool CanBuildCube(Vector3 cubePosition) (){…}

    public void ChangeDestroyModeState(){…}

    public void Lock(){…}

    public void Unlock(){…}
}
```

Obrázok 28: BuildCube.cs

Ďalšou triedou zabezpečujúcou interakciu používateľa s aplikáciou, konkrétne otáčanie budovy, je *CameraMove* (obrázok 29). Názov sa môže zdať trochu mätúci, no v skutočnosti nehýbeme budovou, ale kamerou, ktorá nám zabezpečuje vidieť budovu. Na súradniciach 0,0,0 sa nachádza prázdny objekt *Pivot*. *Pivot* je rodičovským objektom *Main Camera*. *Main Camera* má určené relatívne súradnice podľa svojho rodičovského objektu. *CameraMove* mení rotáciu *Pivotu* a vďaka tomu sa otáča aj kamera.

Keďže je aplikácia určená pre dotykové zariadenia, odchytávame udalosť *Input.GetTouch(0)*. *Input.GetTouch(int index)* vracia objekt typu *Touch*. Z neho vieme zistiť rôzne vlastnosti, my využívame fázu dotyku a *deltaPosition*, čo je rozdiel súčasnej pozície a pozície v predchádzajúcom update.

Aby bola rovnako dobre funkčná a ovládateľná aj na počítači, rozhodli sme sa odchytávať aj udalosti myši. Funkcia *Input.GetMouseButon(0)* vráti hodnotu *true*, pokiaľ je stlačené ľavé tlačidlo na myši, keďže sme zadali parameter 0. Potom vďaka funkciám *Input.GetAxis("Mouse X")* a *Input.GetAxis("Mouse Y")* zistíme ako sa pohla myš a vďaka tomu vieme meniť rotáciu správnym smerom.

```
public class CameraMove : MonoBehaviour{
    public float intensity = 0.3f;
    public float dempening = 10.0f;
    private Touch touch;
    private Rect touchZone;
    private Vector3 rotation;

    private void Start(){
        touchZone = new Rect(0, 0, Screen.width*0.6f, Screen.height);
    }

    void Update() {

        if(Input.touchCount > 0){
            touch = Input.GetTouch(0);
            if(touch.phase == TouchPhase.Moved && touchZone.Contains(touch.position)){
                rotation.y += touch.deltaPosition.x * intensity;
                rotation.x -= touch.deltaPosition.y * intensity;
            }
            if (rotation.x < 0f) rotation.x = 0f;
            else if (rotation.x > 90f) rotation.x = 90f;
        }
        else if(Input.GetMouseButton(0) && (Input.GetAxis("Mouse X")!= 0 || Input.GetAxis("Mouse Y")!= 0){
            rotation.y += Input.GetAxis("Mouse X") * intensity * 50;
            rotation.x -= Input.GetAxis("Mouse Y") * intensity * 50;
            if (rotation.x < 0f) rotation.x = 0f;
            else if (rotation.x > 90f) rotation.x = 90f;
        }
        Quaternion rotationXY = Quaternion.Euler(rotation.x, rotation.y, 0);
        this.transform.parent.rotation = Quaternion.Lerp(this.transform.parent.rotation, rotationXY,
Time.deltaTime*dempening);

    }
}
```

Obrázok 29: CameraMove.cs

Zvyšok aplikácie je ovládaný už pomocou tlačidiel, ktoré sa nachádzajú v objekte *Canvas*. Všetky tlačidlá obsahujú predprogramovaný *Button Script*, v ktorom sa nachádza element *On Click()*, kde sa môže zadať objekt a z neho vybrať *Script* a funkcia, ktorá sa má po stlačení tlačidla vykonať.

3.3. Implementácia úrovní

Všetky úrovne pozostávajú z veľmi podobných komponentov. Každá úroveň je tvorená jednou scénou, v ktorej sa nachádzajú objekty *DataControl, EventSystem, Canvas, Ground, Pivot, Building, Background* a *Background Cam*. *Pivot* a *Canvas* obsahujú ešte ďalšie objekty. V *Pivote* sa nachádza *Main Camera*, vďaka ktorej sa zobrazuje stavba.

V *Canvase* sa nachádzajú všetky tlačidlá, *ProgressBar*, objekty, do ktorých sa vykresľujú plány, tabuľky a *notificationImage*, ktorý je viditeľný len v určitých situáciách, teda všetky 2D objekty okrem pozadia. Keďže *Canvas* sa vykresľuje úplne navrchu, keby mal nastavené pozadie, tak by sme nemohli vidieť stavbu. Toto sme vyriešili objektami *Background* a *Background Cam*. *Background Cam* je kamera, ktorá má v zábere obrázok *Background*. Parameter *Depth* kamery *Background Cam* sme nastavili na nižší ako na kamere *Main Camera* a vďaka tomu sa jej obsah zobrazuje za obsahom *Main Camera*. Takto teda máme pozadie vzadu a stavbu pred ním.

Ground je kváder, ktorý slúži ako podložka pre stavbu. Jeho veľkosť sa mení podľa toho, akú veľkú budovu generujeme.

Building je prázdny objekt, do ktorého generujeme kocky, aby sme nimi v prípade potreby mohli jednoduchšie prechádzať.

DataControl je objekt so skriptom *Data*, ktorý slúži na prenášanie dát medzi scénami. Funkcia *Awake()* (obrázok 30) v skripte *Data* zabezpečuje, že ak už objekt *DataControl* existuje, novovytvorený objekt sa zničí a uchová sa pôvodný z predchádzajúcej scény, vďaka čomu sa prenesú údaje uložené v *Data*. *Data* načítava súbor *data.txt*, v ktorom je päť čísel. Prvé štyri označujú koľko úloh je v jednotlivých úrovniach vyriešených. Piate označuje to, ktorá najvyššia úroveň je prejdená. *Data* teda uchováva informácie o stave jednotlivých úrovní a tiež slúži na prenášanie dát pri vytvorení vlastnej úlohy v editore.

```
void Awake()
{
    if (DATA == null)
    {
        DontDestroyOnLoad(gameObject);
        DATA = this;
    }
    else if (DATA != this)
    {
        Destroy(gameObject);
    }
}
```

Obrázok 30: funkcia Awake() v skripte Data.cs

3.4. Generátor a kontrola úloh

O generovanie úloh sa stará trieda *Building*, keďže základom každej úlohy je budova. Táto trieda obsahuje funkciu *Start()*, ktorá sa vykoná hneď pri načítaní scény úlohy. V tejto funkcii sa najskôr vygeneruje plán stavby, ktorý je typu *Plan*.

Plan obsahuje funkcie ako G*enerate(int size, int floors, int min, int max), isBuilding(int[,] buildingPlan, int min, int max), rotateRandom(int[,] plan), equals(int[,] plan1, int[,] plan2)* a ďalšie. Konštruktor *Plan(int size, int floors, int progress)* vygeneruje plán budovy *int plan[,]* o veľkosti *size* a *size* zavolaním funkcie *Generate()*. Keďže stavba je dobrá len vtedy, ak sú všetky kocky prepojené stenou, je potrebné kontrolovať, či je vygenerovaný plán správny, pretože generujeme tak, že do poľa náhodne uložíme čísla od 0 do *floors*. Potom zavoláme funkciu *isBuilding(plan)*, ktorá skontroluje, či vygenerovaný plán je správny. Táto funkcia najskôr skontroluje, či plán budovy pozostáva z požadovaného intervalu kociek a potom zavolá rekurzívnu funkciu *check()*, ktorá funguje na princípe prehľadávaní do hĺbky. Funkcia *Generate()* generuje *plan* dovtedy, kým nie je správny.

Ak sa jedná o prvú úroveň, *PlanGenerator* vytvorí ďalšie tri plány modifikáciou plánu stavby. Zabezpečí, aby každý plán bol odlišný. Funkcie na modifikáciu plánu sa nachádzajú v triede *Plan*. Po vygenerovaní plánov sa zavolá funkcia *Draw()* z triedy *PlanDrawer*, ktorá vykreslí plány do objektov *buildingPlan* v scéne prvej úrovne.

V prvej úrovni a každej párnej úlohe druhej, tretej a štvrtej úrovne sa postaví budova. Na to slúži funkcia *Build()* v triede *Building*. Táto funkcia inicializuje objekty *Cube* na správnych súradniciach podľa plánu. V druhej úrovni sa vygeneruje aj plán do objektu *buildingPlan* s prázdnymi políčkami, na ktoré sa dá klikať a meniť im hodnotu. O to sa stará funkcia *InitializeEmpty()* v triede *PlanDrawer*. V tretej a štvrtej úrovni sa v tabuľkách *Table*

zobrazia tlačidlá, ktorými sa môžu meniť hodnoty v riadkoch tabuľky. Po kliknutí sa zavolá funkcia *increase()* alebo *decrease()* zo skriptu *Floors*.

V nepárnych úlohách druhej úrovne sa vygenerovaný plán vykreslí do objektu *buildingPlan*. V tretej a štvrtej úrovni sa do tabuliek zapíšu hodnoty, odpovedajúce vygenerovanému plánu.

Na kontrolu správnosti vyriešenej úlohy slúži skript *Confirm*, ktorý obsahuje funkcie *CheckLevel1(), CheckLevel2(), CheckLevel3()* a *CheckLevel4()*. Každá funkcia slúži na kontrolu jednej úrovne. Funkcie sú volané tlačidlom *checkButton* v scéne úrovne. Funkcia skontroluje správnosť úlohy a zavolá funkciu *show(bool correct)* zo skriptu *Notification* s parametrom *correct* s hodnotou *true*, ak je úloha vyriešená správne a *false*, ak nesprávne. Funkcia *show* nastaví objektu *notificationImage* parameter *enabled* na hodnotu true, vďaka čomu sa obrázok zobrazí. Podľa parametra *correct* sa zobrazí buď obrázok s oznámením o správne alebo nesprávne vyriešenej úlohe.

3.5. Editor

Skript *OwnTask* sa stará o správne fungovanie editora vlastných úloh. Funkcia *Start()* v editore pre prvú a druhú úroveň inicializuje v objektoch *buildingPlan* dvadsaťpäť štvorcov obsahujúcich skript *Squere*, ktorého funkcia *Click()* zabezpečuje po kliknutí menenie obrázku v štvorci. *OwnTask* tiež obsahuje funkciu *LoadCreatedTask()*, ktorá načíta vytvorenú úlohu, pokiaľ je vytvorená správne. Správnosť úlohy kontroluje funkcia *IsValidLevel(int level)*.

4. Testovanie

Aby sme zaistili, či je aplikácia dobre ovládateľná a funkčná, je veľmi dôležité jej otestovanie čo najviac užívateľmi, najlepšie v takej vekovej kategórii, pre akú je určená. Naša aplikácia mala byť dva krát testovaná v triede so žiakmi prvého stupňa. Prvýkrát počas vývoja a druhý krát po dokončení a spracovaní pripomienok žiakov. Keďže boli školy zavreté, prvé testovanie sa nepodarilo uskutočniť. Toto testovanie sme nahradili tak, že sme aplikáciu dávali viackrát testovať známym a podľa ich pripomienok sme aplikáciu upravovali. Takéto testovanie samozrejme nedokáže nahradiť testovanie s deťmi, pre ktoré je aplikácia určená, a ktoré ju môžu vnímať úplne inak. Nakoniec sa nám podarilo uskutočniť aspoň druhé testovanie.

4.1. Priebeh testovania

Testovanie prebiehalo cez konferenčný hovor so žiakmi šiesteho ročníka. Toto testovanie nebolo ideálne, pretože žiaci, ktorí aplikáciu testovali, boli starší ako cieľová skupina. Napriek obavám z toho, že bude pre nich jednoduchá a nudná, sme boli prekvapení ako na ňu príjemne reagovali.

Testovali sme s dvomi triedami a obidve testovania prebiehali veľmi podobne. V jednej skupine si žiaci hneď všimli, že v hovore sa nachádza niekto, kto nie je ich spolužiak a boli zvedaví, kto je to. Keď im pani učiteľka povedala, že budú testovať hru, podobne ako aj pred týždňom, veľmi sa potešili. Keď sme dostali slovo, najskôr sme im predstavili našu prácu a vysvetlili im, čo by sme od nich potrebovali. Žiaci, ktorí mali Android zariadenia, si aplikáciu stiahli z Google Play. Tí, ktorí takúto možnosť nemali, no mali aspoň počítač s operačným systémom Windows, si do počítača stiahli exe súbor. Zvyšní, ktorí nemali ani jednu z možností, riešili úlohy s nami, vďaka zdieľanej obrazovke. Podľa ich inštrukcií sme aplikáciu ovládali. Keď nevedeli, čo majú presne robiť, buď si poradili navzájom, alebo sme im pomohli my. Žiaci boli veľmi šikovní, za necelých dvadsať minút sme prešli všetky úrovne. Kvôli krátkemu času sme už nestihli vyskúšať editor na tvorbu vlastných úloh.

Aplikácia sa im veľmi páčila. Pochválili ju, že je veľmi pekne urobená, dobre sa im ovláda a že ich aj bavila. No dali nám aj veľmi dobré pripomienky na zmenu. Všetci sa zhodli na tom, že by aplikácia pri spustení novej úlohy mala obsahovať návod, najlepšie interaktívny alebo aspoň popis. K tomu sme dostali aj pripomienku, že by mal byť po

slovensky, ale aj anglicky a mohol by byť aj v nemčine. Tiež by sa im páčilo, keby aplikácia obsahovala hudbu a zvukové efekty. Ďalším nápadom na zlepšenie od žiakov bolo, aby sme v štvrtej úrovni v tabuľke zapísali podlažia naopak, teda jednotku dole a štvorku hore, aby to bolo prirodzenejšie.

4.2. Výsledky

Počas testovania sme zistili, že žiakom nebolo úplne jasné, čo majú v jednotlivých úrovniach robiť. Prvá úroveň bola celkom jasná. Tam rýchlo pochopili, že majú vybrať plán stavby. V druhej úrovni tiež nemali problém s prvou úlohou, kde mali nakresliť plán stavby. Druhá úloha im však robila problém. Nepochopili, že majú podľa plánu postaviť budovu, ale snažili sa odstrániť štvorce z plánu, no to im nešlo. Po objasnení, aké je zadanie, úlohu s ľahkosťou vyriešili. Tiež im trochu robilo problém tlačidlo na stavanie a búranie kociek. Bolo pre nich mätúce, že keď kocky stavajú, nachádza sa na ňom mínus a keď ich búrajú je na ňom plus. V tretej a štvrtej úrovni väčšina nerozumela, čo treba do tabuľky vypĺňať. Po krátkom vysvetlení však veľmi rýchlo úlohy vyriešili.

Vďaka testovaniu sme vypozorovali, že deťom veľmi chýbal návod. V prvých úlohách úrovní mali dosť veľké problémy s tým, že nevedeli čo majú robiť. Vďaka návodu by tento problém zanikol.

Testovanie bolo pre nás veľmi príjemnou skúsenosťou. Vidieť aplikáciu plniť svoj účel a dostávať podnetné nápady od detí bolo pre nás veľmi dôležitou spätnou väzbou. Veľmi nás potešilo, že sa aplikácia páčila starším deťom, pre aké bola určená.

Záver

Cieľom našej práce bolo vytvorenie výukovej mobilnej aplikácie, ktorá vychádza z princípov Hejného metódy a spracúva prostredie Stavby z kociek. Aplikáciu sme tvorili po naštudovaní si Hejného metódy a taktiež sme navštívili hodinu, kde touto metódou vyučujú, aby sme videli, ako funguje v praxi. Preštudovali sme si, ako by mal vyzerať dobrý edukačný softvér a zanalyzovali sme podobné predchádzajúce systémy. Vďaka týmto informáciám sme najskôr vytvorili návrh úrovní a vzhľad aplikácie. Následne sme podľa návrhu začali aplikáciu programovať. Naša aplikácia mala byť počas vývoja dvakrát testovaná na žiakoch základnej školy, no keďže boli školy zavreté, túto možnosť sme nemali. Namiesto toho sme aplikáciu dávali priebežne testovať rodinným príslušníkom a na základe ich pripomienok, sme aplikáciu upravovali. Po dokončení sme aplikáciu zverejnili v službe Google Play. Vďaka tomu sa naša aplikácia stala verejne prístupná a konečne sme mohli aplikáciu otestovať aj na deťoch. Testovanie prebehlo cez konferenčný hovor so žiakmi šiesteho ročníka základnej školy. Táto skúsenosť bola pre nás veľmi obohacujúca. Žiaci nám dali veľmi dobré rady na vylepšenie aplikácie a ich záujem a radosť nás veľmi potešil a dal zmysel našej práci.

V budúcnosti by sme chceli do aplikácie pridať viac úrovní s ďalšími typmi úloh. Tiež by sme chceli pridať interaktívny návod, ktorý by sa spustil po prejdení na novú úroveň. V neposlednom rade by sme medzi jednotlivé úrovne chceli pridať krátke animácie, ktoré by znázorňovali príbeh a zvukové efekty, aby tak bola aplikácia pre deti pútavejšia.

Zdroje

[1] Virtual Lab: Premeny triedy [online]. [cit. 4.5.2019]. Dostupné online na internete: http://www.virtual-lab.sk/claroline/claroline/backends/download.php?url=L1ByZW1lbmFfdHJpZWR5XzIucGRm&cidReset=true&cidReq=M_DT

[2] Co je to „Hejného metoda“? [online]. [cit. 4.5.2020]. Dostupné na internete: https://www.h-mat.cz/hejneho-metoda

[3] 12 klíčových principů [online]. [cit. 4.5.2020]. Dostupné na internete: https://www.h-mat.cz/principy

[4] Krychlove stavby [online]. [cit.5.5.2020]. Dostupné na internete: http://www.zslns.cz/userfiles/file/2015/hejn%C3%BD/ks.pdf

[5] Úlohy z matematiky pre deti na základných školách [online]. [cit. 5.1.2019]. Dostupné online na internete: https://www.matika.in/sk/

[6] Júlia Gablíková: Softvérová podpora vyučovania matematiky Hejného metódou – prostredie Bilandia, FMFI UK Bratislava, 2019

[7] Andrea Spišáková: Softvérová podpora vyučovania matematiky Hejného metódou – prostredie Parketovanie, FMFI UK Bratislava, 2018

[8] Scripting in Unity for experienced programmers [online]. [cit.7.5.2020]. Dostupné na internete: https://unity.com/how-to/programming-unity

[9] Introduction to the C# language and the .NET Framework [online]. [cit. 7.5.2020]. Dostupné online na internete: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[10] Rider [online]. [cit. 8.5.2020]. Dostupné online na internete: https://www.jetbrains.com/rider/

Prílohy

1. Zdrojový kód aplikácie, spustiteľné verzie .exe a .apk dostupné na internete: http://davinci.fmph.uniba.sk/~vlckova66/Bakalarka/#download

2. Verziu pre Android je možné priamo nainštalovať cez: https://play.google.com/store/apps/details?id=com.company.Kocky