UNIVERZA V LJUBLJANI
FAKULTETA ZA UPRAVO
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Diplomsko delo

RAZVOJ INTERAKTIVNE IZOBRAŽEVALNE APLIKACIJE ZA OTROKE

Matija Ozebek

Ljubljana, september 2014

UNIVERZA V LJUBLJANI
FAKULTETA ZA UPRAVO
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

DIPLOMSKO DELO

RAZVOJ INTERAKTIVNE IZOBRAŽEVALNE APLIKACIJE ZA OTROKE

Kandidat: Matija Ozebek
Vpisna številka: 04038923
Študijski program: univerzitetni študijski program Upravna informatika prva stopnja

Mentor: doc. dr. Matija Marolt

Ljubljana, september 2014

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Matija Ozebek, študent univerzitetnega študijskega programa Upravna informatika prva stopnja, z vpisno številko 04038923, sem avtor diplomskega dela z naslovom: Razvoj interaktivne izobraževalne aplikacije za otroke.

S svojim podpisom zagotavljam, da:

- je priloženo delo izključno rezultat mojega lastnega raziskovalnega dela;
- sem poskrbel, da so dela in mnenja drugih avtorjev oz. avtoric, ki jih uporabljam v predloženem delu, navedena oz. citirana v skladu s fakultetnimi navodili;
- sem poskrbel, da so vsa dela in mnenja drugih avtorjev oz. avtoric navedena v seznamu virov, ki je sestavni element predloženega dela in je zapisan v skladu s fakultetnimi navodili;
- sem pridobil vsa dovoljenja za uporabo avtorskih del, ki so v celoti prenesena v predloženo delo, in sem to tudi jasno zapisal v predloženem delu;
- se zavedam, da je plagiatorstvo – predstavljanje tujih del, bodisi v obliki citata bodisi v obliki skoraj dobesednega parafraziranja bodisi v grafični obliki, s katerimi so tuje misli oz. ideje predstavljene kot moje lastne – kaznivo po zakonu (Zakon o avtorski in sorodnih pravicah, Uradni list RS, št. 21/95), kršitev pa se sankcionira tudi z ukrepi po pravilih Univerze v Ljubljani in Fakultete za upravo;
- se zavedam posledic, ki jih dokazano plagiatorstvo lahko predstavlja za predloženo delo in za moj status na Fakulteti za upravo;
- je elektronska oblika identična s tiskano obliko diplomskega dela ter soglašam z objavo dela v zbirki »Dela FU«.

Diplomsko delo je lektorirala: Veronika Blažič.

Ljubljana, 03. 09. 2014

Podpis avtorja:

POVZETEK

Tema diplomske naloge je razvoj interaktivne izobraževalne aplikacije za otroke, ki vsebuje vse elemente kakovostnega gradiva za izobraževanje otrok. Najprej sem raziskal področje razvoja iger v izobraževalne namene ter predstavil njihove glave značilnosti skupaj z elementi, ki jih takšne igre morajo vsebovati. Nato sem na podlagi študija literature predstavil prednosti in slabosti izobraževanja otrok z uporabo iger v primerjavi s klasično obliko izobraževanja. Pripravil sem tudi pregled izobraževalnih gradiv za otroke, ki so na voljo na spletu v slovenskem jeziku, ter jih kritično ovrednotil. Sledi opis tehnologij, ki sem jih uporabil za razvoj aplikacije. Kot glavno razvojno orodje sem si izbral pogon za igre Unity, programsko kodo sem napisal v jeziku C# z uporabo razvojnega okolja MonoDevelop. Uporabil sem še program GIMP za manipulacijo s slikami ter Audacity za urejanje zvoka. Razvil sem aplikacijo na temo ljudske pravljice Janček Ježek, s katero se uporabnik spoznava preko celotne vsebine gradiva. Aplikacija je sestavljena iz uvodne scene, petih interaktivnih in dinamičnih glavnih scen, ki predstavljajo prizore iz zgodbe, štirih izobraževalnih iger za razvijanje otrokovih mentalnih sposobnosti, besedila in zvočnega zapisa zgodbe ter petih animacij, ki zgodbo predstavijo še v obliki interaktivnih risank. Razvita aplikacija je prilagojena otrokom ter spodbuja njihovo domišljijo, ustvarjalnost, smisel za raziskovanje in samostojnost pri reševanju nalog.

Ključne besede: izobraževanje otrok, interaktivnost, razvoj iger, Unity.

SUMMARY

DEVELOPMENT OF INTERACTIVE EDUCATIONAL APPLICATION FOR CHILDREN

The topic of this thesis is development of interactive educational application for children, that contains all the elements of high quality material for the education of children. Firstly I explored the area of game development for educational purposes and presented their main characteristics along with the elements, that such games must contain. Then I presented, based on literature study, main advantages and disadvantages of children's education with the use of games, compared to classical form of education. I have also prepared an overview of educational materials for children, that are available online in the Slovenian language, and made a critical evaluation. The following chapter contains description of technologies, which I used for application development. As the main development tool I chose Unity game engine, I wrote programming code in C# using MonoDevelop development environment. I've used GIMP program for image manipulation and Audacity for audio editing. I developed an application based on folk tale Janček Ježek, which the application user gets to know through entire content of this learning material. Application consists of the title scene, five interactive and dynamic main scenes, representing parts of the story, four educational games, that develop children's mental ability, story in text and audio format and five animations that presents the story in a form of an interactive cartoons. Developed application is adjusted to children's use and encourages their imagination, creativity, a sense of exploration and independence in problem solving.

Key words: children's education, interactivity, game development, Unity.

KAZALO

KAZALO PONAZORITEV

KAZALO SLIK

KAZALO PRIMEROV PROGRAMSKE KODE

SEZNAM UPORABLJENIH KRATIC

2D	dve dimenziji
3D	tri dimenzije
EUR	Evro – evropska denarna valuta
USD	Dolar – ameriška denarna valuta

SEZNAM TUJIH IZRAZOV

Animation controller	upravljavec animacij
Force over lifetime	sila tekom življenjske dobe
Particle system	sistem delcev
Rotation over lifetime	rotacija tekom življenjske dobe

1 UVOD

Večino otrok računalnik spremlja že od malih nog. Otroci uporabljajo računalnike predvsem za igranje računalniških iger. Posledično manj berejo in so bolj navajeni na interaktivne vsebine, kar marsikateremu učencu povzroča težave s koncentracijo in motivacijo pri klasični obliki učenja ter s sodelovanjem pri pouku. Ta težava se vse bolj upošteva tudi pri pripravi gradiva za pouk v šolah, kjer se vedno bolj uveljavlja učenje in utrjevanje snovi s pomočjo izobraževalnih aplikacij za otroke, najpogosteje v obliki računalniških iger, ki naredijo pouk za otroke bolj zanimiv in privlačen, učenci so bolj motivirani in se učijo bolj samostojno kot pri tradicionalni obliki učenja.

Problem, ki sem si ga zadal v okviru diplomske naloge, je raziskati področje izobraževalnih aplikacij za otroke ter odkriti njihove glavne značilnosti, prednosti in slabosti. Pridobljeno znanje sem nato uporabil pri razvoju izobraževalne aplikacije, ki otrokom na zabaven in interaktiven način pomaga pri učenju in razvijanju umskih sposobnosti. Aplikacija otroka izobražuje tudi na literarnem področju, saj temelji na slovenski ljudski pravljici Janček Ježek, s katero se otrok tekom uporabe aplikacije dodobra spozna. Aplikacija vsebuje besedilo in zvočni zapis pravljice, interaktivne scene iz zgodbe, animirane risanke ter izobraževale igre. Vsi elementi aplikacije so prilagojeni otrokom in v njih vzbujajo zanimanje, kar jim daje dodatno motivacijo za igranje.

Osnovni namen diplomskega dela je bil razviti interaktivno aplikacijo, ki vsebuje vse potrebne elemente za kakovostno učno gradivo za otroke, saj jim na zanimiv in privlačen način omogoča zabavno učenje in razvijanje sposobnosti ter jim predstavi del slovenske literature preko ljudske pravljice Janček Ježek. Namen sem dosegel tako, da sem najprej s pomočjo raziskovanja po literaturi iz tega področja ugotovil dobre in slabe prakse pri razvoju aplikacij za izobraževanje otrok. Ugotovitve sem nato uporabil kot podlago pri razvoju aplikacije, ki otroku ponuja zanimiv in kakovosten pripomoček pri izobraževanju, vsebuje vse priporočljive elemente izobraževalnih iger za otroke, neprimernim elementom pa se aplikacija izogne.

Pri izdelavi diplomske naloge sem dal poudarek na praktični del, torej na samo izdelavo aplikacije, ki daje temu delu tudi neko praktično vrednost. Vendar vseeno ne gre zanemariti dela diplomske naloge, ki je teoretične narave, torej študij literature s tega področja, saj sem s tem dobil osnove in smernice za izdelavo aplikacije. Pri pisnem delu diplomske naloge sem uporabil znanstvene metode, kot so deskripcija, primerjava obstoječih aplikacij, analiza stanja področja. Podatke o ustreznosti idej ter same razvite aplikacije sem zbiral s pomočjo intervjujev ciljne skupine uporabnikov. Pri razvoju sem uporabil tudi svoje lastne izkušnje iz tega področja, saj se tudi sicer ukvarjam z razvojem iger za otroke.

V pisnem delu diplomske naloge sem najprej predstavil pregled področja, v katerega sem vključil kratek zgodovinski pregled, tako pri nas kot tudi po svetu. Nato sem opisal značilnosti izobraževalnih aplikacij za otroke skupaj z glavnimi elementi, ki jih takšne aplikacije morajo vsebovati. Sledi primerjava med tradicionalno obliko izobraževanja ter

izobraževanja z uporabo računalniških aplikacij, na koncu pa sem pripravil še pregled in kritično oceno obstoječih gradiv, ki spadajo v to kategorijo in so na voljo v slovenskem jeziku. V naslednjem poglavju sem predstavil vse tehnologije, ki sem jih uporabil pri izdelavi aplikacije. Pri vsakem opisu sem podal glavne značilnosti, dodal pa sem tudi razloge, zakaj sem se za neko tehnologijo odločil in kakšne so njene prednosti pred konkurenti. Sledi poglavje o razvoju aplikacije, v katerem natančno predstavim vsebino aplikacije ter kako je razvoj potekal, predvsem v tehničnem smislu. V tem delu diplomske naloge se lahko bralec podrobneje seznani, kako je aplikacija zgrajena, katere elemente vsebuje in na kakšen način so implementirani. Zadnje poglavje predstavi rezultate izdelane aplikacije, torej ali aplikacija zadostuje zadanim ciljem ter ali je primerna za uporabo pri pedagoških dejavnostih.

2 PREGLED PODROČJA

2.1 ZGODOVINSKI PREGLED

Uporaba informacijskih tehnologij v izobraževalne namene je prisotno že kar nekaj časa. Prva izobraževalna aplikacija se je pojavila že leta 1950 v obliki računalniškega simulatorja letenja, ki so ga uporabljali za urjenje pilotov. Leta 1959 so začeli na eni izmed osnovnih šol v New Yorku uporabljati računalnike za učenje računanja z binarnimi števili. Velik porast uporabe izobraževalnih aplikacij se je zgodil leta 1977, ko so se v šolah začeli pojavljati prvi mikroračunalniki, predhodniki današnjih osebnih računalnikov. Še en zelo pomemben dogodek za to področje se je zgodil leta 1994, ko se je pojavil svetovni splet. Po letu 2000 je sledil razcvet svetovnega spleta, s tem pa so se tudi začeli pojavljati koncepti, kot sta učenje preko spleta oziroma učenje na daljavo (Roblyer, 2006).

V Sloveniji se informacijska tehnologija v šolah začne pojavljati po letu 1970. Leta 1980 postane računalništvo v srednjih šolah obvezen predmet, v osnovnih šolah pa izbirni predmet (Mori, 2004). V zadnjih letih se pri pouku v osnovnih šolah pojavlja uporaba tabličnih računalnikov, na katerih otroci rešujejo naloge pri različnih predmetih.

2.2 ZNAČILNOSTI IZOBRAŽEVALNIH APLIKACIJ ZA OTROKE

Izobraževalne aplikacije za otroke se praviloma pojavljajo v obliki računalniških iger, saj so te za otroke najbolj privlačne. Izobraževalne igre se od klasičnih računalniških iger razlikujejo glede na same cilje, ki jih igra pri igralcu skuša doseči. Pri klasičnih računalniških igrah je namreč primarni cilj igralca zabavati, pri izobraževalnih igrah pa je primarni cilj igralca izobraževati, vendar navadno ostaja tudi zabavnost eden izmed pomembnih ciljev (Michael in Chen, 2005, str. 17).

Pri razvoju izobraževalnih iger za otroke je treba paziti na primerno izbiro zvrsti igre. Ta omejitev sicer na splošno velja za vse igre, ki so namenjene otrokom. Najbolj neprimerna tema je zagotovo nasilje, ki je v klasičnih računalniških igrah precej pogost pojav. Druga zelo pomembna stvar, ki jo je treba upoštevati pri razvoju izobraževalnih iger za otroke, pa je izbira jezika, kar navadno pomeni, da mora biti igra v otrokovem maternem jeziku, saj sicer ni primerna za pedagoški proces (Ger Moreno in dr., 2008).

Glede same vsebine izobraževalnih iger za otroke so pomembni naslednji elementi:

Cilji in pravila

Vsaka igra mora imeti cilje, ki dajejo igri neki smisel. Tipični cilji iger so reševanje nalog in izzivov ali doseganje čim večjega števila točk. Cilji morajo biti dobro definirani in merljivi. Pravila definirajo omejitve, znotraj katerih je treba doseči željen cilj. Cilji so zelo pomembni, saj igralca želja po doseganju cilja dodatno motivira za igranje (Brainard in dr., 2010).

Težavnost

Pri izobraževalnih igrah je pomembno, da je za dosego cilja potreben določen trud, tako iz motivacijskega smisla, kot tudi iz izobraževalnega smisla. Ker pa so otroci še posebno težavna skupina in zelo hitro izgubijo motivacijo, sploh če je igra zanje pretežka, je pomembno, da je težavnost igre dobro uravnovešena. Torej igra ne sme biti ne prelahka ne pretežka. Težava se pojavi še zaradi razlike v znanju in sposobnosti otrok različnih starosti. Dober način, da naredimo igro primerno otrokom različnih starosti in sposobnosti je ta, da uvedemo različne stopnje težavnosti igre.

Povratna informacija

Povratna informacija je zelo pomembna že pri klasični obliki izobraževanja, enako pa velja tudi za izobraževalne igre. Prav ta povratna informacija, ki otroku pove, kako napreduje in koliko je v igri že dosegel, omogoča otroku učečo izkušnjo. Na tak način se namreč otrok zaveda svojih napak in se iz njih lahko uči, obenem pa je nagrajen, v primeru da dobro rešuje zadane naloge (Charles in dr., 2009). Povratna informacija je eden izmed glavnih elementov interaktivne igre.

Interaktivnost

Interaktivnost v kontekstu iger pomeni komunikacijo med človekom in računalnikom. Interaktivna igra je zgrajena tako, da igralcu omogoča interakcijo z igro, torej igralec med igranjem ni pasiven, hkrati pa igra sama na to interakcijo na neki način zna odgovoriti. Pomembno je, da je interakcija obojestranska. Gre torej za funkcionalnost igre, ki zagotavlja neko reakcijo na akcijo igralca (Markopoulos in dr., 2008, str. 22).

Igra za otroke, ki ima visoko stopnjo interaktivnosti, poveča otrokovo motivacijo za igranje in izboljša njegovo igralsko izkušnjo. Hkrati pa lahko z interaktivnimi elementi v igri spodbujamo otrokovo domišljijo in možgansko dejavnost, kar je za izobraževalne igre seveda zelo pomembno. Za igre, ki so namenjene otrokom, je priporočljivo tudi to, da z interaktivnimi elementi v igri spodbujajo otrokovo radovednost in ustvarjalnost.

2.3 PREDNOSTI IN SLABOSTI IZOBRAŽEVALNIH APLIKACIJ ZA OTROKE

Ker se izobraževalne aplikacije za otroke v osnovnih šolah vse pogosteje pojavljajo in v vedno večji meri nadomeščajo klasično obliko učenja, se pojavlja veliko različnih mnenj, katera oblika je za otroke bolj primerna. Vsekakor imata obe obliki učenja svoje prednosti in slabosti.

Najbolj očitna razlika med tradicionalnim učenjem in učenjem s pomočjo računalniških aplikacij oziroma iger je v tem, da se otroci z igrami igrajo, kar pa pri tradicionalnem učenju navadno ne drži (Klopfer in dr., 2009, str. 4). Posledično so otroci bolj zainteresirani in motivirani za učenje, saj je tak način dela nov in zanimiv, za učence pa je manj naporen in predvsem bolj zabaven. Učenci so bolj aktivni in postanejo samostojnejši pri reševanju nalog. Ker tak način pouka ni tako strogo voden s strani učiteljev, kot je tradicionalna oblika,

omogoča učencem bolj interaktivno in raziskovalno učenje. S tem pa se učenci naučijo razmišljati bolj samostojno in inovativno, spodbuja se tudi otrokova domišljija. Učenci z igrami dobijo takojšnjo individualno informacijo o tem, kako dobro so neko nalogo rešili, kar pri klasični obliki pouka ni možno vedno zagotoviti. Napake se ne kaznujejo, je pa učenec nagrajen za dobro reševanje, kar ustvari prijetno in spodbujajoče okolje za učenje.

Mora pa biti uporaba izobraževalnih iger pri pouku nadzorovana s strani učiteljev. Tudi tak način pouka zahteva dobro pripravo učitelja, ki mora otrokom razložiti osnovna pravila in zagotoviti, da ostane učenje na prvem mestu. Učitelj mora take vsebine smiselno vključiti v pouk. (Gerlič, 2000). Pomembno je tudi, da so izobraževalne igre dobro sestavljene in primerne za starost otrok. Tak način učenja naj bi bil bolj primeren za sposobnejše učence, saj imajo taki učenci s samostojnim razmišljanjem manj težav. Nekoliko manj sposobni učenci pa potrebujejo bolj voden pouk, saj imajo lahko težave že s samo organizacijo pri samostojnem reševanju nalog (Krnel, 2008, str. 8). Učiteljeva naloga je, da takim otrokom pomaga. Jasno je torej, da je učiteljeva vloga tudi pri takem načinu učenja še vedno zelo pomembna. Otroci namreč pri učenju še vedno potrebujejo vodenje. Tak način učenja je odlična popestritev klasičnega načina učenja in ima veliko pozitivnih učinkov na učence, ne more pa v celoti nadomestiti klasičnega pouka.

2.4 PREDSTAVITEV OBSTOJEČIH APLIKACIJ

V tem poglavju bom naredil pregled izobraževalnih aplikacij za otroke, ki so na voljo v slovenskem jeziku. Sicer je kar nekaj izobraževalnih vsebin, ki se uporabljajo pri pouku v osnovnih šolah plačljivih, tako da jih nisem imel priložnosti v celoti preizkusiti. Obstaja pa tudi nekaj prosto dostopnih, predvsem v obliki spletnih portalov.

Puhčeva interaktivna igralnica (dostopna na: http://www.modri-jan.si/igre-in-zabava/puhceva-interaktivna-igralnica)

Na tej spletni strani, katere avtor je slovensko podjetje Holding Slovenske elektrarne, najdemo veliko zbirko brezplačnih poučnih interaktivnih iger za otroke v zgodnjem in srednjem otroštvu, v katerih se pojavlja glavni junak Puhec. Uvedba glavnega junaka je vsekakor dobra ideja, saj se otroci na junaka navežejo, kar jih še dodatno motivira za igranje. Izbira iger je zelo pestra, igre pa so narejene kvalitetno, tako grafično kot tudi vsebinsko. Najdemo lahko igre, ki otroke ozaveščajo o težavah iz realnega sveta (onesnaženje, pomanjkanje pitne vode), simulacije raznih opravil (lovljenje rib, trgatev), najdemo pa tudi vrsto iger, ki trenirajo spomin, logiko, besedišče, ustvarjalnost ter splošno razgledanost iz področja živali, rastlin, vesolja in drugih tem. Igre imajo na voljo več stopenj oziroma težavnosti, tako da so primerne za različne starostne skupine. Vse igre ponujajo visoko stopnjo interakcije, so zabavne in otroku nudijo povratne informacije o njegovi uspešnosti. Puhčeva interaktivna igralnica ponuja res odličen pripomoček pri izobraževanju otrok.

Slika 1: Primer igre na Puhčevi interaktivni igralnici

Vir: HSE (2014)

Okolje in jaz 3 (dostopno na: http://www.modrijan.si/modrijan-fl/)

Okolje in jaz 3 so prosto dostopne interaktivne prosojnice, ki so namenjene izobraževanju otrok na področju ljudi in okolja. Avtor prosojnic je založba Modrijan. Prosojnice omogočajo izbiro med dvajsetimi temami, ki so potem podrobneje opisane. Za vsako temo je na voljo več različnih dejstev, zanimivosti in nalog v obliki interaktivnih vsebin, kot so na primer povezovanje pravilnih odgovorov. Prosojnice so bogate z večpredstavnimi vsebinami, vendar so kljub temu manj privlačnega videza, saj so dokaj statične in zastarele. Tudi interaktivnost je na nizki ravni. Pri reševanju nalog manjka povratna informacija, naloge pa imajo samo eno stopnjo težavnosti. Gradivo Okolje in jaz 3 je precej pomanjkljivo in manj privlačno za otroke, način dela pa je podoben tradicionalni obliki pouka. Kljub temu je lahko tudi to gradivo uporaben pripomoček za učenje in obnavljanje snovi.

Slika 2: Interaktivne prosojnice Okolje in jaz 3

Vir: Modrijan (2014)

Dežela Lilibi (dostopna na: http://www.lilibi.si/)

Dežela Lilibi je spletni portal za otroke, ki vsebuje več kot 1300 vsebin, kot so interaktivne naloge, interaktivne zgodbice, avdio in video vsebine, pesmice, igrice in učne liste (Založba Rokus Klett, 2014). Avtor spletnega portala je Založba Rokus Klett. Vsebine so plačljive, cena za enoletni dostop za šole in vrtce je 9 EUR na učenca, za fizične osebe pa 29 EUR. Obstaja tudi možnost tridnevnega brezplačnega preizkusa portala. To, da je portal komercialne narave, se pozna pri kakovosti spletnega portala, saj je tako vizualno kot tudi vsebinsko res vrhunsko sestavljen. Vsebuje štiri ločene dele, v vsakem delu pa najdemo po temah ali po šolskih predmetih razdeljenih ogromno interaktivnih vsebin. Vsak del predstavlja tudi različno raven težavnosti vsebin. Uporabnika vseskozi spremljajo simpatične animacije, zvočni efekti ter glasovi, ki otroka vodijo skozi portal. Igre na portalu vsebujejo vse elemente, ki so potrebni za dobro izobraževalno igro za otroke. Široka ponudba zelo kakovostnih tem in vsebin predstavlja odlično dopolnilo k pouku, ki bo za otroke zabavno in tudi zelo koristno.

Slika 3: Spletni portal Lilibi

Vir: Rokus Klett (2014)

Spletni portal Lek za otroke (dostopen na: http://www.lek.si/otroci/)

Na spletnih straneh podjetja Lek lahko najdemo prosto dostopni spletni portal za otroke, ki vsebuje šest interaktivnih izobraževalnih iger. Otroke ob igranju spremlja animiran govoreč lik kače, preko katerega poteka vsa interakcija igre z otrokom. Lik kače na igriv način otroka vodi med igranjem, mu razlaga navodila, nudi povratno informacijo in kar je še najbolj dobrodošlo, otroka med igranjem motivira, saj ga ob vsaki pravilni potezi pohvali in zanj navija, ob nedejavnosti igralca pa lik kače zaspi. Na portalu najdemo igre kot so štetje, sestavljanka, spomin, prepoznavanje likov in barv ter skrivalnice. Portal je dobro narejen in privlačen za otroke, animiran govoreč lik, ki otroka motivira pa je odlična ideja za takšne portale. Vendar portalu manjkajo stopnje težavnosti, predvsem pa več vsebine, torej več izobraževalnih iger iz različnih tem in področij.

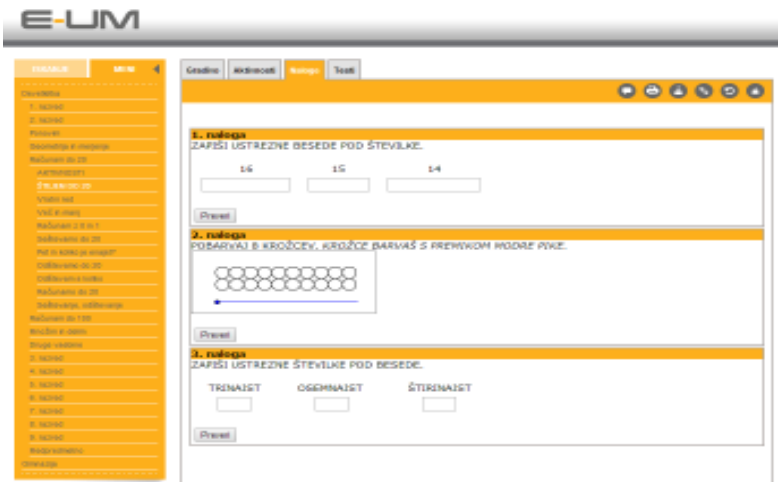
Slika 4: Spletni portal za otroke podjetja Lek

Vir: Lek (2014)

Spletni portal E-um (dostopen na: http://www.e-um.si/)

Spletni portal E-um je prosto dostopna zbirka učnih gradiv, namenjenih tako učencem kot tudi učiteljem. Gradiva so razdeljena po razredih in so namenjena osnovnošolskim in srednješolskim otrokom. Vsebin je veliko, od interaktivnih nalog, razlag z animacijami in zvočnim zapisom, pa do testov za preverjanje znanja. Vsebine pa niso predstavljene v obliki igre, ampak so bolj podobne klasičnim učbenikom in delovnim zvezkom, zato ta portal za otroke ni preveč privlačen. Problematična je tudi sama navigacija po vsebinah, saj je le ta podana v obliki klasičnih spletnih strani in ni prilagojena otrokom, zato se znajo pojaviti težave pri samostojnem delu mlajših otrok.

Slika 5: Naloge na portalu E-um



Vir: E-um (2014)

Glede na pregled nekaterih obstoječih interaktivnih izobraževalnih aplikacij za otroke lahko ugotovimo, da je ponudbe veliko, vendar niso vsi izdelki enako kakovostni in primerni za izobraževanje otrok. Sam bi glede na raziskavo priporočal brezplačni portal Puhčeva interaktivna igralnica ter plačljivi portal Dežela Lilibi.

3 OPIS UPORABLJENIH TEHNOLOGIJ

V tem poglavju bom opisal tehnologije in ogrodja, ki sem jih uporabil pri izdelavi aplikacije. Med tehnologijami sem se odločal glede na njihovo splošno uporabnost, primernost zadani nalogi ter glede na moje predznanje v teh tehnologijah. V poštev so prišle samo tiste tehnologije in ogrodja, ki so brezplačna oziroma tista, ki imajo na voljo tudi brezplačno različico.

3.1 UNITY

Najpomembnejša odločitev je bila vsekakor izbira tehnologije, v kateri bo aplikacija narejena. Na voljo je veliko programskih ogrodij in pogonov za igre, ki bi lahko prišli v poštev, zato je bilo treba preučiti lastnosti teh ogrodij in na podlagi tega narediti selekcijo. Glavni kriterij pri odločanju je bil možnost izdelave aplikacije za več različnih platform, torej za splet, mobilne naprave in osebne računalnike. Kriteriji za odločanje so bili še podpora za razvoj 2D iger, zmogljivost ogrodja, hitrost razvoja ter enostavnost uporabe. Tem kriterijem najbolj ustreza pogon za igre Unity, s katerim sem že imel nekaj izkušenj pri predhodnih projektih in bil z njim zelo zadovoljen. Zato sem se odločil, da za izdelavo igre izberem ta pogon.

Podjetje Unity Technologies je pogon za igre Unity prvič predstavilo leta 2005, vendar je takrat bila razvijalcem na voljo le različica za operacijski sistem Mac OS X. Od svojega začetka pa do danes se je Unity vztrajno razvijal in trenutno podpira kar 11 različnih platform: Windows, Mac OS X, Linux, Unity Web Player (spletni brskalnik), iOS, Android, BlackBerry 10, Windows Phone 8, Windows Store Apps, Xbox 360 in PS3, pripravlja pa se tudi podpora za Xbox One, Wii U, PlayStation 4, PlayStation Vita, PlayStation Mobile in Tizen (Wikipedia, 2014). Unity je eden izmed redkih pogonov, ki omogoča tako široko podporo. To razvijalcem prinaša veliko vrednost, saj lahko igro z minimalnimi spremembami izdajo na več različnih platformah, kar je lahko ob uporabi raznih drugih tehnologij zelo težavno in zamudno.

Unity se ponaša z zelo dobro načrtovanim uporabniškim vmesnikom in načinom dela, ki omogoča uporabnikom hitro razumevanje delovanja pogona, kljub temu da je ta zelo zmogljiv in kompleksen. Posledično je Unity eden izmed najbolj popularnih pogonov za igre, ki ima trenutno registriranih že okoli 3 milijone razvijalcev (Unity, 2014). Uporabljajo ga tako neodvisni razvijalci kot tudi velika podjetja. Ta pogon so uporabili pri izdelavi nekaterih svetovno znanih uspešnic iz področja iger, kot so: Bad Piggies (Rovio), Subway Surfers (Sybo Games), Need for Speed: World (Electronic Arts) in Temple Run 2 (Imangi Studios) (Unity, 2014). Unity je primeren za razvoj vseh zvrsti iger, kar tudi dokazuje pestra galerija že dokončanih iger v tem pogonu.

Slika 6: Uporabniški vmesnik pogona za igre Unity

Vir: Unity (2014)

3.1.1 UNITY FREE IN UNITY PRO

Unity ima na voljo brezplačno Unity Free in plačljivo Unity Pro verzijo. Osnovna Unity Pro verzija stane 1500 USD oziroma 75 USD na mesec, medtem ko je za razširitvi, ki sta namenjeni izdelavi mobilnih iger (iOS Pro in Android Pro), potrebno odšteti še dodatnih 1500 USD oziroma 75 USD na mesec. Razširitev, ki omogoča ekipam skupinsko delo na Unity strežnikih, pa stane še dodatnih 500 USD oziroma 20 USD na mesec. V primeru kupljenega programa je za posodobitev na novo verzijo treba odšteti od 600 USD naprej, v primeru mesečne naročnine pa so posodobitve brezplačne (Unity, 2014).

Razlika med Unity Free in Unity Pro je predvsem v tem, da Unity Pro ponuja še nekaj dodatnih funkcionalnosti. Kljub temu pa tudi Unity Free verzija uporabnikom ponuja res veliko funkcij in omogoča izdelavo zelo kakovostnih iger. Za vrhunski občutek in videz igre pa je vendarle potreben nakup Unity Pro verzije, ki poleg vseh funkcionalnosti iz Unity Free verzije ponuja še dodatne možnosti, kot so uporaba video posnetkov v igrah, 3D teksture, mehke sence v realnem času, razni efekti pri izrisu grafike, izboljšane luči in zvok, boljša učinkovitost pri izrisu velikih scen, ter še vrsto drugih. Unity Free ima še eno dodatno omejitev, in sicer to, da brezplačne verzije programa ne smejo uporabljati podjetja oziroma organizacije, ki imajo letni bruto prihodek višji od 100,000 USD (Unity, 2014).

Pri izdelavi aplikacije bom uporabil Unity Free verzijo, ki ponuja za potrebe zadane naloge več kot dovolj funkcionalnosti.

3.1.2 UNITY 2D IN UNITY 3D

Pri Unity Technologies so novembra 2013 izdali novo verzijo programa Unity, verzijo 4.3. Ob tej posodobitvi so v Unity vključili veliko novost, in sicer podporo za izdelovanje 2D iger. Unity je namreč bil vse od svojega začetka pa do te verzije namenjen le razvoju 3D iger. Vsekakor pa je mogoče narediti tudi 2D igro z uporabo 3D pogona, vendar je razvoj take igre zelo neroden in neučinkovit, predvsem pri izrisu grafike, pri uporabi fizike ter pri animacijah. Obstajajo tudi razne Unity razširitve oziroma dodatki, kot so na primer 2D

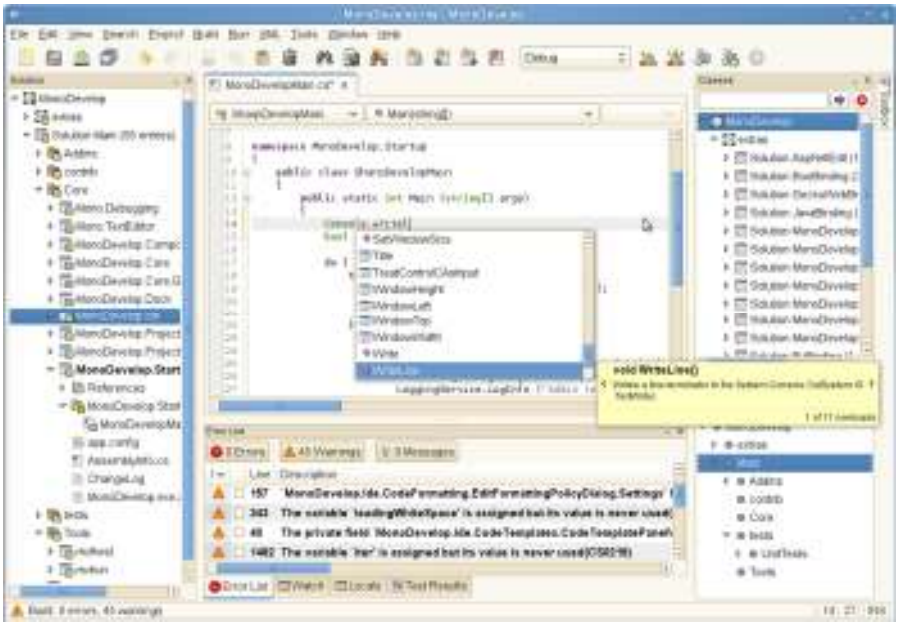
Toolkit in Orthello 2D Framework, ki ponujajo funkcionalnosti za lažje delo z 2D grafiko. Te razširitve so večinoma na voljo na uradni Unity trgovini na spletu Unity Asset Store. So pa ti dodatki večinoma plačljivi, dokaj pomanjkljivi in omejeni v svojih zmožnostih. Zato je podpora za 2D igre zelo dobrodošla novost v pogonu Unity in bo tudi meni prišla zelo prav pri izdelavi moje aplikacije, ki bo v celoti zgrajena na Unity 2D pogonu.

Tako kot pri izdelavi 3D iger, imajo sedaj Unity uporabniki tudi za 2D igre na voljo praktično vse funkcionalnosti, ki so potrebne za izdelavo vrhunske igre. V Unity 4.3 je na novo vpeljana podpora za uvoz, uporabo in izris 2D slik, integriran je zelo zmogljiv 2D pogon za fiziko Box2D, omogočeno pa je tudi zelo preprosto izdelovanje 2D animacij. Pri uporabniškem vmesniku je novost ta, da je na voljo tudi pogled, ki je prilagojen razvoju 2D iger. Unity omogoča tudi preprosto mešanje 2D in 3D elementov v igri (Unity, 2014).

3.2 MONODEVELOP

MonoDevelop je brezplačno odprtokodno razvojno okolje, ki je primarno namenjeno programiranju v programskem jeziku C#, podpira pa tudi vrsto drugih programskih jezikov, kot so Java, C, C++, Python, Visual Basic.NET in druge. Ogrodje deluje na operacijskih sistemih Windows, Linux in Mac OS X (Wikipedia, 2014). To ogrodje je v prilagojeni različici že vključeno v Unity. Ponuja kvalitetno razvojno okolje, ki je zelo intuitivno in pregledno. Ima tudi vrsto uporabnih funkcij za programerje, kot so samodejno dopolnjevanje kode in prikaz razpoložljivih metod oziroma funkcij, obarvanje programske kode ter možnost razhroščevanja. Programiranje aplikacije je v celoti potekalo v tem razvojnem okolju.

Slika 7: Uporabniški vmesnik programa MonoDevelop



Vir: Wikipedia (2014)

3.3 PROGRAMSKI JEZIK C#

Unity omogoča programiranje v treh programskih jezikih, in sicer v C#, UnityScript in Boo. Sam sem se odločil, da bom programiral v programskem jeziku C#, saj je kot programski jezik najbolj jasen in odlično dokumentiran. Za ta programski jezik je na voljo veliko primerov kode na spletu, kar je zelo uporabno, saj nam to lahko prihrani veliko časa pri reševanju programerskih problemov. Večina uporabnikov in poznavalcev pogona za igre Unity priporoča programiranje v programskem jeziku C#.

C# je objektno usmerjen programski jezik, ki so ga leta 2000 razvili pri podjetju Microsoft. Je zelo splošno uporaben, preprost, zmogljiv in moderen programski jezik. Njegova sintaksa je podobna nekaterim starejšim jezikom, kot so Java, C in C++ (C# Station, 2014). C# je zelo popularen programski jezik, ki je bil leta 2013 na četrtem mestu po številu projektov, narejenih v tem jeziku (za Javo, PHP, in C++), v rasti popularnosti pa je bil v letu 2013 celo na prvem mestu (Ramel, 2013).

3.4 GIMP

GIMP je brezplačen odprtokodni program za manipulacijo s slikami, ki je na voljo za Windows, Linux in Mac OS X operacijske sisteme. Projekt sta leta 1995 začela Spencer Kimball in Peter Mattis na ameriški univerzi Berkeley, trenutno ga vzdržuje skupina prostovoljcev, razvoj pa se financira preko donacij (Wikipedia, 2014). GIMP je program, ki je zelo zmogljiv in ima na voljo veliko funkcionalnosti. Nudi celovit set pripomočkov za risanje, omogoča pa tudi rezanje in urejanje slik, sestavljanje kolažev, pretvarjanje slik v različne formate in še veliko drugega.

Toliko funkcionalnosti posledično pomeni, da program ni ravno najbolj preprost za uporabo, zato je delo v tem programu za začetnike lahko precej težavno. Sam sem se odločil za GIMP, ker sem v preteklosti že delal s tem programom, tako da med uporabo nisem imel nobenih težav. Program sem uporabljal predvsem za obrezovanje slik, prilagajanje velikosti, ter za izvajanje raznih popravkov na slikah.

Slika 8: Urejanje slike prizora iz igre v programu GIMP

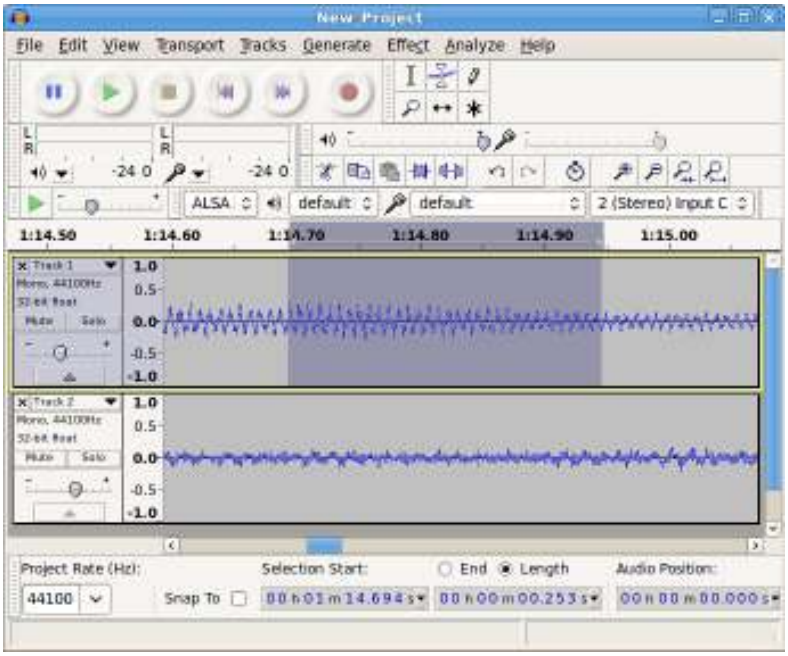
Vir: lasten

3.5 AUDACITY

Audacity je prosto dostopna odprtokodna programska oprema namenjena snemanju in manipulaciji z zvoki. Na voljo je za Windows, Linux in Mac OS X ter tudi za ostale operacijske sisteme. Program Audacity sta razvila Dominic Mazzoni in Roger Dannenberg leta 2000 na univerzi Carnegie Mellon v ZDA. Razvoj programa teče naprej preko prostovoljcev, financira pa se preko donacij in oglasov na spletni strani. Audacity je z več kot 80 milijoni uporabnikov eden izmed najbolj priljubljenih programov s področja manipulacije z zvoki (Audacity, 2014). Glavne funkcionalnosti programa so snemanje zvokov, urejanje (rezanje, kopiranje in združevanje) zvokov v različnih formatih, spreminjanje hitrosti predvajanja, spreminjanje kvalitete zvokov ter pretvorba v različne formate. Ponuja tudi široko paleto zvočnih efektov.

Audacity sem izbral, ker je dokaj preprost in enostaven za uporabo, kljub temu pa je zelo zmogljiv in ponuja veliko funkcionalnosti. Program sem uporabljal že pri drugih projektih. Pri izdelavi aplikacije sem program uporabil pri prilagajanju zvokov ter za spreminjanje kakovosti zvokov in formata zapisa.

Slika 9: Manipulacija z zvokom v programu Audacity



Vir: Audacity (2014)

4 RAZVOJ APLIKACIJE

V tem poglavju bom podrobneje predstavil razvoj aplikacije, ki je potekal približno pol leta. Pri razvoju sem sodeloval s študentko podiplomskega študijskega programa Grafične in interaktivne komunikacije na Naravoslovnotehnični fakulteti Anjo Kidrič. Anja je izdelala ilustracije za aplikacijo in poskrbela za grafično oblikovanje, sam sem poskrbel za implementacijo aplikacije, medtem ko je vsebina aplikacije nastajala v medsebojnem sodelovanju. Ravno to sodelovanje je omogočilo dosego enega izmed glavnih ciljev te diplomske naloge, in sicer izdelavo kakovostne in otrokom privlačne aplikacije. Pogoj za uspešno aplikacijo je namreč ta, da je aplikacija tako tehnično dobro izdelana, kot tudi vizualno privlačna. Še toliko bolj je to pomembno pri aplikacijah, ki so narejene v obliki igre in pri katerih so ciljna skupina uporabnikov mlajši otroci.

4.1 OPIS APLIKACIJE

Aplikacija, ki sem jo razvil, je interaktivna izobraževalna igra na temo slovenske ljudske pravljice. Ciljna skupina uporabnikov aplikacije so otroci v prvi polovici osnovne šole, je pa velik del aplikacije zanimiv tudi za nekoliko mlajše oziroma starejše otroke. Aplikacija v celoti temelji na slovenski ljudski pravljici Janček Ježek (poglavje 4.1.1), to pravljico so si namreč otroci izbrali kot njim najljubšo. Uporabnik aplikacije bo imel priložnost spoznati zgodbo pravljice na različne načine, in sicer preko samega besedila zgodbe, preko interaktivnih scen iz pravljice ter preko risanih animacij oziroma risank.

Aplikacija je razdeljena na pet glavnih delov oziroma prizorov, med katerimi lahko uporabnik izbira. Vsak prizor predstavlja en odlomek iz pravljice. Prizori so interaktivni in dinamični (Slika 10). Dinamičnost scen naredi aplikacijo bolj zanimivo in privlačno, otroku pa omogoča, da se v sceno bolj vživi. Dosegel sem jo z elementi, kot so premikanje oblakov, sijanje zvezd in padanje listov v gozdu. Interaktivnost pri uporabniku spodbuja ustvarjalnost in domišljijo, saj mora uporabnik v igri sam odkrivati skrite funkcionalnosti, hkrati pa ga interakcija tudi bolj angažira in s tem izboljša njegovo uporabniško izkušnjo. Na osnovne scene sem implementiral veliko skritih interaktivnih elementov, kot so na primer dim, ki se kadi iz hiške, prižiganje in ugašanje luči ter razne animacije oseb in živali na sceni.

Znotraj aplikacije najdemo štiri različne igre, ki so namenjene izobraževanju in razvijanju otrokovih umskih sposobnosti. Vse te igre se navezujejo na glavno zgodbo. Igre spodbujajo sposobnosti, kot so spomin, prepoznavanje in razvrščanje v skupine, natančno opazovaje več objektov naenkrat ter hitrost odločanja. Igre so sestavljene iz več stopenj, tako da so primerne za različne starostne skupine. Igre ves čas nudijo otroku povratno informacijo o uspešnosti reševanja nalog. Povratna informacija je podana tako v grafični kot tudi v zvočni obliki. Med igranjem otrok nabira točke, na podlagi katerih prejme na koncu posamezne igre nagrado v obliki zlatih jabolk, ki jih otrok zbira skozi celotno aplikacijo.

Slika 10: Primer interaktivnosti in dinamičnosti – premikanje oblakov, dim in let ptičkov

Vir: lasten

Vsaka izmed scen ima gumb za izbiro knjige, ki ob pritisku prikaže knjigo z besedilom zgodbe, ki se nanaša na trenutno sceno (Slika 11). Na ta način lahko uporabnik prebere celotno besedilo pravljice. Na knjigo sem dodal še gumb za zvok, ki ob kliku sproži zvočni posnetek besedila, namenjen otrokom, ki še ne znajo brati, ostalim pa še dodatno obogati celotno izkušnjo.

Slika 11: Prikaz knjige z besedilom

Vir: lasten

Uporabniku je na vsaki sceni na voljo tudi animacija, ki v obliki risanke prikaže tisti del pravljice, ki se nanaša na trenutno sceno. Animacije omogočajo pogled na zgodbo še iz drugačne perspektive. Med animacijo uporabnik aplikacije ni pasiven, saj so tudi animacije interaktivne in na določenih mestih zahtevajo uporabnikovo sodelovanje.

Aplikacija na vsaki sceni ponuja tudi možnost izbire med dnevnim in nočnim prizorom (Slika 12). Prizora prikazujeta isto okolje, vendar enkrat ponoči, drugič pa podnevi, kar da zelo zanimiv učinek. Med obema prizoroma se razlikujejo skriti interaktivni elementi, ki omogočajo uporabniku še dodatne možnosti za raziskovanje.

Slika 12: Dnevni in nočni prizor

Vir: lasten

4.1.1 PRAVLJICA JANČEK JEŽEK

Janček Ježek je slovenska ljudska pravljica, ki izhaja iz Prekmurja. Govori o dečku Jančku, ki želi pomagati materi mesiti kruh, a ga ta zapodi z besedami: »Beži no, ježek!«. Janček se ob materinih besedah čudežno spremeni v ježka. Ker pa ježki nimajo prostora v hiši, odide Janček v gozd, kjer si pod hruško izkoplje jamo in v njej prebiva sedem let. Nekega dne pa Janček Ježek v gozdu sreča izgubljenega grofa in mu ponudi, da mu pomaga najti pot domov, a za plačilo zahteva eno od njegovih hčera. Ker grof nima izbire, v ponudbo privoli in ježek mu pokaže pot do doma. Grof pa obljube ne drži in ko pride Janček naslednji dan do gradu na ogled hčera, grof naroči hčerkam zapreti vsa vrata in okna. Ježek pa skoči petelinu na hrbet in ta skupaj z njim poleti do okna, kjer stojijo grofove hčerke. Starejši dve hčerki ježka nočeta za moža, najmlajša pa se ježka usmili, ker je pomagal rešiti njenega očeta. Ko pa prideta ženin in nevesta pred oltar, se ježek spremeni nazaj v prekrasnega mladeniča. Takoj ko sestri to vidita, si premislita in si želita poročiti z Jančkom, vendar jih ta zavrne, saj hoče le najmlajšo hčerko, ker ga je ta rešila iz ježkove kože (Wikipedia, 2014).

Pravljica je bila prvič objavljena kot samostojna knjiga leta 1982, je pa tudi del različnih zbirk ljudskih pravljic, kot so na primer Babica pripoveduje, Slovenske ljudske pravljice ter Pesmice in pravljice zate (Wikipedia, 2014).

4.2 RAZVOJ UPORABNIŠKEGA VMESNIKA

Razvoj uporabniškega vmesnika je treba skrbno načrtovati, saj mora uporabniku ponujati vse potrebne funkcionalnosti, hkrati pa mora biti preprost in za uporabnika razumljiv, še posebej v primeru, ko so ciljna skupina otroci. Zato sem se pri načrtovanju ravnal po Mandelovih principih (Mandel, 1997), ki veljajo za nekakšna zlata pravila pri razvoju

uporabniških vmesnikov. Mandelovi principi so sestavljeni iz treh točk, vsaka točka pa sestoji iz več elementov, ki jih moramo zagotoviti.

Prvi princip: zagotovi nadzor uporabnika

Prvi element tega principa je omogočanje prekinitve danih opravil. Ker nas pri uporabi aplikacije lahko zmoti veliko dejavnikov, je pomembno, da aplikacija omogoča prekinitev trenutnih akcij in shranjevanje stanja za poznejše nadaljevanje. V primeru moje aplikacije za to poskrbi že sam pogon za igre Unity, ki v primeru, da igra izgubi fokus, trenutno stanje v igri zaustavi in ga shrani, tako da igralcu omogoča poznejše nadaljevanje igranja.

Drugi element je prikazovanje besedila oziroma obvestil s pomočjo uporabniku. S tem zagotovimo, da uporabnik zna uporabljati uporabniški vmesnik in da ve, zakaj se je neka akcija v aplikaciji sprožila. Pomembno je, da so obvestila do uporabnika vljudna in da ga ne grajajo. V aplikacijo sem na uvodno sceno dodal navodila za igranje ter legendo simbolov v igri, prav tako se navodila prikažejo na začetku posamezne igre, med in ob koncu posamezne igre pa sem implementiral obvestila o stanju.

Slika 13: Navodilo na začetku igre

Vir: lasten

Zagotoviti je treba tudi preglednost uporabniškega vmesnika, ki mora biti jasen in čist. To sem dosegel z dobro načrtovano postavitvijo elementov uporabniškega vmesnika. Elementi so enakomerno porazdeljeni po robovih scene in postavljeni intuitivno glede na obstoječe prakse. Posledično uporabnik nima problemov z razumevanjem vmesnika.

Zelo pomemben element, ki omogoča uporabniku nadzor nad aplikacijo, je možnost izhoda, ki mora biti ves čas na voljo. S tem zagotovimo, da uporabnik ni ujet v aplikaciji in lahko kadarkoli prekine neko akcijo, v primeru da si premisli, oziroma če je to akcijo izvedel pomotoma. Ta element pride zelo v poštev pri moji aplikaciji, saj sestoji iz več dolgih animacij in iger, zato sem povsod implementiral gumb za izhod, ki uporabnika vrne nazaj na glavno sceno.

Zadnji element tega principa je odzivnost. Vsaka akcija mora imeti takojšen učinek, nuditi pa mora tudi povratno informacijo. Tega sem se držal tudi pri razvoju moje aplikacije, saj vsak uporabnikov pritisk na gumb sproži takojšno reakcijo. Povratno informacijo sem implementiral na preprost, a učinkovit način, in sicer z uporabo zvočnega efekta ob pritisku na gumb.

Drugi princip: reduciraj obremenitev uporabnikovega spomina

Prvi element tega principa je razbremenjevanje kratkotrajnega spomina. Vsak podatek v aplikaciji, ki si ga aplikacija sama lahko zapomni, si ga tudi mora, še posebej ker uporabniki velikokrat počnejo več stvari hkrati, tako da ne smemo predpostavljati, da si bo uporabnik neki podatek zapomnil. To pravilo pride bolj v poštev pri raznih poslovnih aplikacijah, v igrah pa nekoliko manj, saj po navadi bolj redko pride do takšnih situacij.

Naslednji element je hitrost oziroma bližnjice. Dober uporabniški vmesnik mora uporabniku zagotoviti hiter dostop do želenega cilja. Zato sem v aplikacijo na vsako sceno implementiral gumbe za prehod med scenami, gumb za izklop glasbe ter gumb za izhod na uvodno sceno. S temi bližnjicami omogočam uporabniku hitro navigacijo po aplikaciji.

Del tega principa so tudi metafore iz realnega sveta. To pomeni, da lastnosti nekih objektov iz realnega sveta prenašamo v uporabniški vmesnik. S tem dosežemo, da lahko uporabnik na podlagi podobnosti elemente uporabniškega vmesnika prepozna, namesto da bi si jih moral predhodno zapomniti. Metafore torej razbremenjujejo uporabnikov spomin. Pomembno je, da so metafore razumljive, saj lahko sicer uporabnika zmedejo. V moji aplikaciji je večina navigacijskih gumbov narejenih na ta način, na primer gumb za zvok, na katerem je narisan zvočnik, na gumbu za animacijo je narisan filmski trak, na gumbu za igro je slika zlaganja kock, gumbi za izbiro scen imajo zaklenjeno oziroma odklenjeno ključavnico, gumb za nočni prizor pa ima narisane zvezde in luno (Slika 14).

Slika 14: Uporaba metafor iz realnega sveta pri elementih uporabniškega vmesnika

Vir: lasten

Uporabnikov spomin razbremenimo tudi z dobro organiziranostjo elementov, zato je pomembno, da gumbe smiselno razvrstimo po nekem pravilu, najbolje glede na njihov vsebinski pomen. Zato sem v aplikaciji gumbe za izbiro scen jasno razvrstil v svojo skupino, gumbe za izbiro animacije, igre in zgodbe pa v drugo.

Tretji princip: Zagotovi konsistentnost

Ohranjanje konteksta uporabnikovih opravil je prvi element tretjega principa. To pomeni, da moramo uporabniku omogočiti, da akcije izvaja vedno na enak način. Ni torej

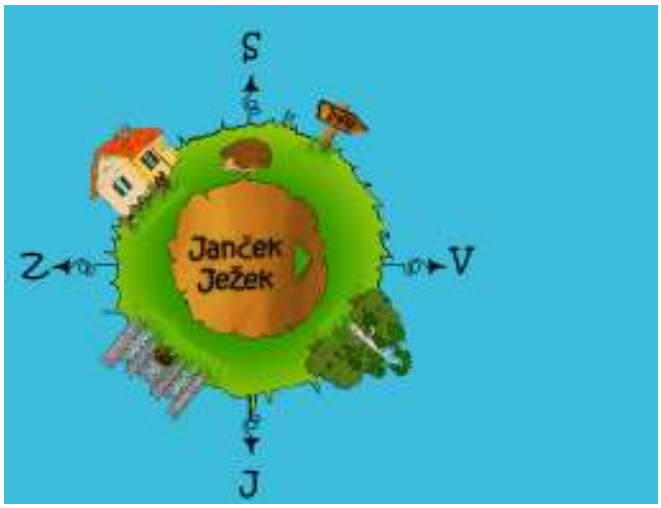
priporočljivo, da sredi programa zamenjamo način interakcije na primer iz miške na tipkovnico. Zagotoviti moramo tudi, da neka akcija uporabnika vedno prinese enak rezultat.

Drugi element je ohranjanje enovitosti v predstavitvi uporabniškega vmesnika. Ta se torej ne sme bistveno spreminjati, ne v vizualnem ne v logičnem smislu. Tudi obnašanje uporabniškega vmesnika mora ostati enako. To omogoča uporabniku, da ko enkrat osvoji osnovne koncepte, lahko to znanje prenaša skozi celotno aplikacijo. Zato ostaja v moji aplikaciji uporabniški vmesnik skozi vse scene enak, prav tako pa to velja tudi za vse igre znotraj aplikacije.

Za ohranjanje konsistentnosti moramo zagotoviti tudi estetsko privlačnost in popolnost. Elementi uporabniškega vmesnika se tudi med seboj ne smejo razlikovati glede na vizualni slog. Ohranjati moramo enake barve in pisave, če želimo zagotoviti privlačen in popoln uporabniški vmesnik.

Zadnja točka pa je spodbujanje raziskovanja. Uporabniški vmesnik mora biti zasnovan tako, da uporabniku omogoča raziskovanje vmesnika brez strahu po negativnih posledicah. Primer spodbujanja raziskovanja pri moji aplikaciji je naslovna scena, ki je sestavljena iz več objektov, na katere mora uporabnik pritisniti, da se pokažejo razne informacije o igri, oziroma da uporabnik lahko nadaljuje na glavne scene. Nobena akcija uporabnika ne prinese negativnih posledic.

Slika 15: Naslovna scena - primer spodbujanja raziskovanja



Vir: lasten

V pogonu za igre Unity se elemente uporabniških vmesnikov implementira deloma s programsko kodo, deloma pa kar znotraj samega pogona. V programski kodi se izris elementov uporabniških vmesnikov ter odzive na dogodke, kot je na primer pritisk na gumb, programira znotraj funkcije OnGUI. Spodnji primer (Primer kode 1) prikazuje izris preprostega gumba na koordinatah x=10 in y=10, na njem pa je izpisan napis »I am a button«. Ob pritisku na ta gumb se v konzolo programa izpiše napis »You clicked the button!«. Na tak način lahko poleg gumbov izrisujemo tudi napise in teksture, ti elementi

pa se izrisujejo brez posebnega oblikovanja in zato kot taki niso primerni za uporabo v resnejših projektih. Zato sem pri razvoju aplikacije uporabil instanco razreda GUIStyle, ki omogoča prilagajanje lastnosti teh elementov. Pri napisih lahko spreminjamo lastnosti, kot je vrsta pisave, velikost ter barva, gumbom pa lahko dodamo lastne slike, ki jih lahko celo menjamo glede na uporabnikove akcije oziroma glede na stanje gumba. V moji aplikaciji so vsi gumbi implementirani na tak način, da se ob dogodku, ko uporabnik z miškinim kazalcem prekrije gumb, temu zamenja sličica z bolj svetlo, kar daje občutek, da se je gumb zasvetil. Gumb s knjigo pa se, ko postane aktiven, spremeni iz zaprte knjige v odprto. Vseh teh lastnosti in efektov ni treba sprogramirati, ampak jih preprosto nastavimo kar znotraj pogona Unity, v programski kodi pa le dodamo to instanco razreda.

Primer kode 1: Izris preprostega guma v pogonu Unity

```
void OnGUI() {
    if (GUI.Button(new Rect(10, 10, 150, 100), "I am a button"))
        print("You clicked the button!");

}
```

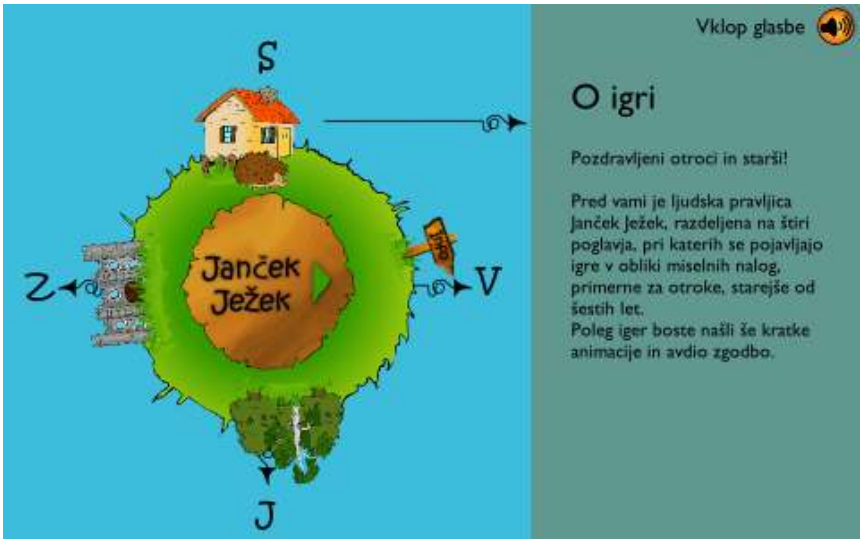
Vir: Unity (2014)

4.3 RAZVOJ UVODNE SCENE

Že uvodna scena aplikacije da uporabniku vedeti, kakšno aplikacijo lahko v nadaljevanju pričakuje. Scena je namreč interaktivna in dinamična, od uporabnika pa zahteva aktiviranje domišljije in smisla za raziskovanje. Na uvodni sceni uporabnik najprej zagleda majhen svet, na katerem stoji animiran ježek, na svetu pa so še štirje objekti, in sicer hiška, smerokaz, grad in gozd. Ti štirje objekti imajo na sebi komponento za zaznavanje trkov 2d objektov, Box Collider 2D, ter skripto, v kateri je funkcija OnMouseDown. To funkcijo sistem pokliče takrat, ko zazna miškin pritisk na komponento za zaznavanje trkov. Na tak način se torej odzovemo na uporabnikove akcije. Ob pritisku na enega od teh objektov se zgodi rotacija celega sveta, da pride tisti objekt, na katerega je uporabnik pritisnil, na vrh zaslona, kjer je tudi ježek. Med rotacijo se izvaja ježkova animacija hoje, tako da dobimo občutek, kot da se ježek sprehodi do objekta. Smer rotacije je odvisna glede na razdaljo od ježka do objekta, svet se torej vedno rotira v tisto smer, pri kateri je razdalja krajša, saj je tako tudi čas rotacije manjši. Razdaljo računam v programski kodi ob pritisku na objekt. Rotacija se izvaja znotraj funkcije Update, ki se kliče pred izrisom grafike, s klicem funkcije Rotate, ki ji kot parameter podam smer ter hitrost rotacije.

Ob koncu rotacije se glede na pritisnjen objekt na desni strani zaslona pojavi ločeno polje z besedilom. V primeru, da izberemo hiško, se pojavi besedilo z osnovnimi podatki o igri, v primeru gradu se izpišejo navodila za igranje, pri gozdu se izpiše celotna ekipa, ki je sodelovala pri izdelavi aplikacije, smerokaz pa prikaže legendo vseh simbolov, ki jih najdemo v aplikaciji. Na sredini sveta se nahaja še napis Janček Ježek in simbol za začetek, ki v primeru, da uporabnik pomakne miškin kazalec nad simbol, zažari. Ob pritisku na ta napis oziroma simbol nas aplikacija pelje na prvo glavno sceno.

Slika 16: Izpis podatkov o igri na naslovni sceni



Vir: lasten

V igri, narejeni v pogonu Unity, prehajamo med različnimi scenami v programski kodi z ukazom »Application.LoadLevel«, kot parameter pa podamo ime tiste scene, ki jo želimo prikazati. Ker se ti prehodi izvajajo brez kakšnega posebnega učinka, vizualno niso ravno najbolj privlačni, saj so prehodi iz ene scene v drugo zelo grobi. Zato sem se odločil v aplikacijo implementirati poseben učinek, kjer stara scena najprej postopoma izgine, nato pa se nova postopoma prikaže. S tem učinkom dobimo zelo lep in tekoč prehod med scenami. Učinek sem sprogramiral v skripti AutoFade, ki vsebuje funkcijo LoadLevel, katero kličemo namesto običajnega ukaza za prikaz nove scene. Ta funkcija nato sproži postopno prikazovanje črnega kvadrata preko celotne scene s pomanjšanjem transparentnosti, ki je na začetku popolna, torej kvadrat na začetku sploh ni viden. Na točki, ko postane kvadrat popolnoma viden in se scena pod njim ne vidi več, se kliče običajen Unity ukaz za zamenjavo scene, nato pa začnemo črnemu kvadratu postopoma zopet povečevati transparentnost, dokler ne popolnoma izgine. Ta metoda je sicer dokaj preprosta, vendar daje zelo dober rezultat. Ob klicu ukaza lahko po želji nastavljamo, koliko časa bo prehod trajal ter s katero barvo se bo prehod izvedel (v mojem primeru je to črna).

4.4 RAZVOJ GLAVNIH SCEN

V aplikaciji je pet glavnih scen, ki predstavljajo pet ključnih prizorov iz zgodbe Janček Ježek. Preko teh glavnih scen tudi dostopamo do ostalih vsebin, kot so igre, avdio in tekstovni zapisi zgodbe ter animacije. Te scene torej predstavljajo nekakšno izhodišče za nadaljnje raziskovanje trenutnega dela zgodbe.

Vseh pet glavnih scen vsebuje gumb s knjigo. Ob pritisku na ta gumb se na desni strani odpre knjiga, na kateri se izpiše besedilo zgodbe. Knjiga je navadna slika, učinek odpiranja in zapiranja pa sem dosegel s pomočjo rotacije slike ter animacije. V primeru, da sliko rotiramo za 90 stopinj po y osi oziroma po širini, postane ta v 2D prostoru nevidna, saj na sceno vedno gledamo iz le enega kota. Ko pa to sliko v animaciji postopoma rotiramo nazaj

na 0 stopinj, dobimo učinek odpiranja knjige. Za zapiranje knjige seveda postopek samo ponovimo v obratni smeri. V skripti, ki kontrolira obnašanje vseh glavnih scen, se v funkciji OnGUI preverja, ali je knjiga odprta, in v primeru da je, nanjo izpišemo besedilo zgodbe ter dodamo gumb za zvočni posnetek in gumb za izhod oziroma zapiranje knjige.

Slika 17: Rotacijo slike knjige se najbolje opazi pri 3D pogledu na sceno

Vir: lasten

Na scenah je implementirana tudi možnosti preklapljanja med dnevnim in nočnim prizorom trenutne scene. Ta učinek sem dosegel tako, da sem postavil na vsako sceno dve sliki, od katerih ena prikazuje dnevni prizor, druga pa nočni. Vsak element na sceni v pogonu Unity je obravnavan kot objekt v igri oziroma GameObject. Do teh objektov lahko dostopamo v programski kodi preko imena ali preko oznake objekta. Referenco na objekt dobimo preko ukaza »GameObject.Find« oziroma »GameObject.FindWithTag«. Ko imamo v programski kodi referenco na neki objekt na sceni, lahko z njim manipuliramo. V primeru dnevnega in nočnega prizora objektom v kodi nastavljam stanje aktivnosti z ukazom »GameObject.SetActive«. V primeru da objektu nastavimo, naj ne bo aktiven, se ta na sceni ne izrisuje več, prav tako pa tudi vsi objekti, ki so del tega objekta (podrejeni objekti). Na ta način se v aplikaciji menjata nočni in dnevni prizor.

Glavne scene imajo v desnem kotu gumbe s številkami, ki predstavljajo zaporedne prizore od ena do pet. Ob pritisku na enega izmed teh gumbov se torej scena zamenja in prikaže se nov prizor. Na gumbih je narisana tudi ključavnica, ki je lahko zaklenjena ali odklenjena (Slika 18). V primeru, da je prizor zaklenjen, ga seveda ne moremo izbrati. Prizore lahko odklenemo po zaporednem vrstnem redu tako, da si pogledamo animacijo na vsaki sceni, saj na ta način sledimo zgodbi iz enega prizora v drugega. Vsaka animacija se namreč začne na trenutnem prizoru in konča na naslednjem. Ko enkrat pridemo skozi animacijo, ostanejo prizori odklenjeni in lahko prosto preklapljamo med njimi. Podatek o odklenjenih prizorih se hrani v skripti LevelController, v kateri je statična spremenljivka s številom odklenjenih scen. Do spremenljivk, ki so statične (imajo oznako static), se lahko dostopa iz vseh skript, ne da bi imeli instanco na razred, kjer je ta spremenljivka definirana. S tem lahko neki podatek prenašamo preko več scen in skript, podatek pa se ob tem ne spreminja.

Slika 18: Primer gumba za odklenjen in zaklenjen prizor

Vir: lasten

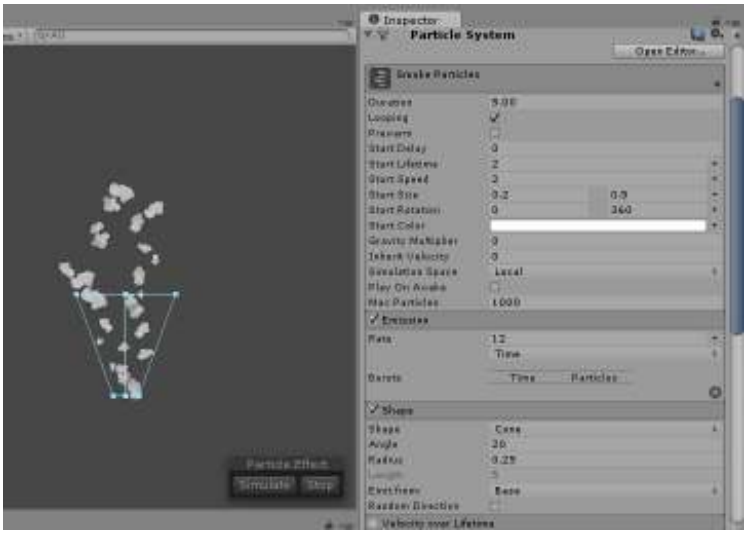
V nadaljevanju bom podrobneje predstavil posamezne scene in katere interaktivne elemente vsebujejo.

4.4.1 SCENA 1

Na prvi sceni je prikazana hiška, kjer stanuje Janček skupaj s svojo materjo, poleg hiške je še veliko drevo ter steza, ki pelje od hiše v gozd (Slika 10). Na sceno sem v ozadje postavil oblake, ki se počasi premikajo preko scene in s tem prizor poživijo. Oblake sem realiziral tako, da sem na sceno postavil dve zaporedni sličici, na kateri sem dal skripto, ki sličice v Update funkciji počasi premika preko scene. Ko pride prva sličica preko celotne scene in ni več vidna, je na sceni že druga sličica, prva sličica pa se premakne za drugo. S tem dobimo neskončno premikanje oblakov preko scene.

Na sceno sem dodal še dva skrita interaktivna objekta. Prvi je hiška, iz katere se ob pritisku nanjo skozi dimnik začne kaditi dim, ki sem ga realiziral s sistemom delcev (particle system). Unity ima odlično podporo za izdelavo sistemov delcev, saj lahko kar znotraj pogona nastavljamo želene parametre in sproti opazujemo spremembe (Slika 19). Dim sem realiziral z več različnimi sličicami, ki se nato naključno prikazujejo in potujejo navzgor. Zelo uporabna funkcionalnost v pogonu Unity je tudi to, da lahko vsem objektom, ki se izrisujejo na sceni, nastavimo vrstni red pri izrisu, kar vpliva na to, kateri objekt je izrisan spredaj in kateri zadaj. Na ta način se na primer oblaki izrisujejo v ozadju, dim pa za hiško, kar da občutek, kot da je zares prišel iz dimnika.

Slika 19: Nastavljanje lastnosti sistema delcev



Vir: lasten

Drugi skriti interaktivni objekt pa je drevo, iz katerega ob pritisku uporabnika začne letati animiran ptiček naključne barve. Za bolj realističen videz leta ptičkov, ti ne letijo naravnost, ampak med letom ves čas spreminjajo smer. Ta učinek sem dosegel tako, da znotraj Update funkcije ptičkom glede na neki časovni interval, ki je določen naključno, spremenim hitrost gibanja po x in y osi. Naključno vrednost dobimo s klicem funkcije »Random.Range«, kateri kot parametra posredujemo minimalno in maksimalno možno vrednost. Omejevanje vrednosti je zelo pomembno, saj bi sicer med letom lahko prišlo do preveč sunkovitega gibanja. V skripti preverjam še, ali je ptiček izven ekrana, saj ga v tem primeru s scene odstranim, da s tem aplikaciji sprostim spomin.

Primer kode 2: Programska koda za letenje ptičkov

```
void Update () {
        transform.position = new Vector3(transform.position.x-speedX,
  transform.position.y-speedY, transform.position.z);
    if(transform.position.x < -14f){
        Destroy(transform.gameObject);
    }
    timeLeft -= Time.deltaTime;
    if(timeLeft < 0){
        timeLeft = Random.Range(0.2f, 2f);
        speedY += Random.Range(-0.02f, 0.02f);
        speedX = Random.Range(0.04f, 0.05f);
    }
}
```

Vir: lasten

Pri nočnem prizoru pa lahko uporabnik prižge luči v hiški, kar sem dosegel s preprosto menjavo slike hiše, drugi interaktivni element pa so zvezde, ki so razporejene po nebu. S pritiskom na posamezno zvezdo se namreč začne izvajati neskončna animacija pravljičnega utripanja zvezde.

4.4.2 SCENA 2

Na drugi sceni je v ospredju hruška, pod katero si je Janček Ježek izkopal jamo, v kateri je nato prebival. V ozadju lahko vidimo elemente iz prejšnjega prizora, torej hiško in drevo, vmes pa je narisana steza (Slika 20). S tem se ta scena navezuje na prejšnjo, tako da ima uporabnik aplikacije občutek povezave med prizori, ki so sestavni del zgodbe in ne ločena enota. Ker je tudi pri tej sceni velik del slike nebo, sem v ozadje ponovno postavil dinamične oblake, ki se pomikajo preko scene. Interaktivna elementa na sceni sta zajček in krtina. Pri obeh elementih se ob pritisku na njiju sproži animacija. Kako se v pogonu Unity dela z animacijami, bom predstavil kasneje v podpoglavju o razvoju animacij (poglavje 4.6). Animacija krtine sestoji iz dveh delov, najprej se krtina zatrese, nato iz nje za kratek čas pokuka krtek, ki se nato skrije nazaj v krtino. Zajček naredi tri poskoke preko scene, najprej na levo stran, ob naslednjem pritisku pa na desno. Pri nočnem prizoru se na nebu zopet pojavijo zvezde, ki ob pritisku začnejo utripati, ob pritisku na krtino se ta zatrese, vendar krtek ponoči ne pokuka iz nje.

Slika 20: Prizor iz druge glavne scene

Vir: lasten

4.4.3 SCENA 3

Tretja scena se nahaja v gozdu, glavni lik na sceni je grof, ki zaskrbljen in žalosten sedi pod drevesom. To drevo je seveda prav ježkova hruška, s tem se ta scena tudi navezuje na prejšnjo. Ker se prizor dogaja v gozdu, sem na sceno dodal listje, ki pada z vrha scene proti dnu. Ta učinek sem realiziral s sistemom delcev, ki nad sceno rojeva naključne liste različnih dreves. Da bi dosegel bolj realistično padanje listov, sem v lastnostih sistema delcev dodal nastavitvi Force over Lifetime (sila tekom življenjske dobe) ter Rotation over Lifetime (rotacija tekom življenjske dobe), ki listom med padanjem dodajata naključne sile ter rotacijo. S tem dosežemo učinek, kot da listi zares letajo po zraku in da na njih delujejo sile težnosti ter veter.

Interaktiven element na tej sceni je grof, ki ob pritisku nanj pomaha in s tem nakazuje, da potrebuje pomoč. Na nočnem prizoru grof spi, ob pritisku pa se za trenutek zbudi, zazeha in nato zopet zaspi. Na prizoru je še sova, ki se ob pritisku premakne in zaskovika.

Slika 21: Nočni prizor tretje scene

Vir: lasten

4.4.4 SCENA 4

Na četrti sceni je osrednji element grofovski grad. Pri dnevnem prizoru je interaktiven objekt animiran metuljček, ki ob uporabnikovem pritisku nanj zleti na drugo lokacijo, nato pa spet pristane. Otrok se lahko z metuljčkom igra v nedogled, saj ta nikoli ne izgine s scene. Mehanika letenja je podobna kot pri ptičkih na prvi sceni, vendar pa sem metuljčku definiral sedem različnih točk, na katere lahko pristane. Za definiranje točk sem uporabil prazne objekte (Empty Game Object), ki so v aplikaciji nevidni, imajo pa svoje koordinate na sceni. Te prazne objekte nato v programski kodi shranim v tabelo objektov, ob uporabnikovem pritisku na metulja pa se izžreba naključni objekt iz te tabele. Pri tem pazim, da nov objekt ni isti kot objekt, na katerem se metulj trenutno nahaja, saj se v tem primeru metulj ne bi premaknil. Ko se v programski kodi izbere nov objekt, se ta določi kot tarča, proti kateri metulja premikam v Update funkciji z ukazom »Vector3.MoveTowards«, ki kot argumente dobi lokacijo metulja, lokacijo tarče ter hitrost premikanja.

Primer kode 3: Premikanje metulja proti ciljni lokaciji

```
void Update ()  {
    if (move){
        transform.position = Vector3.MoveTowards(transform.position,
target.position, Time.deltaTime * speed);
        if (Vector3.Distance(transform.position, target.transform.position)
        < 0.2f){
            move=false;
            anim.SetBool("animate", false);
        }
    }
}
```

Vir: lasten

Pri nočnem prizoru je interaktiven element grad, na katerem se ob pritisku nanj prižigajo in ugašajo luči pri naključnem oknu. Naključni vrstni red prižiganja in ugašanja sem dosegel tako, da sem lokacije vseh oken shranil v seznam in te elemente seznama nato naključno razporedil v funkciji ShufflePositions (Primer kode 4). Zatem za sliko gradu, ki ima transparentna okna, glede na lokacije iz seznama dodajam temno oziroma svetlo sliko, ki predstavlja ugasnjeno oziroma prižgano luč.

Primer kode 4: Naključno razvrščanje elementov seznama

```
    private void ShufflePositions () {
        for (int i = 0; i < positions.Count; i++) {
            GameObject temp = positions[i];
            int randomIndex = Random.Range(i, positions.Count);
            positions[i] = positions[randomIndex];
            positions[randomIndex] = temp;
        }
    }
```

Vir: lasten

4.4.5 SCENA 5

Peta scena se nekoliko razlikuje od ostalih, saj je mišljena kot končna scena aplikacije in ne vsebuje nočnega prizora ter igre. Še vedno pa lahko uporabnik prebere oziroma posluša zgodbo ter pogleda animacijo. Ponovno je na prizoru grad, pred gradom pa stojita nejevoljni starejši grofovi hčerki, ki spremljata najmlajšo hčerko, kako se skupaj z ježkom v kočiji odpravlja na poroko. Na sceni lahko vidimo tudi potko, po kateri se kočija v animaciji odpelje proti kraju poroke. Na tej sceni so trije interaktivni elementi, in sicer obe starejši hčerki ter kočija. Pri vseh treh elementih se ob uporabnikovem pritisku na njih sproži krajša animacija premika.

Slika 22: Prizor iz pete scene

Vir: lasten

4.5 RAZVOJ IGER

V aplikaciji lahko torej preko glavnih scen dostopamo do štirih izobraževalnih iger za otroke. Vse igre imajo identičen uporabniški vmesnik, ki nudi možnost izhoda iz igre ter izbiro ponovnega začetka igre, ob tem pa še tri polja za zlata jabolka, ki služijo kot nagrada za uspešno reševanje iger. Te nagrade so v aplikaciji implementirane zato, da ima otrok pri igranju neki cilj, ki ga spodbuja k čim boljšemu reševanju nalog. Zlata jabolka se podeljujejo ob koncu igre glede na doseženo število točk v igri. Število potrebnih točk za podelitev zlatega jabolka je pri vseh igrah enakovredno, saj je pogojeno s številom napak, ki jih igralec lahko v posamezni igri napravi. Ko igralec v določeni igri prejme zlato jabolko, si aplikacija to zapomni in igra ob naslednjih poskusih to tudi prikaže. Prazno polje se torej zapolni s sliko zlatega jabolka. Število doseženih zlatih jabolk v posamezni igri si aplikacija zapomni na enak način kot število odklenjenih scen, torej z uporabo statične spremenljivke v razredu LevelController. Razlika je le v tem, da je ta spremenljivka tabela, ki za vsako zlato jabolko vsebuje podatek o tem, ali je bilo doseženo ali ne. Otrok je torej za dobro reševanje igre trajno nagrajen, na koncu pete animacije, kjer se zgodba in s tem tudi

aplikacija konča, pa dobi otrok še potrditev o njegovi uspešnosti, saj se na zaslon izpiše število zlatih jabolk, ki jih je med igranjem osvojil.

Pri vsaki igri se najprej izpiše kratko navodilo za igranje, igro pa začnemo s pritiskom na gumb z napisom start. V nadaljevanju bom podrobneje predstavil vsako izmed iger.

4.5.1 IGRA 1

Prva igra se nahaja znotraj hiške, kjer živi Janček s svojo materjo. S tem se igra navezuje na prizor iz prve scene ter na prvo animacijo. Gre za igro spomina, v kateri si mora igralec najprej zapomniti predmete na sceni, nato pa prepoznati razlike. Igra je torej namenjena treniranju otrokovega spomina, natančnega opazovanja scene ter koordinacije v prostoru. Igra poteka tako, da se na sceni najprej pokažejo različni predmeti oziroma sadje. Predmeti niso postavljeni na naključna mesta, ampak so smiselno razporejeni glede na pohištvo v prostoru. Igralec ima na voljo nekaj sekund, da si predmete in njihove položaje zapomni, nato se predmeti oziroma njihovi položaji spremenijo, igralec pa mora te spremembe prepoznati in jih označiti. Možni sta dve vrsti sprememb, in sicer zamenjava predmeta na isti lokaciji ali pojavitev novega predmeta na novi lokaciji. Igra je sestavljena iz petih stopenj, ki se med seboj razlikujejo po številu predmetov na sceni, številu sprememb ter po času, ki ga ima igralec na voljo, da si predmete zapomni. V primeru, da se igralec zmoti in označi napačen predmet, se trenutna stopnja neuspešno zaključi, vendar pa ima igralec na voljo več možnosti ponovitve stopnje. Igralec napreduje v naslednjo stopnjo v primeru, da pravilno označi vse spremembe na sceni. Ves čas igranja je igralcu na voljo podatek o točkah, času ter o številu sprememb, vsaka pravilna oziroma napačna igralčeva odločitev pa se označi s kljukico ali s križcem.

Slika 23: Prizor iz prve igre

Vir: lasten

Igro sem implementiral tako, da sem najprej na sceno smiselno postavil predmete na vse možne lokacije. Na predmete sem dodal še skripto, ki hrani vrsto predmeta in njegovo lokacijo. Ta dva podatka potrebujem zato, da se na istem mestu ne bi pojavila dva predmeta naenkrat, oziroma da ob spremembi predmeta le tega ne bi zamenjal z istim predmetom.

Vsak predmet ima na sebi še komponento za zaznavanje uporabnikovega pritiska na njih. Vse te predmete v programski kodi shranim v seznam ter jih naredim neaktivne. Na sceno postavim želeno število predmetov tako, da najprej pridobim naključno številko lokacije, za katero preverim, ali je že zasedena. V primeru, da ni zasedena, izberem to lokacijo, sicer pa izberem naslednjo prosto lokacijo. Nato pridobim še eno naključno število, ki mi služi kot začetni indeks pri preiskovanju seznama vseh predmetov. Od tega indeksa naprej v seznamu poiščem prvi predmet, ki se nahaja na prej izbrani lokaciji. Ta predmet aktiviram in ga dodam na seznam originalnih predmetov. Spremembe na sceni generiram tako, da na sceno preko funkcije, ki sem jo uporabil že pri originalnih predmetih, dodam nov predmet, oziroma tako da obstoječ predmet iz seznama originalnih predmetov zamenjam z drugim na isti lokaciji. Nove predmete dodam na seznam spremenjenih predmetov, na podlagi katerega se potem odločam o pravilnosti izbire igralca med reševanjem naloge.

Primer kode 5: Dodajanje predmetov v prvi igri

```
private void SpawnObjects(int number, List<GameObject> list){
    for(int i=0; i<number; i++){
        int randomPosition = Random.Range(1,positionsNum);
        while(takenPositions.Contains(randomPosition)){
            randomPosition++;
            if(randomPosition>=positionsNum) randomPosition=1;
        }
        takenPositions.Add(randomPosition);
        int randomIndex = Random.Range(0,objectsNum);
        for(int j=0; j<objectsNum; j++){
            if(objectsPositions[randomIndex]==randomPosition){
                objectsArray[randomIndex].SetActive(true);
                list.Add(objectsArray[randomIndex]);
                break;
            }
            randomIndex++;
            if(randomIndex>=objectsNum) randomIndex=0;
        }
    }
}
```

Vir: lasten

4.5.2 IGRA 2

Druga igra se nahaja v ježkovem brlogu pod hruško, gre pa za igro razvrščanja predmetov in sadja. Otrok se ob igri uči ločevanja med predmeti, tropskim in domačim sadjem. Osnovno navodilo otroku naroča, naj na sceni obdrži le objekte, ki spadajo v brlog, tako da mora otrok sam prepoznati, kateri objekti se v ježkovem brlogu običajno nahajajo. Igra poteka tako, da se na sceni pojavi enako število objektov iz vseh treh skupin, otrok pa mora nato tujke na sceni razvrstiti v primerno škatlo, ki je namenjena predmetom oziroma tropskemu sadju. Domače sadje pa mora ostati na sceni, saj spada v ježkov brlog. Igra je časovno omejena in se neuspešno konča v primeru, da otroku v določenem času ne uspe razvrstiti vseh objektov na sceni. V primeru da otrok razporedi vse predmete dovolj hitro, se igra nadaljuje na naslednji stopnji. V igri so tri stopnje, med seboj pa se razlikujejo glede na število objektov na sceni ter glede na čas, ki je na razpolago za rešitev naloge. Igro se igra tako, da mora igralec na objekt pritisniti in ga povleči v škatlo, ki se odpre, če je ta

pravilna, objekt pa izgine v njej, sicer pa ostane zaprta in objekt se vrne na svoje originalno mesto. Za dodatno povratno informacijo in za povečanje otrokove motivacije pri igranju je na sceni tudi ježek, ki glede na pravilne oziroma napačne odgovore spreminja obrazne izraze od veselega do žalostnega.

Slika 24: Prizor iz druge igre

Vir: lasten

Tudi ta igra ima vnaprej določene lokacije, na katerih se objekt lahko pojavi. S tem poskrbim za lepo in enakomerno postavitev vseh objektov na sceni. V programski kodi sem nato definiral tri sezname. Prvi seznam hrani podatke o lokacijah, drugi hrani različne objekte, ki se lahko na sceni pojavijo, tretji pa hrani vse aktivne objekte, ki so trenutno na sceni. Tokrat objektov nimam že vnaprej postavljenih na sceno, ampak jih po potrebi ustvarjam v programski kodi s pomočjo tipa Prefab. Prefab je koncept, ki ga definira Unity, in ki hrani vse podatke o nekem objektu v igri (GameObject). Pred začetkom posamezne stopnje seznama vseh objektov ter lokacij premešam, tako da vedno dobim različne objekte in lokacije. Glede na število objektov v trenutni stopnji generiram nove objekte in jih dodam na seznam aktivnih objektov. Ti objekti imajo na sebi komponento za zaznavanje trka, ki omogoča zaznavanje igralčevega pritiska na objekt, ter skripto z imenom DragObject. V to skripto sem dodal funkcijo OnMouseDrag, v kateri je implementirano sledenje objekta miškinemu kazalcu (Primer kode 6), v Update funkciji pa sem dodal mehaniko, ki predmet vrne na izhodiščno lokacijo (Primer kode 7). Za vsak objekt se v skripti hrani še tip objekta (domače sadje, tropsko sadje ali predmet), saj se na podlagi tega odločam o pravilnosti igralčevega odgovora. Odgovore zaznavam tako, da sem tudi škatlam dodal komponento za zaznavanje trkov, poleg tega pa še komponento Rigidbody 2D, ki objektu v igri definira fiziko. To komponento potrebujemo za zaznavanje trka med škatlo in objekti v igri, paziti pa moramo na to, da komponenti izklopimo gravitacijo. Objektom v igri nato določimo, da je njihova komponenta za zaznavanje trka sprožilec (polje Is Trigger), na trke pa se odzivamo s pomočjo funkcije OnTriggerEnter2D.

Primer kode 6: Funkcija, ki omogoča vlečenje objekta z miško

```
void OnMouseDrag(){
    if(!goBack){
        sprite.sortingOrder = 5;
        Vector3 point =
        Camera.main.ScreenToWorldPoint(Input.mousePosition);
        point.z = gameObject.transform.position.z;
        gameObject.transform.position = point;
    }
}
```

Vir: lasten

Primer kode 7: Vračanje objekta na originalno lokacijo

```
void Update(){
    if(goBack){
        float targetX = transform.position.x;
        float targetY = transform.position.y;
        float targetZ = transform.position.z;
        targetX = Mathf.Lerp(targetX, startX, 5f * Time.deltaTime);
        targetY = Mathf.Lerp(targetY, startY, 5f * Time.deltaTime);
        transform.position = new Vector3(targetX, targetY, targetZ);
    }
}
```

Vir: lasten

4.5.3 IGRA 3

Pri tretji igri mora otrok pomagati ježku poiskati hrano, igra pa se, tako kot tretji prizor, dogaja v gozdu. Deloma se navezuje na prejšnjo, saj je ponovno treba izbirati med objekti, vendar tokrat le med tropskim in domačim sadjem. Otroku, ki je že pri prejšnji igri osvojil potrebno znanje o sadju, ki ga ježek je, bo pri igranju te igre lažje. Tudi po sami mehaniki sta si igri podobni, vendar tokrat pravilno sadje povlečemo na ježka, nepravilno pa pustimo na sceni. Pri igranju je potrebno hitro odločanje, saj je vsaka izmed treh stopenj časovno omejena, prav tako pa je potrebno tudi dobro opazovanje scene, saj so nekateri objekti dokaj majhni in zato manj opazni. Povratna informacija je med drugim podana tudi preko glavnega junaka Jančka Ježka, saj se ta ob nepravilni izbiri sadja skrije v klopčič. V primeru pravilne izbire se sadje nabode na bodice in tam ostane, ježek pa ob uspešni rešitvi naloge skupaj s sadjem odkoraka s scene.

Slika 25: Prizor iz tretje igre

Vir: lasten

4.5.4 IGRA 4

Tretja igra se nahaja v grofovskem gradu, predstavlja pa igro iskanja parov. Najprej se na oknih gradu prikažejo različni objekti, kot so sadje, predmeti, živali ter osebe, ki jih srečujemo skozi celotno aplikacijo. Ti objekti so prikazani nekaj sekund, nato pa jih zakrijejo polkna. Zatem se na vratih en za drugim prikazujejo objekti, igralec pa mora izbrati tisto okno, za katerem se skriva enak objekt. Gre torej za igro, ki trenira predvsem kratkotrajni spomin. V igri so dodane animacije odpiranja oken in vrat, pravilen oziroma napačen odgovor pa označita kljukica ali križec. Igra je sestavljena iz treh stopenj, ki se razlikujejo glede na število objektov na sceni ter po času, ki ga ima igralec na voljo, da si te objekte zapomni. Igra je popustljiva do napak, saj se ob napačnem odgovoru igra ne zaključi, ampak se igralcu le odštejejo točke, kar vpliva na končen rezultat in s tem na število prejetih zlatih jabolk. V naslednjo stopnjo igralec napreduje v primeru, da uspešno najde vse pare.

Slika 26: Prizor iz četrte igre

Vir: lasten

Igro sem realiziral tako, da sem najprej za vsako sliko gradu definiral lokacije vseh oken in vrat. Lokacije nato v programski kodi shranim v seznam, tako kot tudi vse možne objekte, ki se v igri lahko pojavijo. Seznam objektov nato zmešam, da s tem dobim naključni vrstni red, po katerem objekte dodajam na sceno. Vsak objekt dodam na sceno dvakrat, prvič ga dodam na okno, drugič pa na vrata, tako da dobim pare objektov. Da bi bil vrstni red iskanja neodvisen od lokacij, premešam še seznam objektov na vratih, nato pa te objekte enega za drugim prikazujem. Ker imajo vsi objekti na sebi skripto, ki hrani njihov tip, lahko na podlagi tega preverjam pravilnost igralčevih odgovorov.

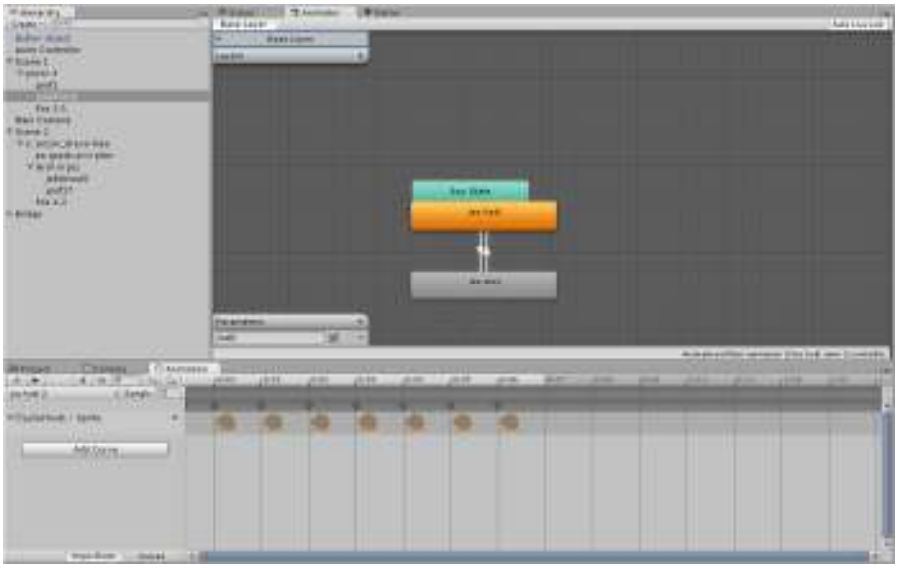
4.6 RAZVOJ ANIMACIJ

Unity ima vgrajeno odlično podporo za razvoj animacij, ki omogoča preprosto in hitro izdelavo tako 2D kot tudi 3D animacij. Vsakemu objektu na sceni lahko dodamo animacijo. 2D objekte animiramo na dva načina. Na prvi način animacijo objekta sestavimo tako, da v

pogonu Unity na sceno preprosto povlečemo zaporedje slik, Unity pa zna iz tega zaporedja sam sestaviti animacijo. Drugi način pa je ta, da sami ustvarimo novo animacijo in jo dodamo na objekt, potem pa animaciji dodajamo lastnosti, za katere želimo, da se bodo med animacijo spreminjale. Objektu lahko animiramo lastnosti, kot so lokacija, rotacija, velikost, prikazana slika, barva, transparentnost in še vrsto drugih. Ko izberemo lastnost, ki jo želimo animirati, na časovnem traku določimo ključne vrednosti, ki določajo vrednost neke lastnosti ob določenem času. Vmesne vrednosti pa zna Unity določiti sam. Oba načina za izdelavo animacij lahko tudi kombiniramo, saj tudi prvi način animacijo zgradi z določanjem ključnih vrednosti na časovnem traku, le da to Unity stori sam in s tem razvijalcu prihrani nekaj časa.

Shranjeni animaciji nato določimo, naj se ta izvaja ponavljajoče ali pa samo enkrat, nastavimo pa lahko tudi hitrost izvajanja. Objektu, ki ga animiramo, moramo dodati komponento Animator, ki zahteva definiranje upravljavca animacije (Animator controller). V upravljavcu določimo, katera animacija se bo izvajala na objektu, lahko pa definiramo tudi graf animacij s prehodi in pogoji za prehode. S tem omogočimo, da lahko objektu menjamo animacije glede na dogodke v aplikaciji. Kot pogoj za prehod med animacijami lahko določimo neki čas, ali pa dodamo parametre, ki jih lahko nastavljamo iz programske kode ter tako pridobimo popolno kontrolo nad izvajanjem animacij. Parametri so lahko števila, logični operatorji ali sprožilci. Nekemu objektu, ki ima definiran prehod med animacijami s sprožilcem z imenom »next animation«, sprožimo naslednjo animacijo z ukazom: »GetComponent<Animator>().SetTrigger("next animation ");«.

Slika 27: Izdelava animacije ježka



Vir: lasten

4.6.1 ANIMACIJA 1

Prva animacija prikazuje, kako se je Janček spremenil v ježka in odšel od hiše v gozd. Dogajanje se začne na prizoru prve glavne scene, kjer se kamera začne približevati hiški.

Približevanje kamere dosežem tako, da kameri dodam animacijo, v kateri manjšam lastnost Orthographic size, kar kameri zmanjša površino, ki jo pokriva, s tem pa objekti na sceni navidezno postanejo večji. Ko se animacija kamere konča, se vrata na hiški zasvetijo in začnejo utripati, kar nakazuje, da mora uporabnik na njih pritisniti. Ob pritisku se scena zamenja in pokaže se notranjost hiške. Med scenami menjam tako, da vse elemente scene združim pod en prazen objekt, ki mu potem nastavljam aktivnost oziroma neaktivnost. V hiški se nahajata Janček ter mati, ki mesi kruh. Janček nekaj časa gleda mati pri delu, nato pa vstane in se pomakne bližje, vendar ga mati z gesto prežene. V tem trenutku se zabliska in Janček se spremeni v ježka, ki nato odkoraka s scene. Za prikaz Jančka imam ves čas samo en objekt, kljub temu, da se ta spremeni iz fanta v blisk in nato še v ježa. To mi omogoča prehajanje med animacijami, saj se lahko v posamezni animaciji objektu spreminja tudi prikazano sliko. Zatem se scena zopet zamenja in sledi še sprehod ježka od hiške do gozda.

Slika 28: Prizor iz prve animacija, ko se Janček spremeni v ježka

Vir: lasten

4.6.2 ANIMACIJA 2

Pri drugi animaciji se kamera najprej približa vhodu v ježkov brlog pod hruško. Vhod se zasveti, ob pritisku nanj se scena spremeni in prikaže se notranjost brloga. V brlogu se nahaja ježek, okoli njega pa je razporejeno sadje. Na tej sceni se lahko otrok malce poigra, saj se ob pritisku na sadje ježek sprehodi do njega in ga poje, ob tem pa postane nekoliko večji. To dosežem tako, da ob pritisku na sadje določim lokacijo tega sadja kot tarčo, proti kateri pomikam ježka. Ker imata tako ježek kot tudi sadje na sebi komponento za zaznavanje trka, lahko v programski kodi ugotovim, kdaj se ježek dotakne sadja in v tistem trenutku objekt sadja uničim, na ježku pa sprožim animacijo rasti. Ob pritisku na tablo, ki nakazuje izhod, pa se ježek sprehodi ven iz brloga, kjer potem sledi še ježkov sprehod skozi gozd. Slika prizora sprehoda je sestavljena iz dveh delov, za ježkom se nahaja ozadje gozda, pred njim pa je podrast, kar da sliki učinek globine. Ob koncu sprehoda ježka pospremi še animacija zajčka, ki poskakuje preko prizora.

Slika 29: Učinek globine pri drugi animaciji

Vir: lasten

4.6.3 ANIMACIJA 3

Tretja animacija prikazuje srečanje med ježkom in grofom ter njuno potovanje do grofovega gradu. Najprej se kader približa grofu, ki sedi pod hruško. Do njega pristopi ježek, s katerim nato skupaj nadaljujeta sprehod proti gradu. Zatem sledi interaktivna scena, na kateri v ozadju vidimo grofovski grad, pot do njega pelje preko lesenega mostu čez potok, vendar most ni prehoden, saj mu manjkajo deske. Manjkajoče deske so naključno razporejene po sceni, otrok pa mora ugotoviti, da je za prehod preko mostu treba deske postaviti na pravo mesto. To stori tako, da na desko z miškinim kazalcem pritisne, jo povleče k mostu ter nato spusti, ob tem se deska sama postavi na pravo mesto, ki je definirano vnaprej. Ko je most sestavljen, se pojavi sistem delcev z zvezdicami, ki nakazujejo uspešno opravljeno nalogo, nato se grof in jež sprehodita preko mostu. Sliki ježa in grofa med sprehodom manjšam, da dosežem učinek hoje v globino.

Slika 30: Sprehod preko mostu v tretji animaciji

Vir: lasten

4.6.4 ANIMACIJA 4

V četrti animaciji lahko vidimo dogajanje v gradu, kjer ježek izbira svojo nevesto. Najprej se kader približa gradu, na sceno pa prikoraka ježek Janček. Vrata gradu se zasvetijo, ob pritisku na njih se s pomočjo animacije odprejo, da lahko ježek odkoraka v grad. Za prikaz hoje v globino sem ponovno uporabil postopno manjšanje velikosti slike. Nato se scena zamenja, prikaže se notranjost gradu skupaj s tremi grofovskimi hčerkami ter ježkom. Na sceni je implementirana skrita funkcionalnost po vzoru glavnih scen, saj se ob pritisku na luči dnevni prizor zamenja z nočnim. Cilj te scene je, da si ježek izbere pravo nevesto. Ob pritisku na katero izmed hčera se namreč sproži animacija, v kateri hčerka z gesto pokaže, ali si ježka želi za moža. Starejši dve ježka zavrneta, ob izbiri najmlajše pa ta prikima in nato skupaj z ježkom odide iz gradu.

Slika 31: Prizor izbiranja neveste v četrti animaciji

Vir: lasten

4.6.5 ANIMACIJA 5

V zadnji animaciji je prikazana poroka med ježkom in najmlajšo grofovo hčerko ter Jančkova preobrazba iz ježka v mladeniča. Animacija se začne tako, da se kamera približa kočiji z ježkom in hčerko. Kočija se nato odpelje po prizoru naprej, medtem ko kamera kočiji sledi. Tudi ta prizoru je sestavljen iz dveh delov, in sicer iz slike ozadja ter iz trave in rož v ospredju, kar daje občutek globine. Ko pride kočija do prostora, kjer se bo zgodila poroka, hčerka in ježek iz nje izstopita, ta pa se odpelje s prizora. Nato se ponovno prikaže blisk in ježek se spremeni nazaj v mladeniča. Mladoporočenca si podasta roke, scena pa se konča s poljubom. Za bolj čaroben konec sem dodal še sistem delcev iz zvezdic, ki obkrožajo grofovo hčerko in Jančka. Celotna animacija je implementirana z uporabo različnih animacij, ki si zaporedno sledijo. Ker je končni prizor pete animacije hkrati tudi končni prizor celotne aplikacije, na tem mestu izpišem napis s čestitkami ter podatek o zbranih zlatih jabolkih.

Slika 32: Končni prizor aplikacije

Vir: lasten

4.7 IMPLEMENTACIJA ZVOKA IN GLASBE

V aplikacijo sem implementiral več zvočnih efektov, ki jih aplikacija predvaja ob uporabnikovih akcijah in so večinoma del povratne informacije. Zvočni efekti, ki spadajo v okvir uporabniškega vmesnika so zvok ob pritisku na katerikoli gumb, zvok za začetek igre oziroma napredovanje v višjo stopnjo, zvoka za pravilen in napačen odgovor v igrah ter zvok, ki označuje na novo odklenjeno sceno v aplikaciji. Poleg tega so v igri še zvočni efekti, ki se predvajajo ob pritisku na interaktivne elemente na glavnih scenah, zvočni zapis besedila zgodbe ter glasba, ki se predvaja v ozadju. Tako na naslovni sceni kot tudi na vseh glavnih scenah ima uporabnik aplikacije ves čas na voljo gumb, ki glasbo v ozadju preneha oziroma začne ponovno predvajati.

Zvoke sem implementiral tako, da sem jih najprej prilagodil potrebam s programom Audacity (poglavje 3.5), nato pa sem jih uvozil v pogon Unity. Znotraj pogona lahko zvokom prilagajamo kakovost in s tem zmanjšamo velikost igre. Če želimo zvok predvajati v igri, moramo nekemu objektu na sceni dodati komponento Audio Source, na to komponento pa dodati izbran zvok. Nastavimo lahko še glasnost, ponavljanje zvoka ter možnost takojšnega predvajanja oziroma čakanja na ukaz iz skripte. Referenco na zvok, ki je dodan objektu, dobimo v z ukazom »GetComponents<AudioSource>()«, če pa je na nekem objektu več zvokov, moramo pri tem ukazu dodati še oglate oklepaje z indeksom želenega zvoka. Tako lahko zvok predvajamo, ustavljamo ter mu spreminjamo glasnost kar iz programske kode. Primer spreminjanja glasnosti med izvajanjem aplikacije je med predvajanjem zvočnega zapisa zgodbe, saj v tem primeru znižam glasnost glasbe v ozadju, tako da je zgodba lažje poslušljiva, po koncu predvajanja pa glasnost glasbe zopet povečam. Pri glasbi v ozadju nastane še ena posebnost, saj so zvoki vezani na neki objekt in so ob menjavi scene skupaj z njimi uničeni, torej se pojavi težava, kako glasbo predvajati čez več različnih scen. Težavo sem rešil tako, da sem na uvodni sceni ustvaril objekt, nanj dodal glasbo ter skripto, v kateri na ta objekt kličem funkcijo DontDestroyOnLoad, ki zagotovi, da objekt kljub zamenjavi scene ne bo uničen. Da lahko tudi iz drugih skript dostopam do glasbe, sem tej skripti dodal še statično referenco na zvok in funkcije, ki s tem zvokom manipulirajo. Pomembno je še to, da ob nalaganju naslovne scene vedno preverim, ali instanca na glasbo že obstaja, saj bi v nasprotnem primeru prišlo do podvajanja glasbe.

Primer kode 8: Skripta za upravljanje z glasbo v ozadju

```
public class SoundScript : MonoBehaviour {

    private static SoundScript instance = null;

    public static SoundScript Instance {
        get { return instance; }
    }

    void Awake() {
        if (instance != null && instance != this) {
            Destroy(this.gameObject);
            return;
        } else {
            instance = this;
        }
        DontDestroyOnLoad(this.gameObject);
    }

    public void StopPlay(){
        audio.Stop ();
    }

    public void StartPlay(){
        audio.Play ();
    }

    public void SetVolume(float volume){
        audio.volume = volume;
    }

    public bool IsSoundPlaying(){
        return audio.isPlaying;
    }
}
```

Vir: lasten

5 ZAKLJUČEK

Računalniške aplikacije se v izobraževalne namene uporabljajo že od leta 1950, od takrat se področje neprestano razvija, njihova uporaba pa je vedno bolj pogosta. Zadnja leta je v osnovnih šolah postala popularna uporaba tabličnih računalnikov, s pomočjo katerih se otroci učijo ob uporabi interaktivnih izobraževalnih iger. Primarni cilj teh iger mora biti izobraževanje otrok. Zagotoviti moramo tudi, da imajo igre primerno težavnost, otroku morajo nuditi povratno informacijo, vsebovati morajo določene cilje, ki jih otrok skuša doseči ter da so igre interaktivne. Uporaba igre s temi lastnosti ima prednost pred tradicionalno obliko pouka v tem, da otroka bolj motivira, spodbuja njegovo radovednost, inovativnost in ustvarjalnost, omogoča bolj samostojno delo, otrok pa se med učenjem bolj zabava. Težave se lahko pojavijo pri manj sposobnih učencih ter pri nenadzorovani uporabi teh iger, ki ne morejo v celoti nadomestiti tradicionalnega učenja, vendar so lahko odličen pripomoček za učenje in popestritev pri pouku. Na slovenskem področju obstaja kar nekaj takšnih vsebin, vendar niso vse dovolj kakovostne za uporabo pri pouku.

Izbor tehnologij, ki sem jih uporabil pri razvoju aplikacije, je potekal glede na moje predznanje, splošno uporabnost tehnologij ter njihovo primernost zadani nalogi. Za implementacijo aplikacije sem izbral pogon za igre Unity, saj nudi odlično podporo za učinkovit in hiter razvoj 2D iger, ki delujejo na različnih platformah. Uporabil sem brezplačno različico programa. Programsko kodo sem v celoti napisal v programskem jeziku C#. Skripte sem pisal v razvojnem okolju MonoDevelop, saj je to okolje del pogona Unity in je prilagojeno za delo v tem pogonu. Za manipulacijo s slikami sem uporabil program GIMP, za urejanje zvokov pa program Audacity.

Aplikacija, ki je namenjena otrokom v zgodnjem in srednjem otroštvu, je razdeljena na pet glavnih prizorov, ki predstavljajo odlomke iz pravljice Janček Ježek. Prizori so dinamični in interaktivni, saj uporabniku aplikacije ponujajo skrite elemente, ki znajo reagirati na uporabnikovo akcijo. Iz glavnih prizorov nudi aplikacija dostop do besedila in zvočnega zapisa zgodbe, do petih animacij, ki v obliki interaktivne risanke prikažejo del zgodbe ter do štirih izobraževalnih iger, ki razvijajo otrokove mentalne sposobnosti. Pri razvoju uporabniškega vmesnika aplikacije sem se ravnal po Mandelovih principih in ga prilagodil za uporabo otrok. V aplikacijo sem implementiral vse dobre prakse pri izdelavi izobraževalnih iger ter iger na splošno. Za dodatno izboljšanje uporabniške izkušnje sem dodal še zvoke ter glasbo v ozadju. Z izdelano aplikacijo sem zadovoljen, saj izpolnjuje vse zastavljene cilje.

Kljub temu, da je aplikacija že v tem trenutku zaključena celota, še vedno obstajajo tudi možnosti izboljšav oziroma dodatnih funkcionalnosti. V aplikaciji je še nekaj prostora za dodatne skrite interaktivne elemente na glavnih scenah. Zvočna podpora bi lahko bila bogatejša. Dobrodošla bi bila tudi možnost shranjevanja napredka glede na uporabnikov profil, saj se trenutno napredek shranjuje le med tekočim igranjem in se ob zaprtju igre

izbriše. Da bi aplikacija dosegla večje občinstvo, bi lahko implementirali še podporo različnim jezikom in bi s tem predstavili slovensko ljudsko pravljico Janček Ježek svetu.

Že med razvojem je bila aplikacija testirana na učencih v osnovni šoli in otrokom je bila igra zelo všeč. Dodatno potrdilo, da je aplikacija kakovostna in primerna tudi za uporabo pri pouku smo dobili iz strani ene izmed največjih slovenskih založb, ki je za aplikacijo pokazala zanimanje, saj bi jo radi vključili med njihova interaktivna izobraževalna gradiva za osnovnošolske otroke. Ta možnost bi bila tudi najboljša, kar se tiče objave igre. Druge možnosti so še objava v okviru enega izmed obstoječih portalov z izobraževalnimi igrami za otroke, ali pa samostojna objava, vendar bi bila ta možnost najslabša, saj je na tak način zelo težko aplikacijo posredovati do končnih uporabnikov brez posebne promocije.

Glede na to, da je bila aplikacija zelo dobro sprejeta tako pri otokih kot tudi pri založbah, ki izdajajo interaktivna gradiva za otroke, verjamem, da lahko tudi ta aplikacija postane eno izmed gradiv, ki se množično uporablja za izobraževanje otrok tako pri pouku v šolah kot tudi doma.

LITERATURA IN VIRI

LITERATURA

- Brainard, Katie, Nettles, Tachelle, Peters, Lindsay (2010). *Educational Gaming Design*. Kent State University, Ohio, ZDA.
- Calabrese, Dave (2014). *Unity 2D Game Development*. Packt Publishing, Birmingham, Velika Britanija.
- Charles, Marie-Therese, Bustard, David, Black, Michaela (2009). Game Inspired Tool Support for e-Learning Processes. *Electronic Journal of e-Learning*. 7, št. 2, str. 101-110.
- Ger Moreno, Pablo (2008). Educational game design for online education. *Computers in Human Behavior*. 24, št. 6, str. 2530-2540.
- Gerlič, Ivan (2000). *Sodobna informacijska tehnologija v izobraževanju*. DZS, Ljubljana, Slovenija.
- Klopfer, Eric, Osterweil, Scot, Salen, Katie (2009). *Moving learning games forward: obstacles, opportunities, & openness*. The Education Arcade, Massachusetts, ZDA.
- Krnel, Dušan (2008). Uporaba informacijsko-komunikacijske tehnologije (IKT) pri pouku v nižjih razredih osnovne šole. *Naravoslovna solnica*. 13, št. 1, str. 6-12.
- Mandel, Theo (1997). *The Elements of User Interface Design*. John Wiley & Sons, Michigan, ZDA.
- Markopoulos , Panos, Read, C. Janet, MacFarlane, Stuart, Hoysniemi ,Johanna (2008). *Evaluating Children's Interactive Products: Principles and Practices for Interaction Designers*. Morgan Kaufmann, Burlington, ZDA.
- Michael, R. David, Chen, Sande (2006*). Serious Games: Games That Educate, Train, And Inform*. Thomson Course Technology, London, Velika Britanija.
- Mori, Ivana (2004). Učenje in poučevanje z računalnikom na razredni stopnji osnovne šole. *Razredni pouk*. 7, št. 1, str. 32-38.
- Roblyer, D. Margaret (2006). *Integrating Educational Technology into Teaching*. Pearson/Merrill Prentice Hall, New Jersey, ZDA.

VIRI

- Audacity (2014). *O Audacity*. Privzeto 20.7.2014 iz http://audacity.sourceforge.net/about/.
- C# Station (2014). *Welcome*. Privzeto 20.7.2014 iz http://csharp-station.com/.
- E-um (2014). *Spletni portal E-um*. Privzeto 25.8.2014 iz http://www.e-um.si.
- Holding Slovenske elektrarne (2014). *Puhčeva interaktivna igralnica*. Privzeto 25.8.2014 iz: http://www.modri-jan.si/igre-in-zabava/puhceva-interaktivna-igralnica.
- Lek (2014). *Igre za otroke*. Privzeto 24.8.2014 iz http://www.lek.si/otroci/.
- Modrijan (2014). *Okolje in jaz 3. Interaktivne prosojnice*. Privzeto 25.8.2014 iz: http://www.modrijan.si/modrijan-fl/.

- Ramel, David. (2013). *Microsoft's C# Programming Language Chosen Most Popular*. Privzeto 20.7.2014 iz: http://visualstudiomagazine.com/articles/2013/01/16/c-sharp-programming-language-wins-honor.aspx.
- Rokus Klett (2014). *Dežela Lilibi*. Privzeto 25.8.2014 iz: http://www.lilibi.si/.
- Unity Technologies (2014). *2d Power*. Privzeto 21.7.2014 iz: http://unity3d.com/pages/2d-power.
- Unity Technologies (2014). *Fast Facts*. Privzeto 21.7.2014 iz: http://unity3d.com/public-relations.
- Unity Technologies (2014). *License Comparisons*. Privzeto 21.7.2014 iz: http://unity3d.com/unity/licenses.
- Unity Technologies (2014). *Showcase*. Privzeto 21.7.2014 iz: http://unity3d.com/showcase.
- Unity Technologies (2014). *Unity Documentation*. Privzeto 21.7.2014 iz: http://unity3d.com/learn/documentation.
- Wikipedia (2014). *GIMP*. Privzeto 20.7.2014 iz http://en.wikipedia.org/wiki/GIMP/.
- Wikipedia (2014). *Ježek Janček*. Privzeto 25.7.2014 iz http://sl.wikipedia.org/wiki/Ježek_Janček/.
- Wikipedia (2014). *MonoDevelop*. Privzeto 20.7.2014 iz http://en.wikipedia.org/wiki/MonoDevelop/.
- Wikipedia (2014). *Unity (game engine)*. Privzeto 20.7.2014 iz http://en.wikipedia.org/wiki/Unity_(game_engine)/.