VISOKO UČILIŠTE ALGEBRA

DIPLOMSKI RAD

Primjena specijalnih mehanika putovanja kroz vrijeme i prostor u razvoju video igara

Filip Ivanko

Zagreb, veljača 2018.

Predgovor

Razvoj računalnih igara je zanimljivo i unosno područje koje je danas pristupačnije nego ikada. Većina alata koji se koriste su besplatni, a razni programski okviri omogućavaju brz i jednostavan razvoj. Za ovu temu sam se odlučio jer mene, kao i mnoge, fascinira jednostavnost razvoja i lakoća ulaska na ovo tržište. Meni je osobito vrlo zanimljiva činjenica da korištenjem suvremenih besplatnih alata jedna osoba može u potpunosti samostalno razviti računalnu igru ili pomoću dobro razvijenih korisničkih zajednica (engl. *community*) koje postoje oko pojedinih alata vrlo lako naći suradnike. Ovaj spektar mogućnosti mnogima daje priliku okušati se u razvoju igara. Kao posljedicu smo dobili veliki broj najraznovrsnijih igara koje izdaju timovi i pojedinci. Danas je izazov nalaženje kreativnih ideja kako bismo se izdvojili iz sve većeg broja igara koje svakodnevno izlaze. Cilj ovog rada je pronaći igre na tržištu koje koriste inovativne mehanike putovanja neprostornim dimenzijama, naći njihove zajedničke osobine i implementirati ih u prototip igre koju ću razviti.

Prilikom uvezivanja rada, umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme diplomskog rada koji ste preuzeli u studentskoj referadi

Sažetak

U ovom radu planiram istražiti zajedničke značajke računalnih igara koje koriste specijalne mehanike kretanja neprostornim dimenzijama, bilo putovanje kroz vrijeme ili nekom drugom vrstom neprostornih dimenzija. Kroz istraživanje koje ću provesti odredit ću zajedničke značajke ovakvih igara i implementirati ih u razvoj prototipa računalne igre. U ovom radu postavljam tri hipoteze. Prva je da igre ovog tipa na manjem prostoru stvaraju veću kompleksnost i stoga imaju manje nivoe. Druga hipoteza je da koriste manji broj grafičkih resursa (grafika, animacija i sl.). Treća hipoteza je da su korišteni resursi jednostavniji, a samim tim je i kreacija resursa brža.

Prototip igre razvijat ću korištenjem Unity razvojne okoline. Do stadija prototipa implementirat ću igru s glavnim značajkama koje ću utvrditi istraživanjem. Sve grafičke resurse (ilustracije, animacije), kao i svu glazbu te razvoj softverskog dijela ove računalne igre napravit ću sam kako bi uz istraživanje i na primjeru potvrdio ranije navedene hipoteze. Sve hipoteze upućuju da je moguća ušteda vremena u razvoju ako se u igru implementiraju spomenute vrste igraćih mehanika. Tehničku implementaciju prototipa računalne igre planiram dokumentirati u poglavljima koja dolaze nakon istraživanja kako bih pojasnio postupak izrade same igre i pojedinosti razvojne okoline Unity.

Ključne riječi: računalne igre, Unity, mehanike računalnih igara

Summary

In this paper I plan to explore the common attributes of computer games that make use of player movement in nonspecial dimensions, whether traveling through time or through a higher spatial dimension.I'm going to be conducting a research to discover the common traits of games that define this genre and implement these traits in the prototype of a computer game that I will be developing.I propose a hypothesis that games of this type create more complexity in a smaller space and as a result have smaller levels. The second hypothesis that I will try to prove it that the games that use this kind of mechanics use less graphical resources (graphics, animations etc.). The third hypothesis is that the resources used for this kind of games are simpler.

The computer game prototype will be developed using Unity framework. I will be developing a game with the main characteristics determined by the results of the before mentioned research. All the graphical resources (graphics and animations) as well as all the music and programming on the game prototype will be done by me personally. By doing this I will try to test the before mentioned hypothesis that including this sort of mechanic shortens the time needed for development. The technical implementation of the game prototype will be documented in the chapters following the chapters containing the research. In the chapters on the technical implementation I plan to explain the process by which the prototype was created and the details of the Unity framework.

Keywords: computer games, Unity, game mechanics.

Sadržaj

1 Uvod

Korištenjem lako dostupnih programskih okvira danas je razvoj računalnih igara dostupniji nego ikad širokom broju zainteresiranih. Trenutna eksplozija broja inovativnih nezavisnih (engl. *indie*) računalnih igara dostupnih kroz digitalne distribucijske platforme je izravna posljedica ove činjenice. Kao glavnu temu ovog rada planiram obraditi razvoj računalne igre korištenjem Unity programskog okvira. Odabir karakteristika igre obavit ću kroz istraživanje. Namjeravam istražiti igre koje koriste specijalne mehanike kretanja kroz vrijeme i prostor. Glavna karakteristika igara koje sam odabrao za svoje istraživanje je mogućnost igrača da se kreće kroz neprostorne dimenzije, kao na primjer putovanje kroz vrijeme, putovanje kroz višu dimenziju prostora (npr. kroz treću dimenziju ako je igra 2D), kroz alternativne dimenzije (npr. dimenzije u kojima vladaju neki drugi fizikalni zakoni) ili neki drugi način kretanja u prostoru koji je van geometrijske dimenzionalnosti igre. Mehanike ove vrste stvaraju veliku kompleksnost na malom prostoru. U ovom radu postavljam tri hipoteze. Prva hipoteza je da su nivoi manji zbog povećanja kompleksnosti igre. Druga je da se korištenjem ove vrste mehanike smanjuje potreba za velikim brojem resursa kao što su ilustracije ili animacije jer se isti resursi češće ponavljaju. Treća hipoteza je da su ilustracije jednostavnije jer je fokus na rješavanju geometrijskih problema ili zagonetki, a ne na vjernosti prikaza. Istraživanje ću provesti na petnaest igara koje koriste ove vrste mehanika. Bilježit ću parametre koji su prisutni i u kojem intenzitetu. Rezultate istraživanja ću implementirati u izradi prototipa računalne igre korištenjem Unity razvojne tehnologije. Razvoj ću dokumentirati s tehničke strane kako bih prezentirao način razvoja igara korištenjem Unity programskog okvira. U drugom poglavlju ću dokumentirati podatke dobivene istraživanjem i opisati metodologiju provođenja koju sam sam osmislio dok ću u trećem poglavlju interpretirati podatke. Poglavlja četiri i pet odnose se na razvoj prototipa računalne igre i implementaciju karakteristika utvrđenih istraživanjem. Poglavlje četiri odnosi se na razvoj osnovnih mehanika poput kretanja ili kamere dok se poglavlje pet odnosi na kompleksnije mehanike koje čine igru zanimljivijom. Nakon istraživanja i implementacije ocijenit ću jesu li ranije navedene hipoteze potvrđene i donijeti zaključak do kojeg sam došao tijekom izrade ovog rada.

2 Istraživanje

Kako bih utvrdio zajedničke karakteristike igara koje koriste mehanike dimenzijskog ili vremenskog kretanja (u daljnjem tekstu specijalna mehanika), proveo sam istraživanje. U uzorak istraživanja sam uzeo petnaest igara koje kao barem jednu mehaniku igre koriste neprostorno kretanje igrača. Igre uključene u istraživanje navedene su u tablici 2.1. Metodologija istraživanja uključuje igranje svake igre minimalno tri sata i bilježenje promatranih karakteristika prema mom subjektivnom doživljaju. Metodologiju provođenja istraživanja u potpunosti sam samostalno osmislio za potrebe ovog rada. Za svaku igru zabilježio sam jedanaest podataka. Neke značajke ću bilježiti na skali od 1 do 10, npr. veličinu nivoa, broj korištenih grafičkih resursa, detaljnost resursa. Ocjena 1 bi označavala iznimno mali ili nepostojeći značaj parametra u igri dok bi 10 označavala iznimno velik. Također ću bilježiti koliko su specijalne mehanike važne u pojedinoj igri, naprimjer radi li se o osnovnoj mehanici oko koje je koncipirana igra ili se samo ograničeno koristi. Ovaj podatak koristit ću kao težinu za pojedinu igru. Podatci igara s većom težinom imat će veću težinu u istraživanju. Neke značajke ću bilježiti binarno tako da odredim posjeduje li igra neku karakteristiku ili ne. Dio značajki ću bilježiti po kategorijama, kao na primjer umjetnički stil (pripada li kategoriji realističnog ili stiliziranog).

Podatci koji se bilježe u istraživanju:

1. Važnost specijalne mehanike za igru (od 1 do 10)
2. Veličina nivoa (na ljestvici od 1 do 10)
3. Količina različitih resursa koji se koriste (na ljestvici od 1 do 10)
4. Detaljnost resursa (na ljestvici od 1 do 10)
5. Umjetnički stil (realističan, stiliziran)
6. Koristi li igra *inventory* mehaniku (skupljanje predmeta) (da/ne)
7. Važnost neprijatelja i borbe u igri (na ljestvici od 1 do 10)
8. Opis specijalne mehanike (Vraćanje i manipulacija vremena, Promjena vremena ili dimenzije, Ponavljanje istog perioda vremena, Kretanje nevidljivim dimenzijama)
9. 2D/3D (radi li se o 2D ili 3D prikazu)
10. Strateška kompleksnost problema koji se unutar nivoa rješavaju (na ljestvici od 1 do 10)

11. Temeljna mehanika kretanja (2D platformer, 3D platformer, *point and click*, pucačka igra igrana u prvom licu (engl. *first person shooter*, skraćeno FPS).

Specijalne mehanike u svim igrama obuhvaćenim istraživanjem nisu identične. U uzorku obuhvaćenom istraživanjem primijetio sam kako se specijalne mehanike korištene u igrama mogu klasificirati u četiri kategorije: Vraćanje i manipulacija vremena - igrač može vraćati vrijeme (naprimjer spriječiti pad ili pogibiju); Promjena vremena ili dimenzije - igrač može mijenjati dimenziju ili vrijeme u kojem se nalazi (npr. igrač ne može proći prema nekom dijelu nivoa u sadašnjosti, ali ako se prebaci u neko drugo vremensko razdoblje, naprimjer 100 godina u prošlost, prepreka ne postoji); Ponavljanje istog perioda vremena - igra se odvija unutar ograničenog perioda vremena koje se konstantno vraća na početno vrijeme; Kretanje nevidljivim dimenzijama - kretanje u prostoru kroz višu dimenziju koja se ne vidi (naprimjer kretanje kroz višu dimenziju u igri Portal).

Strateška kompleksnost problema koji se unutar nivoa rješavaju označava težinu zagonetki koje igrač mora riješiti.

Ne postoje igre koje se u potpunosti sastoje od zagonetki jer su igre bazirane na zagonetkama uvijek spojene i s nekim drugim žanrom kako bi se podignula privlačnost same igre. Zagonetke ne čine samo igranje igre (engl. *gameplay*). Zagonetke su specifični problemi. Razvoj igara uključuje razvoj sistema unutar kojih se generiraju zagonetke (Rollings et al., 2004).

Za kategoričke varijable ću odrediti koliko igara spada u pojedinu kategoriju te težinu igara određene kategorije. Za numeričke varijable ću odrediti vaganu prosječnu vrijednost s težinama određenim prema varijabli važnost specijalne mehanike (težinama igara u istraživanju). Odredit ću Pearsonov koeficijent korelacije i nagib regresijskog pravca kako bih utvrdio postoji li korelacija između promatranih varijabli i važnost specijalne mehanike za igru te u kojem intenzitetu i kojem smjeru. Ove mjere razmatram da vidim je li pojedini parametar povezan s važnosti specijalne mehanike (naprimjer ako vrijednost važnosti specijalne mehanike raste, raste li i vrijednost parametra), u kojem intenzitetu (nagib regresijskog pravca) i koliko pouzdano (Pearsonov koeficijent korelacije). U odnosu na ove dvije varijable donijet ću zaključak da li implementirati pojedini rezultat u prototipu (ako pojedini pokazatelj nije ni u kakvoj statističkoj povezanosti sa specijalnom mehanikom u prototipu, rezultati neće biti implementirani).

Regresija je statistička metoda kojom se za određeni skup podataka računa povezanost dviju varijabli i intenzitet te povezanosti; ako jedna raste, raste li ili pada druga i obrnuto. Regresijski pravac (linija trenda) predstavlja linearnu aproksimaciju svih točaka skupa podataka (središnju liniju između točaka) pomoću koje se može pretpostaviti linearna ovisnost jedne varijable o drugoj. Što je pravac strmiji, to zavisna varijabla jače reagira na promjenu nezavisne varijable. U mom istraživanju nezavisna varijabla je uvijek važnost specijalne mehanike. Nagib regresijskog pravca može biti pozitivan ili negativan, ovisno jesu li varijable pozitivno ili negativno povezane. Vrijednost mu se može kretati između ekstrema od nula ako je regresijski pravac paralelan s osi x do beskonačnosti ako je paralelan s osi y.

Koeficijent korelacije jedna je numerička karakteristika dvodimenzionalnog slučajnog vektora koja može poslužiti za analizu zavisnosti među njegovim komponentama (Benšić et al., 2013).

Za procjenu koeficijenta korelacije možemo koristiti nekoliko procjenitelja. Ovdje ćemo spomenuti samo procjenitelja koji se zove Pearsonov korelacijski koeficijent i koristi se kod neprekidnih slučajnih varijabli. Ako su (x1, y1), ..., (xn, yn) parovi nezavisnih realizacija slučajnog vektora (X, Y), onda se iznos Pearsonova korelacijskog koeficijenta računa pomoću izraza (1) (Benšić et al., 2013).

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x}_n)(y_i - \overline{y}_n)}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x}_n)^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y}_n)^2}} \qquad (1)$$

Pearsonov koeficijent korelacije je statistička mjera kojom se mjeri korelacija, tj. koliko blisko točke prate regresijski pravac (liniju trenda). Vrijednost se kreće od 1 do -1 ovisno radi li se o pozitivnoj ili negativnoj korelaciji. Ukoliko je koeficijent jednak 1 ili -1, radi se o savršenoj pozitivnoj ili negativnoj korelaciji. To znači da se sve točke nalaze na pravcu. Ako je vrijednost koeficijenta jednaka nula, to znači da nema korelacije te se točke ne mogu grupirati ni oko jednog pravca.

U sklopu ovog istraživanja postavljam tri hipoteze koje ću ispitati u toku istraživanja. Prva hipoteza je da igre koje koriste specijalne mehanike imaju manje nivoe od prosjeka. Druga hipoteza je da igre koje koriste specijalne mehanike koriste manje grafičkih resursa. Treća hipoteza je da igre koje koriste specijalne mehanike koriste grafičke resurse s manje detalja

od prosjeka nezavisnih igara na tržištu (prosjek po mojem subjektivnom događaju). Za sve ove hipoteze pretpostavljam da postoji opće pravilo da što se igra jače oslanja na specijalne mehanike, da su ove pretpostavljene karakteristike jače izražene.

	Naziv	Razvojni studio	Izdavač	Godina
1.	Braid	Number None	Number None	2008.
2.	Chronology	Bedtime Digital Games	Bedtime Digital Games	2014.
3.	Day of the tentacle remastered	Duble fine production	Duble fine production	2016.
4.	Evoland	Shiro games	Shiro games	2013.
5.	Evoland 2	Shiro games	Shiro games	2015.
6.	Portal	Valve	Valve	2007.
7.	Quantum conundrum	Archlight games	Square enix	2012.
8.	Seasons after fall	Swing swing submarine	Swing swing submarine	2016.
9.	The sexy brutale	Cavalier games, Tequila works	Tequila works	2017.
10.	The silent age	House on fire	Median 4	2012.
11.	Tick Tock isle	Squiddershins	Squiddershins	2015.
12.	Zelda: Link betwen worlds	Nintendo	Nintendo	2013.
13.	Zelda: Ocarina of time	Nintendo	Nintendo	1998.
14.	Zelda: Mejoras mask	Nintendo	Nintendo	2000.
15.	Prince of Persia: sands of time	Kudosoft	Ubisoft	2003.

Tablica 2.1 Igre uključene u istraživanje

2.1 Važnost specijalne mehanike

Ovaj pokazatelj je ključan u provođenju istraživanja. Pokazuje koliko je specijalna mehanika (mehanika povezana s kretanjem dimenzijama ili vremenom) važna za način igranja u pojedinoj igri. Bilježi se u rasponu od 1 do 10. Vrijednosti za svaku igru navedene su u tablici 2.2. Ako je pokazatelj 1, tada je specijalna mehanika korištena samo marginalno, u malom broju slučajeva i nije ključna za način igranja. Pokazatelj je 10 ako je mehanika ključna za igranje, koristi se konstantno i igra je građena oko same mehanike. Zbroj rezultata važnosti svih igara u istraživanju je 82, a postotak važnosti tj. težina svake igre je postotni udio svake igre u ukupnoj težini svih igara uključenih u istraživanje.

Postotak važnosti specijalne mehanike nam govori koliko je pojedina igra važna za istraživanje. Što u pojedinoj igri specijalna mehanika ima važniju ulogu, time je i težina svih njezinih karakteristika veća za istraživanje. Ova vrijednost se koristi pri vrednovanju svih parametara pojedine igre (npr. veličine nivoa). Koristi se kao težinski faktor (ako se vrijednosti brojčano iskazuju) jer su za istraživanje važnije vrijednosti parametara igara koje intenzivno koriste specijalne mehanike od vrijednosti parametara igara koje ih koriste samo usputno.

	Naziv igre	Postotak važnosti (Težina za ostale varijable)	Važnost specijalne mehanike
1.	Braid	10.98%	9
2.	Chronology	7.32%	6
3.	Day of the tentacle	8.54%	7
4.	Evoland	1.22%	1
5.	Evoland 2	2.44%	2
6.	Quantum connundrum	9.76%	8
7.	Seasons after fall	7.32%	6
8.	The sexy brutal	9.76%	8
9.	The slent age	10.98%	9
10.	Tick tock isle	10.98%	9
11.	Legend of Zelda Ocarina of time	3.66%	3
12.	Legend of Zelda Link in time	1.22%	1
13.	Legend of Zelda Majoras Mask	3.66%	3
14.	Portal	9.76%	8
15.	Prince of Persia Sands of time	2.44%	2
		100.00%	82

Tablica 2.2 Važnost specijalne mehanike

2.2 Veličina nivoa

Veličina nivoa je jedan od ključnih faktora koji određuju vrijeme razvoja kroz povećavanje vremena potrebnog za dizajn nivoa i količinu rada potrebnu da se nivo ispuni resursima. Veličine nivoa svake igre prikazane su u tablici 2.3. U ovom istraživanju postavljam tezu da se igre koje koriste specijalne mehanike definirane u ovom radu u pravilu sastoje od malih nivoa. Ovu tezu temeljim na pretpostavci da se korištenjem specijalnih mehanika povećava kompleksnost tako da je moguće na manjem prostoru kreirati više sadržaja korištenjem istih resursa.

	Naziv igre	Postotak važnosti (Težina)	Veličina nivoa	Veličina nivoa puta važnost
1.	Braid	10.98%	1	0.11
2.	Chronology	7.32%	5	0.37
3.	Day of the tentacle	8.54%	3	0.26
4.	Evoland	1.22%	6	0.07
5.	Evoland 2	2.44%	5	0.12
6.	Quantum connundrum	9.76%	2	0.20
7.	Seasons after fall	7.32%	6	0.44
8.	The sexy brutal	9.76%	4	0.39
9.	The slent age	10.98%	2	0.22
10.	Tick tock isle	10.98%	2	0.22
11.	Legend of Zelda Ocarina of time	3.66%	4	0.15
12.	Legend of Zelda Link in time	1.22%	8	0.10
13.	Legend of Zelda Majoras Mask	3.66%	3	0.11
14.	Portal	9.76%	1	0.10
15.	Prince of Persia Sands of time	2.44%	2	0.05
		100.00%		2.89

Tablica 2.3 Veličina nivoa

Prema napravljenom istraživanju prosječna vagana ocjena za veličinu nivoa je 2.89. Kao težine korištene su vrijednosti postotka važnosti posebne mehanike. Prosječna vagana ocjena je 2.89 od 10. Pearsonov koeficijent korelacije veličine nivoa uz važnost specijalne mehanike u promatranom uzorku iznosi -0.6470779448, a nagib regresijskog pravca -0.4287819253.

2.3 Količina resursa

Kao i veličina nivoa, količina resursa direktno utječe na vremensko trajanje razvoja računalne igre i troškove razvoja. Razmatranjem prosječne količine resursa u istraživanom

uzorku planiram doći do određenog pravila koje nam govori na koji način količina resursa ovisi o specijalnoj mehanici. Vrijednosti ove varijable za svaku igru navedene su u tablici 2.4. Također kao i u primjeru veličine nivoa, postavljam pretpostavku da igre sa specijalnim mehanikama, koje se u pravilu koncentriraju na logiku, koriste manji broj resursa.

	Naziv igre	Postotak važnosti (Težina)	Količina resursa	Količina resursa puta važnost
1.	Braid	10.98%	4	0.44
2.	Chronology	7.32%	5	0.37
3.	Day of the tentacle	8.54%	4	0.34
4.	Evoland	1.22%	5	0.06
5.	Evoland 2	2.44%	6	0.15
6.	Quantum connundrum	9.76%	3	0.29
7.	Seasons after fall	7.32%	4	0.29
8.	The sexy brutal	9.76%	5	0.49
9.	The slent age	10.98%	3	0.33
10.	Tick tock isle	10.98%	2	0.22
11.	Legend of Zelda Ocarina of time	3.66%	5	0.18
12.	Legend of Zelda Link in time	1.22%	6	0.07
13.	Legend of Zelda Majoras Mask	3.66%	4	0.15
14.	Portal	9.76%	3	0.29
15.	Prince of Persia Sands of time	2.44%	8	0.20
		100.00%		3.87

Tablica 2.4 Količina resursa

Prema napravljenom istraživanju prosječna vagana ocjena za količinu resursa je 3.87, kao težine korištene su vrijednosti postotka važnosti posebne mehanike. Prosječna vagana ocjena je 3.87 od 10. Pearsonov koeficijent korelacije količine resursa uz važnost specijalne mehanike u promatranom uzorku iznosi -0.7506752118, a nagib regresijskog pravca -0.3629666012.

2.4 Detaljnost resursa

Pri implementaciji igre možemo se odlučiti za grafičke resurse s više ili manje detalja. Ovo je još jedna metrika koja uvelike utječe na vrijeme i troškove razvoja. Kao i za prethodne dvije metrike, i za ovu postavljam istu tezu. Postavljam pretpostavku da igre koje koriste specijalne mehanike mogu koristiti manje detaljne resurse jer se fokus pomiče sa samih vizuala na rješavanje zagonetki. Vrijednosti ove varijable za svaku igru navedene su u tablici 2.5.

	Naziv igre	Postotak važnosti (Težina)	Detaljnost resursa	Detaljnost resursa puta važnost
1.	Braid	10.98%	5	0.55
2.	Chronology	7.32%	6	0.44
3.	Day of the tentacle	8.54%	3	0.26
4.	Evoland	1.22%	2	0.02
5.	Evoland 2	2.44%	3	0.07
6.	Quantum connundrum	9.76%	4	0.39
7.	Seasons after fall	7.32%	3	0.22
8.	The sexy brutal	9.76%	7	0.68
9.	The slent age	10.98%	1	0.11
10.	Tick tock isle	10.98%	2	0.22
11.	Legend of Zelda Ocarina of time	3.66%	4	0.15
12.	Legend of Zelda Link in time	1.22%	3	0.04
13.	Legend of Zelda Majoras Mask	3.66%	4	0.15
14.	Portal	9.76%	3	0.29
15.	Prince of Persia Sands of time	2.44%	8	0.20
		100.00%		3.78

Tablica 2.5 Detaljnost resursa

Prema napravljenom istraživanju prosječna vagana ocjena za detaljnost resursa je 3.78, kao težine korištene su vrijednosti postotka važnosti posebne mehanike, prosječna vagana ocjena je 3.78 od 10. Pearsonov koeficijent korelacije detaljnosti resursa uz važnost specijalne mehanike u promatranom uzorku iznosi -0.08433078011, a nagib regresijskog pravca -0.05206286837.

2.5 Umjetnički stil

U ovom dijelu istraživanja razmatram umjetnički stil igara u istraživanju kako bih utvrdio kakav umjetnički stil prevladava u ovom žanru videoigara. Umjetnički stil za svaku igru naveden je u tablici 2.6. Ovdje utvrđen umjetnički stil primijenit ću u prototipu igre koji ću izraditi kao dio ovog rada.

	Naziv igre	Postotak važnosti (Težina)	Umjetnički stil
1.	Braid	10.98%	Stiliziran
2.	Chronology	7.32%	Stiliziran
3.	Day of the tentacle	8.54%	Stiliziran
4.	Evoland	1.22%	Stiliziran
5.	Evoland 2	2.44%	Stiliziran
6.	Quantum connundrum	9.76%	Stiliziran
7.	Seasons after fall	7.32%	Stiliziran
8.	The sexy brutal	9.76%	Stiliziran
9.	The slent age	10.98%	Realističan
10.	Tick tock isle	10.98%	Stiliziran
11.	Legend of Zelda Ocarina of time	3.66%	Stiliziran
12.	Legend of Zelda Link in time	1.22%	Stiliziran
13.	Legend of Zelda Majoras Mask	3.66%	Stiliziran
14.	Portal	9.76%	Realističan
15.	Prince of Persia Sands of time	2.44%	Realističan
		100.00%	
	stiliziran	76.83%	12
	realističan	23.17%	3

Tablica 2.6 Umjetnički stil

Najveći dio igara koristi stiliziran umjetnički stil. Dvanaest igara upotrebljava resurse izvedene u stiliziranom dok samo tri u realističnom stilu. Ukupna težina igara koje koriste stilizirani umjetnički stil je 76.83 % dok je ukupni težinski faktor igara koje koriste realističan stil 23.17 %.

2.6 Važnost mehanike borbe

Borbeni sistem je jedan od najvažnijih mehanika u igrama te uvelike određuje način igranja i atmosferu igre. Borbene mehanike su u nekim žanrovima osnova načina igranja dok se u drugima koriste kao dopuna neke druge temeljne mehanike. U tablici 2.7 navedene su vrijednosti važnosti borbenog sistema za svaku igru uključenu u istraživanje.

	Naziv igre	Postotak važnosti (Težina)	Važnost CS	Važnost CS puta važnost
1.	Braid	10.98%	1	0.11
2.	Chronology	7.32%	1	0.07
3.	Day of the tentacle	8.54%	1	0.09
4.	Evoland	1.22%	4	0.05
5.	Evoland 2	2.44%	3	0.07
6.	Quantum connundrum	9.76%	1	0.10
7.	Seasons after fall	7.32%	1	0.07
8.	The sexy brutal	9.76%	1	0.10
9.	The slent age	10.98%	1	0.11
10.	Tick tock isle	10.98%	1	0.11
11.	Legend of Zelda Ocarina of time	3.66%	4	0.15
12.	Legend of Zelda Link in time	1.22%	6	0.07
13.	Legend of Zelda Majoras Mask	3.66%	4	0.15
14.	Portal	9.76%	1	0.10
15.	Prince of Persia Sands of time	2.44%	6	0.15
		100.00%		1.49

Tablica 2.7 Važnost mehanike borbe

Prema napravljenom istraživanju prosječna vagana ocjena za važnost borbene mehanike je 1.49, kao težine korištene su vrijednosti postotka važnosti posebne mehanike. Prosječna vagana ocjena je 1.49 od 10. Pearsonov koeficijent korelacije detaljnosti resursa uz važnost specijalne mehanike u promatranom uzorku iznosi -0.8937858807, a nagib regresijskog pravca -0.5510805501.

2.7 Inventory

Inventory je mehanika koja podrazumijeva prikupljanje objekata u nivoima, njihovo spremanje, prenošenje i korištenje na drugom mjestu ili više drugih mjesta. Igre koje koriste *inventory* se, u pravilu, značajno razlikuju od igara koje ga ne koriste. Tablica 2.8 prikazuje koje igre u uzorku koriste *inventory* mehaniku, a koje ne. U istraživanju obilježja igara u ovom žanru odredio sam broj igara u promatranom uzorku koje koriste *inventory* mehaniku.

	Naziv igre	Postotak važnosti (Težina)	Inventory
1.	Braid	10.98%	ne
2.	Chronology	7.32%	ne
3.	Day of the tentacle	8.54%	da
4.	Evoland	1.22%	da
5.	Evoland 2	2.44%	da
6.	Quantum connundrum	9.76%	ne
7.	Seasons after fall	7.32%	ne
8.	The sexy brutal	9.76%	da
9.	The slent age	10.98%	da
10.	Tick tock isle	10.98%	da
11.	Legend of Zelda Ocarina of time	3.66%	da
12.	Legend of Zelda Link in time	1.22%	da
13.	Legend of Zelda Majoras Mask	3.66%	da
14.	Portal	9.76%	ne
15.	Prince of Persia Sands of time	2.44%	ne
		100.00%	
	da	52.44%	9
	ne	47.56%	6

Tablica 2.8 *Inventory* mehanika

Veći dio igara u uzorku koristi *inventory* mehaniku. Devet igara koristi *inventory* dok ju šest igara ne koristi. Zbroj težinskih faktora igara koje koriste *inventory* mehaniku je 52.44 % dok je zbroj težinskih faktora igara koje ju ne koriste 47.56 %

2.8 Opis specijalne mehanike

Specijalne mehanike su definirajuća značajka ovog žanra. U istraživanju su definirane četiri različite specijalne mehanike. U ovom dijelu ću utvrditi koja je specijalna mehanika najrasprostranjenija, tj. najpopularnija u ovom tipu igara. Tablica 2.9 navodi vrste specijalnih mehanika igara uključenih u istraživanje.

	Naziv igre	Postotak važnosti (Težina)	Opis specijalne mehanike
1.	Braid	10.98%	Vraćanje i manipulacija vremena
2.	Chronology	7.32%	Promjena vremena ili dimenzije
3.	Day of the tentacle	8.54%	Promjena vremena ili dimenzije
4.	Evoland	1.22%	Promjena vremena ili dimenzije
5.	Evoland 2	2.44%	Promjena vremena ili dimenzije
6.	Quantum connundrum	9.76%	Promjena vremena ili dimenzije
7.	Seasons after fall	7.32%	Promjena vremena ili dimenzije
8.	The sexy brutal	9.76%	Ponavljanje istog perioda vremena
9.	The slent age	10.98%	Promjena vremena ili dimenzije
10.	Tick tock isle	10.98%	Promjena vremena ili dimenzije
11.	Legend of Zelda Ocarina of time	3.66%	Promjena vremena ili dimenzije
12.	Legend of Zelda Link in time	1.22%	Promjena vremena ili dimenzije
13.	Legend of Zelda Majoras Mask	3.66%	Ponavljanje istog perioda vremena
14.	Portal	9.76%	Kretanje nevidljivim dimenzijama
15.	Prince of Persia Sands of time	2.44%	Vraćanje i manipulacija vremena
		100.00%	
	Vraćanje i manipulacija vremena	13.41%	2
	Promjena vremena ili dimenzije	63.41%	10
	Ponavljanje istog perioda vremena	13.41%	2
	Kretanje nevidljivim dimenzijama	9.76%	1

Tablica 2.9 Opis specijalne mehanike

Najrasprostranjenija mehanika je Promjena vremena ili dimenzije. Ona je primijenjena u 10 igara u uzorku i s težinskim faktorom od 63.41 %. Vraćanje i manipulacija vremena je primijenjena u dva slučaja s težinskim faktorom od 13.41 %. Ponavljanje istog perioda vremena je također primijenjeno u dva slučaja s težinskim faktorom od 13.41 % dok je kretanje nevidljivim dimenzijama implementirano samo u jednom primjeru s težinom od 9.76 %.

2.9 Perspektiva

Perspektiva je jedna od osnovnih značajki videoigara. Je li igra izvedena kao 2D ili 3D igra ima veliki utjecaj na način igranja. Sama tehnička izvedba igre bitno se razlikuje u svakom od slučajeva. Tablica 2.10 daje pregled perspektive igara uključenih u istraživanje.

	Naziv igre	Postotak važnosti (Težina)	Perspektiva
1.	Braid	10.98%	2D
2.	Chronology	7.32%	2D
3.	Day of the tentacle	8.54%	2D
4.	Evoland	1.22%	2D/3D
5.	Evoland 2	2.44%	2D/3D
6.	Quantum connundrum	9.76%	3D
7.	Seasons after fall	7.32%	2D
8.	The sexy brutal	9.76%	3D
9.	The slent age	10.98%	2D
10.	Tick tock isle	10.98%	2D
11.	Legend of Zelda Ocarina of time	3.66%	3D
12.	Legend of Zelda Link in time	1.22%	2D
13.	Legend of Zelda Majoras Mask	3.66%	3D
14.	Portal	9.76%	3D
15.	Prince of Persia Sands of time	2.44%	3D
		100.00%	
	2D	57.32%	7
	3D	39.02%	6
	2D i 3D	3.66%	2

Tablica 2.10 Perspektiva

Igre u promatranom uzorku koriste 2D i 3D perspektivu otprilike u jednakom broju, ali težine vuku prema 2D perspektivi. U sedam slučajeva u igrama se koristi 2D perspektiva s težinskim faktorom 57.32 %. 3D perspektiva koristi se u šest slučajeva s težinskim faktorom 39.02 %. Dvije igre u uzorku koriste u nekom dijelu igre 2D perspektivu, a u drugom 3D perspektivu i one čine 3.66 % ukupne težine.

2.10 Kompleksnost zadataka

Gotovo sve igre u sebi sadrže rješavanje određene vrste logičkih problema. Kompleksnost tih problema je bitna odrednica u dizajnu igre. Sve igre iz uzorka koriste određene vrste zadataka, bilo da je riječ o problemima koji se vrte oko omogućavanja igraču da dosegne neku poziciju, interakciji igrača s okolinom na način da skuplja i koristi objekte u nivoima ili rješavanje nekog drugog logičkog problema. Sve igre u uzorku koriste specijalne mehanike kako bi pomoću njih rješavale neke vrste logičkih problema, težine tih zadataka su pokazane u tablici 2.11.

	Naziv igre	Postotak važnosti (težina)	Kompleksnost zadataka	Kompleksnost zadataka puta važnost
1.	Braid	10.98%	8	0.88
2.	Chronology	7.32%	5	0.37
3.	Day of the tentacle	8.54%	7	0.60
4.	Evoland	1.22%	2	0.02
5.	Evoland 2	2.44%	2	0.05
6.	Quantum connundrum	9.76%	8	0.78
7.	Seasons after fall	7.32%	6	0.44
8.	The sexy brutal	9.76%	7	0.68
9.	The slent age	10.98%	6	0.66
10.	Tick tock isle	10.98%	7	0.77
11.	Legend of Zelda Ocarina of time	3.66%	4	0.15
12.	Legend of Zelda Link in time	1.22%	2	0.02
13.	Legend of Zelda Majoras Mask	3.66%	5	0.18
14.	Portal	9.76%	8	0.78
15.	Prince of Persia Sands of time	2.44%	2	0.05
		100.00%		6.43

Tablica 2.11 Kompleksnost zadataka

Prema napravljenom istraživanju prosječna vagana ocjena za kompleksnost zadataka je 6.43, kao težine korištene su vrijednosti postotka važnosti posebne mehanike. Prosječna vagana ocjena je 6.43 od 10. Pearsonov koeficijent korelacije detaljnosti resursa uz važnost specijalne mehanike u promatranom uzorku iznosi 0.9309624878, a nagib regresijskog pravca 0.7008840864.

2.11 Mehanika kretanja

Mehanika kretanja opisuje kako se igrač kreće kroz nivo i kakvu interakciju ostvaruje s igrom. Velika je razlika između doživljaja igre s „uperi i klikni“ (engl. *point and click*) mehanikom gdje igrač mora samo kliknuti mišem gdje želi da se lik u igri pomakne i platformera gdje mora ocijeniti daljinu i visinu svakog skoka. Vrste kretanja koje koriste igre u uzorku prikazane su u tablici 2.12. Ako igre u ovom žanru preferiraju određenu mehaniku kretanja, u ovom dijelu ćemo to otkriti.

	Naziv igre	Postotak važnosti (Težina)	Mehanika kretanja
1.	Braid	10.98%	Platformer
2.	Chronology	7.32%	Platformer
3.	Day of the tentacle	8.54%	Point and click
4.	Evoland	1.22%	Izometrijski
5.	Evoland 2	2.44%	Izometrijski
6.	Quantum connundrum	9.76%	FPS
7.	Seasons after fall	7.32%	Platformer
8.	The sexy brutal	9.76%	Izometrijski
9.	The slent age	10.98%	Pont and click
10.	Tick tock isle	10.98%	Platformer
11.	Legend of Zelda Ocarina of time	3.66%	Platformer
12.	Legend of Zelda Link in time	1.22%	Izometrijski
13.	Legend of Zelda Majoras Mask	3.66%	Platformer
14.	Portal	9.76%	FPS
15.	Prince of Persia Sands of time	2.44%	Platformer
		100.00%	
	Platformer	46.34%	7
	Point and click	19.51%	2
	Izometrijski	14.63%	4
	FPS	19.51%	2

Tablica 2.12 Mehanika kretanja

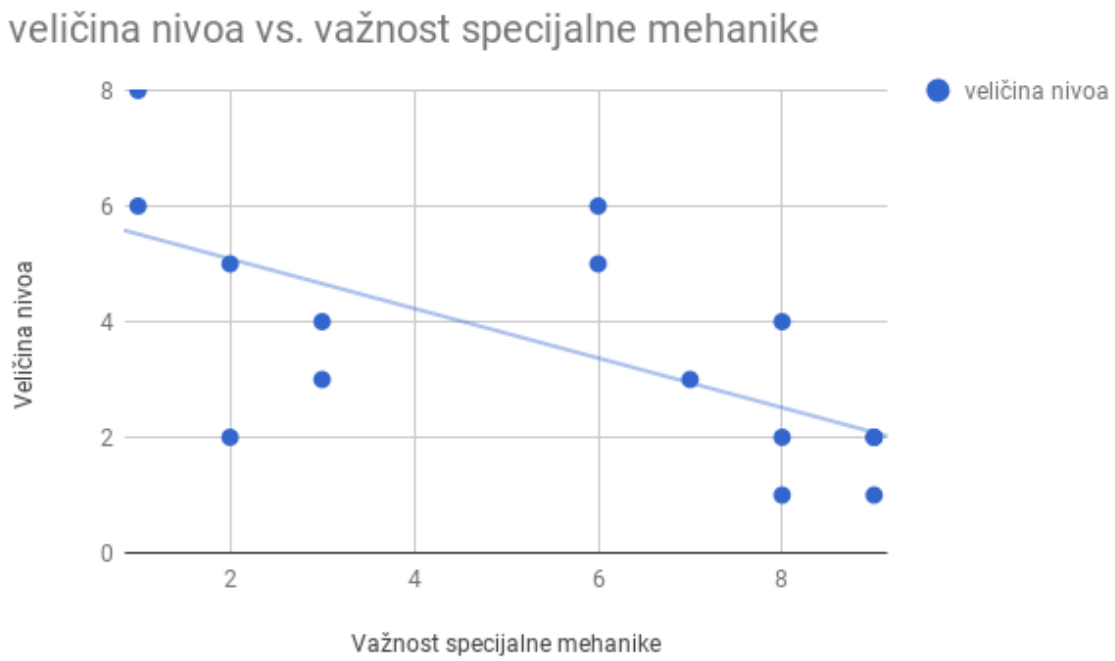
Platformer je najčešći tip igre u promatranom uzorku. Sedam igara iz promatranog uzorka koristi platformer kao način kretanja. Ukupan težinski faktor tih igara je 46.36 %. Četiri igre koriste izometrijski način s ukupnom težinom 14.63 %. Dvije igre koriste *point and click* mehaniku kretanja sa zbrojem težinskih faktora 19.51 %. FPS mehaniku koriste dvije igre s težinskim faktorom 19.51 %.

3 Analiza rezultata

Nakon provedenog istraživanja u ovom dijelu ću donijeti zaključke. Prema zaključcima istraživanja ću odrediti kako ću značajke implementirati u prototipu računalne igre koju ću razviti u sklopu ovog rada. Za kategoričke varijable kao što je vrsta specijalne mehanike u realizaciji igre implementirat ću kategoriju koja se pokaže kao najzastupljenija u istraživanju. Varijable s brojevnim ocjenama implementirat ću u prototipu u intenzitetu koji odgovara vaganom prosjeku ocjena svih igara uključenih u istraživanje. Iz rezultata istraživanja zaključit ću jesu li potvrđene tri pretpostavke koje sam postavio u uvodnom dijelu.

3.1 Analiza rezultata veličine nivoa

Vagana prosječna vrijednost veličine nivoa je 2.89 od 10. Ovakva ocjena pokazuje da su nivoi u igrama značajno manji od prosjeka. Ovakva ocjena potvrđuje moju prvu hipotezu. U pravilu, što je specijalna mehanika važniji dio igre, to su nivoi manji. Nadalje, jasno je vidljiva korelacija između važnosti specijalne mehanike i veličine nivoa koju jasno demonstrira graf na slici 3.1.

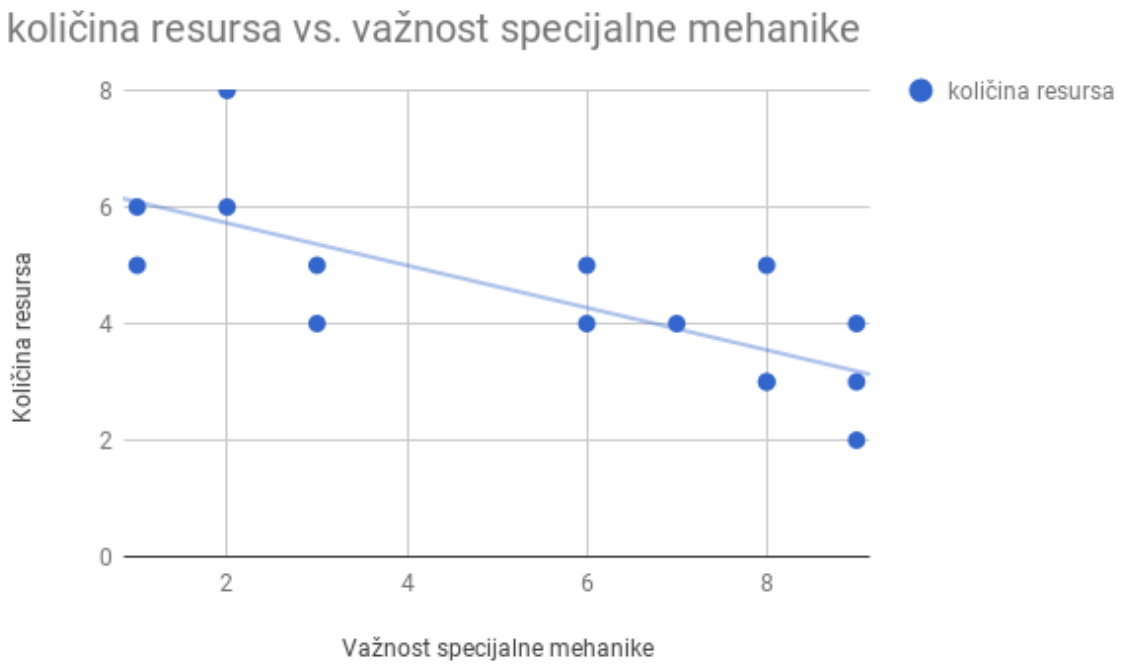


Slika 3.1 Graf opažanja ovisnosti veličine nivoa o važnosti specijalne mehanike

Pearsonov koeficijent korelacije iznosi -0.6470779448. Ova vrijednost koeficijenta upućuje na jaku negativnu korelaciju između varijabli. Ove činjenice govore da su mali nivoi jedna od definirajućih odlika igara u ovom žanru i da je veličina nivoa usko povezana s ovom mehanikom. U implementaciji prototipa igre prilagodit ću veličinu nivoa tako da bude u skladu s rezultatima istraživanja. Dizajnirat ću nivoe maksimalne veličine dotriFHD rezolucije ekrana horizontalno i vertikalno.

3.2 Analiza rezultata količine resursa

Vagana prosječna vrijednost količine resursa iznosi 3.87 od 10. Pearsonov koeficijent korelacije iznosi -0.7506752118, a nagib regresijskog pravca iznosi -0.3629666012.



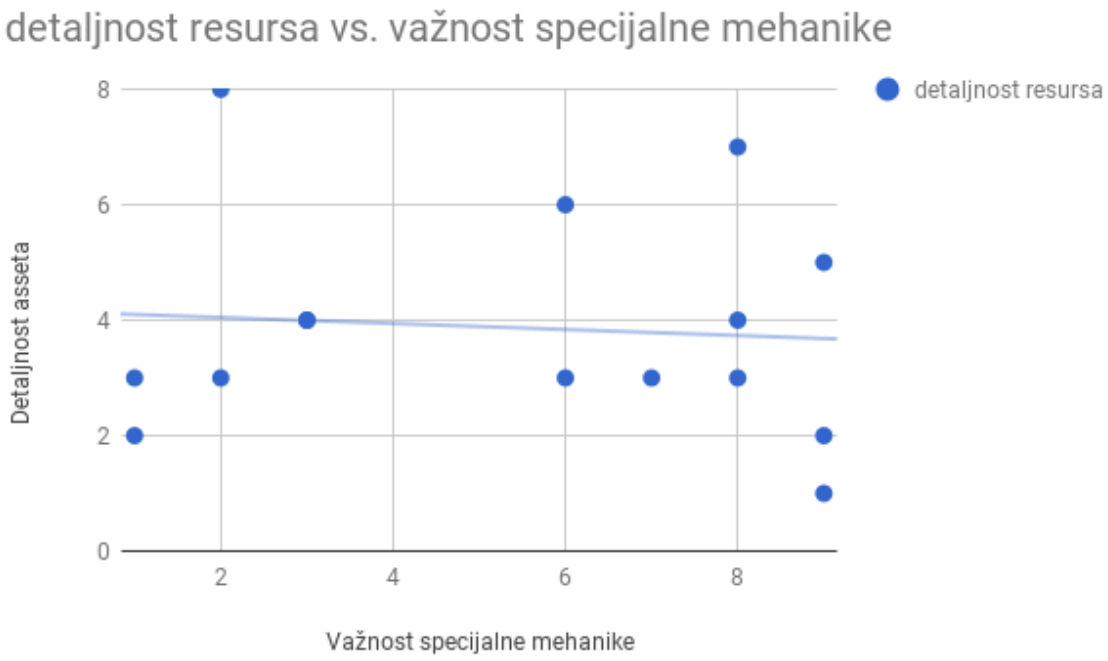
Slika 3.2 Graf opažanja ovisnosti količine resursa o važnosti specijalne mehanike

Visoka negativna vrijednost koeficijenta korelacije govori nam da postoji jaka korelacija između varijabli te da su očitanja usko grupirana oko regresijskog pravca kako je i pokazano na grafu na slici 3.2. Ali ako uzmemo u obzir i relativno nisku vrijednost nagiba regresijskog pravca, dolazimo do zaključka da vrijednosti količine resursa ne odstupaju snažno od srednje vrijednosti te ne reagiraju intenzivno na promjenu važnosti specijalne mehanike. Prosječna vagana vrijednost od 3.87 govori nam da igre koje koriste specijalne mehanike koriste nešto manje resursa od prosjeka igara na tržištu. Iako nije jako izražena, možemo reći da postoji

blaga negativna veza između važnosti specijalnih mehanika i količine resursa, ali zbog malog uzorka i slabe izraženosti ne možemo definitivno potvrditi ovu hipotezu iz uvoda. U implementaciji prototipa računalne igre upotrijebit ću raznolike resurse s ponavljanjem određenih resursa na pojedinim mjestima tako da pokušam dobiti raznolikost resursa nešto manju od prosjeka igara na tržištu.

3.3 Analiza rezultata detaljnosti resursa

Prosječna vagana vrijednost detaljnosti resursa igara u uzorku je 3.78 od 10. Pearsonov koeficijent korelacije iznosi -0.08433078011, a nagib regresijskog pravca iznosi -0.05206286837.



Slika 3.3 Graf opažanja ovisnosti detaljnosti resursa o važnosti specijalne mehanike

Zbog izrazito malog koeficijenta korelacije i jednako malog nagiba regresijskog pravca možemo reći da korelacija ne postoji i da nam sama vagana prosječna vrijednost ne govori ništa. Na grafu na slici 3.3 vidimo da opažanja nisu grupirana oko regresijskog pravca niti ga prate na neki smislen način. Zbog ovih činjenica mogu utvrditi da ranija pretpostavka vezana za detaljnost resursa nije potvrđena. Pri kreiranju resursa za prototip igre stremit ću k postizanju nekog prosjeka detaljnosti u današnjim nezavisnim igrama.

3.4 Analiza rezultata umjetničkog stila

Stiliziran umjetnički stil koristi dvanaest igara iz uzorka koje čine 76.83 % ukupne težine. Realističan umjetnički stil koriste tri igre iz uzorka koje čine 23.17 % ukupne težine.

	Težina	Broj
stiliziran	76.83 %	12
realističan	23.17 %	3

Tablica 3.1 Zbroj težina umjetničkog stila

Stilizirani umjetnički stil se pokazao kao jedna od izraženih karakteristika ovog žanra igara. Izrazito prevladava kako brojem igara u uzorku koje ga koriste, tako i zbrojem težinskih faktora tih igara, kako je vidljivo iz tablice 3.1. Iz tog razloga pri razvoju prototipa igre koji će pratiti ovaj rad, kreirat ću resurse u stiliziranom (više crtanom) stilu.

3.5 Analiza rezultata inventory mehanike

Kako je vidljivo u tablici 3.2, većina promatranih igara koristi *inventory* mehaniku. Zbroj težina igara koje koriste ovu mehaniku tek minimalno nadmašuje zbroj težina igara koje ju ne koriste. Unatoč tome, možemo reći da igre ovog žanra imaju tendenciju korištenja *inventory* mehanike.

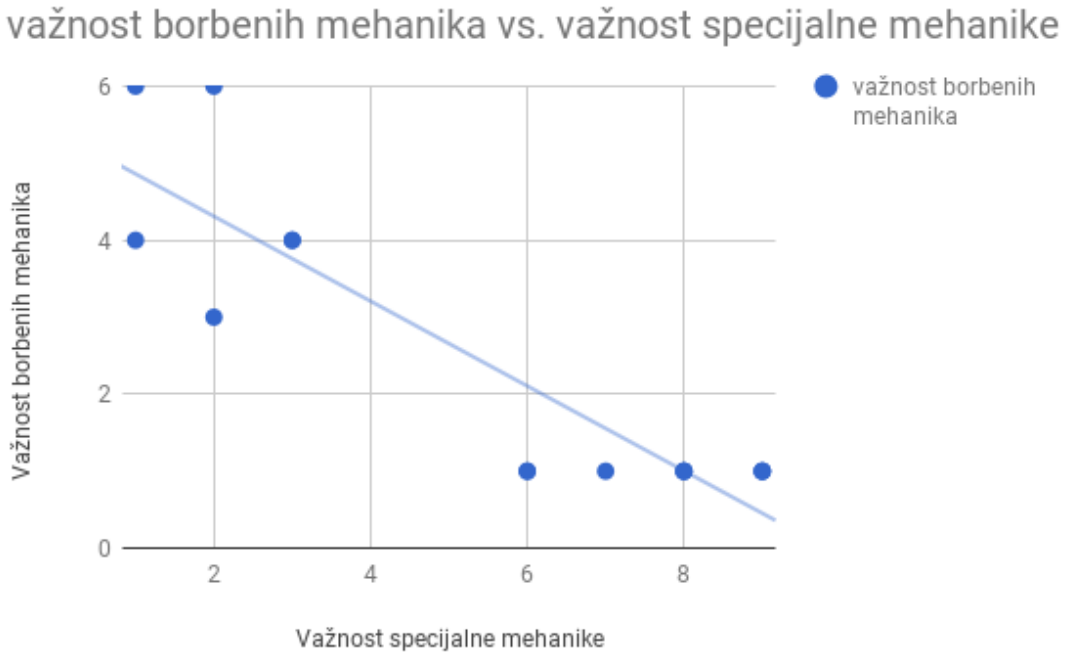
	Težina	Broj
koristi	52.44%	9
ne koristi	47.56%	6

Tablica 3.2 Zbroj težina *inventory* mehanika

Do ovog sam zaključka došao jer su i broj igara i težina na strani korištenja inventoryja. Uzevši u obzir da *inventory* nije standardna mehanika u većini igara, možemo zaključiti da u igrama koje koriste specijalne mehanike postoji tendencija korištenja iste. Zbog ovih razloga pri razvoju prototipa računalne igre implementirat ću *inventory* kao jednu od mehanika koje će igra koristiti.

3.6 Analiza rezultata važnosti borbenih mehanika

Prosječna vagana ocjena je 1.49 od 10. Pearsonov koeficijent korelacije detaljnosti resursa uz važnost specijalne mehanike u promatranom uzorku iznosi -0.8937858807, a nagib regresijskog pravca -0.5510805501.



Slika 3.4 Graf opažanja ovisnosti važnosti borbenih mehanika o važnosti specijalne mehanike

Postoji jaka negativna korelacija između intenziteta korištenja specijalnih mehanika i važnosti borbenih mehanika. Na grafu na slici 3.4 jasno vidimo kako igre koje imaju visoku važnost specijalne mehanike imaju niske vrijednosti važnosti borbenih mehanika i obrnuto. Vrlo niska vagana prosječna vrijednost kao i visok negativan nagib regresijskog pravca jasno upućuju na činjenicu da se igre koje u značajnoj mjeri koriste specijalne mehanike vrlo malo oslanjaju na borbene mehanike ili ih ne koriste uopće. U implementaciji prototipa igre neću koristiti borbene mehanike jer postoji izrazita tendencija da igre u ovom žanru koje se intenzivnije oslanjaju na specijalne mehanike većinom ne koriste borbene mehanike ili ih koriste u nekom vrlo rudimentarnom obliku.

3.7 Analiza rezultata vrsta specijalnih mehanika

U promatranom uzorku najzastupljenija vrsta specijalnih mehanika je „Promjena vremena ili dimenzije“, u deset igara koje ju koriste ukupne težinske vrijednosti 63.41 %. Navedeno

možemo vidjeti u tablici 3.3. Mehanike „Vraćanje i manipulacija vremena“ i „Ponavljanje istog perioda vremena“ zastupljene su u jednakom broju i težini s po dvije igre koje ih koriste, s 13.41 % ukupne težinske vrijednosti. „Kretanje nevidljivim dimenzijama“ je korišteno samo u jednoj igri u promatranom uzorku, s težinskim faktorom 9.76 %.

Specijalna mehanika	težina	broj
Vraćanje i manipulacija vremena	13.41%	2
Promjena vremena ili dimenzije	63.41%	10
	13.41%	2
Kretanje nevidljivim dimenzijama	9.76%	1

Tablica 3.3 Zbroj težina vrsta specijalnih mehanika

„Promjena vremena ili dimenzije“ je najčešća i najzastupljenija vrsta specijalne mehanike u ovoj vrsti igara. U promatranom uzorku dominira i brojem igara koje ju koriste i ukupnom težinom tih igara. U implementaciji prototipa koristit ću „promjenu vremena ili dimenzije“ kao specijalnu mehaniku jer je najraširenija i čak bi se moglo reći karakteristična za ovaj žanr.

3.8 Analiza rezultata perspektive

U tablici 3.4 prikazani su rezultati analize. Dvodimenzionalne i trodimenzionalne igre su po broju otprilike jednako zastupljene u promatranom uzorku, 2D sa sedam instanci, a 3D sa šest. Kombinacija 2D i 3D igranja se pojavljuje u dva slučaja.

	težina	broj
2D	57.32%	7
3D	39.02%	6
2D i 3D	3.66%	2

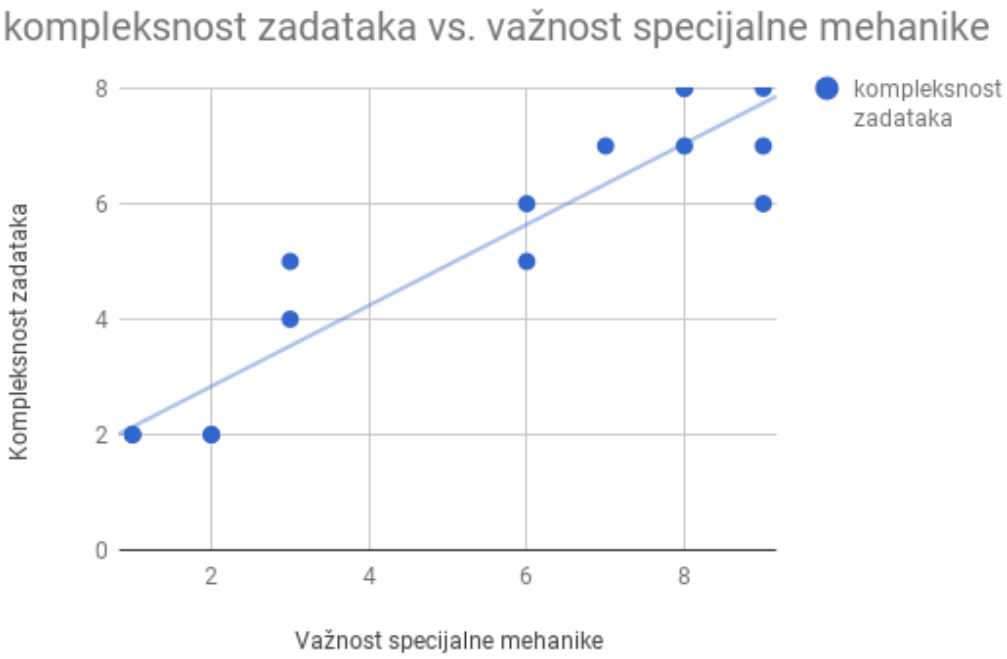
Tablica 3.4 Zbroj težina perspektiva

Ako pogledamo ukupne težinske faktore igara koje su dizajnirane u pojedinoj perspektivi, vidimo kako težine naginju prema 2D perspektivi s 57.32 % ukupne težine. Nasuprot tome, igre koje čine 39.02 % težine svih igara koriste 3D perspektivu. Obje perspektive u istoj igri

koriste igre koje čine 3.66 % težine igara. Iz ovih podataka zaključujem da postoji tendencija prema dvodimenzionalnosti kod igara koje koriste specijalne mehanike. Ova tendencija nije snažno izražena, ali je definitivno prisutna. Uzevši u obzir ovu činjenicu, prototip igre koji razvijam implementirat ću u 2D perspektivi.

3.9 Analiza rezultata kompleksnosti zadataka

Prosječna vagana ocjena je 6.43 od 10. Pearsonov koeficijent korelacije detaljnosti resursa uz važnost specijalne mehanike u promatranom uzorku iznosi 0.9309624878, a nagib regresijskog pravca 0.7008840864.



Slika 3.5 Graf opažanja ovisnosti kompleksnosti zadataka o važnosti specijalne mehanike

Ove vrijednosti ukazuju da postoji jasna pozitivna korelacija između važnosti specijalnih mehanika u pojedinoj igri i kompleksnosti zadataka koji se stavljaju pred igrača. Igre u prosjeku implementiraju vrlo kompleksne zadatke. U pravilu, što se igre više oslanjaju na specijalne mehanike, koriste sve kompliciranije logičke zadatke. Ovakav rezultat je smislen jer igre koje se jače oslanjaju na specijalne mehanike moraju imati kompliciranije zadatke jer se same specijalne mehanike u pravilu i koriste za rješavanje zadataka. Na grafu na slici 3.5 jasno je vidljivo kako vrijednosti parametara za igre u istraživanju vrlo blisko prate regresijski pravac. U implementaciji igre planiram dizajnirati zadatke visoke razine

zahtjevnosti kako bi razvijeni prototip bio u skladu s pravilima koja sam utvrdio istraživanjem.

3.10 Analiza rezultata mehanike kretanja

Platformer je najčešća mehanika kretanja koju koriste igre u promatranom uzorku. U tablici 3.5 prikazani su rezultati mehanika kretanja. Sedam igara iz uzorka koristi ovaj način kretanja s ukupnom težinom važnosti 46.34 %. Izometrijski način kretanja koriste četiri igre s ukupnom težinom 14.63 %, FPS i *point and click* načine kretanja koriste po dvije igre i oba imaju težinske faktore 19.51 %.

	težina	broj
Platformer	46.34%	7
Point and click	19.51%	2
Izometrijski	14.63%	4
FPS	19.51%	2

Tablica 3.5 Zbroj težina mehanika kretanja

U promatranom žanru vidljiva je očita preferencija prema platformerskom načinu kretanja, i broj igara koje su implementirane kao platfromeri i njihova ukupna težina su definitivno dominantni u odnosu na druge u uzorku. Iz tog razloga igru koju ću kreirati u prilogu ovog rada, dizajnirat ću kao platformer.

4 Dizajn igre i osnovne mehanike

Računalna igra je pojam koji može označavati široki spektar raznih kreacija. Što je računalna igra, možemo definirati na sljedeći način.

Igra je zatvoreni formalni sistem koji subjektivno predstavlja podskup stvarnosti. Promotrimo svaki od termina podrobnije. Pod zatvoreni podrazumijevamo da je igra potpuna i samodostatna konstrukcija. Formalnost označava da igra ima eksplicitna pravila. Riječ sistem govori da se sastoji od skupa povezanih dijelova. Subjektivna stvarnost znači da igre ne rekreiraju objektivno stvarne događaje, ali su igraču ipak subjektivno stvarni (Crawford, 1997).

Prototip igre sam implementirao korištenjem Unity programskog okvira. Glavne značajke izvedbe igre odredio sam u istraživanju koje sam opisao u prethodnom poglavlju. Ilustracije, animacije, glazbu i programiranje sam samostalno izradio kako bih prošao sve korake u izradi računalnih igara. U sljedećim poglavljima dokumentirat ću i objasniti kako sam izveo pojedine dijelove igre. Igru sam izradio u skladu sa zaključcima istraživanja. Stoga je rezultat 2D platformer s malim nivoima. U izradi resursa sam koristio stilizirani umjetnički stil s umjerenom količinom detalja. Ravnajući se prema rezultatima istraživanja, nisam implementirao borbene mehanike. Implementirao sam *inventory* mehaniku. Kao specijalnu mehaniku koristio sam mehaniku „Promjena vremena ili dimenzije“ sa zadatcima visoke složenosti jer je istraživanje pokazalo da su to norme u ovom žanru.

4.1 Kratko o Unityju

Unity je platforma za razvijanje 2D, 3D, virtualna stvarnost (engl. *virtual reality*, skraćeno VR) i proširena stvarnost (engl. *augmented reality*, skraćeno AR) igara i aplikacija. Na popisu kompanija koje koriste Unity nalaze se, između ostalih, i Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square Enix, Ubisoft, Obsidian, Insomniac i Warner Bros (unity3d.com, 2017).

Rad u Unityju kombinira radnje u Unity pregledniku (engl. *editor*) i pisanja skripti pisanih u C# ili javascript programskom jeziku. U grafičkom pregledniku objektima se dodaju elementi Unity programskog okvira: objekti kao što su okviri za detekciju sudara (engl. *collider),* objekti koji reproduciraju zvuk (engl. *audio source*) ili koji dodaju fizikalna

svojstva kao gravitaciju, inerciju (rigid body) i slično. U skriptama se definira ponašanje objekata korištenjem ranije definiranih metoda koje se izvršavaju u zadanom redoslijedu i prema određenim pravilima. Većina se razvoja može riješiti implementacijom Unity skripti, ali korištenje editora uvelike olakšava razvoj omogućavajući bolji pregled razvoja, nadzor nad objektima te lakše nalaženje grešaka (engl. *debugging*) i brže izvođenje promjena.

4.2 Opis igre

Razvijeni prototip igre je 2D puzzle platformer sa specijalnim mehanikama. Priča koja se provlači kroz igru uključuje igrača koji je lik u knjizi u misiji spašavanja princeze. U svojoj misiji igrač mora naći put kroz platformerski nivo skupljajući objekte (engl. *pick up*) koje pohranjuje u *inventoryju*. Objekti iz *inventoryja* mu omogućavaju daljnje napredovanje (npr. otvaraju neka vrata ili omogućavaju prelazak neke prepreke). Igra se odvija kao priča u knjizi u kojoj je igrač lik, on može prelaziti s jedne stranice na drugu i tako putovati unazad u vremenu ako ide na prethodnu stranicu i unaprijed ako ide na sljedeću. To je osnova specijalne mehanike koju sam primijenio. Kad igrač putuje kroz vrijeme ako promjeni ključnu stvar na prethodnoj stranici, utječe na događaje na sljedećoj stranici. Igrač se može u svakom trenutku vratiti na stranicu koju je već posjetio i ponovo promijeniti ključni događaj tako vraćajući sljedeću stranicu u prethodno stanje. Na taj način se razvija stablo mogućih budućnosti po kojem se igrač može kretati. Na primjeru od tri stranice možemo vidjeti kako jedna promjena na prvoj stranici generira dvije alternativne budućnosti (naprimjer ako na prvoj stranici posadimo sjemenku, na sljedećoj će izrasti drvo, a ako sjemenku izvadimo iz zemlje, drva neće biti). Ako ovaj princip s jednim ključnim događajem primijenimo i na te dvije opcije druge stranice, dobivamo četiri opcije za treću stranicu. Na taj način iz jednog nivoa dobivamo sedam različitih varijanti po kojima se igrač može kretati i biti u interakciji. Ostale mehanike također se koncentriraju na stvaranje zagonetki, a njihovo rješavanje igraču omogućava napredak u igri. Igrač može vući razne objekte da ih postavi na poziciju gdje može dosegnuti neku platformu. Od određenih površina može se odbiti i odskočiti. Može pokupiti razne objekte koje pohranjuje u *inventoryju*. Te objekte kasnije može koristiti kako bi napredovao, naprimjer uklonio neke prepreke ili preskočio neki jaz koji inače ne bi mogao. Igra također koristi i mehaniku interakcije igrača s objektima u nivou tako da se isti objekt može staviti u različita stanja, naprimjer otvoren ili zatvoren otvor na podu. Ako je otvoren, tada igrač može skočiti kroz njega na niži kat, a ako je zatvoren, igrač može prijeći preko na ostatak kata do kojeg ne bi mogao doći da je poklopac

otvoren. Sve mehanike djeluju zajedno u kreiranju zadataka i zagonetki koje igrač mora obaviti i riješiti. Nelinearnim putovanjem kroz vrijeme i istraživanjem alternativnih budućnosti mora spasiti princezu i završiti igru.

4.3 Dizajn svijeta

Dizajn svijeta važna je odrednica svake računalne igre. Adekvatnim dizajnom okoline uvlačimo igrača u svijet igre, a lošim dizajnom dobivamo suprotan efekt. Kako bi okolinu nivoa učinio što interaktivnijom, implementirao sam dvije često korištene metode u 2D igrama: paralaksu i otkrivanje unutrašnjosti zgrada tek kad igrač u njih uđe.

4.3.1 Paralaksa

Paralaksa je efekt koji pojačava uvjerljivost kretanja i dojam prostora tako što se u odnosu na kameru objekti koji su dalje od fokusa kamere u pozadini kreću sve sporijom brzinom u smjeru kretanja kamere. Linije perspektive sve su manje udaljene jedna od druge što su bliže točki nedogleda perspektive. Fokus kamere je na igraču i kamera se najčešće kreće brzinom igrača. Ovaj efekt podrazumijeva da se objekti koji se nalaze između igrača i kamere (engl. *foreground*) moraju kretati u suprotnom smjeru od kretanja igrača jer kamera ne prati liniju dna ekrana nego liniju po kojoj hoda igrač.

Pozadinu sam podijelio u četiri sloja različitih udaljenosti od kamere i dodao sam jedan sloj *foregrounda*. Sve Unity objekte koji čine pojedini sloj dodao sam u krovni objekt tog sloja kojemu sam pridružio skriptu koja ga pomiče u odnosu na kameru.

```
void Update () {
GetComponent<Transform>().position =
new Vector3(
transform.position.x +
(Kamera.XIznosPomakaKamere*postotakPomaka),
transform.position.y +
(Kamera.YIznosPomakaKamere*postotakPomaka),
transform.position.z);}
```

Kod 4.1 `Update` funkcija paralaxSkripte

Iz koda 4.1 je vidljivo da svaki objekt s paralax skriptom ima referencu na kameru, tj. na kamera skriptu na glavnoj kameri. Svi objekti slojeva pozadina prate pomak kamere koji se

računa u kamera skripti, a intenzitet pomaka se može pojedinačno korigirati postavljanjem javne varijable `postotakPomaka` u Unity editoru na objektu svakog sloja pozadine i tako prilagođavati slojeve da se dobije željena brzina i smjer kretanja slojeva pozadine i prvog plana.

4.3.2 Nestajanje prednje strane zgrada

Ovaj efekt dodaje volumen i uvjerljivost 2D zgradama tako što otkriva unutrašnjost pojedine zgrade tek kada igrač u nju uđe i ponovno ju sakrije kad igrač iz nje izađe. Na ovaj način se odvaja unutrašnji prostor od vanjskog iako se oba nalaze na istoj 2D površini. Ovaj efekt se postiže kombinacijom načina konstrukcije Unity objekata koji predstavljaju zgrade i modificiranjem sprite renderera, točnije alpha kanala njegovog `color` svojstva (engl. *property*).

Efekt je implementiran na način prikazan u kodu 4.2. U `OnTriggerEnter2D` funkciji KatNestajanjePrvogZidaSkripte povjerava je li igrač u koliziji s objektom koji bi trebao nestati postavljanjem varijable `vanjskiZidVidliv`.

```
void OnTriggerEnter2D(Collider2D coll)
{
if(coll.gameObject.tag == "Igrac"){
vanjskiZidVidliv = false;}
}
```

Kod 4.2 `OnTriggerEnter2D` funkcija katNestajanjePrvogZidaSkripte

Nakon što je `vanjskiZidVidliv` postavljen, u funkciji, koja je prikazana u kodu 4.3, postupno se mijenja alpha kanal `color` svojstva sprite renderera objekta koji predstavlja vanjski zid građevine dok ne dođe na nulu. Kad je igrač izvan collidera objekta, `vanjskiZidVidliv` je postavljen na `true`, a vidljivost objekta se ponovno povećava dok ne dođe do 1 i ponovno sakrije unutrašnjost.

```
void Update() {
if (vanjskiZidVidliv){
if (vidljivostVanjskogZida < 1) {
foreach (GameObject vanjskiZid in vanjskiElementi) {
vanjskiZid.GetComponent<SpriteRenderer>().color =
new Color(1, 1, 1, vidljivostVanjskogZida);
vidljivostVanjskogZida += 0.04f;}
}...
```

Kod 4.3 Dio `Update` funkcije u katNestajanjePrvogZidaSkripti

4.4 Mehanika kretanja

Mehanika kretanja je možda i najvažnija mehanika u svakoj igri. To je definitivno mehanika koju igrač najviše koristi. U prototip igre sam implementirao 2D mehaniku kretanja koja je većini igrača najpoznatija i najbliža, takozvanu Super Mario mehaniku. Osnova ove vrste kretanja je da se prilikom skoka igračem može upravljati i u zraku. To je veliki odmak od realnosti i fizike, ali je u platformerima vrlo čest. Najveća prednost ovog načina upravljanja je omogućavanje lakšeg kretanja i preciznijih skokova tako da igrači ne moraju ponavljati skokove i ponovno prelaziti platforme.

Uz kretanje igrača najuže je vezano i kretanje kamere kao i stroj stanja koji upravlja animacijama igrača pa ću u sljedećem dijelu pojasniti kako sam i njih izveo.

4.4.1 Kamera

Ponašanje kamere shvatio sam kao dio igre o kojem je najbolje razmišljati ergonomski. Što se manje primjećuje gibanje kamere, to je bolje jer se igrač manje umara.

Implementaciju pomicanja kamere napravio sam u `lateUpdate` funkciji prikazanoj u kodu 4.4 u skripti koja je pridružena kameri. `LateUpdate` je korišten jer za ispravno pomicanje kamere pozicija igrača i ostalih objekata mora biti prethodno ažurirana. Radi tečnijeg kretanja kamere odvojio sam pozicije na kojima se kreće igrač i na kojima se kreće kamera. U središnjem djelu ekrana se kreće igrač, a kamera stoji. Kad igrač dođe do određene granice na ekranu, kamera se počinje micati zajedno s igračem i strogo ga prati. Također sam odvojio brzo kretanje kamere od polaganog. Ako se igrač teleportira po osi x, daleko izvan središta

kamere, kamera će velikom brzinom krenuti prema njemu dok ne dođe na poziciju u kojoj prati igrača i dalje ga nastaviti normalno pratiti. Isti je princip i s osi y uz jednu razliku. Zbog platformerske naravi igre kako bi se kod skokova izbjegli nagli pokreti, kamera je implementirana tako da kad igrač prijeđe određenu prvu vertikalnu granicu, kamera mu se krene približavati tek kad doskoči na platformu. Nadalje, ako prijeđe i drugu širu vertikalnu granicu, kao naprimjer kod pada s visine, kamera ga počinje strogo pratiti kako ne bi izišao s ekrana.

```
void LateUpdate(){
provjeriKojiObjektSePrati();
zabiljeziPrethodeVrijednostiXiYPozicija();
if(Igrac.GetComponent<igracSkripta>().IgracAktivan
|| ObjektKojiSePrati != Igrac) {
provjeriUdaljenostOdIgracaPoXOsi();
pomakniKameruAkoSeIgracPrebacioDaleko();
pomakniKameruAkoIgracHoda();
pomakniKameruDoEkstremaIgracaAkoJeIgracStaoNaMjestoZaTo();
provjeriVertikalnuBrzinu();
PomicanjeKamerePoYOsi();
pozicijaKamere = transform.position.y;
pozicijaIgracaPlus5 = ObjektKojiSePrati.transform.position.y
+ dolnjaTockaPomakaKamerePremaGore;
}
izracunajPomakKamere();}
```

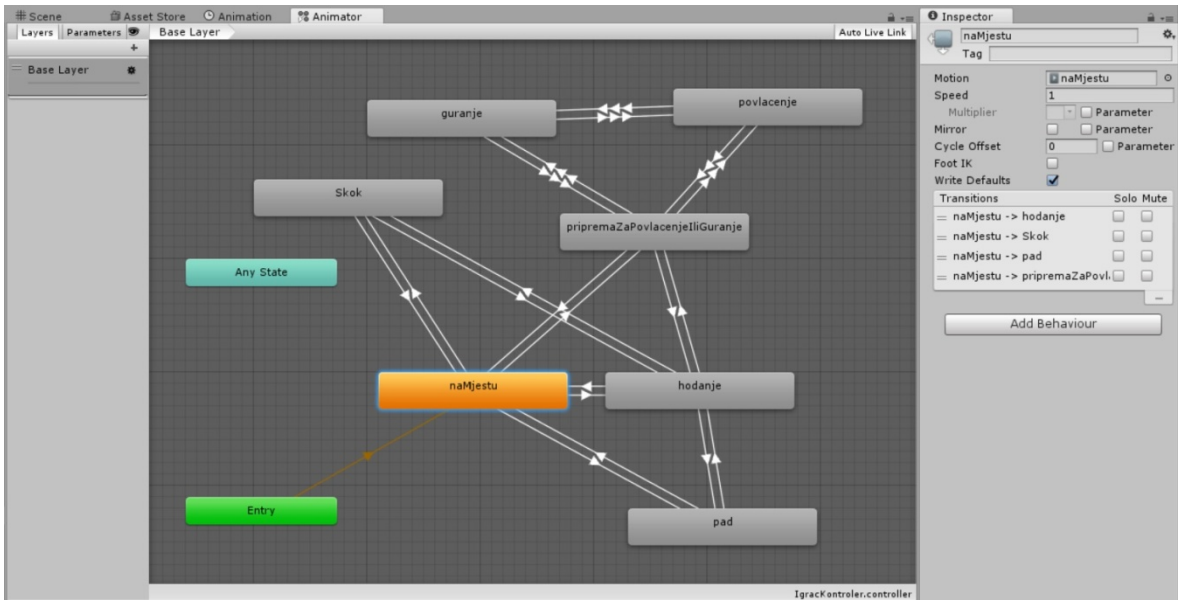
Kod 4.4 `LateUpdate` funkcija kameraSkripte

Ponekad je potrebno igraču skrenuti pozornost na neku lokaciju koja je udaljena od igrača i pozicije kamere. Zato je kamera postavljena tako da ne prati direktno igrača, nego prati objekt koji joj je postavljen za praćenje. Većinu vremena je to igrač osim kad je potrebno skrenuti pozornost na nešto. U tom trenutku se promijeni varijabla `ObjektKojiSePrati` u kameraSkripti i kamera se ponaša kao da je taj objekt igrač što znači da će u režimu brzog kretanja doći do njega. Ovakva je mehanika vrlo elegantna jer se dobiva *pan* efekt i sprječava skok kamere s jednog kadra na drugi koji bi se dobio da se samo promijeni x i y pozicija kamere.

4.4.2 Animator i controller

Unity koristi stroj stanja kako bi odredio koju animaciju prikazati u kojem trenutku za pojedini objekt. Animacijama upravlja animator objekt kojem možemo pristupiti preko grafičkog sučelja (sučelje na slici 4.1). Animator sadrži kontroler komponentu koja djeluje kao stroj stanja. Kontroler se sastoji od pojedinačnih stanja u kojima se objekt može nalaziti. Svakom stanju je pridružena animacija i tranzicija između stanja. Unutar tranzicija su definirani uvjeti koji određuju prijelaz iz jednog stanja u drugo. Na primjer, ako igrač krene hodati udesno, njegova brzina se promijeni s nule na neku negativnu vrijednost. Ako je horizontalna brzina uvjet za promjenu stanja iz stajanja u hodanje, njezinom promjenom s nule na neku negativnu vrijednost, promijenit će se i stanje, a s njim i animacija.

Kod implementacije prototipa videoigre složene animacije sam koristio samo na objektu igrača. Kontoler na animatoru igrača ima sedam stanja: na mjestu, hod, skok, pad, priprema za povlačenje ili guranje, povlačenje i guranje. Tranzicije su određene horizontalnom i vertikalnom brzinom, blizinom određenih objekata (naprimjer kola za guranje) i određenim komandama igrača (npr. pritisak gumba da igrač uhvati kola). Odvojio sam animacije za skok i pad. U oba se stanja može ući i iz stanja mirovanja i iz stanja hoda, ali su uvjeti drukčiji. U stanje guranja i vuče objekata se može ući samo iz stanja priprema za povlačenje ili guranje te se iz stanja guranja može ući u stanje vuče i obrnuto. Igrač može i vući i gurati s obje strane objekt i to u oba smjera x osi što komplicira implementaciju. Zato ova stanja imaju po dva prijelaza između sebe i drugih stanja.



Slika 4.1 Pregled stanja i prijelaza igrača u editoru animator komponente

4.4.3 Kretanje

Unity nudi mnogobrojne opcije kako razviti kretanje igrača. Kod dodavanja fizike u dvije dimenzije najjednostavnija opcija je korištenje rigidbody2d komponente. Ova komponenta objedinjuje dio fizikalnih svojstava koja se tiču gravitacije, rotacije, inercije, brzine i svojstava materijala. Rigidbody2d s pridruženim *colliderima* (objektima koji opisuju koji prostor igrač zauzima) opisuje fizičko prisustvo igrača u svijetu igre.

Kod tehničkog izvođenja kretanja igrača zamrznuo sam rotaciju po z osi kako bi se igrač predvidljivo kretao bez prevrtanja. Igraču su pridružena četiri *collidera*. Dva *collidera* određuju geometriju igrača. Jedan *box collider* za gornji dio igrača i jedan *circle collider* za donji dio. Druga dva *collidera* su *triggeri* što znači da prolaze kroz druge *collidere* i samo bilježe je li se kolizija dogodila ili traje. Jedan *trigger* služi za bilježenje blizine igrača nekom objektu koji se može vući. Ako je u koliziji s objektom koji se može vući, igrač je u stanju da ga povuče. Drugi *trigger* se koristi za kontrolu udarca glavom koja služi da se u slučaju udarca igrača glavom u platformu kontrolira vertikalna brzina igrača kako bi promjena brzine kod udarca izgledala prirodnije. `OdTriggerEnter2D` funkcija u skripti igrača prikazana u kodu 4.5 određuje s koje strane objekta koji se gura se igrač nalazi kad je kraj objekta (u poziciji da gura ili vuče) kako bi se aktivirala pravilna animacija.

```
void OnTriggerEnter2D(Collider2D coll) {
if (coll.gameObject.layer == 8) {
animator.SetBool("nalaziSeSaDesneStraneObjekta", true);
nalaziSeSaDesneStraneObjekta = true;}...
```

Kod 4.5 Funkcija `OnTriggerEnter2D` u igracSkripti

Sve animacije su izvedene s igračem okrenutim udesno, a pri okretanju u lijevo `localScale` igrača se mijenja kako je pokazano u kodu 4.6 tako da se promijeni predznak x komponente.

```
void OkreniIgraca() {
okrenutNaDesno = !okrenutNaDesno;
Vector3 orjentacijaIgraca = transform.localScale;
orjentacijaIgraca.x = orjentacijaIgraca.x * -1;
transform.localScale = orjentacijaIgraca;}
```

Kod 4.6 Funkcija `OkreniIgraca` u igracSkripti

4.4.4 Skok

Kod gibanja kad igrač nije na zemlji, važno je razlikovati skok od pada. Animacije se razlikuju kad igrač skače i kad pada. Kod skoka u funkciji `Skoci` postavljaju se varijable animatora kako bi se u skoku prikazala ispravna animacija. Skok sam izveo tako da sam postavio vertikalnu brzinu na određenu vrijednost u trenutku skoka. Izvedba je vidljiva u kodu 4.7. Dodavanje brzine svakom skoku daje jednaku visinu skoka dok kod dodavanja sile, rezultati nisu uvijek ujednačeni.

```
void Skoci() {
uSkoku = true;
animator.SetBool("naZemlji", false);
animator.SetBool("pad", false);
animator.SetBool("skok", true);
rigidBody2DIgraca.velocity = new
Vector2(rigidBody2DIgraca.velocity.x, jacinaSkoka);}
```

Kod 4.7 Funkcija `Skoci` u igracSkripti

5 Napredne mehanike

U ovom poglavlju ću pojasniti mehanike koje čine osnovicu igre. Pomoću ovih mehanika igrač vrši interakciju s igrom. One su oruđe pomoću kojeg su izrađene logičke zagonetke koje igrač rješava kako bi napredovao kroz igru.

Najelegantnije mehanike su one koje su toliko posebne da otvaraju potpuno nove načine igranja. Nije dovoljno kreirati varijacije na postojeće vrste interakcija. Suprotno tome, potrebno je istraživati mehanike koje uvode nove strategije i načine istraživanja koji nisu ranije postojali (Sylvester, 2013).

Unity ima puno pozitivnih karakteristika koje ubrzavaju razvoj. Neke od njih su korištenje karakteristika programskog okvira za implementaciju fizike ili korištenje grafičkog sučelja pri razvoju. Nasuprot tome, pri razvoju kompleksnih scena i mehanika pokazuju se i neki nedostatci.

Kod kompleksnih scena lako je izgubiti orijentaciju, teško održati predodžbe koje objekti imaju pridružene kao specifične komponente. Unity u sebi sadrži funkcionalnost za pronalazak pridruženih skripti, ali bi ta funkcionalnost mogla biti robusnija jer se i uz ovu funkcionalnost javljaju situacije u kojima je potrebno provjeriti sve unutar scene kako bi se pronašle poveznice među skriptama (Hocking, 2015).

5.1 Inventory mehanika

Inventory sam razvio na način da se prikazuju objekti u četiri prozorčića na korisničkom sučelju. U jednom trenutku se može vidjeti maksimalno četiri objekta u *inventoryju*. Inputom korisnika cijeli *inventory* prolazi preko ta četiri prozorčića i pokazuju se četiri po četiri *inventory* objekta. Objekt koji na sebi ima BazaItemaSkriptu u sebi sadrži sve *pick up* objekte koji se pojavljuju u igri. Kad igrač skupi određeni objekt, odgovarajući objekt se doda u *inventory* iz BazaItemaSkripte. Odabir određenog objekta u *inventoryju* vrši se pomicanjem *inventory* kursora na odgovarajući objekt u sučelju i pritiskom na gumb za korištenje *inventoryja*.

5.1.1 Pick up objekti

Objekti koje igrač može pokupiti raspoređeni su po nivoima. Igrač mora naći način kako doći do njih i pokupiti ih. Nakon što ih pokupi, odgovarajući se objekt pojavljuje u njegovom *inventoryju* i može ga koristiti. U kodu 5.1 je prikazano kako se kod kolizije aktivira dodavanje u *inventory*. *Pick up* objekt predstavlja prikaz *inventory* objekta u nivou i ujedno označava mjesto do kojeg igrač mora doći kako bi dobio odgovarajući *inventory* objekt. Igrač mora doći u koliziju s *pick up* objektom kako bi dobio pripadajući objekt u *inventoryju* nakon čega *pick up* objekt postaje neaktivan. Svaki *pick up* objekt ima `audiosource` koji se oglašava nakon što igrač pokupi objekt. `Audiosource` se nalazi na odvojenom *game* objektu kako bi se zvuk čuo nakon što je *pick up* objekt deaktiviran.

```
void OnTriggerEnter2D(Collider2D coll) {
if(coll.gameObject.tag == "Igrac"){
inventorij.dodajUInventorij(bazaItema.GetComponent<BazaItemaS
kripta>().SviItemi[indexItemaUBaziKojiSeDodajeUinventorij]);
audioSource.Play();
gameObject.SetActive(false);
}}
```

Kod 5.1 Funkcija `OnTriggerEnter2D` u pickUpSkripti

5.1.2 Inventory i inventory objekti

Inventory igraču prikazuje samo četiri *inventory* objekta u bilo kojem trenutku. Kod 5.2 prikazuje kako se upravlja prikazom *inventoryja*. Kod svake promjene u *inventoryju*, potrebno je ponovno prikazati objekte u izborniku *inventoryja*. Igrač kada kursorom dođe do kraja izbornika, više ne pomiče kursor nego objekte u *inventoryju*, a kada iskoristi neki potrošni objekt, on nestane iz prikaza i njegovo mjesto zauzme objekt koji je bio iza njega.

U *inventory* skripti nalazi se lista *inventory* objekata (itemiUInventoriju u kodu) koja predstavlja objekte koje igrač sa sobom nosi. Odabir objekata koji se prikazuju u četiri mjesta za prikaz određen je varijablom `indexPrviPokazaniItem`. Odabrani objekt je osim njom određen i varijablom `indexOdabranogItema`. `IndexPrviPokazaniItem` određuje koji indeks u listi *inventory* objekata se prikazuje na prvom mjestu prikaza dok

`indexOdabranogItema` određuje koji je prozorčić od četiri (lista `slotovi` u kodu) odabran na prikazu. Zbrojem tih dviju vrijednosti dobivamo indeks odabranog objekta u *inventoryju*.

```
private void pokaziPrikazaneItemeIzbornika() {
for (int i = 0; i < 4; i++){
if (itemiUInventoriju.Count > 4
&& itemiUInventoriju.Count - indexPrviPokazaniItem < 4){
indexPrviPokazaniItem--;
}
if (itemiUInventoriju.Count > indexPrviPokazaniItem + i) {
Slotovi[i].transform.GetChild(0).GetComponent<SpriteRenderer>
().sprite =
itemiUInventoriju[indexPrviPokazaniItem + i].sprite;
}else {
Slotovi[i].transform.GetChild(0).GetComponent<SpriteRenderer>
().sprite = null;}}
}
```

Kod 5.2 Funkcija `PokaziPrikazaneItemeIzbornika` u inventorySkripti

Korisničkom komandom *inventory* objekt se može koristiti na određenim mjestima u nivou. Korištenje je moguće ako je jedan od igračevih *trigger collidera* u koliziji s objektom koji na sebi nosi AktiviranjeObjekataPomocuItemaSkripta skriptu. Kod 5.3 prikazuje kako objekti s kojima je igrač u interakciji u AktiviranjeObjekataPomocuItemaSkripti nose liste objekata koje se aktiviraju i deaktiviraju (aktivacijom jedne serije objekata i deaktivacijom druge, naprimjer objekt prozora se zamijeni objektom razbijenog prozora ako igrač na njemu iskoristi objekt čekić iz *inventoryja*). U tom slučaju *inventory* objekt nestane ako je namijenjen za jednokratnu uporabu ili ostane u inventoryju ako je namijenjen da se koristi više puta.

```
if (Igrac.IgracAktivan && mozeSeKoristiti &&
brojacDoSlijedecegKoristenja >=
vrijemeDoSlijedecegKoristena){
if (Inventory.indexOdabranogItema +
Inventory.indexPrviPokazaniItem
<Inventory.itemiUInventoriju.Count
&& Inventory.itemiUInventoriju[Inventory.indexOdabranogItema
+ Inventory.indexPrviPokazaniItem].id ==
idItemaKojiAktiviraDrugiObjekt
&& Input.GetButtonDown("KoristiItem")){
pokreniAnimacijuIZvuk();
foreach (GameObject d in DrugiGameObjekti) {
d.SetActive(true);}
foreach (GameObject p in PrviGameObjekti) {
p.SetActive(false);}
```

Kod 5.3 Dio `Update` funkcije skripte aktiviranjeObjekataPomocuItemaSkripta

5.2 Mehanike interakcije

Mehanike interakcije su posebno važne za igre koje ne koriste borbene mehanike. To su mehanike koje određuju kako igrač djeluje na svijet igre oko sebe i obrnuto. Mehaniku djelovanja na predmete u svijetu igre pomoću objekata u *inventoryju* obradio sam u prethodnom dijelu pa tu mehaniku neću ponovo objašnjavati. Mehanike koje ću objašnjavati su: povlačenje objekata, penjanje po ljestvama i konopcima, odskakanje od elastičnih površina i korištenje objekata koji se aktiviraju bez *inventory* objekta.

5.2.1 Objekti koji se vuku

U igri igrač može vući razne objekte, kola kutije i slično, kako bi ih dovukao u poziciju u kojoj ih može iskoristiti, bilo da s njih dosegne neku platformu ili nešto drugo. Svaki objekt koji se može vući na sebi ima dva *collidera* koji označavaju mjesta na kojima igrač može "primiti" objekt. Ako je igračev *trigger collider* u koliziji s jednim od ta dva *collidera,* igrač može dati komandu da krene vući (ili gurati) taj objekt. Razlog zašto su dva *collidera* na objektu je činjenica da igrač može prići objektu s obje strane i na svakoj strani ga uhvatiti i gurati ili vući što aktivira različite animacije.

Svaki objekt koji se može vući na sebi ima guranje ObjektaSkripta skriptu koja kod kolizije s igračem igracSkripti signalizira koji objekt se vuče. Guranje ili vuča objekta riješena je u igracSkripti implementacijom funkcije guraj objekt (Kod 5.4).

```
private void gurajObjekt() {
rigidBodyObjektaKojiSeVuce.velocity = new Vector2(
rigidBody2DIgraca.velocity.x,
rigidBodyObjektaKojiSeVuce.velocity.y);
}
```

Kod 5.4 `GurajObjekt` funkcija u igracSkripti

5.2.2 Ljestve i direktni prelazak s platforme na platformu

Na određenim mjestima u igri igrač mora prijeći s jedne platforme na drugu i kad je udaljenost između njih veća nego ona koju igrač može dosegnuti skokom. Za takav slučaj predvidio sam objekte koji ga mogu prebaciti: ljestve, konope i slično. Izvedba tih objekata se temelji na principu da se igračeva pozicija promijeni na unaprijed zadanu lokaciju. Kod 5.5 je dio koda na ljestveSkripti koji kontrolira je li prebacivanje moguće i zove funkciju igrača koja ga prebacuje.

```
if (premaGore&&LjestveSeMoguKoristiti
&&Igrac.GetComponent<igracSkripta>().IgracAktivan
&&Igrac.GetComponent<igracSkripta>().naZemljiZaSkok
&& Input.GetButtonDown("Koristi")) {
kamera.TelePortTimer =
0;Igrac.GetComponent<igracSkripta>().promjeniPoziciju(GornjaT
ocka.transform.position);
LjestveSeMoguKoristiti = false;
}
```

Kod 5.5 Dio `Update` funkcije u ljestveSkripti

Kod 5.6. je igračeva funkcija koja upravlja prebacivanjem pozicije. Kako bi ova promjena izgledala elegantnije, izvedena je na sljedeći način. Igrač se ne može kretati dok traje

“teleportacija” i postepeno nestaje tako da alpha kanal smanjuje na nulu. Kad je u potpunosti transparentan, promijeni mu se lokacija i alpha kanal se povećava do 1.

Nakon završetka ovog postupka igrač se može ponovo kretati. Smjer kretanja je određen s obzirom na stranu na kojoj se igrač nalazi u odnosu na vertikalnu sredinu objekta ljestava. Isti princip je implementiran za objekte koji igrača teleportiraju horizontalno i one koji to rade vertikalno. Kako se naglo dogodi promjena pozicije igrača, pomak kamere je odgođen dok igrač nije potpuno vidljiv i dok mu se ponovno ne omogući kretanje.

```
public void promjeniPoziciju(Vector3 pozicija) {
novaPozicija = pozicija;
Nestajanje = true;
}
```

Kod 5.6 `PromijeniPoziciju` funkcija u igracSkripti

5.2.3 Elastični odskok

U određenim situacijama rješavanje zadataka zahtijeva da se igrač od neke površine odbije kao od trampolina. U toj situaciji, što igrač ima veću vertikalnu brzinu u jednom smjeru, kod odbijanja će imati veću brzinu u suprotnom. Uz ovaj efekt aktivira se i zvuk koji igraču daje do znanja što se događa. Igrač ima *trigger collider* na donjoj strani objekta koji testira koliziju s podlogom. Kad elastični objekt detektira ovaj *collider* igraču promijeni vertikalnu brzinu i pokrene zvuk odbijanja s glasnoćom zvuka proporcionalnom jačini udarca (vertikalnoj brzini) kako je demonstrirano u kodu 5.7.

```
void OnTriggerEnter2D(Collider2D coll){
if (coll.gameObject.tag == "IgracTestDno"){
UlaznaBrzina =
Igrac.GetComponent<igracSkripta>().MaksimalnarzinaPadanja;
if (Mathf.Abs(UlaznaBrzina) >
Igrac.GetComponent<igracSkripta>().jacinaSkoka) {
audioPlayer.volume = 1;
} else {
audioPlayer.volume = Mathf.Abs(UlaznaBrzina)/
Igrac.GetComponent<igracSkripta>().jacinaSkoka;}
audioPlayer.Play();
Igrac.GetComponent<Rigidbody2D>().velocity =
new Vector2(Igrac.GetComponent<Rigidbody2D>().velocity.x, -1f
*
(UlaznaBrzina + 2f));}}
```

Kod 5.7 `OnTriggerEnter2D` funkcija u bounceSkripti

5.2.4 Interakcija s posebnim objektima

Igrač može djelovati na neke objekte i bez korištenja *inventory* objekta. Ove objekte igrač može staviti u jedno od dva stanja i vratiti nazad u prvotno stanje, npr. otvorena ili zatvorena vrata. Ovaj mehanizam također koristi i specijalna mehanika kod promjene uvjeta za putovanje na kroz vrijeme, naprimjer ako neki objekt zapalimo u prošlosti u sadašnjosti će izgorjeti, a ako se vratimo u prošlost i vatru ugasimo u sadašnjosti, objekt će biti neoštećen. Ovaj postupak možemo ponavljati i putovati u bilo koju verziju budućnosti koliko god puta nam je to potrebno.

Ova mehanika funkcionira na istom principu kao i mehanika koja koristi inventory objekte. Također koristi listu *game* objekata koje aktivira i listu objekata koje deaktivira npr. deaktivira objekt zatvorena vrata i aktivira objekt otvorena vrata. Igrač tako stekne dojam da je otvorio vrata, ovaj mehanizam je prikazan u kodu 5.8. Ako promjena mijenja objekt nivoa (engl. *level item*), tada utječe i na specijalnu mehaniku, dakle na događaje u budućnosti, a ako ne utječe, tada samo mijenja stanje u trenutnom vremenu.

```
if (Igrac.IgracAktivan && mozeSeKoristiti &&
Input.GetButtonDown("Koristi") &&
brojacDoSlijedecegKoristenja >=
vrijemeDoSlijedecegKoristena){
if (mjenjaLevelItem){
LevelManager.LevelItemManager.AktivniLevelItem.SmjerPremaGor
=
!LevelManager.LevelItemManager.AktivniLevelItem.SmjerPremaGor
e;
pokreniAnimacijuIZvuk();
} else{
foreach (GameObject d in DrugiGameObjekti){
d.SetActive(true);}
foreach (GameObject p in PrviGameObjekti) {
p.SetActive(false);}
pokreniAnimacijuIZvuk();}
brojacDoSlijedecegKoristenja = 0;}
```

Kod 5.8 Dio `Update` funkcije aktiviranjeObjektaAkcijomSkripte

5.3 Specijalna mehanika

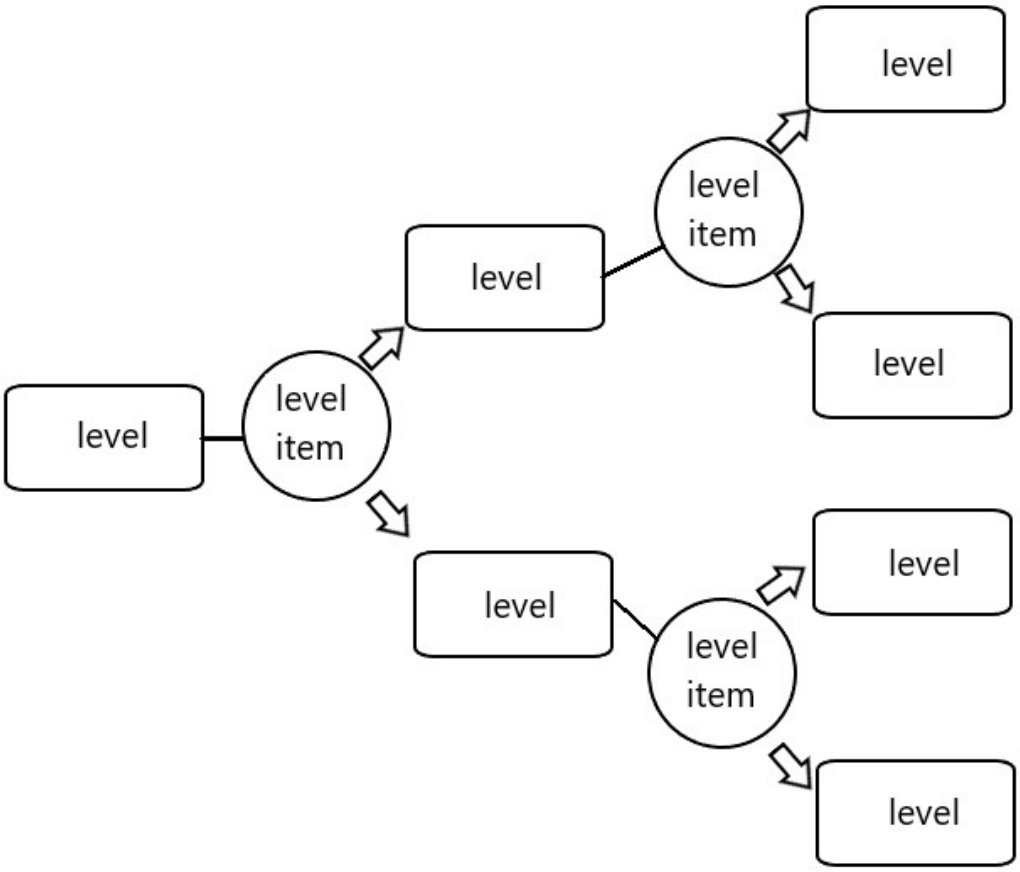
Specijalna mehanika je mehanika putovanja kroz vrijeme. Ona je ključni dio igre. Po svojoj zamisli, kao i po svojoj izvedbi, najkompliciranija je od svih mehanika u igri. Specijalna mehanika igre koju razvijam uz ovaj rad temelji se na putovanju kroz vrijeme. Rješavanje zagonetki temeljenih na ovoj mehanici je temelj progresije kroz igru.

Moguće je programirati igru tako da svaki problem ima jednu akciju koja ga rješava. Klasične igre progresije kao što su tekst-avanture funkcioniraju na ovaj način - svaki problem je jedna zagonetka koju rješava jedan odgovor ili akcija. U većini suvremenih igara barem neke od akcija ili izazova su izvedene na drugi način (Adams et al., 2012).

Iako je progresija u igri vezana striktno za pojedinačne događaje, mehanika ipak ostavlja igraču određenu slobodu. Promjena ključnog događaja u prošlosti utječe na stanje u sadašnjosti, a promjena ključnog događaja u sadašnjosti utječe na stanje u budućnosti. Igrač ne može putovati u budućnost dok ne otkrije ključni događaj u vremenskom periodu u kojem se nalazi. Svaki od tih događaja može se mijenjati neograničen broj puta, a igrač može nakon što otkrije ključni događaj u svakom trenutku putovati u budućnost ili prošlost. Nivo može pokrivati tri vremenska perioda. Prvi vremenski period (najstariji) je uvijek isti. Drugi

vremenski period (srednji) ima dvije verzije, ovisno o stanju ključnog događaja u prvom. Svaka verzija drugog perioda ima svoj ključni događaj i generira svoje dvije budućnosti (treći period). Treći period ima četiri verzije. Po dvije za svaku verziju iz drugog perioda. Verzija se aktivira ovisno o stanju ključnog događaja u srednjem periodu iz kojeg se došlo do njega. Na ovaj način se stvara stablo alternativnih budućnosti kojima se igrač može kretati.

Izvedbu specijalne mehanike riješio sam korištenjem nekoliko objekata koji kontroliraju promjene koje se događaju u nivou. Promjena vremena je riješena tako da se objekti koji su posebni za svaki vremenski period aktiviraju i deaktiviraju ovisno o stanju objekata koji drže stanje igre. Nekoliko vrsta ključnih objekata uvjetuje stanje i ponašanje nivoa. Na svakom nivou je jedan objekt koji nosi LevelManagerSkriptu i upravlja stanjem nivoa. On određuje koje stanje će nivo poprimiti i upravlja ostalim objektima. Objekti koji na sebi nose LevelItem skriptu služe za čuvanje putanje u vremenu. Postoje maksimalno tri ovakva objekta i njihovo se stanje održava putem LevelManagerSkripte (kod 5.10).



Slika 5.1 Struktura navigacije kroz vrijeme

Svaki objekt nivoa (u kodu `levelitem`) je kao jedna skretnica, promjenom stanja objekta nivoa mijenjamo budućnost koja nam je u tom trenutku dostupna (u kodu `level`). Putovanje u budućnost postaje dostupno kad se `levelitem` aktivira (kad igrač otkrije koji ključni događaj mijenja budućnost). `Levelitem` čuva stanje ključnog događaja i određuje koja budućnost je aktivna. `Level` sadrži samo ime vremena koje opisuje i listu Game objekata koji sačinjavaju to vrijeme. `Levelitemom` kao i `level` objektima koji predstavljaju budućnosti upravlja LevelManagerSkripta. Na svakom nivou može biti samo jedan objekt s LevelManagerSkriptaom, tri s LevelItem skriptom i sedam s Level skriptom (slika 5.1).

```
private void PreskociNaLevel(){
if (Igrac.IgracAktivan && levelSeMozePromjeniti &&
Input.GetAxisRaw("StranicaPlusMinus") > 0 &&
AktivniLevelItem.Aktivan && indexTrenutnogLevela < 2){
levelSeMozePromjeniti = false;
foreach (GameObject objektKojiOdlazi in
trenutnoAktivniObjekti){
objektKojiOdlazi.SetActive(false);
}
if (AktivniLevelItem.smjerPriveBuducnosti &&
AktivniLevelItem.buducnostPrva != "") {
foreach (GameObject objektKojiDolazi in
ObjektiPoBuducnostima[AktivniLevelItem.buducnostPrva]){
objektKojiDolazi.SetActive(true);
}
trenutnoAktivniObjekti =
ObjektiPoBuducnostima[AktivniLevelItem.buducnostPrva];
}
if (!AktivniLevelItem.smjerPriveBuducnosti &&
AktivniLevelItem.buducnostDruga != ""){
foreach (GameObject objektKojiDolazi in
ObjektiPoBuducnostima[AktivniLevelItem.buducnostDruga]){
objektKojiDolazi.SetActive(true);
}
trenutnoAktivniObjekti =
ObjektiPoBuducnostima[AktivniLevelItem.buducnostDruga];
}
indexTrenutnogLevela++;
}
```

```
if (Igrac.IgracAktivan && levelSeMozePromjeniti &&
Input.GetAxisRaw("StranicaPlusMinus") < 0 &&
indexTrenutnogLevela > 0 && AktivniLevelItem.proslost != ""){
levelSeMozePromjeniti = false;
foreach (GameObject objektKojiOdlazi in
trenutnoAktivniObjekti){
objektKojiOdlazi.SetActive(false);

}
if (indexTrenutnogLevela == 2){
foreach (GameObject objektKojiDolazi in
ObjektiPoBuducnostima[AktivniLevelItem.sadasnjost]){
objektKojiDolazi.SetActive(true);
}
trenutnoAktivniObjekti =
ObjektiPoBuducnostima[AktivniLevelItem.sadasnjost];}
else{
foreach (GameObject objektKojiDolazi in
ObjektiPoBuducnostima[AktivniLevelItem.proslost]){
objektKojiDolazi.SetActive(true);
}
trenutnoAktivniObjekti =
ObjektiPoBuducnostima[AktivniLevelItem.proslost];
}
indexTrenutnogLevela--;}}
```

Kod 5.9 `PreskociNaLevel` funkcija u levelManagerSkripti

Promjena vremena se događa kad se mijenja aktivni `level` prema stanju aktivnog `levelitem` objekta (kod 5.9).

```
private void PromjeniLevelItem(){
if (Input.GetButtonDown("LeveItemGumb")
&& indexTrenutnogLevela < 2
&& coolDownTimer > coolDown)
{
AktivniLevelItem.smjerPriveBuducnosti =
!AktivniLevelItem.smjerPriveBuducnosti;
coolDownTimer = 0;
}
if (indexTrenutnogLevela == 0){
AktivniLevelItem = PrviLevelItem;
}else{
if (PrviLevelItem.smjerPriveBuducnosti &&
indexTrenutnogLevela == 1){
AktivniLevelItem = DrugiLevelItemPrvaBuducnost;
}
else if (indexTrenutnogLevela == 1 &&
!PrviLevelItem.smjerPriveBuducnosti){
AktivniLevelItem = DrugiLevelItemDrugaBuducnost;}}}
```

Kod 5.10 `PromjeniLevelItem` funkcija u levelManagerSkripti

5.4 Korisničko sučelje

Grafičko korisničko sučelje u prototipu igre koji sam razvio sastoji se od dva izbornika na ekranu. Vertikalni izbornik na desnoj strani ekrana prikazuje *inventory* i omogućava igraču pregled i korištenje objekata koje je kroz igru sakupio. Horizontalni izbornik u donjem lijevom kutu je izbornik specijalne mehanike koji omogućava igraču pregled stanja ključnih događaja u svakom vremenu i putovanje iz jednog vremena u drugo.

Preferirana metoda primanja komandi korisnika je kontroler (engl. *gamepad*). Opisano standardnim rasporedom kontrolera komande su: A - skakanje, X - korištenje, Y - promjena stanja level objekata, B - korištenje *inventory* objekata. D-pad vertikalna os upravlja promjenama *inventory* objekata, D-pad horizontalna os promjenom vremena (prošlost-budućnost), lijeva analogna kontrola (engl. *analog stick*) kretanjem (lijevo - desno).

Komande i način igranja su dosta kompleksni pa je stoga nužno da su početni nivoi jednostavniji i da se pojedine komande uvode postupno da ih igrač nauči kako bi mogao pristupiti kompleksnijim zadacima.

6 Zaključak

Provedenim istraživanjem pokušao sam potvrditi tri hipoteze koje sam postavio u prvom dijelu rada. Od tri hipoteze koje sam postavio za jednu mogu reći da je mojim istraživanjem u cijelosti potvrđena. To mogu utvrditi za hipotezu da igre koje koriste specijalne mehanike u prosjeku imaju male nivoe. U istraživanju se jasno vidi veza između veličine nivoa i intenziteta korištenja specijalnih mehanika, a u prilog ovoj tvrdnji ide i vrlo niska vagana prosječna težina vrijednosti veličine nivoa. Nasuprot tome mogu tvrditi da moja hipoteza kako ove vrste igara koriste resurse male detaljnosti nije potvrđena i da moje istraživanje nije pokazalo nikakve dokaze za ovu tvrdnju. Treća hipoteza koju sam iznio je da igre sa specijalnim mehanikama u svom dizajnu koriste mali broj resursa. Iako bi se prema rezultatima istraživanja moglo reći da je ova tvrdnja potvrđena, zbog činjenice da rezultati pokazuju samo vrlo blagu tendenciju u tom smjeru i činjenice da je uzorak vrlo mali, moram ustvrditi da ni ova tvrdnja nije potvrđena.

Sve značajke koje sam mjerio u istraživanju pokušao sam što bolje implementirati u prototipu igre koju sam razvio kao dodatak ovom radu. Pri izradi sam se uvjerio u veliku korisnost i jednostavnost primjene Unity-3D razvojnog okruženja. Kod razvoja igara koje koriste nestandardne mehanike ovakvi programski okviri su vrlo korisni. Štede puno vremena tako što se osoba koja razvija igru može koncentrirati na one mehanike koje su posebne za njihovu igru. Za fiziku i ostale generičke stvari brine sam programski okvir. Uštede vremena ove vrste omogućile su mi da u potpunosti sam izvedem sve dijelove prototipa igre, sve programiranje, ilustraciju, animaciju i glazbu. Korištenjem programskog okvira poput Unity-3D razvoj igara je postao pristupačan širokom spektru zainteresiranih, među njima i meni kao studentu računarstva. Razvojem ove igre mogu reći da sam puno naučio i da mi je cijeli proces bio iznimno zabavan. Također se nadam da ću nekada u budućnosti ponovno biti u prilici razvijati neku računalnu igru.

Popis kratica

VR	*Virtual reality*	virtualna stvarnost
AR	*Augmented reality*	proširena stvarnost
FPS	*First person shooter*	pucačka igra igrana u prvom licu
FHD	*Full high definition*	rezolucija 1920×1080 piksela

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] Rollings, A. Morris, D. Game Architecture and Design:A New Edition: New riders, 2004.

[2] Crawford, C. *The Art of Computer Game Design*: Washington State University, 1997.

[3] Hocking ,J. *Unity in Action*: Manning Publications Co., 2015.

[4] Sylvester, T. *Designing Games: A Guide to Engineering Experiences*: O'Reilly Media, Inc., 2013.

[5] Adams, E., Dormans, J. *Game Mechanics: Advanced Game Design*: New Riders Games, 2012.

[6] https://unity3d.com/public-relations, studenti 2017.

[7] Benšić, N., Šuvaks, J. *Primijenjena statistika*: Sveučilište J. J. Strossmayera, Odjel za matematiku, 2013.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 19.2.2018.

Filip Ivanko