SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Gloria Babić**

Izrada avanturističke računalne igre tipa "usmjeri i klikni"

ZAVRŠNI rad

Varaždin, 2017.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Gloria Babić

Matični broj: 43317/14-R

Studij: Informacijski sustavi

Izrada avanturističke računalne igre tipa "usmjeri i klikni"

ZAVRŠNI RAD

Mentor:

Mladen Konecki, mag. inf.

Varaždin, rujan 2017.

**Sadržaj**

[1. Uvod 1](#_Toc493085372)

[2. Unity 2](#_Toc493085373)

[2.1. C# 3](#_Toc493085374)

[3. Point and click 4](#_Toc493085375)

[4. Izrada igre 6](#_Toc493085376)

[4.1. Scena Ljeto 6](#_Toc493085377)

[4.2. Kretanje 8](#_Toc493085378)

[4.2.1. Navigation System 9](#_Toc493085379)

[4.2.1.1. Pomicanje Svemirke 10](#_Toc493085380)

[4.3. Torba 14](#_Toc493085381)

[4.4. Interaktivni sustav 15](#_Toc493085382)

[4.4.1. Uvjeti 16](#_Toc493085383)

[4.4.2. Reakcije 19](#_Toc493085384)

[4.4.3. Interaktivni predmeti 22](#_Toc493085385)

[4.5. Početna scena 24](#_Toc493085386)

[5. Izvoz igre 26](#_Toc493085387)

[6. Zaključak 27](#_Toc493085388)

[7. Literatura 28](#_Toc493085389)

[8. Popis slika 29](#_Toc493085390)

1. Uvod

Inspirirana svojom najdražom igrom svih vremena – Syberijom – odlučila sam ispuniti svoj san i pokazati svoje vještine stečene tijekom studija na izradi igre. Syberia me oduševila svojim zapletom, zanimljivim likovima, prekrasnim okolišem, a iznad svega tijekom igre. Naime, igra se sastoji od mnogobrojnih zagonetki i slagalica koje se uklapaju u priču i tvore nevjerojatno zanimljivu cjelinu. Iako je igra dosta stara i prešla sam je bezbroj puta, moje oduševljenje nije jenjalo i dan danas je rado igram. Druge igre zbog kojih sam zavoljela ovaj način upravljanja likovima su igre iz serije Agatha Christie gdje otkrivamo počinitelje zločina. Sve u svemu, krenula sam se i sama okušati u izradi jedne igre na tom principu.

Naravno, moja igra je skromnijeg karaktera i kraćeg trajanja od navedenih ali nadam se da je uspješno utjelovila avanturistički duh i misterioznost. Radi se o neobičnom ženskom liku – Svemirki – koja je dospjela na zemlju kako bi našla slavno gusarsko blago.

Za izradu igre sam odlučila koristiti jedan od najpopularnijih alata, a još bitnije svima dostupan, Unity. Primarno sam ga odabrala zbog preporuke kolega koji se također bave razvijanjem igrica i već su se njim koristili. Najveća prednost mu je široka zajednica koja je već odgovorila na sva pitanja što su me zanimala tijekom izrade. Osnove sam naučila preko Udemy tečaja: „The Complete Unity 5 Guide: Unity Game Development Made Easy“, a princip „usmjeri i klikni“ na „Adventure Game Tutorial“ na službenoj Unity stranici. Modele sam uglavnom preuzimala sa „Unity Asset Store“, a zvukove sa raznih open source stranica te u vlastitoj izradi. Sav kod je pisan u jeziku C# s kojim sam se imala priliku susresti na kolegiju „Programsko inženjerstvo“, ali pravu inspiraciju za završni sam dobila na kolegijima „Algoritmi“ i „Uvod u formalne metode“, te na radionici „Izrada računalnih igara kroz primjere“. Dio skripti za kretanje lika i interakciju sa predmetima je također preuzet sa službene Unity stranice i prilagođen za ovaj specifični slučaj, ali to ću sve detaljno dokumentirati kroz rad.

1. Unity

Unity je „game engine“ ili alat za izradu igara koji je prvobitno bio namijenjen za OS X platformu, a danas podržava 27 platformi uključujući naravno Windows za koji je rađena ova igra. Unity podržava izradu 2D, 3D, VR i AR igara i aplikacija, a kodiranje je moguće u C#-u i Javascriptu. Unity je potpuno samostalan alat u kojem se može dizajnirati i isprogramirati kompletna igrica. Podržava uvoz spriteova za 2D igrice i ima napredan prikaz svijeta dok za 3D podržava dotjerivanje tekstura i rezolucije za svaku platformu zasebno, pruža „bump mapping“, „reflection mapping“, „parallax mapping“, „screen space ambient occlusion“ (SSAO), „shadow maps“, „render-to-texture“ i „full-screen post-processing effects“.

Unity se hvali kao vodeći softver za izradu igara u svijetu. Najjači su zapravo na području mobilnih igara gdje je čak 34% najboljih igara napravljeno u Unityiju. Također, industrija u kojoj se probijaju je rastuća Virtual Reality gdje je čak 90% Samsungovih i 53% Oculus Riftovih igara Unity izrade.

Osim samog alata, Unity nudi razne druge resurse i pomagala pri stvaranju igara. Neki od njih su Unity Asset Store, Unity Cloud Build, Unity Analytics, Unity Ads, Unity Everyplay, i Unity Certification. Također, Unity je dostupan u nekoliko paketa te svatko može pronaći nešto što mu odgovara; od velikih kompanija do amatera i studenata. Tako postoji Personal namijenjen početnicima i studentima pa je ujedno i besplatan. Plus je za one ozbiljnije koji žele ponešto i zaraditi od svoje igre. Zatim postoji Pro za stručnjake kojima je potrebna fleksibilnost i napredna prilagodba, a za one najveće postoji i Unity for Enterprise koji jednostavno nema granica. (Unity, 2017)

Ovaj projekt je rađen u verziji Unity 2017.1, u 3D načinu i korišten je C# za programiranje.

* 1. C#

C# je moćan objektno-orijentirani jezik čija je sintaksa prepoznatljiva svima koji su se susreli sa C, C++-om ili Javom, a u isto vrijeme je jednostavan za učenje. Podržava razne napredne koncepte koji su korišteni i u ovom radu poput enumeracije, polimorfizam, nasljeđivanja i slično. Likovi u igri su lako povezani koristeći klase, a rad nad likovima je postignut korištenjem metoda unutar klasa. Svi konstrukti i koncepti će biti detaljnije objašnjeni u nastavku. (C#, 2017)

Od dva moguća jezika u kojima se može raditi u Unityiju, odabrala sam C# zbog lakoće razumijevanja i iskustva u radu na predmetu „Programsko inženjerstvo“.

1. Point and click

Radije nego klasično kretanje likova na strelice ili druge tipke na tipkovnici, igre „usmjeri i klikni“ se u potpunosti oslanjaju na kontroliranje igre mišem. Ovakve igre čak i ne zahtijevaju glavnog lika već mogu biti scene na kojima igrač traži neke predmete poput . U novije vrijeme, naravno, i klikanjem po dodirnom ekranu. Smisao ovakvog načina upravljanja igrom je poticanje igrača na logičko razmišljanje i pobuđivanje avanturističkog duha. Igre su većinom „first person“, a dominira horror ili misteriozna radnja. Često se uz to velika pozornost obraća na izradu okoliša kako bi se što bolje infiltriralo igrača u svijet igre (Point-and-Click Game, 2017). Divan primjer za to je Syberia koja ima očaravajuće prizore.



Slika : Prizori iz Syberije

Kad se već spominje Syberia, kao što je ranije navedeno, ona je jedna od inspiracija za samu izradu ovoga rada kao i općenita inspiracija za izradu igara. Syberia je fantastična priča o odvjetnici koja samo želi sklopiti kupoprodajni ugovor, a uvuče se u neobičnu avanturu potrage za vlasnikom tvornice te na kraju legendarnog otoka Syberije. Igra je prepuna zadivljujućih prizora, zanimljivih likova i nevjerojatnih mehaničkih strojeva koji doprinose čari igre. Igra je originalno sastavljena u dva dijela, a treći dio je tek pokušaj nadovezivanja na prijašnje igre – ne slijedi istu priču.

Iz Syberije je preuzet koncept mijenjanja pokazivača za različite radnje, kao i mijenjanje kamere unutar iste scene.



Slika : Syberia pokazivači

Druga igra koja je spomenuta kao inspiracija ovom radu je Agatha Christie: Murder on the Orient Express. Igra je napravljena prema popularnom romanu istog imena. Igrač je u ulozi slavnog detektiva Hercule Poirota i istražuje ubojstvo. Igra pruža dojam kao da se uistinu nalazite u luksuznom vlaku te cijelo vrijeme imate osjećaj da ste u opasnosti jer vreba ubojica.



Slika : Agatha Christie: Murder on the Orient Express

Ova igra je bila inspiracija kako se iz već gotove priče može složiti izvanredna igra što znači da igra ne mora imati originalnu priču da bi bila zanimljiva. Igrajući ovu igru sam vidjela kako postoje i druge igre poput Syberije te su gotovo jednako zanimljive pa sam još više odlučila i sama kreirati nešto slično.

1. Izrada igre

Kako je tema ovog rada avanturistička igra, odlučila sam se za klasičnu potragu za blagom. Igra nema kartu s blagom ni ikakve druge upute pa igrač sam mora otkriti kako se kreće i kuda treba ići s ciljem igre da se što teže pronađe blago. Igra se sastoji od tri scene: Početne, Torbe i Ljeto. Početna scena sadrži okoliš iz igre u pozadini i dva interaktivna gumba. Gumb Igraj učitava iduću scenu, a gumb Izađi gasi aplikaciju. Scena Torba sadrži „Canvas“ sa četiri polja, tj. prazna mjesta za predmete koje ćemo skupljati tijekom igre. Unutar scene Torba se učitava scena Ljeto kako bismo u sceni Ljeto mogli vidjeti koje predmete imamo u Torbi. U nastavku ću detaljno opisati izradu svake scene.

* 1. Scena Ljeto

U ovoj sceni sam prvo počela sa izradom terena. Zamislila sam pustinjske dine pa sam prema tome tražila i prikladne elemente.

Prvo sam stvorila novi 3D element „Terrain“ koji je stvoren upravo za oblikovanje okoliša. Na terenu sam oblikovala uzvisine koje će kasnije predstavljati dine, a ostavila sam ravne dijelove za kretanje igrača kako ne bih imala problema sa pozicioniranjem što ću kasnije još objasniti kod kodiranja. Na textures.com sam našla Seamless teksturu pustinjskog pijeska tj. teksturu bez ruba koja se može ponavljati. Unity nudi Standard Assets iz kojeg sam preuzela Environment pack koji sadrži palme, travu i vodu koju sam iskoristila za oazu. Palme sam učitala kao sastavni dio terena kako bi zauzimale manje memorije nego kad bi svako stablo zasebno učitavala. Isto sam učinila i sa travom. Na Unity Asset Store sam našla Cactus Pack iz kojeg sam iskoristila razne modele kaktusa i pustinjskih kamenja. Od tih kamenja sam načinila puteljak koji olakšava igraču snalaženje u prostoru, špilju u kojoj se nalazi ključ te prepreke kako igrač ne bi vidio vatru i špilju odmah s početka igre.



Slika : Oaza

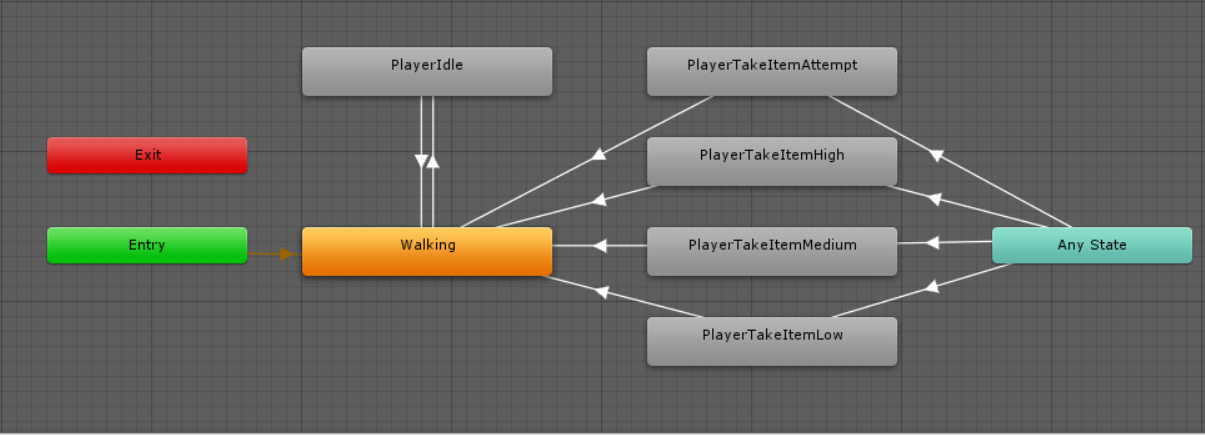
Sa Asset Storea sam preuzela i škrinju iz Fantasy Treasure Chest paketa, kantu iz Medieval Containers Pack with Wagon, ključ iz Handpainted Keys te grmlje i stablo iz Free Desert Plants paketa.

Vatru i dim sam sama kreirala koristeći „Particle System“ i teksture dima i vatre preuzete slijedno sa: <http://johnstejskal.com/wp/how-to-create-realistic-smoke-in-unity/> i <http://jvunity.weebly.com/uploads/4/7/6/0/47606749/985133_orig.jpg>.

Svijet je kreiran koristeći „cubemap“ način, a teksturu sam preuzela sa: <https://www.pinterest.com/pin/239042692703500908/>.

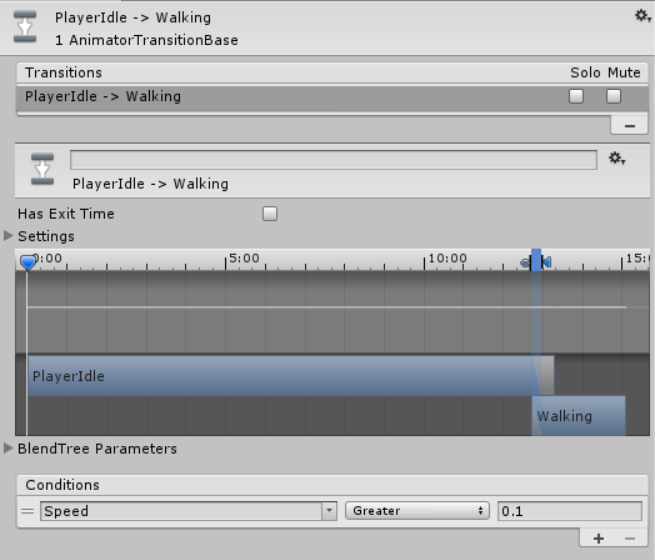
* 1. Kretanje

Kako je ovaj rad baziran na principu „usmjeri i klikni“ upravo tako se kreće i junakinja igre – Svemirka. Svemirka je preuzeta iz već spomenutog tutorijala „Adventure Game Tutorial“ zajedno sa skriptom za kretanje. Model Svemirke je animirani humanoid što je idealno za Unity koji nudi potporu za baš takve modele. Njeno kretanje je definirano animatorom koji joj je dodijeljen.



Slika : Animator Svemirke

Na slici možemo vidjeti razna stanja tj. animacije u kojima se lik može nalaziti. Osim hodanja i stajanja Svemirka ima i animacije za uzimanje predmeta. PlayerTakeItemLow se koristi kod uzimanja kante i kupljenja vode iz jezera. Sve animacije imaju određene uvjete za okidanje tj. na idućoj slici vidite parametre za pokretanje animacija. Na osnovu veličine parametra Speed, točnije ako je veći od 0.1 Svemirka prelazi u stanje Walking, u suprotnom je u stanju PlayerIdle. Sva PlayerTake stanja se okidaju na vlastiti „trigger“ koji se poziva u kodu.

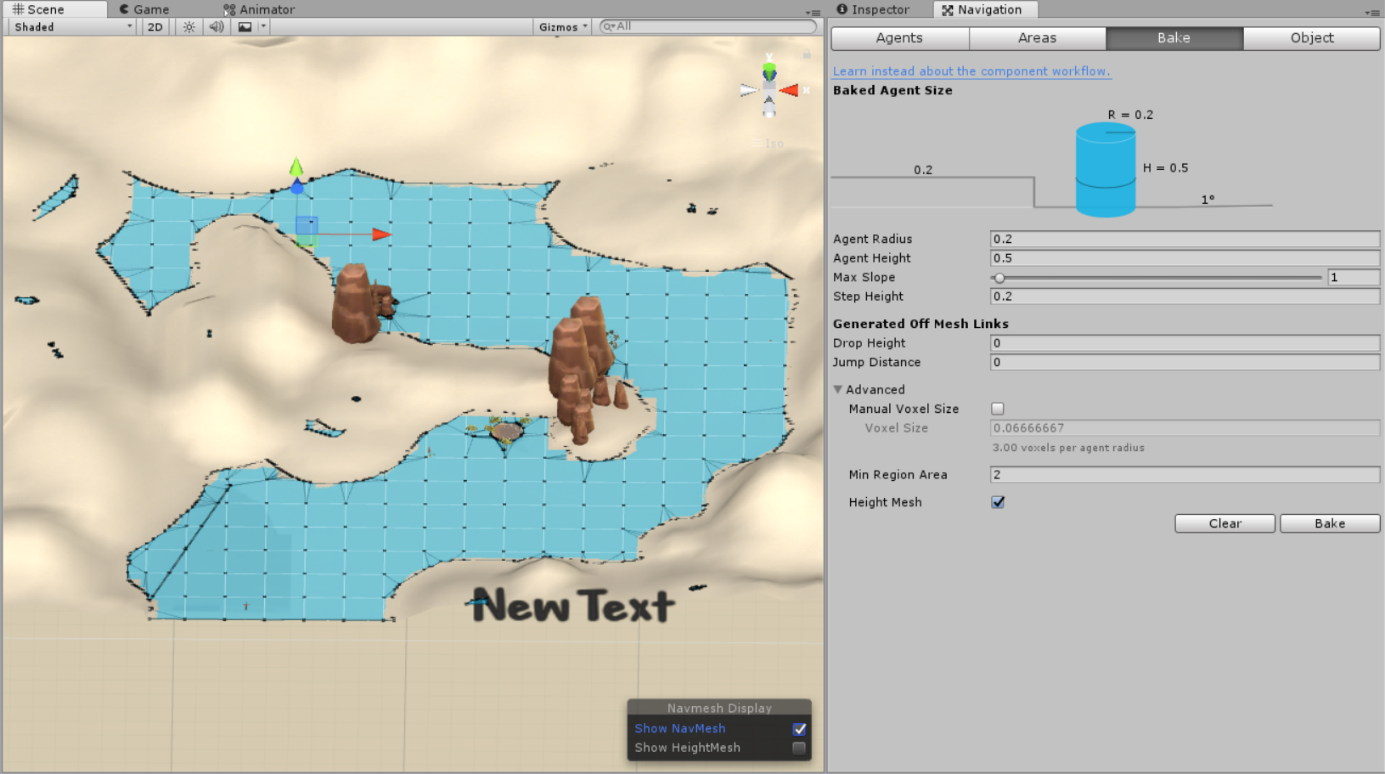


Slika : Prijelazno stanje iz stajanja u hodanje

Kad smo pomoću Animatora definirali animacije pri kretanju glavnog lika, možemo krenuti na konkretno pomicanje lika po terenu.

* + 1. Navigation System

Unity sadrži izvanredan sustav za kretanje pod nazivom Navigation System. Bilo koja površina se može postaviti (Bake ili Build) kao NavMesh (Navigation Mesh). NavMesh je struktura podataka koja opisuje površine u svijetu igre po kojima se može hodati i određivati put od jedne površine do druge. U ovom slučaju je NavMesh postavljen na sami teren korištenjem Bake tehnike. Teren je postavljen kao Navigation Static kako bi Unity prepoznao da se teren neće pomicati tijekom igre. To je uvelike bitno zbog toga što Unity samo jednom mora računati sve koordinate po kojima se može kretati.



Slika : Bake NavMesh

Svemirka ima NavMesh Agent komponentu koja stvara likove koji se kreću po svijetu igre koristeći NavMesh te izbjegavaju prepreke i jedni druge. Kao što vidimo na slici 4, Svemirka se može kretati po plavom dijelu terena. (Navigation System in Unity, 2017)

Kako bi igra prepoznala klik mišem na teren ili na određeni predmet, potrebno je dodati dvije „stvari“. Prva od njih je EventSystem koji prepoznaje određene događaje baš poput klika mišem, a druga je Physics Raycaster – komponenta koju ćemo dodati na svaku kameru kako bi sustav mogao znati na koju točku želimo pomaknuti Svemirku iz naše perspektive.

Međutim, pravi pokretač Svemirke je naravno C# skripta naziva Pomicanje Svemirke.

* + - 1. Pomicanje Svemirke

Skripta Pomicanje Svemirke dodana je instanci lika Svemirke koji se nalazi u igri kao komponenta. Čitava skripta je bazirana na primjeru skripte „PlayerMovement“ iz „Adventure Game Tutorial“ tutorijala sa službene Unity stranice.

Skripta ima nekoliko javnih varijabli koje se mogu mijenjati izravno iz Unity Inspector kartice. Dvije najvažnije su animator i agent kojima dodjeljujemo Svemirkin animator i samu Svemirku kao NavMesh Agenta. Ostale varijable nam dopuštaju podešavanje kretanja poput usporavanja rotacije i slično.

Dvije privatne varijable koje bi bilo važno spomenuti su hashBrzinaParametar i hashKretanjeTag koje dohvaćamo pomoću StringToHash metode iz animatora. Dohvaćamo parametar Speed iz animatora i tag Kretanje iz animatora koji je postavljen na animacije u kojima se Svemirka kreće.

U metodi Start, koja je predefinira metoda koja se izvršava prilikom pokretanja igre, prvo onemogućavamo automatsko rotiranje Svemirke jer ćemo napraviti prilagođenu rotaciju. Kako ne želimo da se Svemirka kreće prilikom interakcije s nekim predmetom, odmah na početku ćemo dodijeliti vrijednost čekanja prilikom interakcije u sekundama. Također, početna pozicija destinacije na koju želimo doći je zapravo sama pozicija Svemirke jer se još nikamo nismo pomaknuli.

Iduća metoda je također predefinirana OnAnimatorMove. U njoj postavljamo željenu brzinu kretanja Svemirke u odnosu na animator koji onda određuje koju će animaciju pokrenuti (trčanje, hodanje ili stajanje). Naša željena brzina je zapravo udaljenost podijeljena vremenom po svakom okviru (frame).

Nakon toga u metodi Update koja se poziva svaki novi frame. Ako se put još uvijek računa, ne želimo učiniti ništa. U drugom slučaju, put je već izračunat te postavljamo brzinu na željenu. Zatim, ako se nalazimo unutar udaljenosti na kojoj se želimo zaustaviti, pozvat ćemo metodu zaustavljanja, malo dalje od toga usporavanje i konačno kretanje. Na kraju, animatoru prosljeđujemo brzinu preko već spomenute hash varijable.

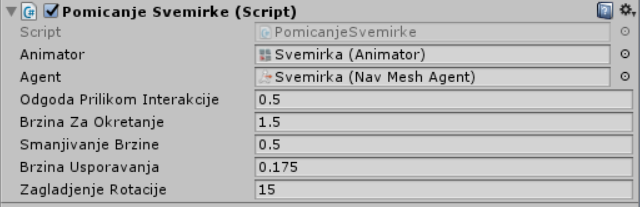
Metoda Zaustavljanje kao parametar ima out float brzina, zato što se u njoj događaju promjene nad brzinom koje želimo zapamtiti. Ona zaustavlja Svemirku te postavlja njenu poziciju točno na željenu poziciju i na kraju postavlja brzinu na 0.

Metoda Usporavanje osim što utječe na brzinu, treba znati i kolika je udaljenost od željene destinacije kako bismo se mogli početi zaustavljati. Kako želimo sami definirati usporavanje Svemirke, baš kao i u zaustavljanju ćemo prvo pozvati agent.Stop(). Novu poziciju Svemirke postavljamo MoveTowards funkcijom i konstantnom brzinom usporavanja. Postavljamo brzinu na linearnu interpolaciju između brzine usporavanja i nule u odnosu na proporcionalnu udaljenost, tako da će se brzina smanjivati kako se Svemirka približava željenoj destinaciji.

Metoda Kretanje zapravo utječe na rotaciju Svemirke. Željena rotacija je kvaternion (Quaternion) u smjeru kojeg nalazimo pomoću statične metode LookRotation iz vektora brzine. Rotiramo Svemirku koristeći interpolaciju između njene trenutne rotacije i željene rotacije koristeći varijablu usporavanje rotacije koja određuje brzinu rotiranja.

Metoda klikNaPod je zaslužna za pokretanje cijelog procesa micanja Svemirke. Prvo se pobrinemo da možemo upravljati unosom, tj. prihvaćati novi klik te da ne vršimo interakciju sa nekim predmetom. Zanima nas gdje smo točno kliknuli, pa iz BaseEventData varijable izvlačimo PointerEventData koji sadrži podatke o kliku mišem. Provjeravamo točku u svijetu koju smo trenutno kliknuli i postavljamo položaj destinacije u tu točku, a u suprotnom nalazimo neku točku u blizini.

Kako bi čitava skripta radila, moramo joj dodijeliti objekt iz igre kojeg želimo pokretati, točnije objekt Svemirke na mjesto Animatora i Agenta.



Slika : Pomicanje Svemirke – Inspector

Osim toga, na sami teren dodajemo komponentu Event Trigger, specifično Pointer Click koji je tipa BaseEventData, komponenti dodjeljujemo instancu Svemirke te za očitavanje podataka koristimo našu metodu klikNaPod kao što smo ranije i opisali.

Međutim, ne želimo samo pomicati Svemirku nego želimo da ima neke interakcije sa predmetima. Za to smo stvorili posebne objekte naziva Interactable koje ćemo opisati malo kasnije, a Pomicanje Svemirke ima posebne metode koje opisuju njeno ponašanje u dodiru s ovim objektima. Ovo je zapravo bitniji dio kretanja Svemirke upravo zbog „zidova“ postavljenih u igri koji na klik tj. prilikom interakcije sa Svemirkom mijenjaju pogled na igru tj. aktiviraju iduću kameru.

Prvo bi se trebali pobrinuti da Svemirka gleda u smjeru predmeta s kojim želi imati interakciju. To činimo u metodi Zaustavljanje iščitavajući rotaciju predmeta iz njegove lokacije te naravno postavljajući rotaciju Svemirke na te iste vrijednosti. Nadalje, bilo bi dobro da Svemirka zapravo vrši interakciju sa kliknutim predmetom pa naravno pozivamo metodu Interact(). Kako bi osigurali da Svemirka vrši interakciju samo jednom, postavljamo trenutni predmet interakcije na null. Trebali bi osigurati da Svemirka „na miru“ obavi interakciju. To ćemo postići blokiranjem bilo kakvog unosa podataka tj. nećemo prihvaćati nove klikove. Želimo li nove klikove ili ne, spremamo u varijablu upravljanjeUnosom koja je na početku naravno postavljena na true kako bi mogli upravljati igrom. Ovdje koristimo našu drugu hash varijablu iz koje iščitavamo da li se animator nalazi u stanju kretanja ili ne.

Koliko dugo trebamo čekati interakciju sa predmetom, kontroliramo metodom CekajInterakciju koja vraća tip IEnumerator. Na početku metode naravno postavljamo upravljanjeUnosom na false jer ne želimo nove unose, a na kraju vraćamo na true. Za čekanje koristimo yield return koji izlazi iz koda i čeka dano vrijeme koje smo mi definirali na početku varijablom cekanjePrilikomInterakcije i odgodaPrilikomInterakcije. Zatim, dok se god ne nalazimo u stanju kretanja čekamo po jedan frame, vraćamo se opet u petlju i provjeravamo uvjet. Ovu metodu pozivamo sa StartCoroutine u Zaustavljanju.

Valjalo bi još spomenuti kako u Usporavanju polako okrećemo Svemirku prema predmetu interakcije interpoliranjem između trenutne rotacije i rotacije predmeta.

Konačno možemo definirati što se zapravo događa kada kliknemo na predmet. Metoda klikNaPredmet će kao parametar prihvaćati varijablu tipa Interactable. Na samom početku ćemo provjeriti da li nam je uopće dopušteno uzimati novi unos tj. klik. Zatim ako jest, postavit ćemo trenutniPredmetInterakcije na proslijeđeni predmetInterakcije. Položaj destinacije ćemo iščitati iz lokacije predmeta, postaviti agenta tj. Svemirku na tu destinaciju i konačno omogućiti ponovno kretanje.

Baš poput kretanja po tlu, predmeti također imaju Event Trigger komponentu u kojoj ovoga puta odabiremo metodu klikNaPredmet za očitavanje potrebnih parametara. Predmete interakcije ćemo kasnije podrobno opisati. (Adventure Game Tutorial, 2017)

* 1. Torba

Kako bi predmeti koje pokupimo u igri mogli biti prikazani na ekranu, dodana je tzv. torba tako da u svakom trenutku igrač bude svjestan što posjeduje. U želji da se igra sastoji od više scena, torba je napravljena kao posebna stalna scena koja unutar sebe učitava druge scene. Za to koristimo posebnu skriptu koja se zove UpravljanjeScenama. Njoj kroz javnu varijablu prosljeđujemo ime scene koju želimo učitati. Kako je ovdje riječ o samo jednoj sceni, učitavamo Ljeto. Kako se učitava jedna scena preko druge, osvjetljenje se nalazi u sceni Torba. Korištena su dva usmjerena svjetla (Directional light). Također je u sceni Torba dodan zvuk na glavnu kameru koji se odmah pokreće i vrti u petlji za „ugođaj“ u igri.

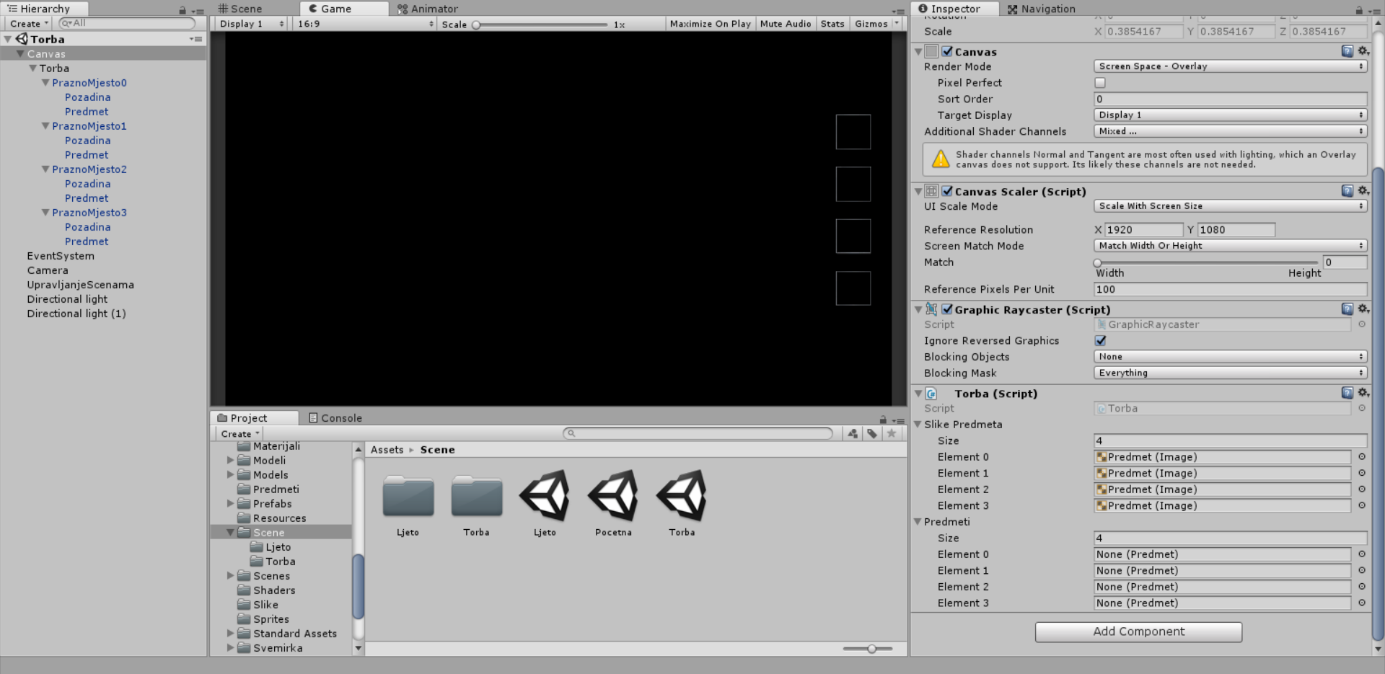
U sceni Torba se nalazi platno (canvas) unutar kojeg se nalaze prazan objekt Torba. Torba se sastoji od točno četiri prazna mjesta za spremanje predmeta. Svako prazno mjesto ima dvije slike: pozadinska slika praznog mjesta i slika pokupljenog predmeta. Pozadinska slika je naravno ista za sva mjesta i ne mijenja se dok je slika predmeta dinamična ovisno o tijeku igre.

Kako bi što lakše upravljali torbom, napravljen je prefab praznog mjesta što znači da ne moramo svaki put ispočetka definirati svako prazno mjesto već samo povučemo novo iz izvora projekta te da možemo odjednom primijeniti neko pravilo na sva prazna mjesta. Imenovali smo ih od nule jer ćemo kasnije kolekciju također imenovati od nule. Mjesta smo poravnali na desni dio ekrana.

Kako bi torba funkcionirala, treba nam naravno nekoliko skripti. Prva od njih je Predmet koja sadrži Sprite tj. samu sliku predmeta i GameObject kao referencu na objekt o kojem se radi. Ova skripta je CreateAssetMenu što znači da možemo kreirati nove predmete direktno iz Unity Asset trake. Skripta Torba je malo kompliciranija. Imamo dvije jednostavne metode za spremanje predmeta u torbu i vađenje predmeta iz torbe. Metoda DodajPredmet sadrži for petlju koja pregledava koje je prvo slobodno mjesto za dodati predmet i postavlja na njega sličicu proslijeđenog predmeta. Na vrlo sličan način i uklanjamo predmet u metodi UkloniPredmet.

Pitanje je, kada pozivamo ove metode? Kada igrač klikne na predmete u svijetu igre, tzv. Interaktivne predmete, želimo da nastane neka reakcija. Dok ćemo sve reakcije kasnije detaljno opisati, ovdje je važno napomenuti da postoje dvije reakcije koje se odnose na Torbu. Te dvije reakcije su PickedUpItemReaction i LostItemReaction. Kada su ispunjeni određeni uvjeti, pokreću se ove reakcije. PickedUpItemReaction prosljeđuje predmet metodi DodajPredmet, a LostItemReaction prosljeđuje predmet metodi UkloniPredmet.

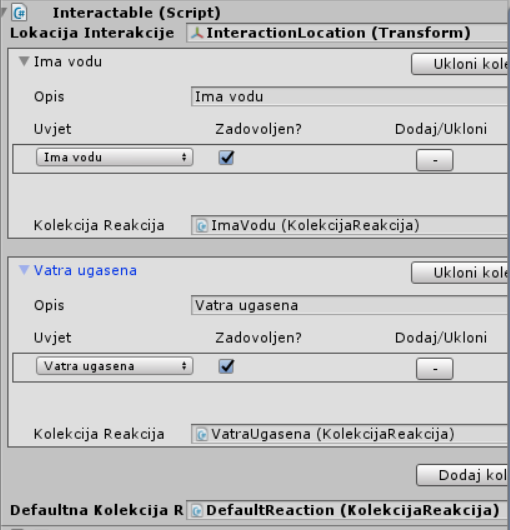
Skriptu Torba dodajemo kao komponentu na platno s početka ovog poglavlja, a PickedUpItemReaction i LostItemReaction dodajemo u kolekciju reakcija na Interaktivni predmet. U inspektoru platna dodjeljujemo slike predmeta, a predmete dodajemo dinamično kroz igru u skriptu Torba kako bi bili povezani sa stvarnim elementima u igri (Adventure Game Tutorial, 2017)



Slika : Torba

* 1. Interaktivni sustav

Kako je već spomenuto, igra sadrži brojne predmete s kojima Svemirka može vršiti interakciju. Cijeli sustav se temelji na načelu uvjeta i reakcija. Naime, kako bi se stvorila određena reakcija: spremljen predmet u torbu, predmet nestaje iz igre, pokrene se zvuk, prikaže se tekst i slično, moraju biti zadovoljeni određeni uvjeti. Uvjeti će biti sličnog karaktera: pokupljen određeni predmet, uklonjen određeni predmet itd. Svaki predmet interakcije će na klik imati određen set reakcija. Ako ne postoje nikakvi uvjeti za pokretanje reakcija, tu kolekciju reakcija nazivamo Default Reaction i nju sadrže svi predmeti interakcije. Nadalje, svaki predmet može sadržati više kolekcija uvjeta od kojih svaka ima svoj set reakcija. Najlakše je objasniti ovaj sustav na primjeru. Predmet interakcije vatra ima tri kolekcije reakcija i dvije kolekcije uvjeta. Prvo se provjerava da li je ispunjen uvjet naziva ima vodu. Ako jest, slijedi kolekcija reakcija naziva ImaVodu. Na isti način se provjerava i uvjet vatra ugašena i okida kolekcija reakcija VatraUgasena. Ako nijedan uvjet nije ispunjen, izvršava se kolekcija DefaultReaction.



Slika : Vatra Interactable

U ovom poglavlju će biti redom opisani uvjeti, reakcije te sami interaktivni predmeti.

* + 1. Uvjeti

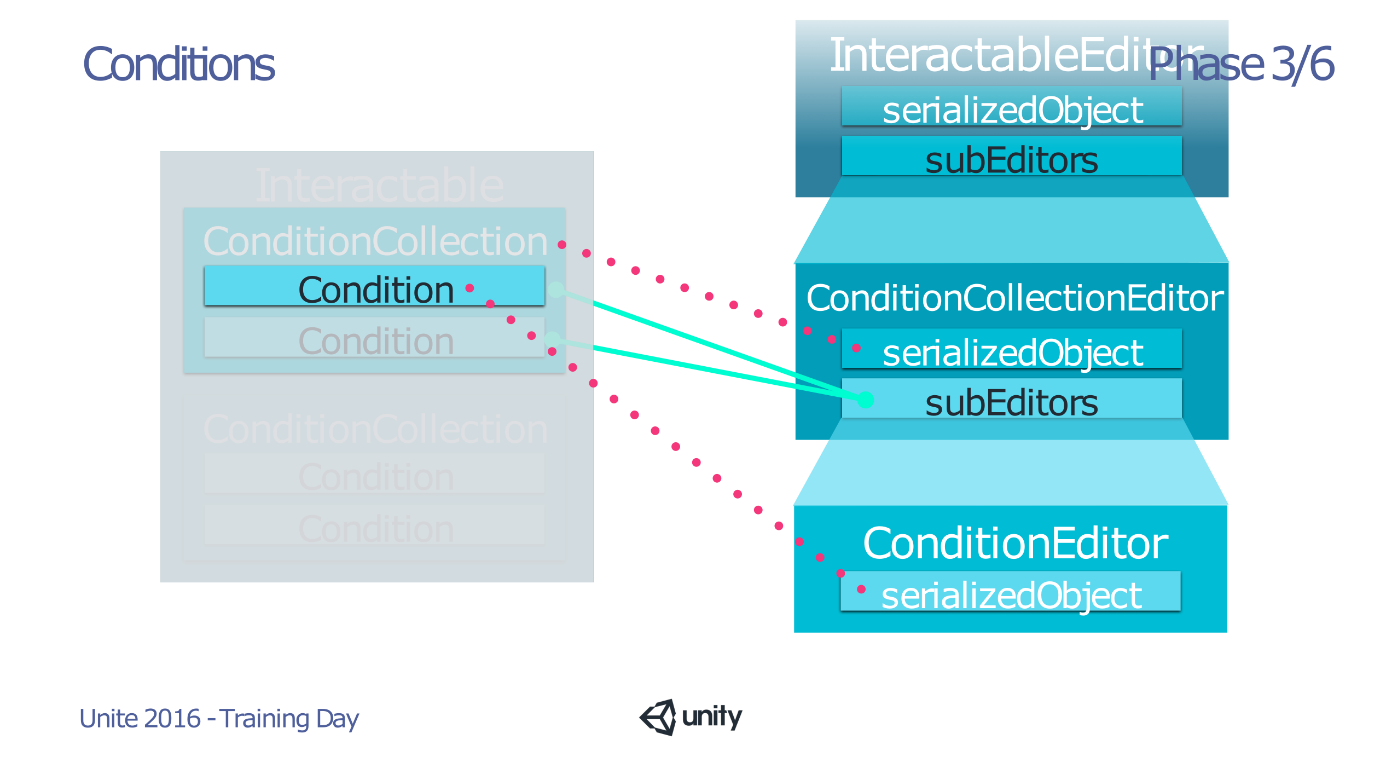
Uvjeti su jednostavni podatkovni objekti koji sadrže opis uvjeta i boolean vrijednost da li je uvjet ispunjen ili ne. Kako ne bismo uspoređivali opise uvjeta, sadrže i hash vrijednost upravo u te svrhe. Spremljeni su kao ScriptableObject jer su to objekti koji se ne moraju pridružiti nekom stvarnom objektu u igri.

SviUvjeti nasljeđuje klasu ResettableScriptableObject kako bi se mogli resetirati na početku svake igre. Sadrži polje uvjeta koje ćemo provjeravati tijekom igre te svojstvo instanca preko kojeg ćemo pristupati uvjetima. Pri dohvaćanju uvjeta provjeravamo da li se nalazi u memoriji, u resursima te ako ne postoji javljamo da ih je potrebno kreirati. Zatim imamo metodu za resetiranje uvjeta na početnu vrijednost, jednostavna for petlja koja sve vraća na false. Konačno, metoda ProvjeriUvjet pronalazi globalni uvjet u SvimUvjetima čija se hash vrijednost poklapa sa zahtijevanim Uvjetom i vraća istinu ili laž ovisno o tome da li je taj uvjet zadovoljen. Ako na primjer želimo da uvjet ImaVodu bude zadovoljen, to je u tom trenutku zahtijevaniUvjet i prosljeđujemo ga kao parametar metodi ProvjeriUvjet. Prvo ćemo u SvimUvjetima tražiti istu hash vrijednost kao i zahtijevaniUvjet i tako ćemo pronađi globalniUvjet istoga naziva: ImaVodu. Konačno vraćamo usporedbu jednakosti globalnog i zahtijevanog uvjeta i tako doznajemo da li je uvjet ImaVodu globalno zadovoljen ili ne.

Iduća skripta vezana uz uvjete je KolekcijaUvjeta. Kolekcija uvjeta nam služi za postavljanje više uvjeta na određene reakcije. Sastoji se od jedne metode koja u for petlji poziva metodu ProvjeriUvjet i tako redom traži barem jedan uvjet koji nije zadovoljen, što znači da čitava kolekcija uvjeta nije zadovoljena, i izlazi iz metode. Ako su svi uvjeti zadovoljeni, pokreće se određena kolekcija reakcija za tu kolekciju uvjeta. Kolekcija uvjeta se kreira dinamično kroz inspector za što postoji posebna editor skripta. Kolekcija reakcija se također dodjeljuje kroz inspector.

Kako bismo omogućili da se sve ovo odvija kroz inspector, potrebno je kreirati malo kompliciranije editor skripte. Ovaj rad nije orijentiran detaljnom prikazu editor skripti pa će one biti samo ukratko opisane u svakom dijelu.

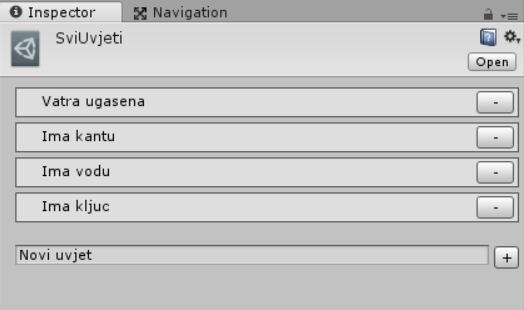
Jedan interaktivni predmet će imati više kolekcija uvjeta od kojih svaka ima više uvjeta. Kako bismo ovo mogli vidjeti u inspectoru koristit ćemo editor sa editorima točnije skriptu EditorWithSubEditors koja je u potpunosti preuzeta sa Adventure Game Tutorial i nije izmjenjivana. U suštini ova apstraktna skripta osigurava da pri nasljeđivanju mogu biti proslijeđeni isključivo editor i ciljani objekt uređivanja, da se stvara niz editora po potrebi te također brišu po potrebi.



Slika : Editori (dostupno 10.9. na <https://unity3d.com/learn/tutorials/projects/adventure-game-tutorial/conditions?playlist=44381>)

Editor za Uvjet je CustomEditor tipa Uvjet i zove se UvjetEditor (ConditionEditor). On ne sadrži nikakve editore unutar sebe pa ne nasljeđuje EditorWithSubEditors već samo Editor. Ova skripta definira tri vrste izgleda editora za uvjete: unutar Asseta SviUvjeti i omogućuje brisanje uvjeta, izgled uvjeta kada se samostalno označi kao Asset te unutar Kolekcije uvjeta.

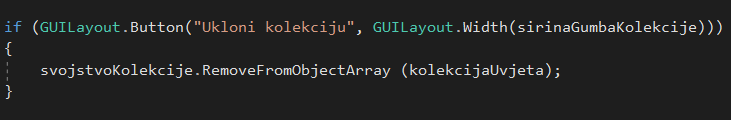
Editor za SviUvjeti (AllConditionsEditor) također nasljeđuje klasu Editor. Stvara Asset SviUvjeti te omogućuje dodavanje novih uvjeta pri čemu stvara i novi UvjetEditor.



Slika :SviUvjeti

Editor za Kolekciju Uvjeta je CustomEditor tipa KolekcijaUvjeta. Nasljeđuje apstraktnu klasu EditorWithSubEditors koja ugnježđuje UvjetEditor, a ciljani objekt je Uvjet. Opisuje izgled samog editora i što se u njemu nalazi. Dakle, unutar ovog editora se nalazi editor za uvjete. Možemo dodavati i brisati kolekcije uvjeta i unutar njih biramo koje uvjete želimo. Prikaz ovog editora može se vidjeti na slici 7.

U spomenutim editorima, a i kasnije, koristi se skripta SerializedPropertyExtensions koja služi kao proširenje za klasu SerializedProperty tj. metode za upravljanje nizovima objekata.



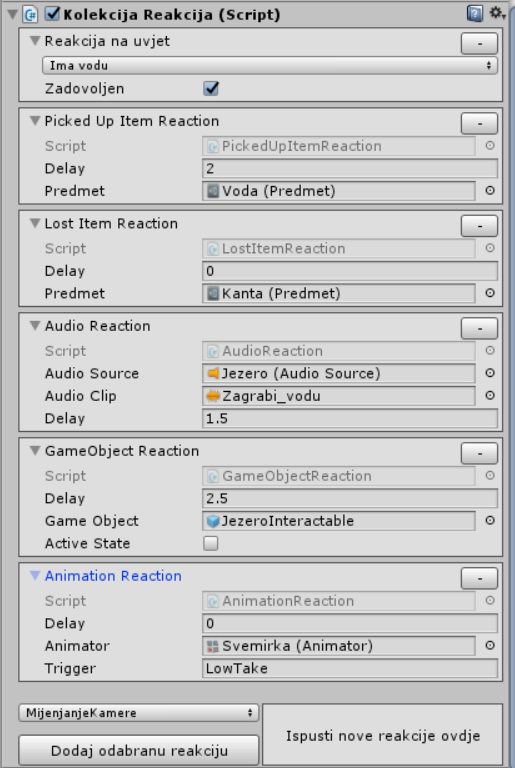
Slika : Primjer korištenja SerializedPropertyExtensions (RemoveFromObjectArray)

SveUvjete resetiramo u UpravljanjeScenama prije svakog učitavanja scene Ljeto.

* + 1. Reakcije

Reakcije će biti nizovi akcija koje će se poduzimati ovisno o ispunjenim uvjetima kada igrač klikne na interaktivne predmete. Reakcije koje ćemo koristiti su pokreni animaciju, pokreni zvuk, prikaži tekst, dodaj predmet u torbu, izbaci predmet iz torbe, izbriši predmet iz scene, promijeni globalni uvjet, promijeni kameru i završi igru. Većina ovih reakcija je preuzeto iz Adventure Game Tutoriala, dok su Mijenjanje kamere i Kraj Igre u potpunosti samostalno napisane.

Neke reakcije imaju opcionalnu odgodu, dok su druge trenutačne. Svaka reakcija je različita te će za svaku biti kreiran posebni editor. Reakcije ćemo grupirati u kolekcije reakcija baš poput kolekcija uvjeta, kako bi odjednom mogli pokrenuti više vrsta reakcija. Primjer je kolekcija reakcija kada je ispunjen uvjet da Svemirka ima kantu. Sastoji se od mnogih različitih reakcija.



Slika : Kolekcija reakcija Ima Vodu

Prvo se pokreće animacija uzimanja predmeta s poda (Animation Reaction - LowTake) i u isto vrijeme se iz Torbe izbaci kanta (Lost Item Reaction – Kanta) te zadovolji globalni uvjet Ima vodu (Reakcija na uvjet). Zatim se čuje zvuk grabljenja vode (Audio Reaction – Zagrabi\_vodu), u Torbi se pokazuje novi predmet – Kanta s vodom (Picked Up Item Reaction – Voda) te naposljetku onemogućujemo objekt JezeroInteractable (GameObject reaction – JezeroInteractable) jer poslije grabljenja vode nemamo više nikakvih akcija vezanih za jezero. Redoslijed pokretanja reakcija vrlo jednostavno postižemo koristeći odgodu kod reakcija u kojima je to omogućeno.

Kako bismo omogućili čitav sustav reagiranja, prvo moramo opisati dvije apstraktne klase. Jedna se koristi za već spomenute trenutačne reakcije, a druga za reakcije sa odgodom. Skripta Reaction je bazna klasa za sve reakcije. Sadrži četiri metode od kojih je jedna inicijalizacijska u kojoj se piše sav kod za pojedine reakcije zajedno sa pozivom iduće metode. Iduća metoda je virtualna i koristi se isključivo u klasama koje nasljeđuju trenutnu. Nadalje imamo metodu React koja pokreće samu reakciju i poziva metodu ImmediateReaction za trenutačne reakcije. U ovom trenutku se ova klasa čini jako nejasna i teško shvatljiva. Međutim, kroz kasnije primjere će sve biti pojašnjeno. Skripta DelayedReaction je također apstraktna klasa i ona je bazna klasa za sve reakcije koje imaju neku odgodu. Ona sama nasljeđuje od skripte Reaction, a razlikuje se po tome što pri inicijalizaciji definiramo odgodu te u samom reagiranju pozivamo usporednu rutinu koja čeka određeno vrijeme i tek onda poziva trenutačnu reakciju.

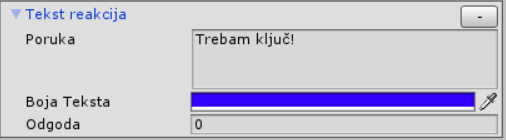
Sad kad smo opisali bazne klase za reakcije, objasnit ćemo redom kako svaka od njih funkcionira. Prvo krećemo od trenutačnih reakcija.

AudioReaction se koristi za pokretanje određenog zvuka od određenog izvora. Sadrži odgodu iz razloga što izvor zvuka sam kontrolira odgodu pa ovu reakciju nije bilo potrebno svrstati u odgođene. Vrlo jednostavna skripta kojoj se zvuk, izvor i odgoda dodjeljuju kroz inspector. To nam naravno omogućuje druga skripta – AudioReactionEditor – koja jednostavno nasljeđuje klasu Reaction Editor.

ReactionEditor je apstraktna klasa koja sadrži osnovni opis izgleda svakog editora reakcija unutar kolekcije reakcija. Na slici 11 možemo vidjeti razne vrste editora. Svaki od njih prikazuje ime reakcije, skriptu koja predstavlja tu reakciju te potrebne varijable za funkcioniranje određene reakcije. Većina ih ima i varijabilnu odgodu.

ReakcijaNaUvjet ima dvije varijable od kojih je jedna uvjet, a druga boolean. Ona vrlo jednostavno postavlja određeni uvjet na istinu ili laž tj. da li je zadovoljen ili ne. U inspectoru je oblikujemo pomoću ReakcijaNaUvjetEditor (ConditionReactionEditor) skripte koja oblikuje listu postojećih uvjeta iz koje možemo odabrati željeni uvjet i radio gumb kojim označavamo da li želimo taj uvjet označiti zadovoljenim.

TextReaction je malo kompleksnija reakcija od ranije spomenute AudioReaction. Sadrži poruku, boju teksta, odgodu te poseban objekt TextManager. TextManager je skripta koja prikazuje tekst na ekranu i u slučaju da se više poruka treba prikazati odjednom, osigurava pravilan poredak. Za oblikovanje ove reakcije u inspectoru služi skripta TextReactionEditor koja osigurava prostor za pisanje poruke, biranje boje i unos odgode.

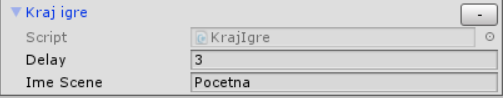


Slika : TextReactionEditor

AnimationReaction nasljeđuje DelayedReaction. Sadrži varijablu za animator te ime okidača koji će pokrenuti željenu animaciju. Kako smo već spominjali da su stringovi nepouzdani pri uspoređivanju, koristimo radije hash vrijednost okidača. Skripta koja oblikuje ovu reakciju u inspectoru je AnimationReactionEditor.

GameObjectReaction je također DelayedReaction. Ima dvije varijable: GameObject i bool. Objekt nas veže uz stvari objekt iz igre kojim želimo upravljati, a bool varijabla određuje da li taj objekt želimo aktivirati ili deaktivirati iz igre. Editor se oblikuje GameObjectReactionEditor skriptom.

KrajIgre (DelayedReaction) je reakcija koja na osnovu imena učitava novu scenu pomoću SceneManagera. U inspectoru je oblikuje KrajIgreEditor.

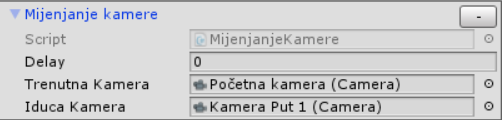


Slika : KrajIgreEditor

LostItemReaction (DelayedReaction) smo već spominjali kod izrade Torbe. Ova skripta traži dani predmet u torbi i uklanja ga. U inspectoru je oblikuje LostItemReactionEditor.

PickedUpItemReaction (DelayedReaction) je također spomenuta pri izradi Torbe. Ona dodaje novi predmet na prvo slobodno mjesto u Torbi tj. okida metodu DodajPredmet iz klase Torba. U inspectoru je oblikuje PickedUpItemReactionEditor.

MijenjanjeKamere je jedna od najvažnijih reakcija u kretanju. Nasljeđuje DelayedReaction kako bismo mogli kontrolirati vrijeme okidanja ove reakcije pomoću odgode. Sadrži dvije javne varijable, obje tipa Camera. Jedna je trenutnaKamera, a druga je iducaKamera. U metodi ImmediateReaction je opisano što točno ova reakcija radi. Prvo provjerava da li je trenutna kamera omogućena i onda onemogućuje trenutnu kameru, postavlja njenu dubinu na 0 te onemogućuje njenu komponentu PhysicsRaycaster. Iduću kameru pak omogućuje, stavlja joj dubinu na 1 i omogućuje sada njenu komponentu PhysicsRaycaster kako bi sada klikovi na ekran bili računati iz novog pogleda nove kamere. U inspectoru je oblikuje MijenjanjeKamereEditor.



Slika : MijenjanjeKamereEditor

Konačno možemo opisati i kolekciju reakcija. Ona sadrži niz reakcija svih tipova. Pošto su sve reakcije ScriptableObject, editor će ih prikazivati zasebno po tipu iako se sve nalaze u istom nizu. ReactionCollectionEditor je još jedan EditorWithSubEditors kojemu je osnovni editor ReactionEditor, a osnovni objekt Reaction. Baš kao što je vidljivo na slici 11, ovaj editor nam omogućuje dodavanje novih reakcija u kolekciju odabirom reakcije sa liste ili dovlačenjem reakcije u za to predviđeno područje.

* + 1. Interaktivni predmeti

Konačno dolazimo do dijela koji povezuje cijelu igru, a to su interaktivni predmeti. Interactable je prazni objekt koji sadrži minimalno dva prazna objekta. Međutim, svaki od njih ima različite komponente koje ih zajedno čine smislenom cjelinom. Interactable sadrži InteractionLocation objekt koji služi isključivo za određivanje lokacije interakcije sa objektom. Primjer može biti vatra s kojom želimo vršiti interakciju podalje od samog objekta kako se ne bi opekli. Interactable obavezno sadrži i objekt nazvan DefaultReaction koji sadrži komponentu kolekcija reakcija. Ovo je osnovna kolekcija koja se odvija kao osnovna, zadana kolekcija kada nema uvjeta ili nisu ispunjeni. Interactable može imati još pod objekata koji sadrže kolekcije reakcija, ovisno o tome koliko različitih kolekcija reakcija taj predmet može izazvati. Tako JezeroInteractable osim DefaultReaction objekta ima i ImaKantuReakcija. Sami Interactable objekt sadrži Collider pomoću kojeg Event Trigger očitava koliziju. Komponenta Event Trigger koristi Pointer Click te metodu klikNaPredmet iz skripte PomicanjeSvemirke kako je ranije već opisano.

Svaki predmet interakcije (Interactable) ima i dvije skripte. Prva je istog naziva: Interactable. Ova skripta sadrži tri javne varijable: lokacijaInterakcije – kojoj pridružujemo ranije spomenuti objekt InteractionLocation, niz kolekcijaUvjeta - koje kreiramo preko editora te defaultnaKolekcijaReakcija – kojoj pridružujemo ranije spomenuti objekt DefaultReaction. Stvarnu akciju obavlja InteractableEditor koji je još jedan EditorWithSubEditors. Ovaj put je pod editor KolekcijaUvjetaEditor, a ciljani objekt je KolekcijaUvjeta. Baš kao što je već rečeno, Interactable ne mora imati nikakve uvjete već se može odmah izvršavati DefaultReaction. Međutim, ako želimo dodati uvjete tj. kolekciju uvjeta, to činimo preko ovog editora.



Slika : KantaInteractable

Metoda Interact u Interactable skripti provjerava kolekcije uvjeta i reagira shodno s tim. Ako su ispunjeni uvjeti neke kolekcije uvjeta, pokreće kolekciju reakcija koja joj je dodijeljena i prekida interakciju, a u suprotnom pokreće DefaultReaction.

Druga skripta koju sadrži svaki predmet interakcije je MjenjanjePokazivaca. Kako bi igra bila jasnija, imamo tri vrste pokazivača. Ako se nalazimo na predmetu interakcije koji ima tag zid, pokazivač će se promijeniti u debeli plavi krug koji bi trebao odavati iluziju da svijetli. Ovo nam poručuje da se pokazivač nalazi na nečemu što možemo kliknuti. Kako su zidovi u igri zamišljeni kao prozirne barijere koje na klik mijenjaju kameru, izgled pokazivača je idealan indikator da uistinu klikamo na zid. U slučaju da prijeđemo mišem preko nekog predmeta interakcije koji ima tag predmet, pokazivač se promijeni u povećalo. Ovo bi trebalo odavati da se ovaj predmet može pokupiti. U svim ostavim slučajevima je pokazivač tanki plavi kružić. Osim ove skripte, za mijenjanje pokazivača je korištena klasa PromjenaPokazivaca. Ona sadrži javne varijable kojima dodjeljujemo sličice koje želimo koristiti kao pokazivač i tri metode koje postavljaju pokazivač na željenu sličicu. Ove metode pozivamo iz skripte MjenjanjePokazivaca.

* 1. Početna scena

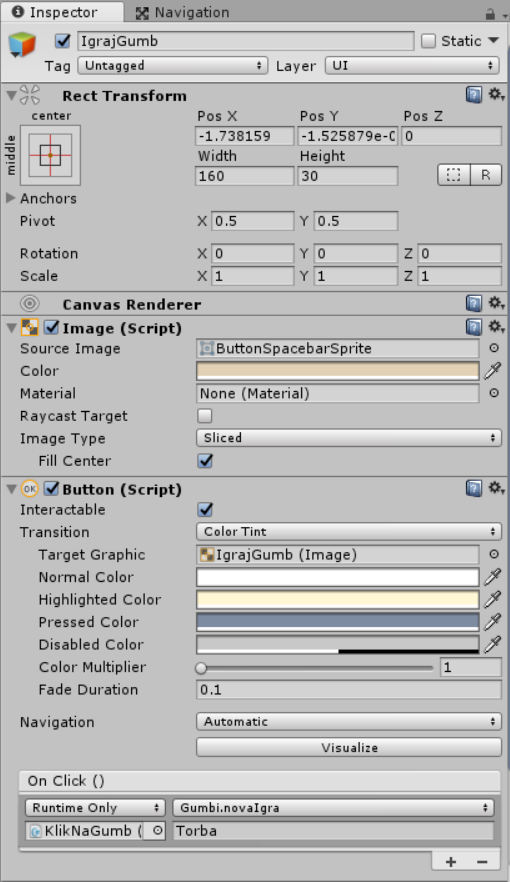
Na samom koncu je bilo potrebno oblikovati nekakav ulaz u igru, reklo bi se, početni izbornik. Nisam htjela komplicirati, pa se tako ova scena sastoji od svega dva gumba.



Slika : Pocetna scena

U scenu sam učitala jedan teren kao što se nalazi i u sceni ljeto, dodala sam Svemirku u njenom idle stanju i škrinju. Za Svemirku sam kreirala posebni animator u kojem se neprestano vrti animacija stajanja u mjestu.

Gumbe sam dodala na Canvas – platno – i pokušala odabrati boje koje se uklapaju u okoliš. Koristila sam komponentu Button s kojom se mogu mijenjati izgledi gumba kada se prijeđe preko njega ili klikne. U toj komponenti sam i definirala što se događa kada se gumb klikne. Koristila sam metode iz skripte Gumbi. Preko objekta klikNaGumb, poziva se metoda novaIgra koja učitava novu scenu.



Slika : IgrajGumb Inspector

Na isti način je kreiran i gumb za izlazak iz igre, samo što on poziva metodu izadjiIzIgre koja izlazi iz same aplikacije.

Zvuk zrikavaca je dodan kao i u sceni Torba, na glavnu kameru.

1. Izvoz igre

Kako bi igra bila dostupna svima, potrebno ju je izvesti tj. exportati u .exe format. U Unityiju je to relativno dobro razrađeno i postiže se kroz par klikova mišem. U Build Settings se odabiru scene koje su uključene u igru, platforma koja nam je ciljna te u Player Settings ikona same igre. Ovaj export stvara ne samo .exe datoteku nego i popratne sadržaje – podatke potrebne za pokretanje. U ovom radu je za ciljanu platformu odabran windowsx86\_64. Nakon ovog exporta je moguće igrati igru na bilo kojem windows računalu. Kako je korištena besplatna verzija Unity alata, na početku se prikazuje ekran Made with Unity, a prije pokretanja igre se može birati i grafičke postave.

1. Zaključak

Cijela igra je bazirana na point and click tj. usmjeri i klikni principu. Lik se može kretati klikanjem po dnu, ali bitniji su klikovi na predmete interakcije. Kako bismo prepoznali te predmete, mijenjamo izgled pokazivača: deblji krug za hodanje i povećalo za uzimanje. Predmeti interakcije mogu i ne moraju sadržavati neke uvjete za izvršavanje reakcije nad njima. Cijeli sustav uvjeta i reagiranja je pomno osmišljen i pojednostavljen korištenjem editora za kreiranje ovih elemenata. Igra je osmišljena da bude avanturistička i da pred igrača stavi izazov pronalaska blaga, uz neke manje hintove. Prikazani su osnovni načini korištenja Unity editora makar on ima brojne druge mogućnosti. Korišteni su neki napredni koncepti C# jezika poput polimorfizma i nasljeđivanja.

Na početku kad sam tek krenula sa izradom sam mislila kako ću napraviti četiri razine i kako će igra biti jako lijepa i duga. Međutim, kada sam vidjela koliko toga je potrebno da se samo napravi jedna funkcionalna scena shvatila sam kako sam previše „zagrizla“ i da ću morati smanjiti opseg svoje igre. Počela sam više cijeniti dizajnere te osobito timski rad jer mi je konstanto nedostajalo drugo mišljenje ili neka svježa ideja sa sasvim drugog pogleda. Otkrila sam puno koncepata u C#-u, a kako je ovo bio prvi put da koristim Unity mogu ponosno reći da sada poznajem osnove ovog alata. Shvatila sam kako funkcioniraju 3D igre i što je zapravo sve potrebno da bi pokrenuli i najmanju sitnicu. Jako mi se sviđa izrada igrica i htjela bih se nastaviti baviti time, a najviše sam se pronašla u samom kodiranju igre.

Na kraju bi se moglo reći kako je ovaj rad samo zagrebao površinu mogućnosti ovoga alata ali i pokazao kako se jednostavno može postići kratka ali zanimljiva igra.

1. Literatura
2. Adventure Game Tutorial, Unity tutorials 2017. Preuzeto 7.9.2017. s <https://unity3d.com/learn/tutorials/projects/adventure-game-tutorial/>
3. C# Guide, 2017. Preuzeto 26.8.2017. s <https://docs.microsoft.com/en-us/dotnet/csharp/csharp>
4. Navigation System in Unity, Unity User Manual (2017.1). Preuzeto 29.8.2017. s <https://docs.unity3d.com/Manual/nav-NavigationSystem.html>
5. Point-and-Click Game, TVTROPES, 2017. Preuzeto 13.9.2017. s <http://tvtropes.org/pmwiki/pmwiki.php/Main/PointAndClickGame>
6. Unity 2017: The world-leading creation engine for gaming. Preuzeto 26.8.2017. s <https://unity3d.com/unity>
7. Popis slika

[Slika 1: Prizori iz Syberije 4](#_Toc493085347)

[Slika 2: Syberia pokazivači 5](#_Toc493085348)

[Slika 3: Agatha Christie: Murder on the Orient Express 6](#_Toc493085349)

[Slika 4: Oaza 7](#_Toc493085350)

[Slika 5: Animator Svemirke 8](#_Toc493085351)

[Slika 6: Prijelazno stanje iz stajanja u hodanje 9](#_Toc493085352)

[Slika 7: Bake NavMesh 10](#_Toc493085353)

[Slika 8: Pomicanje Svemirke – Inspector 12](#_Toc493085354)

[Slika 9: Torba 15](#_Toc493085355)

[Slika 10: Vatra Interactable 16](#_Toc493085356)

[Slika 11: Editori (dostupno 10.9. na https://unity3d.com/learn/tutorials/projects/adventure-game-tutorial/conditions?playlist=44381) 17](#_Toc493085357)

[Slika 12:SviUvjeti 18](#_Toc493085358)

[Slika 13: Primjer korištenja SerializedPropertyExtensions (RemoveFromObjectArray) 18](#_Toc493085359)

[Slika 14: Kolekcija reakcija Ima Vodu 19](#_Toc493085360)

[Slika 15: TextReactionEditor 21](#_Toc493085361)

[Slika 16: KrajIgreEditor 21](#_Toc493085362)

[Slika 17: MijenjanjeKamereEditor 22](#_Toc493085363)

[Slika 18: KantaInteractable 23](#_Toc493085364)

[Slika 19: Pocetna scena 24](#_Toc493085365)

[Slika 20: IgrajGumb Inspector 25](#_Toc493085366)