Razvoj edukativne 2D računalne igre

Sučić, Tomislav

SVEUČILIŠTE U ZAGREBU

GRAFIČKI FAKULTET

Tomislav Sučić

Razvoj edukativne 2D računalne igre

DIPLOMSKI RAD

Zagreb, 2018.

Sveučilište u Zagrebu
Grafički fakultet

Tomislav Sučić

Razvoj edukativne 2D računalne igre

DIPLOMSKI RAD

Mentor:

Doc.dr.sc.Tibor Skala

Student:

Tomislav Sučić

Zagreb, 2018.

Izjavljujem da je ovaj rad moje samostalno djelo izvedeno uz konzultacije sa mentorom doc.dr.sc Tiborom Skalom i neposrednim voditeljem Vladimirom Cviljušcem, mag.ing.

U radu nema ilustracija niti dijelova teksta koji su bilo gdje drugdje objavljeni, a da im izvor nije točno naveden.

Zagreb, 06.09.2018. Potpis:

ZAHVALA

Zahvaljujem svom mentoru doc.dr.sc. Tiboru Skali na strpljenju i pomoći pri izradi ovog diplomskog rada. Također zahvaljujem i neposrednom voditelju asistentu Vladimiru Cviljušcu, mag.ing. koji mi je svojom susretljivosti, stručnim savjetima i znanjem pomogao u izradi ovog diplomskog rada. Zahvaljujem mu i na trudu i vremenu koje je uložio da bih ja napravio ovaj diplomski rad.

Sažetak

Učenje novih stvari nekada zna biti zamorno i dosadno. Pogotovo kada se mora učiti zbog ocjena, a ne iz zabave i želje za znanjem. Ovaj rad će se baviti razvojem edukativne višeplatformne 2D računalne igre namijenjene edukaciji ljudi kroz zabavu i natjecanje. U radu će biti opisani programi iz Adobeovog paketa korišteni za razvijanje vizualnih elemenata igre. Adobe Photoshop i Adobe Illustrator za razvijanje grafičkih elemenata, te Adobe Flash Builder razvojno okruženje (engl. framework) za kreiranje igrica i aplikacija koristeći ActionScript 3.0 objektno orijentirano programiranje (OOP). U radu će se istražiti postojeće računalne igre u kojima se kojima se do cilja dolazi sakupljanjem točnih odgovora te strateške igre bazirane na osvajanju mape svijeta. Nakon teoretskog dijela detaljno će se izložiti proces razvoja igre, od osmišljavanja pravila dovoljno jednostavnih da svima budu jasna, prvih skica pa sve do razvoja programskog koda za računalnu igru. Konačni praktični zadatak bit će realiziran u vidu izrade algoritma koji će poslužiti za razvoj igre. U zaključku će biti opisane prednosti razvijene računalne igre koja spaja kvizaški žanr sa strateškim tako učinkovito da će ljudi imati osjećaj zabave dok će istovremeno učiti nove stvari.

Ključne riječi:

- 2D višeplatformna računalna igra
- Učenje
- Zabava
- ActionScript 3.0
- XML
- SVG

Sadržaj

1. UVOD

U današnje vrijeme mnogo ljudi, i djece i odraslih, ima problema s učenjem novih stvari. Uglavnom im se ne da sjediti i učit jer misle da mogu pametnije iskoristiti to vrijeme. Ili su umorni nakon što obave sve ostale obveze u danu i na kraju dana najradije se udobno smjeste u kauč i odgledaju film ili odigraju igru na računalu. Istraživanja su pokazala da dosta djece ne zna pokazati osnovne stvari na mapi svijeta. U Americi djeca nisu znala pokazati gdje se nalazi Europa kada su ih to pitali. Kada bi im se pružila prilika da uče nove stvari kroz zabavu takvih stvari bi bilo puno manje. U današnjem digitalnom svijetu praktički ne postoji kućanstvo koje nema barem jedno računalo ili čovjek koji se ne koristi pametnim višefunkcionalnom telefonom. Djeca se od malih nogu uče koristiti tehnološki naprednim uređajima te veliki dio vremena ljudi provode igrajući računalne igre na raznim platformama. Upravo to je prilika da se ljude dodatno educira bez da im to ne predstavlja neki problem. Umjesto da gube vrijeme na pametnim telefonima ljudi bi mogli dobivati znanje igrajući igre protiv drugih ljudi.

Interaktivna 2D Edukativna igra Trivia Conquest opisana u ovom diplomskom radu bazirana je na standardnom modelu kviza. Igrač dobije pitanje s četiri ponuđena odgovora. Jedan od ta četiri odgovora je točan. Kada odgovori na sva pitanja igrač dobiva rezultate odgovaranja. Novost u ovoj igri je što igrač odabire države na čija pitanja želi odgovarati simulirajući tako napad na tu državu. Broj napada predstavlja broj života koje korisnik ima. Time se dobilo na težini igre. U radu je opisano razvojno okruženje u kojem je igra razvijena, te cijeli put od ideje do razvitka programskog koda.

2. RAZVOJNO OKRUŽENJE

2.1. Adobe Flash builder

Adobe flash builder je integrirano razvojno okruženje(engl. Integrated Development Environment – IDE) razvijen na Eclipse platformi i služi za brzi razvoj bogatih web aplikacija i nativnih računalnih i mobilnih aplikacija. Flash builder ima ugrađene uređivače teksta za MXML i ActionScript. Omogućava nam brže kodiranje pomoću raznih alata kao što su: javljanje pogrešaka, završavanje naredbi, kodovi u boji za lakše snalaženje, interaktivno debugiranje korak po korak koje omogućava programerima da lakše pronađu greške u kodu, te izbornike za kontroliranje potrošnje memorije.[1]

2.2. Adobe Illustrator

Adobe Illustrator je grafički računalni program razvijen od strane Adobe korporacije u svrhu obrade vektorske grafike. Vektorska grafika se temelji na principima geometrije dok se rasterska grafika temelji na pikselima. Vektorska grafika se koristi za izradu logo-a, fontova, navigacijskih ikona na web stranicama, pakiranjima za proizvode, karta, grafikona, postera itd.[2]

1986 inžinjer Mike Schuster je dobio zadatak da napravi program za crtanje koji je jednostavan za upotebu,a koristi post-script jezik. 1987 godine puštena je prva verzija Adobe-ovog illustratora za Macintosh. 1989 godine puštena je verzija Illustratora i za Microsoft. Jedan od najbitnijih opcija bio je pen tool koji je omogućavao korisniku da crta glatke linije.[3]

2.3. Adobe Photoshop

Adobe photoshop je grafički računalni program koji služi za obradu slike. Photoshop koristi slojeve kako bi omogućio dubinu i fleksibilnost u dizajnu i obradi grafičkih materijala. Također, pruža mnoge moćne alate koji kada se kombiniraju ništa ne ostavljaju nemogućim.

Originalno je razvijen 1987 od strane Thomasa i Johna Knolla. Thomas je napisao program za prikaz slika na svom Macintosh Plusu i nazvao ga Display. Njegov brat John ga je nagovorio da od toga napravi cijeli program za obradu

slika. Počeli su surađivati te su program nazvali Photoshop jer je ime imagePro već bilo zauzeto. 1988. godine licencu je kupio Adobe te počeo s distribucijom programa. Photoshop 1.0 je pušten u prodaju u veljači 1990. i bio je samo za Macintosh. Svaka nova verzija je bila veliko poboljšanje u odnosu na prijašnju verziju te je program brzo postao standard za obradu slike.Posljednja verzija je CC 2018(19.1.6) izdana u srpnju 2018 godine. [4]

2.4. Action script 3.0

ActionScript je objektno orijentirani programski jezik originalno razvijen od strane Macromedie. ActionScript je dobro organizirani jezik koji puno svoje metodologije i sintakse dijeli s ostalim objektno orijentiranim jezicima tako da se iskusni programeri lagano mogu prebaciti na programiranje u njemu. Flash sadržaj se može raditi i bez ActionScript 3.0 jezika, ali to će onda biti samo animacije.[5] ActionScript je neophodan kada se želi napraviti dinamične, responzivne, aplikacije za višekratnu upotrebu i prilagodljive aplikacije. Sa ActionScriptom se mogu napraviti: igre za više igrača, navigacije, MP3 svirači, komunikacija sa Javascriptom, parsiranje XML objekata, ploče s porukama te mnogo drugih zanimljivih stvari.[6]

Objektno orijentirano programiranje je način programiranja u kojem je težište razvitka aplikacije na tome da se konstruiraju manji objekti koji međusobno razmjenjuju poruke. Osnovne vrijednosti objektno orijentiranog programskog jezika su klase, objekti, apstrakcija, nasljeđivanje, učahurivanje ili enkapsulacija i višeobličje ili polimorfizam.

Objekti su gradivni blokovi, a klase su konstruktori koji ih opisuju. Klasa je nacrt za izradu objekta. Svaki objekt je načinjen iz jedne klase. U OOP-u se mogu povući paralele sa živim svijetom. Ako uzmemo npr. Bicikl kao objekt onda je nacrt za sastavljanje bicikla klasa.[5]

```
public class Bicikl {
    public String ime;
    public int brojKotaca;
    public int brojVolana;
    public int brojSjedala;

}
```

Isječak ispisa koda 1: Primjer klase

```
Public var MountainBike:Bicikl = new Bicikl("Mountain bike", 2, 1, 1);
```

Isječak ispisa koda 2: Primjer objekta

Enkapsulacija ili učahurivanje je odvajanje sučelja objekta i same implementacije. Drugi objekti moraju znati samo što objekt s kojim komuniciraju radi, ali ne i kako to radi. To se postiže postavljanjem varijabli unutar klasa na private kako bi ih samo ta klasa mogla mijenjati.

Često se dogodi da jedna klasa ima dosta sličnosti s nekom drugom klasom. U tom slučaju bilo bi suvišno napraviti skoro pa istu klasu sa nekom malom izmjenom. Umjesto toga nova klasa može naslijediti klasu koja je već stvorena prije te se u nju mogu dodati male izmjene koje su potrebne samo toj klasi. To svojstvo se naziva nasljeđivanje.

Polimorfizam je svojstvo objektno orijentiranog programiranja da se podklase tretiraju kao klase iz koje su produžene. Npr. Napravi se klasa MountainBike koja je podklasa klase Bicikl. Ako se treba očitati brzina i jedne i druge klase može se napisati funkcija:

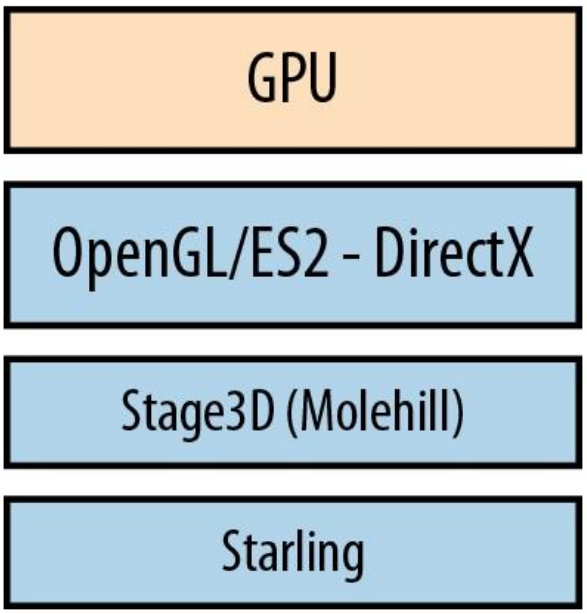
```
pokaziBrzinu(Bicikl, MountainBike);

private function pokaziBrzinu(Bicikl1:Bicikl, Bicikl2:Bicikl):void{
               Bicikl1.brzina();
               Bicikl2.brzina();

}
```

Isječak ispisa koda 3: Primjer polimorfizma

Svojstvom polimorfizma MountainBike će biti očitan kao klasa Bicikl.[7]

2.5. Starling framework i Feathersui

Starling je AS3 okvir otvorenog koda koji se koristi uglavnom za razvijanje 2D igara, ali može biti korišten za izradu bilo kakve grafičke aplikacije. Zahvaljujući Adobe AIR runtime-u aplikacije bazirane na Starlingu mogu se pokrenuti na svim mobilnim i stolnim platformama. Starling oponaša klasičnu arhitekturu display liste od Adobe AIR-a ili Flasha, ali pruža mnogo bolje performanse jer su svi objekti renderirani direktno od strane grafičke procesne jedinice(GPU). Skriva unutrašnjost Stage3D-a od developera, ali omogućuje da joj se lagano pristupi ako je to potrebno.[8] Potreba za Starlingom se javila jer je Flash primarno dizajniran za stolna računala koja imaju jake centralne procesne jedinice(CPU), ali primitivan grafički hardware. Današnji mobilni uređaji rade upravo suprotno – imaju slabe CPU-ove(zbog štednje baterije), ali dosta razvijene grafičke čipove. Adobe je bio svjestan ovog problema te je 2011. godine razvio nižerazredni grafički API Stage3D. U osnovi Stage3D je samo bio omotač oko OpenGL i DirectX-a. Taj API je omogućavao developerima da iskoriste snagu grafičkih čipova. Problem je bio što nije omogućavao developerima jednostavnu upotrebu za razvijanje igara i aplikacija. Starling koristi Stage3D, ali je i dovoljno jednostavan da ga koriste developeri. Jedan od ciljeva autora Starling okvira je da bude što lakši i jednostavniji za upotrebu. Cilj je bio da developeri shvate što se događa iza scene jer se samo tako kod može proširivati i modificirati dok ne postane potpuno odgovarajući za određene potrebe. Najpoznatije igre izrađene sa Starlingom su Angry Birds i Incredipede.[9]



Slika 1: Hijerarhija Starlinga

(izvor:https://www.safaribooksonline.com/library/view/introducing-starling/9781449320904/ch01s04.html**)**

FeathersUI je biblioteka koja proširuje Starling okvir. Također je besplatna i otvorenog koda. Omogućuje korištenje velikog broja unaprijed određenih komponenti za korisničko sučelje(gumbi, ladice, liste, tabovi i mnogo drugih). Komponente iz feathersUI biblioteke su responzivne što uvelike olakšava razvijanje cross-platform aplikacija.[10]

2.6. XML sintaksa

XML je kratica za Extensible Markup Language odnosno jezik za označavanje podataka. Zamišljen je kao jezik koji će biti jednako dobro čitljiv ljudima kao i računalnim programima, a uz to je potpuno neovisan o računalnoj platformi i operacijskom sustavu na kojem se nalazi. XML dokument se najčešće sprema kao tekstualna datoteka koja sadrži sadržaj i oznake. Sadržaj se označava sa dvije oznake – početnom i završnom.[11]

```
<oznakaElementa> Sadržaj Elementa </oznakaElementa>
```

Isječak ispisa koda 4: Primjer elementa

XML spada u grupu jezika za označavanje. Ostali jezici u toj skupini su: HTML, SGML, RTF. Za razliku od Html-a koji služi za prikaz podataka s fokusom na to kako podaci izgledaju, XML je dizajniran za prijenos informacija s fokusom na sadržaj. U HTML-u su oznake već prije definirane i one se moraju koristiti. Korisnik ne može pisati sam svoje oznake. Kada se HTML želi proširiti novim oznakama mora se mijenjati cijeli standard što ga čini nepraktičnim.

Standard generalized markup language(SGML) 1986. Godine objavljen je kao međunarodna norma ISO 8879. SGML-om su autori pokušali stvoriti jezik dovoljno formaliziran da može jamčiti vjernost dokumenta izvorniku, dovoljno strukturiran da se može nositi s kompleksnim dokumentima te dovoljno otvoren da se može nositi s velikom količinom podataka. Problem SGML-a je što je bio ogroman. Autori su pokušali pokriti svaku moguću primjenu jezika te je nastali proizvod bio preopširan i zbog toga skup u upotrebi. Zbog svoje glomaznosti nije bio pogodan za korištenje unutar web preglednika. Bilo je potrebno napraviti jezik koji će biti dovoljno mali i jednostavan za korištenje unutar web preglednika, a s druge strane dovoljno prilagodljiv da se može proširivati korisničkim oznakama.[12] Izrade takvog jezika prihvatio se World Wide Web Consortium. Željeli su razviti jezik koji će objediniti jednostavnost HTML-a i izražajnu snagu SGML-a. Na početku su razvili 10 ciljeva kojih su se pokušali držati u razvoju:

1. XML mora biti izravno primjenjiv preko interneta.
2. XML mora podržavati širok spektar primjena.
3. XML mora biti kompatibilan s SGML-om.
4. Mora biti lako pisati programe koji procesiraju (parsiraju) XML dokumente.
5. Broj opcionalnih "feature-a" u XML-u mora biti apsolutno minimalan, u idealnom slučaju jednak nuli.
6. XML dokumenti moraju biti čitljivi ljudima, te u razumnoj mjeri jednostavni

7. Standard mora biti specificiran što prije
8. Dizajn XML-a mora biti formalan i precizan
9. Kreiranje XML dokumenata mora biti jednostavno
10. Sažetost kod označavanja dokumenta XML-om je od minimalnog značaja [13]

Svaki XML dokument se sastoji od 2 dijela. Prvi je prolog ili zaglavlje.

```
<?xml version="1.0"  encoding="UTF-8"?>
```

Isječak ispisa koda 5: XML zaglavlje

U zaglavlju se navodi verzija XML preporuke prema čijim standardima je dokument napravljen te kodna stranica. Kodna stranica služi kako bi programi koji barataju XML-om znali koje znakove mogu očekivati u dokumentu. Drugi dio XML dokumenta je sadržaj.

```
                <item question="Who was seen as a model of
Christian resistance against the Ottoman Muslims in the middle
ages?">
                    <answer>Gjergj Arianiti</answer>
                    <answer correct="true">Skanderberg</answer>
                    <answer>Hamza Kastrioti</answer>
                    <answer>Gjon Kastrioti II </answer>
                </item>
```

Isječak ispisa koda 6: XML sadržaj

2.7. SVG sintaksa

SVG je jezik za prikazivanje dvodimenzionalne vektorske grafike, bilo statične ili animirane. To je otvoreni standard stvoren od strane World Wide Web Consortiuma. SVG je zapravo samo XML dokument u kojem su opisane točke, linije, krivulje, boje i tekst. Ljudi lagano mogu pročitatit SVG dokument. Za definiranje jednostavnijih oblika dovoljni su elementi poput pravokutnika, kruga, elipse, višekutnika i linija.[14]

```
<svg width="400" height="180">
  <rect x="50" y="20" width="150" height="150"
  style="fill:blue;stroke:pink;stroke-width:5;opacity:0.5" />
  <circle cx="150" cy="150" r="40" stroke="black" stroke-
width="3" fill="red" />
  <ellipse cx="200" cy="280" rx="100" ry="50"
  style="fill:yellow;stroke:purple;stroke-width:2" />
```

Isječak ispisa koda 7: Svg elementi

Kod jednostavnijih oblika dovoljno je odrediti početnu točku i veličinu oblika. Kod složenijih oblika koristi se path element. Sljedeće naredbe su moguće za path element:

- M = pomakni
- L = linija
- H = horizontalna linija
- V = vertikalna linija
- C = zavini
- S = glatko svini
- Q = kvadratična Bézierova krivulja
- T = glatka kvadratična Bézierova krivulja
- A = eliptični luk
- Z = završi putanju [15]

```
<g id="Switzerland">

	<path fill="#E2E2E2" stroke="#01002A" stroke-width="0.25" stroke-linecap="round" stroke-linejoin="round" d="M337.053,389.83
	c1.328,2.54-1.883-0.373-0.84-0.705c1.046-0.333-1.371-2.87-2.543-2.903c-1.614-3.881,1.659-7.105-0.396-5.019 c-1.774,1-1.578-1.143-0.876-0.964c-0.616,0.34-2.882,1.303-3.565,0c-2.004,0.472-7.43,2.429-10.334,2.178 c-1.2-1.528-2-1.308-2.338-1.575c-2.555,2.291-2.862,0.611-4.73,3.322c0.755,0.085-1.278,2.226-0.685,2.673 c-4.558,4.651-5.124,6.431-8.066,10.286c0.373,0.215,0.228,0.484,0.039,0.77c1.297-0.396,1.813-0.569,3.092-1.218 c-0.319-1.66,1.44-1.621,4.19-1.05c1.84,0.384,0.903,3.997,2,4.745c0.91,0.62,4.916-0.106,7.833-1.229 c2.858-1.099,0.978-3.389,2.956-3.938c2.918-0.813,3.383,2.963,7.525,3.397c-0.639,0.925-0.771,0.688-0.639,0.925 c0.93,0.931,0.464-2.354,0.639-0.925c1.022-1.271,2.649-3.828,4.713-4.771c0.888,3.595,4.514,0.997,6.753,2.829
	C339.114,393.234,334.838,390.071,337.053,389.83z"/>

.
```

Isječak ispisa koda 8: SVG path element

Svim elementima moguće je odrediti boju obruba te boju unutrašnjosti. S obzirom da je SVG vektorska grafika, a ne rasterska, nije ovisna o rezoluciji te se može povećavati koliko god treba bez gubitka kvalitete.

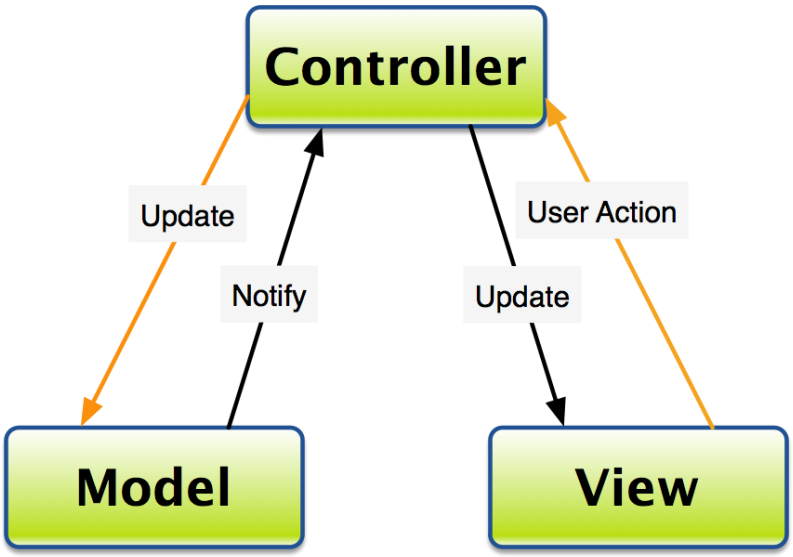
2.8. MVC obrazac

MVC je najčešći obrazac u svijetu programiranja web aplikacija. MVC obrazac dijeli aplikaciju u 3 komponente: model, pogled i kontroler.

Model – Odgovoran je za spajanje na bazu podataka, rad sa podacima, izvršavanje upita, implementaciju poslovnih pravila itd. Model se ne može pozvati direktno već to radi kontroler. Model odgovara na zahtjeve koji stižu iz kontrolera, priprema i obrađuje podatke, te ih vraća kontroleru koji ih onda šalje do pogleda gdje ih vidi krajnji korisnik.

Kontroler – Zadužen je za upravljanje akcijama. Kada korisnik šalje zahtjev sustavu preko pogleda, kontroler to prosljeđuje prema modelu. Nakon što model obradi zahtjev vraća ga kontroleru, te model sukladno tome mijenja pogled koji korisnik vidi.

Pogled – Zadužen je za prikaz podataka korisniku. Nakon što model obradi podatke i pošalje ih kontroleru, kontroler to šalje pogledu i tamo se podaci prikažu korisniku. Pogled je vidljiv od strane korisnika dok model i kontroler nisu.

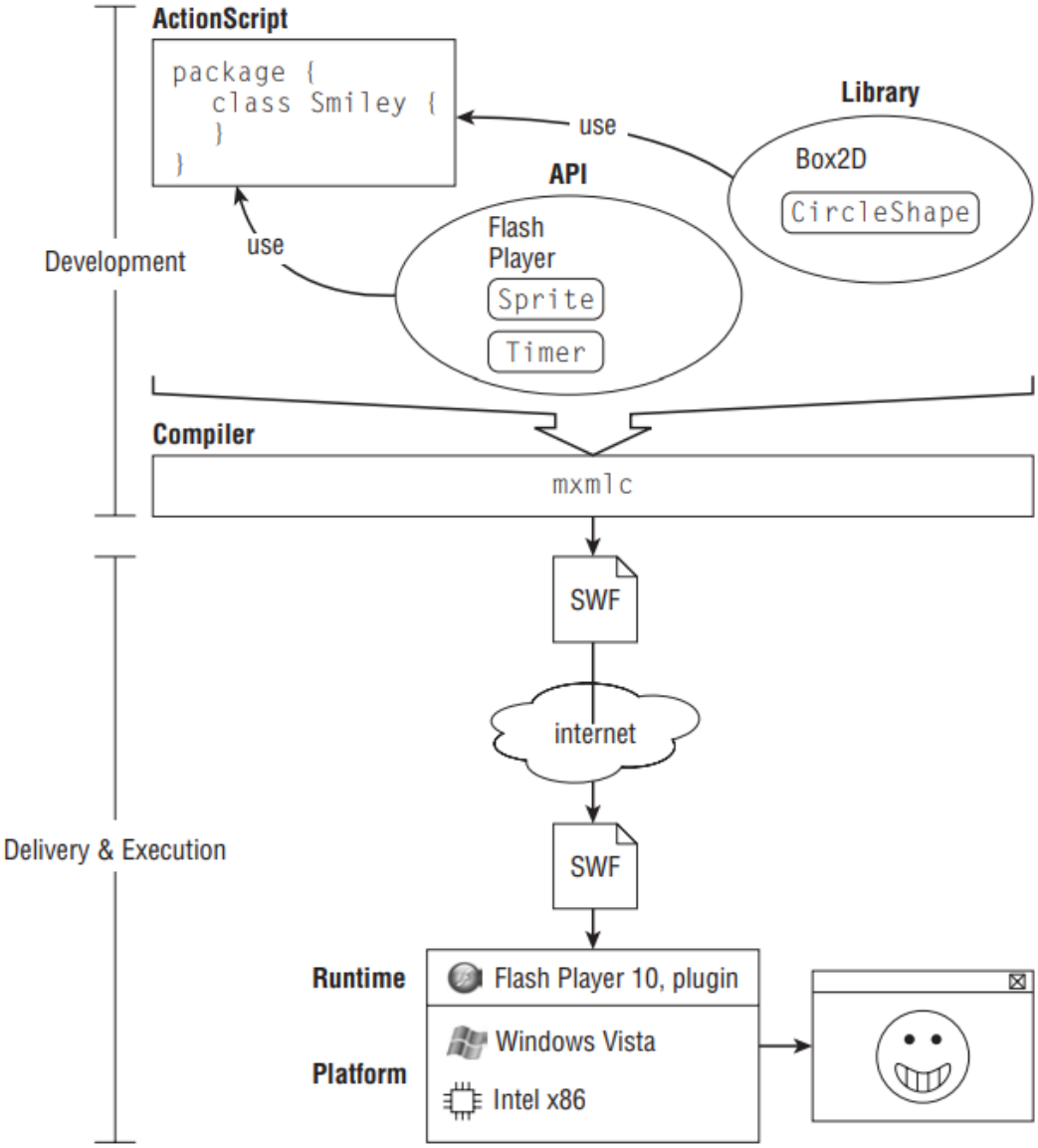


Slika 2: MVC Arhitektura

(izvor: https://cupsofcocoa.files.wordpress.com/2011/08/mvc-diagram1.png)

Prednosti korištenja MVC tehnologije su lakše testiranje i nadograđivanje aplikacije. Omogućen je paralelan razvoj sve 3 komponente(model, pogled i kontroler). Lagano se može mijenjati jedan dio aplikacije bez da se nešto promijeni u ostalim dijelovima. [16]

2.9. Od programera do korisnika u flash okruženju



Slika 3: Od programera do korisnika

(izvor: http://www.sccg.sk/~dadova/wega/ActionScript.pdf)

Programeri pišu kod u programskom jeziku(ActionScript 3.0 u ovom slučaju) koristeći neki od uređivača tekstova ili integriranih razvojnih okolina(Flash builder 4.7). Koriste se kompileri kako bi pretvorili taj kod u datoteku koja se može izvršavati u ciljanom okruženju. Korisnik skine ili nadogradi flash player. Zatim otvori internet preglednik te ode na stranicu sa flash contentom. Iza

scene preglednik skine swf datoteku sa interneta te pokrene plug-in verziju flash playera, te onda flash player pokrene swf datoteku.[5]

3. RAZVOJ APLIKACIJE I POSTUPAK IZVEDBE

3.1. Ideja Aplikacije

Igra opisana u ovom radu zamišljena je kao zabavna interaktivna igra koja bi ljudima omogućila da se u isto vrijeme zabavljaju i uče. S obzirom na to da su pitanja grupirana po kategorijama za svaku državu, igrači znaju koja pitanja mogu očekivati napadom jedne države. Pitanja za svaku državu su grupirana u 5 kategorija: povijest, zemljopis, jučer-danas-sutra, sport i poznate osobe. Napadom na državu igrač dobiva po jedno pitanje iz svake kategorije koje dolaze jedno za drugim. Da bi osvojio državu mora odgovoriti na 3 od 5 pitanja točno. Generator za pitanja je kodiran tako da se pitanja stalno miješaju unutar kategorija tako da igrač nikada ne dobiva isti set pitanja. Isto tako konstantno se miješaju i odgovori u pitanjima pa točan odgovor nikada nije na istom mjestu. Igra je napravljena tako da se lagano mogu dodavati nove mape sa svojim setom pitanja.

Na početku svake igre igrač dobije 5 napada. Svakim neuspjelim napadom na neku državu igrač gubi jedan napad. Svakim uspješnim napadom na državu igrač dobiva jedan napad. Time se razvija i taktičko razmišljanje svakog igrača te ga se prisiljava da prvo napada države o kojima misli da više zna kako bi prikupio što više napada. Osvajanjem svake države na mapi igrač je pobijedio igru, ali ako broj napada koje igrač ima dođe do 0 igrač je izgubio.

3.2. Istraživanje tržišta

Na tržištu postoji mnogo aplikacija za kviz igre. Većina od tih aplikacija ima preko milijun downloada što pokazuje da je potražnja za takvim igrama velika.

Trivia Crack – 100 milijuna downloada

Quiz up – 10 milijuna downloada

Trivia crack kingdoms – 5 milijuna downloada

Trivia 360 – 1 milijun downloada

Trivia quiz – 1 milijun downloada

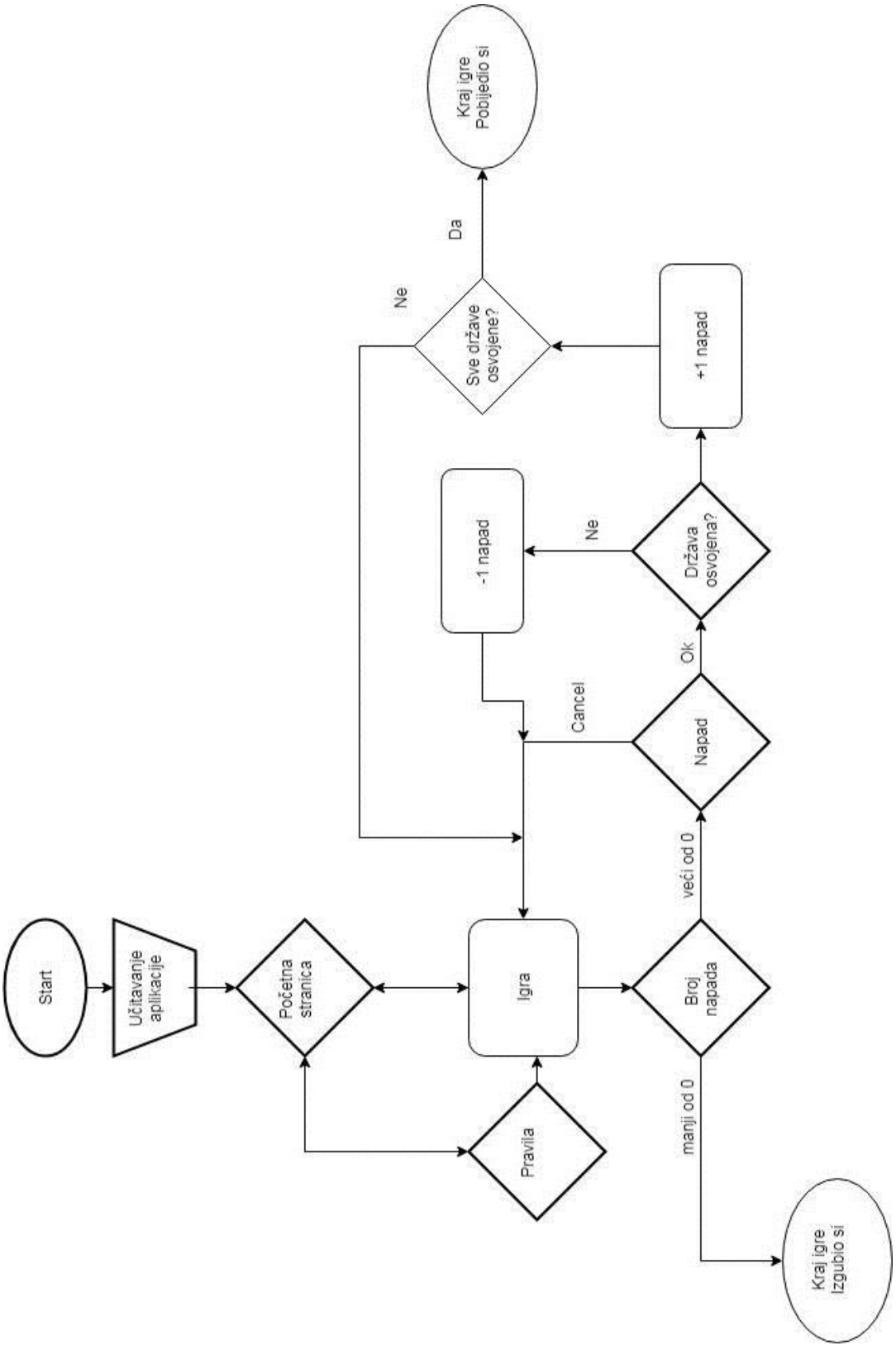
Flags and Capitals od the world quiz – 1 milijun downloada

World Geography – 5 milijuna downloada

Nijedna od navedenih igara se ne bazira na koncentriranom učenju o pojedinoj zemlji te nema sustav napadanja sličan riziku kao igra opisana u ovom diplomskom radu. Postoje kviz igre koje se igraju na kartama svijeta, ali one se fokusiraju uglavnom na znanje o izgledu zastave, te geografskom položaju pojedine države.

3.3. Dijagram toka aplikacije

Dijagram toka je grafički prikaz algoritma. To je pomoćno sredstvo kojim se vizualizira zadatak pa on postaje pregledniji. Sastoji se od niza jednostavnih geometrijskih tijela spojenih smjerenim crtama. Dijagram toka je najjednostavnji način kojim možemo prikazati kako će se aplikacija ponašati kada bude gotova.



Slika 4: Dijagram toka

3.4. Wireframe aplikacije

Nakon izrade dijagrama toka radi se wireframe. Wireframe je pojednostavljeni prikaz dizajna proizvoda. Wireframe se radi prije nego se krene u razvitak aplikacije kako bi se jednostavno prikazao sadržaj aplikacije i raspored elemenata u korisničkom sučelju. Sadrži linije, pravokutnike i sive palete boja. Preko wireframe se može najlakše testirati korisničko iskustvo i vidjeti snalaze li se korisnici u zamišljenom načinu rada aplikacije. Ako nešto nije dobro napravljeno ili testiranja pokažu da se ljudi ne snalaze u zamišljenom načinu rada aplikacije to se u wireframe-u može vrlo lako promijeniti. Postoje mnogo online alata za izradu wireframe-ova. Poznatiji od njih su UXpin, wireframe.cc, proto.io, invision. Za izradu statičnih skica za ovaj diplomski korišten je online alat Mockflow.

Pri pokretanju aplikacije korisnik vidi zaslon za učitavanje. Podaci se učitavaju pri pokretanju aplikacije kako bi se što lakše i brže kretalo unutar aplikacije. Nakon učitavanja korisnik dolazi na početni zaslon. Od tamo može izabrati zaslon sa pravilima jer to svim učine prije nego počnu igrati neku igru, ili može pokrenuti igru. Ako korisnik odabere zaslon sa pravilima na njemu može pročitati pravila, te od tamo može pokrenuti igru ili se vratiti na početni zaslon. Na zaslonu sa igrom korisnik vidi mapu, koliko napada mu je ostalo, te kratki tutorial. Pritiskom na državu na mapi korisniku se pojavljuje zaslon sa pitanjima. U svakom napadu korisnik odgovara na 5 pitanja. Nakon što odgovori na njih pojavljuje mu se zaslon sa rezultatima odgovaranja na pitanja u tom krugu. Nakon što je korisnik osvojio sve države ili nakon što je potrošio sve napade pojavljuje mu se zaslon za kraj igre.

Slika 5: Wireframe aplikacije

3.5. Dizajn aplikacije

3.5.1. Odabir palete boja

Boja je jako važna, kao i u svakodnevnom životu, tako i u dizajniranju aplikacija. Pravilnim odabirom boja privlači se više ljudi da koriste aplikaciju. Svaka boja nosi svoje određeno značenje te pobuđuje određene emocije kod ljudi. Za našu aplikaciju odabrane su smeđa i tamno plava boja. Smeđa boja je odabrana jer podsjeća ljude na povijest, a povijest je majka znanja. Plava boja predstavlja sigurnost, stabilnost, smirenost.[17]Često se koristi u poduzećima i bankama kako bi se napravio osjećaj sigurnosti i povjerenja u brend. Plava boja je boja

broj 1 i kod muškaraca i kod žena. Boje država dok nisu osvojene su sive jer je to neutralna boja.

3.5.2. Tipografija

Odabirom dobre tipografije uvelike se pridonosi estetici i izričaju aplikacije za koju se ta tipografija koristi. Izbor tipografije, u isto vrijeme može privući, ali i odbiti krajnjeg korisnika od aplikacije. Ne preporuča se miješati previše fontova za jednu aplikaciju. Kod ove igre za naslovni font korišten je JSL ancient. JSL ancient je baziran na fontovima koji su se koristili u Engleskoj u sedamnaestom stoljeću te time pridonosi povijesnom, starinskom izgledu aplikacije. U kombinaciji s njim korišten je font Roboto. Roboto je sans-serifni font razvijen od strane Google-a kao font za android sustave.

3.5.3. Izgled aplikacije

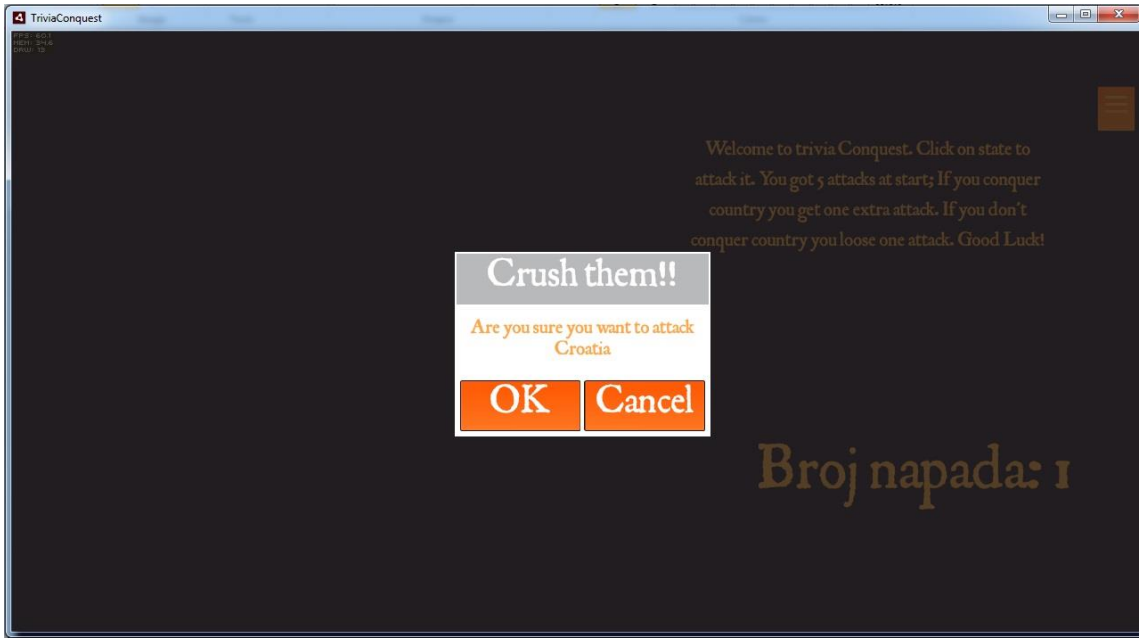
Na slikama je prikazan konačni izgled aplikacije:

Slika 6: Početni zaslon

Slika 7: Zaslon sa pravilima

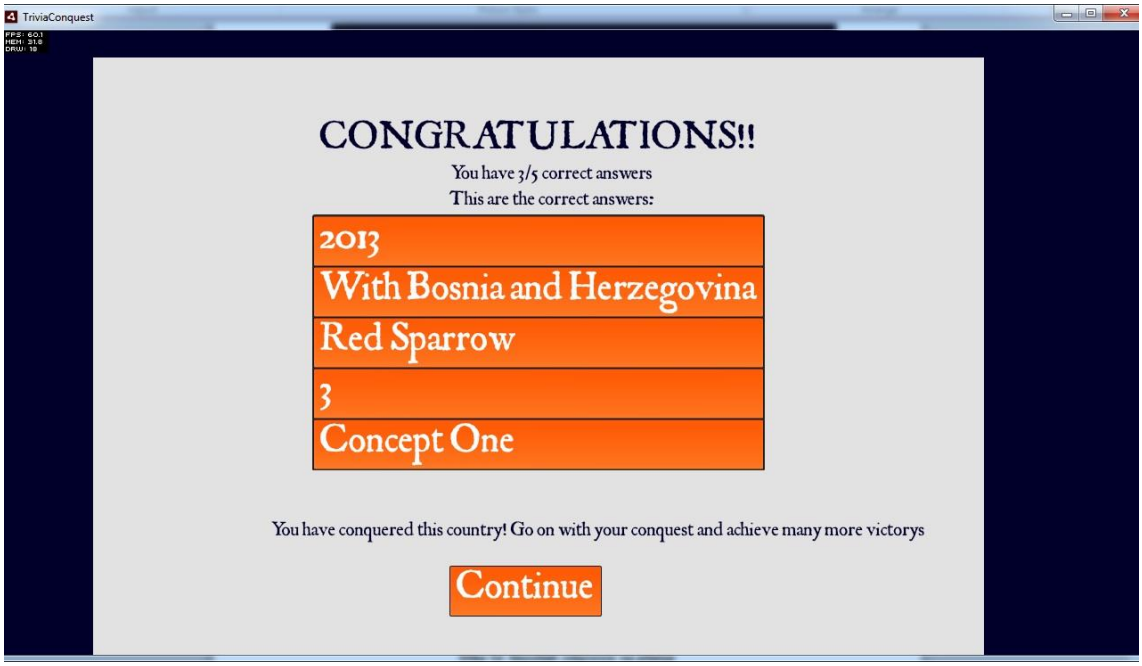
Slika 8: Zaslon sa igrom

Slika 9: Hover zaslon

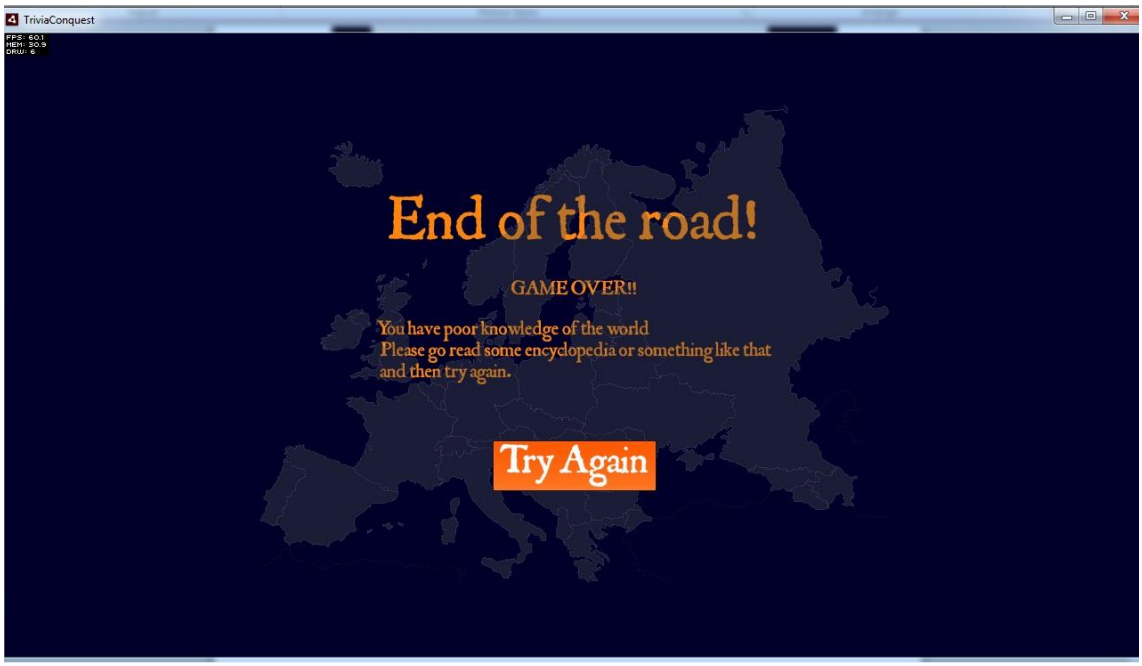


Slika 10: Potvrda napada

Slika 11: Zaslon sa pitanjima



Slika 12: Rezultati odgovora na pitanja



Slika 13: Završetak igre

3.6. Uređivanje mape za aplikaciju

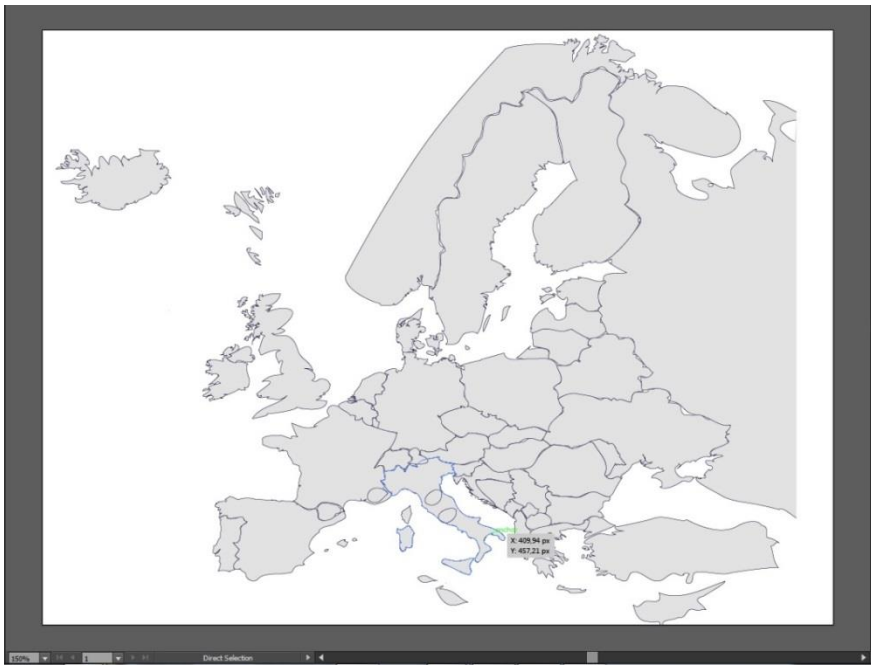
Početna mapa Europe je preuzeta sa ove stranice:

https://freevectormaps.com/world-maps/europe/WRLD-EU-01-0002

Nakon preuzimanja mape otvoren je ai file u ilustratoru.

Slika 14: Preuzeta ai mapa Europe

Za potrebe aplikacije opisane u ovom diplomskom mapa je preuređena. Države San Marino, Vatikan, Lichtenstein, Andora, Monaco, Cipar i Malta su povećane kako bi igrači mogli kliknuti na njih. Zbog brzine učitavanja mape u igri smanjen je broj točaka kojima su države opisane te je zbog toga došlo do neznatne promjene u njihovom izgledu. Mapa je spremljena u svg formatu.



Slika 15: Uređena mapa za igru u svg formatu

3.7. Izrada xml dokumenta sa pitanjima

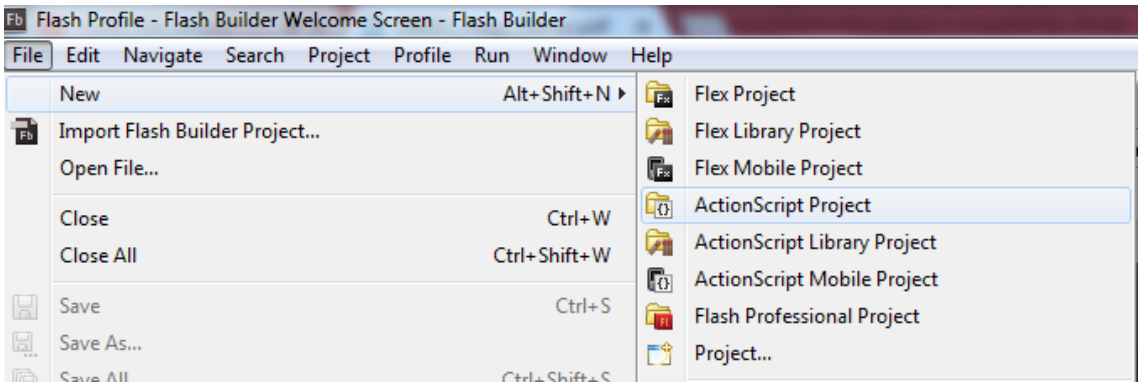
Sva pitanja potrebna za ovu aplikaciju su spremljena u xml datoteci. Pitanja su kategorizirana po državama te po kategorijama. Prije svake države napisan je komentar kako bi se znalo o kojoj državi su pitanja koja slijede. Element naziva country sadrži ime države te podatak za putanju do slike zastave. Slika zastave se pokaže u igri kada se mišem prijeđe preko države. Pitanja su ugniježđena u kategorije. Element Items sadrži ime kategorije. Element item sadrži jedan atribut question u kojem se nalazi tekst pitanja. Svako pitanje ima 4 ponuđena odgovora. Točan odgovor je označen sa correct = „true“.

```
<country name = "Albania" flag="data/flags/Albania.png">
        <!-- Albania History -->
          <items category = "history">
                <item question = "In 1939 Albania was invaded by another country in the region. Which one??">
                        <answer>Ottoman Empire</answer>
                        <answer>Serbia</answer>
                        <answer correct="true">Italy</answer>
                        <answer>Germany</answer>
                </item>
                <item question ="When did Albania gain its independence?">
                        <answer>1907</answer>
                        <answer>1912</answer>
                        <answer correct="true">1912</answer>
                        <answer>1904</answer>
                </item>
                <item question ="For nearly four centuries Albania was part of an Empire. Which?">
                        <answer>Serbian</answer>
                        <answer>Macedonian</answer>
                        <answer correct="true">Ottoman</answer>
                        <answer>Croatian</answer>
                </item>
                <item question ="When was the last Albanian revolt before WW1?">
                        <answer>1915 </answer>
                        <answer>1914 </answer>
                        <answer correct="true">1912</answer>
                        <answer>1913 </answer>
                </item>
                <item question = "The first Albanian Republic lasted for many years?">
                        <answer>4</answer>
                        <answer>7</answer>
                        <answer correct="true">3</answer>
                        <answer>6</answer>
                </item>
          </items>
```

Isječak ispisa koda 9: Primjer pitanja o albanskoj povijesti

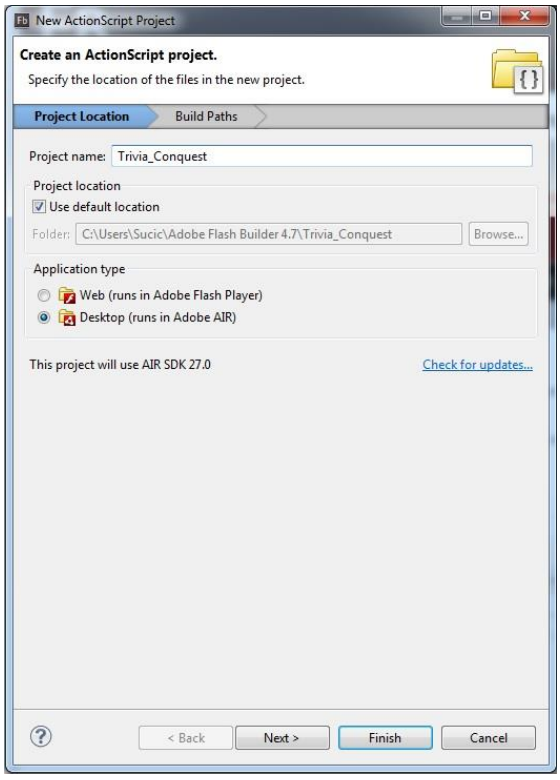
3.8. Stvaranje projekta u Flash builderu 4.7

Prije početka stvaranja projekta u Flash builderu potrebno je skinuti sve potrebne okvire i biblioteke – Starling, Feathers, SVGRenderer. Nakon što je sve skinuto spremimo ih u datoteku libs koja se nalazi u datoteci sa imenom diplomskog rada. U flash builderu se izabere opcija new ActionScript project koja se nalazi pod File => new.

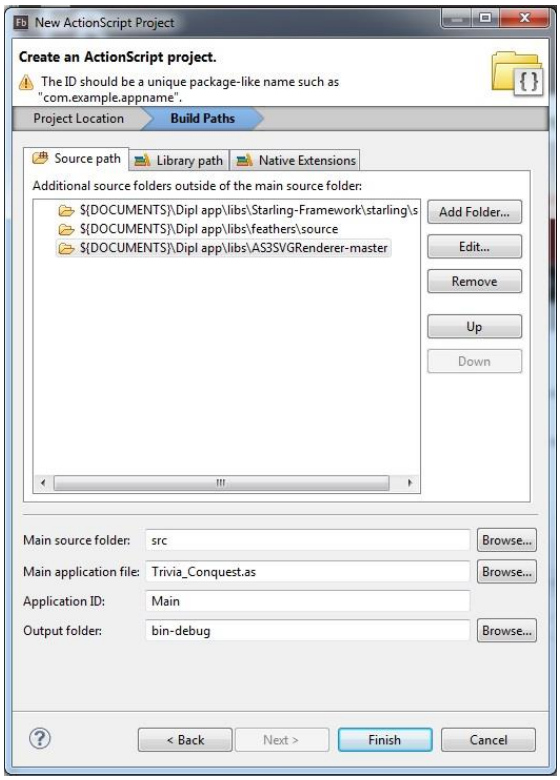


Slika 16: Stvaranje novog AS3 projekta

Odabirom novog AS3 projekta pojavljuje se skočni prozoru u kojem se određuje ime projekta te gdje će se izvršavati. S obzirom da je naša aplikacija dizajnirana za desktop računala izabiremo opciju Desktop(runs in Adobe AIR). Klikom na next pojavljuje se opcija da dodamo biblioteke i okvire.



Slika 17: Odabir vrste aplikacije

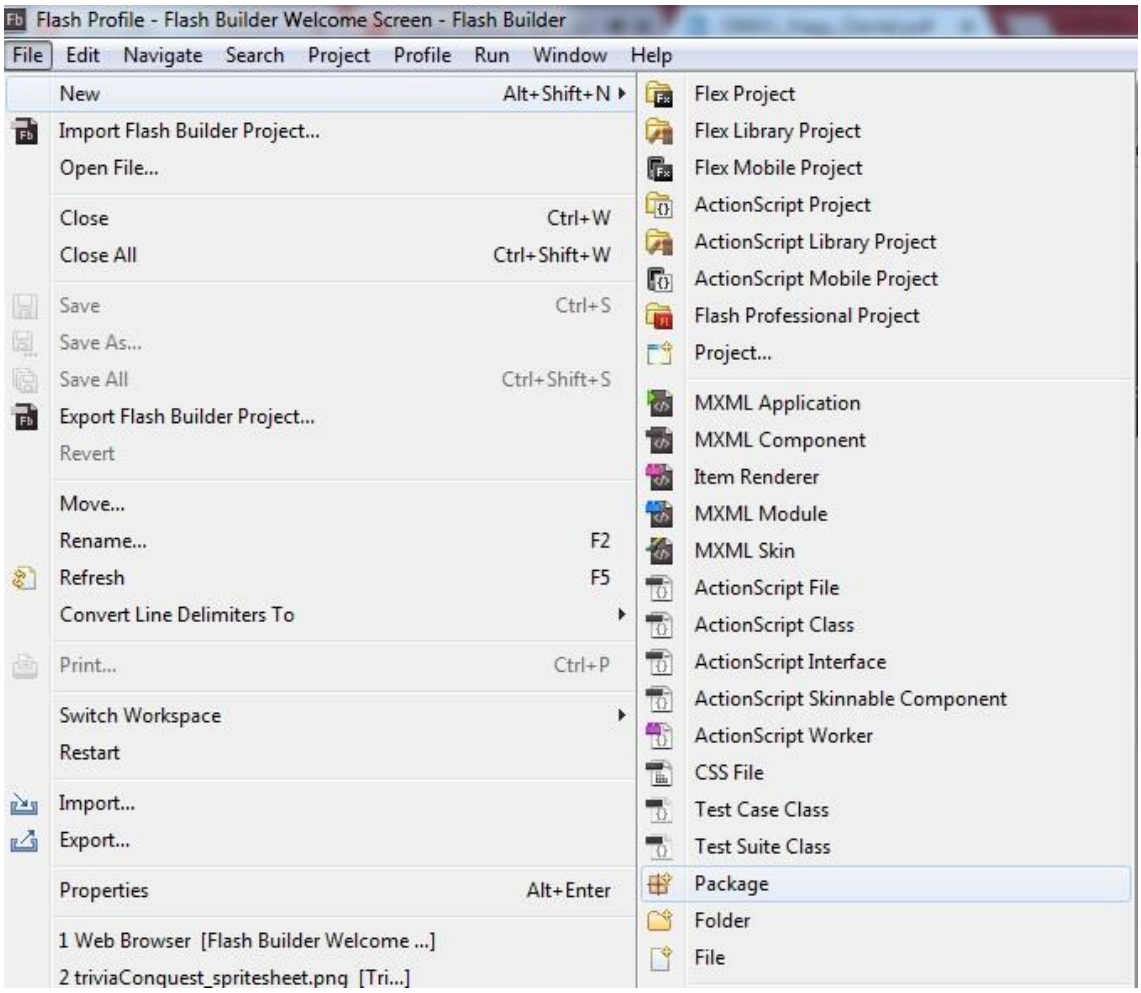


Slika 18: Uvoz biblioteka u projekt

Nakon uspješnog uvoza biblioteka uvozimo fontove i spritesheet-ove. Za potrebe ovog diplomskog koristiti ćemo već unaprijed napravljeni spritesheet od feathers biblioteke u koji smo dodali par slika pomoću Shoebox aplikacije. Fontovi se spremaju u mapu fonts koja se nalazi u mapi assets, a spritesheet se smješta u mapu images koja se također nalazi u mapi assets. XML datoteku sa pitanjima smještamo u mapu data koja se nalazi u mapi bin-debug.

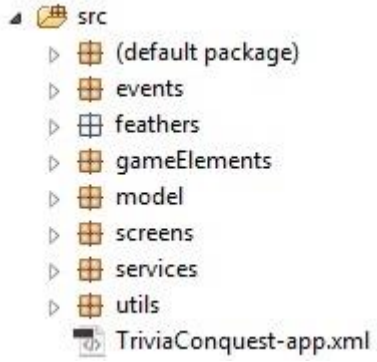
Slika 19: Spritesheet za aplikaciju

Kada je sve uvezeno počinjemo stvarati pakete. Ponekad projekti sadrže stotine, ako ne i tisuće klasa. Organiziranje tih klasa u povezane grupe standardna je praksa.[18] Paketi se stvaraju na isti način kao i projekti samo se u izborniku new izabere package umjesto actionscript project.



Slika 20: Stvaranje novog paketa

Stvorili smo još pakete events, gameElements, model, screens, services i utils. U paketu model ugnijezdili smo još jedan paket vo(value object), a u paket screens pakete menu i plugins. Nadalje, u paket menu dodali smo još jedan paket quiz koji sadrži zaslone za prikazivanje pitanja.



Slika 21: Paketi u aplikaciji

3.9. Isječci programskog koda

3.9.1. Pokretanje Starling-a

Prvo inicijaliziramo Starling okvir. To radimo na početku kako bi se kroz aplikaciju koristio Stage3D, a ne Flash Display Object. Stage3D koristi snagu grafičke procesne jedinice umjesto centralne. Unutar klase je deklarirana varijabla _starling koja je instanca klase Starling. Unutar uglatih zagrada određena je veličina ekrana, te boja pozadine scene. Metoda showStats nam omogućuje da u svakom trenutku vidimo potrošnju memorije i brzinu izvršavanja koda. Stage.align = StageAlign.TOP_LEFT pozicionira scenu u gornji lijevi kut. U konstruktoru Starlinga unosimo ime klase koja služi za prebacivanje u Starling okvir. Metodom _starling.start pokrenut je Starling okvir.

```
package
{

	import com.lorentz.processing.ProcessExecutor;
	import flash.display.Sprite;
	import flash.display.StageAlign;
	import flash.display.StageScaleMode;
	import starling.core.Starling;

	[SWF(frameRate="60",backgroundColor="000000",width="1280", height="700")]
	public class TriviaConquest extends Sprite
	{

		private var _starling:Starling;
		public function TriviaConquest()
		{
			// Init stage.
			stage.scaleMode = StageScaleMode.NO_SCALE;
			stage.align = StageAlign.TOP_LEFT;
			stage.frameRate = 60;

			// No flash.display mouse interactivity.
			mouseEnabled = mouseChildren = false;

			// Init Starling.
			Starling.handleLostContext = false;
			Starling.multitouchEnabled = false;

			_starling = new Starling( Main, stage, null, null, "auto",
"auto" );
			_starling.enableErrorChecking = false;
			_starling.showStats = true;
			_starling.simulateMultitouch = false;
			_starling.start();
			ProcessExecutor.instance.initialize(stage);
		}
	}
}
```

Isječak ispisa koda 10: Pokretanje Starling okvira

3.9.2. Navigator

Aplikacija je postavljena tako da sadrži jedan glavni kontenjer u kojemu su sadržani svi zasloni koje korisnik vidi. U tom kontenjeru stvoren je navigator koji služi za tranziciju između zaslona ovisno o korisnikovim željama. Ovaj dio koda se nalazi u klasi Main.as koji predstavlja starlingovu scenu.

```
// LoadingScreen početni ekran učitavanja
var loadingScreen:StackScreenNavigatorItem = new
StackScreenNavigatorItem(LoadingScreen);
loadingScreen.addPopEvent(Event.COMPLETE);
this.navigator.addScreen(ScreenTypeIDs.LOADING_SCREEN, loadingScreen);

// Main container novi navigator
var mainContainer:StackScreenNavigatorItem = new
StackScreenNavigatorItem(MainContainer);
mainContainer.addPopEvent(Event.COMPLETE);
this.navigator.addScreen(ScreenTypeIDs.MAIN_CONTAINER, mainContainer);

// Eventi koji se osluškuju
loadingScreen.setScreenIDForPushEvent(NavigationEventTypes.SHOW_MAIN_CONTAINER
, ScreenTypeIDs.MAIN_CONTAINER);
mainContainer.setScreenIDForPushEvent(NavigationEventTypes.SHOW_START_SCREEN,
ScreenTypeIDs.START_SCREEN);

// Prvi ekran koji će se prikazati
this.navigator.rootScreenID = ScreenTypeIDs.LOADING_SCREEN;

// Definiranje tranzicije navigatora
this.navigator.pushTransition = Slide.createSlideLeftTransition();
```

Isječak ispisa koda 11: Postavljanje navigatora

Prvo se učitava loading zaslon koji korisnik vidi dok se igrica učitava. Nakon toga se učita glavni kontenjer koji sadrži sve ostale zaslone. Svi ostali zasloni su dodani u klasi MainContainer.as

```
var startScreen:ScreenNavigatorItem = new ScreenNavigatorItem(StartScreen);
this.navigator.addScreen(ScreenTypeIDs.START_SCREEN, startScreen);
var gameScreen:ScreenNavigatorItem = new ScreenNavigatorItem(GameScreen);
this.navigator.addScreen(ScreenTypeIDs.GAME_SCREEN, gameScreen);
var rulesScreen:ScreenNavigatorItem = new ScreenNavigatorItem(RulesScreen);
this.navigator.addScreen(ScreenTypeIDs.RULES_SCREEN, rulesScreen);
var quizScreen:ScreenNavigatorItem = new ScreenNavigatorItem(QuizScreen);
this.navigator.addScreen(ScreenTypeIDs.QUIZ_SCREEN, quizScreen);
var finishScreen:ScreenNavigatorItem = new
ScreenNavigatorItem(FinishScreen);
this.navigator.addScreen(ScreenTypeIDs.FINISH_SCREEN, finishScreen);

transition = new ScreenSlidingStackTransitionManager(this.navigator);
transition.duration = Constants.slideAnimationDuration;
transition.ease = Transitions.EASE_IN_OUT;
// Navigator first screen
this.navigator.showScreen(ScreenTypeIDs.START_SCREEN);
/// dispatched events
this.navigator.addEventListener(CustomEventTypes.SHOW_SCREEN, showScreen);
```

Isječak ispisa koda 12: Događaji za promjenu zaslona

Za odabir željenog zaslona koristimo switch petlju. Njoj kao ulazni parametar prosljeđujemo objekt s imenom zaslona koji želimo vidjeti. Kada dobije ulazni parametar switch petlja provjerava ima li rješenje za takav slučaj. Ako ne nađe rješenje onda javlja grešku. Ako pronađe vrijednost koja odgovara ulaznom parametru petlja korisnika vodi na odabrani ekran.

```
private function showScreen(event:Event, responseObj:Object):void
            {

/// this switch knows whitch screen is currently active and which screen
needs to show
switch(responseObj.screenToShowScreenTypeID)
                  {
            case ScreenTypeIDs.START_SCREEN:
            this.navigator.showScreen(ScreenTypeIDs.START_SCREEN);
            break;
            case ScreenTypeIDs.GAME_SCREEN:
            this.navigator.showScreen(ScreenTypeIDs.GAME_SCREEN);
            break;
            case ScreenTypeIDs.QUIZ_SCREEN:
            this.navigator.showScreen(ScreenTypeIDs.QUIZ_SCREEN);
            break;
            case ScreenTypeIDs.RULES_SCREEN:
            this.navigator.showScreen(ScreenTypeIDs.RULES_SCREEN);
            break;
            case ScreenTypeIDs.FINISH_SCREEN:
            this.navigator.showScreen(ScreenTypeIDs.FINISH_SCREEN);
            break;
            default:
            throw new Error("Nešto nije u redu sa ScreenID-em. Pokušavaš
pozvati screen koji ne postoji!");
            break;
                  }
            }
```

Isječak ispisa koda 13:Switch petlja za promjenu zaslona

3.9.3. Učitavanje mape pomoću AS3svgrenderer biblioteke

Za učitavanje mape korištena je biblioteka AS3SVGRenderer od Lucasa Lorentza. Biblioteka ima klase koje predstavljaju svaki od SVG oblika. Raščlanjuje i pretvara svaki svg element u objekt za Flash display. Tako nam omogućava da selektiramo svaku državu posebno. Nedostatak biblioteke je što je napravljena za Flash display, a ne za Starling display. Kada se ubacuje element koji radi na Flash displayu u scenu koja je napravljena za Starling taj

element je uvijek iznad svega na zaslonu. Zbog toga svaki put kada želimo da se pojavi pop-up kojim ispitujemo želimo li stvarno napasti državu moramo napisati map.visible = false.[19]

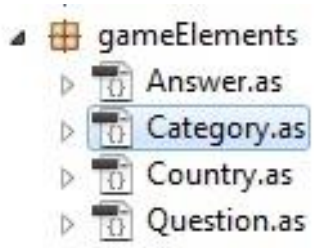
Mapu prvo učitamo u SVGDocument. U tako učitanoj mapi možemo selektirati svaki element. U for petlji se prolazi kroz svaki element i može se s njime raditi što je potrebno.

```
private function drawMap(event:Event):void
            {
                  map = new SVGDocument();
                  map.load("data/MapaEurope.svg");
                  Starling.current.nativeOverlay.addChild(map);
                  map.addEventListener(SVGEvent.VALIDATED, showCountries);
            }

private function showCountries(e:SVGEvent):void
      {
      trace(map);
      var resultVector:Vector.<String> = map.listDefinitions();
      for(var i:int = 0; i < resultVector.length; i++)
            {
      var svgelement:SVGElement = map.getDefinition(resultVector[i]) as
      SVGElement;

      if(GameDataModel.getInstance().isCountryDefeated(resultVector[i]))
            {
            trace("Pobjeđena: " + resultVector[i]);
            svgelement.transform.colorTransform = new ColorTransform(1,
0.8, 0.5, 1, 0x77, 0x51, 0x32);
            }
            else
            {
svgelement.addEventListener(MouseEvent.MOUSE_OVER, svg_mouseOverHandler);
svgelement.addEventListener(MouseEvent.MOUSE_OUT, svg_mouseOutHandler);
svgelement.addEventListener(MouseEvent.CLICK, svgGroup_clickHandler);
            }

            trace(resultVector[i]);
            }

}
```

Isječak ispisa koda 14: Učitavanje mape u igru

3.9.4. Učitavanje pitanja

Za potrebe učitavanja pitanja stvorene su nove klase Answer.as, Country.as, Question.as i Category.as. Te klase se nalaze u paketu gameElements.



Slika 22: gameElements

Pomoću tih klasa xml podatke prosljeđujemo u objekte kako bi ih mogli koristit u aplikaciji. Za dohvaćanje pitanja za svaku državu stvorena je funkcija getQuestionsforCountry.

```
public function getQuestionsForCountry(countryId:String):void
	{
		var result:Array = new Array();
		var len:int = quizData.countryArray.length;

	//// za svaku kategoriju dohvatiti jedno pitanje i ponuđene odgovore
		for(var i:int=0; i<len; i++)
		{
		if(countryId == quizData.countryArray[i].id)
		{
		var catLen:int = quizData.countryArray[i].categories.length;
		for(var j:int = 0; j < catLen; j++)
			{
			var catObj:Object = new Object();
				/// shuffle array s pitanjima
quizData.countryArray[i].categories[j].questionsArray =
Helper.shuffleArray(quizData.countryArray[i].categories[j].questionsArray);
catObj.category = quizData.countryArray[i].categories[j].categoryId;
var randomNumber:int =
Helper.randomRange(0,quizData.countryArray[i].categories[j].questionsArray.
length - 1);
catObj.question =
quizData.countryArray[i].categories[j].questionsArray[randomNumber];
result.push(catObj);
			}
		}
		}
		quizData.selectedQuestions = result;
	}
```

Isječak ispisa koda 15: Dohvaćanje pitanja

Kako pitanja ne bi uvijek dolazila istim redoslijedom korištena je for petlja koja kao ulazni parametar uzima polje s elementima, a vraća te iste podatke ali drugim redoslijedom. Ista ta petlja se koristi i za miješanje odgovora u pitanju kako točan odgovor ne bi uvijek bio na istom mjestu.

```
public static function shuffleArray(inputArray:Array):Array
      {
            var shuffledArray:Array = new Array(inputArray.length);
            var randomPos:Number = 0;

            for (var i:int = 0; i < shuffledArray.length; i++)
                  {
                  randomPos = int(Math.random() * inputArray.length);
                  shuffledArray[i] = inputArray.splice(randomPos, 1)[0];
//splice() vraća Array stoga je potrebno uzeti samo prvi član (u ovome
slučaju vraća samo jedan član )
                  }

                  return shuffledArray;
            }
```

Isječak ispisa koda 16: Petlja za miješanje pitanja

3.10. Završetak igre

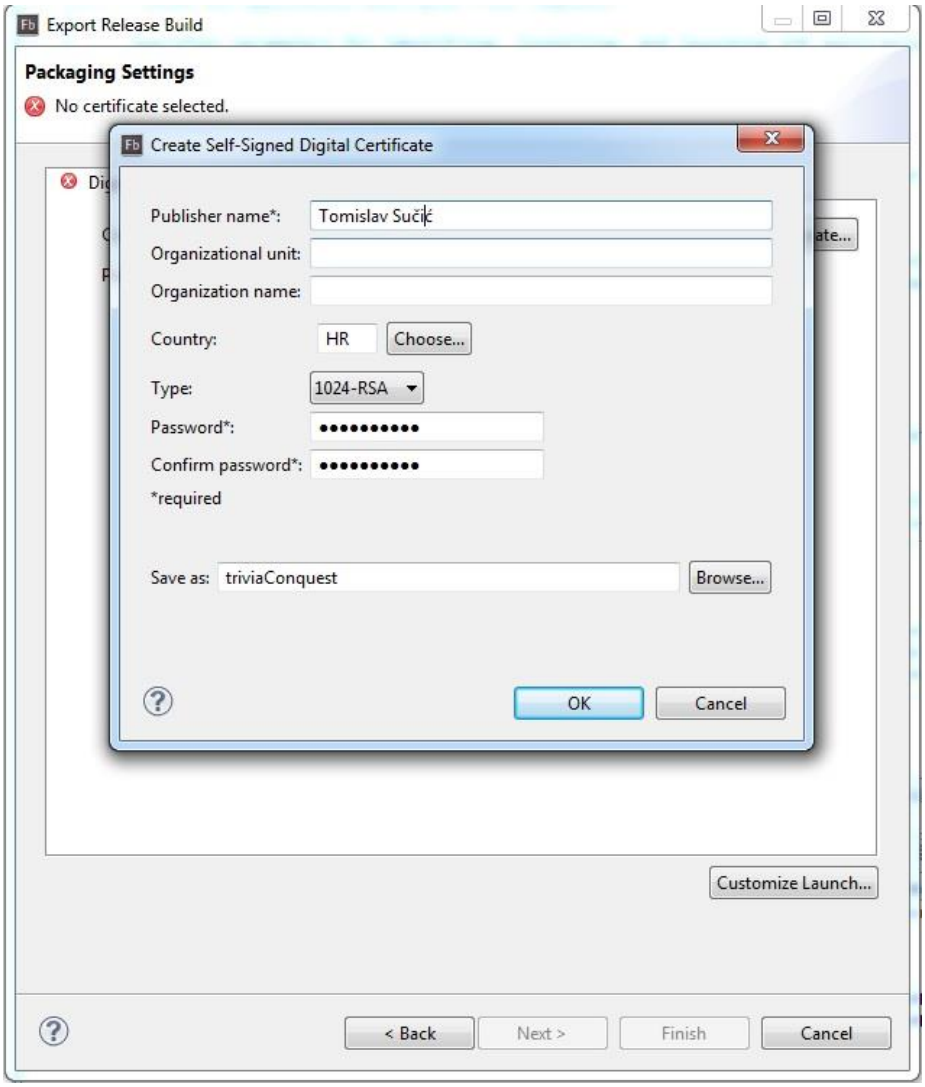
Igra ima 2 moguća ishoda – pobjeda igrača ili gubitak igrača. Igrač pobjeđuje ako je osvojio sve države,a gubi ako je broj napada došao do 0. To ispitujemo u funkciji endGame. Potrebno je sve varijable postaviti na početne vrijednosti kako bi se mogla igrati sljedeća igra sa istim postavkama kao i kad se igra prvi put.

```
private function endGame():void{
      if(GameDataModel.getInstance().GameInfoData.numberOfAttacks <= 0 ||
GameDataModel.getInstance().GameInfoData.defetatedCountries.length == 47)
            {
GameDataModel.getInstance().GameInfoData.numberOfAttacks = 5;
GameDataModel.getInstance().GameInfoData.defetatedCountries = new Dictionary;
var responseObj:Object = new Object;
responseObj.screenToShowScreenTypeID = ScreenTypeIDs.FINISH_SCREEN;
responseObj.currentScreenScreenTypeID = ScreenTypeIDs.QUIZ_SCREEN
this.dispatchEventWith(CustomEventTypes.SHOW_SCREEN, true, responseObj);
                  }
```

Isječak ispisa koda 17: Završetak igre

3.11. Izvoz aplikacije

Prije izvoza aplikacija mora biti validirana. Aplikacija se validira desnim klikom na projekt i odabirom opcije validate. Nakon validacije opet se klikne desnim klikom na projekt i odabere export. Među ponuđenim opcijama odabere se release build. Kada je sve odabrano flash builder nam ponudi da odaberemo ime i mapu u koju želimo spremiti release build. Također moramo napraviti i certifikat za igru. Kada sve to napravimo možemo izvesti igru. Nakon što je igra izvedena u željenoj mapi dobijemo installer za igru.



Slika 23: Opcije certifikata

4. REZULTATI I RASPRAVA

Ovakva vrsta igre ima puno mjesta za napredak. Igra je postavljena tako da se vrlo lako mogu ubacivati nove mape sa svojim pitanjima u nju. U budućnosti je zamišljeno da se igra može igrati na mobilnim uređajima te da se igra protiv drugih igrača. Uvođenjem multiplayer načina igre uvelike bi se pridonijelo na kompetitivnosti igre te bi time ljudi više uživali u igri. Igrači bi imali vojnike u svakoj od svojih država te bi na temelju toga mogli napadati druge igrače na mapi. Takav sustav napadanja koristi se u igri Rizik. Vojnici bi se gubili po postotcima od ukupne vojske koja napada ili se brani ovisno o razlici u broju točnih odgovora. Uspješnim napadom na drugog igrača mogla bi mu se preoteti država. Kao i u Riziku igrači bi dobivali kartice ako osvoje jednu državu u svom krugu. Skupljanjem kartica imali bi mogućnosti korištenja jokera. Uveo bi se i tip igre u kojima bi ljudi uložili novce te mogli imati neku zaradu ako pobijede. S vremenom bi se mogao napraviti i pravi board game u kojem bi ljudi igrali na ploči, a pitanja bi se vrtjela kroz aplikaciju.Iz svega navedenog vidljivo je da bi igra ljudima pružila zabavu, priliku za učenje, priliku za zaradu te priliku da pokažu ljudima koliko su pametni.

5. ZAKLJUČAK

Ljudi puno volje i brže uče i pamte stvari kada imaju volju za to nego kada su na to primorani zbog posla ili zbog dobivanja veće ocjene. Učenje u školama više nije učenje zbog znanja nego učenje zbog prolaza. Većinu toga što se nauči u takvim situacijama ljudi zaborave čim im više nije potrebno. Ali kada ljudi uče zbog vlastite znatiželje i želje za znanjem tada im je to puno lakše. Tako stečeno znanje kod ljudi ostaje puno dulje u pamćenju. Cilj ovog diplomskog rada bio je napraviti igru koja će omogućiti ljudima da uče kroz zabavu i natjecanje. Korištenjem ActionScript 3.0 jezika, XML-a, SVG-a i ostalih alata razvijena je jedna takva aplikacija. Najveći problem kod izrade edukativne 2D interaktivne igre bilo je osmisliti način na koji će se mape i pitanja za mape učitavati u igru, te kasnijim razvitkom lagano dodavati nove mape u igru - svaka sa svojim setom pitanja. Najbolji način za to bio je učitavanje mape u SVG formatu i pripadajućih pitanja u XML formatu. SVG i XML datoteke se lagano i brzo učitavaju pa kasnijim dodavanjem novih mapa nije potrebno mijenjati ActionScript 3.0 kod. Problem kod takvog učitavanja mapa bio je što Action Script 3.0 ne podržava direktno učitavanje SVG-a sa svim elementima koje on sadrži pa smo morali koristiti dodatnu biblioteku za to. U vrijeme izrade diplomskog rada nije bilo dostupnih biblioteka za učitavanje SVG formata u Starling display pa je korištena biblioteka AS3SVGRenderer koja učitava SVG u Flash display. Za razvitak igre u vidu dodavanja vojnika u svaku državu te sustava napadanja koji se koristi u igri Rizik morala bi se razviti biblioteka koja učitava SVG u Starling display. Time bi se pridonijelo na kompetitivnosti igre. Ljudi bi tada još više uživali u igri jer bi mogli pokazivati drugim igračima da su pametniji od njih. Mape ne moraju nužno biti kontinenti. Mogu se ubacivati i mape iz raznih svjetova fantazija (Game of Thrones, Lord of the rings i mnogi drugi). Potencijali su veliki.

Izvorni kod aplikacije i izvršna datoteka nalaze se na cd–u priloženom uz diplomski rad.

6. LITERATURA

1. https://www.adobe.com/products/flash-builder-standard.html - Adobe Flash Builder, 02.08.2018.
2. https://www.vecteezy.com/blog/2015/5/24/the-history-of-adobe-illustrator - povijest Adobe Illustratora, 05.08.2018.
3. https://helpx.adobe.com/illustrator/how-to/what-is-illustrator.html - What is Illustrator, 05.08.2018.
4. https://www.techopedia.com/definition/32364/adobe-photoshop - technopedia photoshop, 05.08.2018.
5. Roger Braunstein(2010): ActionScript 3.0 Bible, 2nd Edition
6. Moock, C. (2002): ActionScript for Flash MX: The Definitive Guide, 2nd Edition
7. http://www.moje-instrukcije.com/index.php?option=com_content&view=article&id=1518:osnove-objektno-orijentiranog-programiranja&catid=178 - osnove objektno orijentiranog programiranja, 23.08.2018.
8. Daniel Sperl: The Starling Manual
9. Imbert Thibault (2012): Introducing Starling, Building GPU Accelerated Applications
10. https://feathersui.com/help/ - Feathers help, 23.08.2018.
11. Dietel, Deitel, Nieto, Lin, Sadhu (2001): XML How to program
12. http://www.fer.unizg.hr/_download/repository/mipro_xml_tekst.pdf - XML tehnologija i primjena u sustavima procesne informatike, 21.08.2018.
13. https://hr.wikipedia.org/wiki/XML - XML, 21.08.2018.

14. https://www.sitepoint.com/svg-101-what-is-svg/ - What is SVG?, 21.08.2018

15. Amelia Bellamy-Royds, J. David Eisenberg(2014): SVG Essentials, 2nd Edition

16. William B. Sanders, Chandima Cumaranatunge - ActionScript 3.0 Design Patterns

17. Faber Birren(2014): Color Psychology and Color Therapy: A Factual Study Of The Influence Of Color On Human Life, Reprint of 1950 Edition

18. https://www.adobe.com/devnet/actionscript/learning/as3-fundamentals/packages.html - Actionscript 3.0 packages, 24.08.2018.

19. https://github.com/lucaslorentz/AS3SVGRenderer - AS3SVGRenderer/LucasLorentz, 25.08.2018.

7. POPIS SLIKA

8. POPIS ISJEČAKA KODOVA