SVEUČILIŠTE U MOSTARU

**FAKULTET STROJARSTVA,RAČUNARSTVA I ELEKTROTEHNIKE**

**ZAVRŠNI RAD**

**Upotreba multimedije u JavaFX okruženju**

|  |  |
| --- | --- |
|  | Jozo Skoko |

Mostar, rujan 2018

SVEUČILIŠTE U MOSTARU

**FAKULTET STROJARSTVA,RAČUNARSTVA I ELEKTROTEHNIKE**

**ZAVRŠNI RAD**

**Upotreba multimedije u JavaFX okruženju**

|  |  |
| --- | --- |
| Mentor: | Student: |
| Tomislav Volarić | Jozo Skoko |

Mostar, rujan 2018.

*Ovim putem zahvaljujem se svom mentoru doc. Tomislavu Volariću na ukazanoj pomoći pri izradi ovog završnog rada,odgovaranju na mailove,davanju literature i ostalim oblicima pomoći.*

*Također bi želio naglasiti njegovu sposobnost koja me je zaintegrirala u svijetu programiranja Jave bez koje ovaj rad ne bi bio moguć.*

Hvala i svim ostalim,prijateljima i obitelji na najmanjoj pomoći u vidu motivacije,savjeta koju su mi pružili tijekom školovanja koji su uvelike pomogli da dođem danas i situaciju da pišem završni rad.

Izjavu dao:

Jozo Skoko

SVEUČILIŠTE U MOSTARU

FAKULTET STROJARSTVA ,RAČUNARSTVA I ELEKTROTEHNIKE

|  |  |
| --- | --- |
| Preddiplomski studij: | Računarstvo |
| Smjer: | - |
| Ime i prezime: | Jozo Skoko |
| Broj indeksa: | 1384/RR |

ZADATAK ZAVRŠNOG RADA

|  |  |  |
| --- | --- | --- |
| Naslov: | Upotreba multimedije u JavaFX okruženju(Using multimedia in JavaFX framework) | |
| Zadatak: | Izraditi projekt u programskom okruženju JavaFX. Projekt je vrste igrice kviza znanja, na kojem će se pokazati upotreba multimedije(slike,zvučnih efekata) i dretvi. Također pokazati prednosti i značajke izradi takve aplikacije u JavaFX-u uz upotrebu MySQL baze podataka i InteliJ IDEA programa za rad s Javom. | |
| Prijava rada: | | *23.05.2018.* |
| Rok za predaju rada: | | *20.09.2018.* |
| Rad predan: | | *01.09.2018.* |

|  |
| --- |
| Mentor: |
| Doc. Tomislav Volarić |

**Upotreba multimedije u JavaFX okruženju**

**Sažetak:**

U ovom radu analizira se razvoj aplikacije u JavaFX frameworku koristeći InteliJ alat, MySQL bazu podataka i Gluon SceneBuilder. Aplikacija će sadržavati dosta multimedije(slika,zvuk) i dretvi koji utječu na interakciju korisnika s aplikacijom. Interakcija će biti detaljno razrađena i vidjet ćemo kako ona utječe na rad aplikacije. Aplikacija je tipa računalne igre kviza znanja.

**Using multimedia in JavaFX framework**

**Abstract:**

This thesis analayze application development in JavaFX framework using InteliJ tool,MySQL database and Gluon SceneBuilder. Application is going to have a lot of multimedia(picture,sound) and threads which have influence on interaction with user. Interaction will be elaborated in details and we will see how it affects on application. Application is computer game type of knowledge quiz.

TABLICA SADRŽAJA...

**1 UVOD**

U ovom radu predstavljena je izrada aplikacije u JavaFX frameworku upotrebljavajaći elemente multimedije i dretvi. Alati koji su korišteni pri izradi aplikacije su InteliJ alat za programiranje u JavaFX, MySQL baza podataka i Gluonov SceneBuilder za izradu grafičkog sučelja. Vidjet ćemo kako postavljanjem multimedije(zvuk,slika) i dretvi možemo postići zanimljivu interakciju korisnika s aplikacijom. Postavljanje multimedije alatom SceneBuilder bit će također posebno pojašnjeno jer je on jedan od alata koji omogućava lakše ubacivanje multimedije. Rad sadržava poglavlja vezana za same karakteristike JavaFX i njegovu povijest kako bi bolje upoznao čitatelja i imao jasnu sliku kako se koristi JavaFX kako bi bolje shvatio temu rada.

Aplikacija je igrica kviza znanja za koju se korisnik treba registrirati kako bi igrao. Ulaskom u igru su mu ponuđena težina pitanja,mogućnost promjene šifre i uvid u tablicu rezultata. Pravila igre su sljedeća :

* Korisnik s obzirom na težinu dobiva pitanja određene težine koja nose više bodova s obzirom na težinu pitanja(Easy – 1 bod,Normal – 2 boda,Hard – 3boda).
* Korisnik ima 3 života, pogrešan odgovor oduzima 1 bod,kada izgubi sve živote sprema se rezultat ako je nadmašio svoj najbolji rezultat ili prvi put igra kviz.
* U gornjem dijelu aplikacije nalazi se vremenska crta koja ističe, tako da korisnik ima ograničeno vrijeme koliko može igrati jedan kviz, nakon toga se sprema rezultat ako je nadmašio svoj najbolji rezulzaz ili prvi put igra kviz.

Treba napomenuti da će se u redu koristit neki stručni engleski izrazi za kojih nema točno formalno definiran prijevod u hrvatskom jeziku.

**2 JAVAFX OKRUŽENJE**

**2.1 Što je JavaFX**

JavaFX je programsko okruženje sačinjeno od grafičkih i medijskih paketa koji omogućavaju programeru da dizajnira,pravi,testira,debugira i izvozi aplikacije koje ispravno rade na različitim platformama. S obzirom da je JavaFX biblioteka napisana kao Java API(Application Programming Interface), kod aplikacije može referencirati na ostale API-je iz Java biblioteke. To znači da JavaFX može koristiti API biblioteke za pristup raznim sustavima kao i sustavima koji su bazirani na spajanju sa serverom i komunikacijom s bazom podataka. Izgled JavaFX-a može se proizvoljno postaviti. CSS(Cascading Style Sheets) se koristi za izgled i stil , te je on posebno odvojen kako bi se programer mogao fokusirati samo na pisanje koda. CSS se primjenjuje na FXML datoteke koje služe za vizualni prikaz aplikacije. FXML datoteka može se ručno uređivati kodom CSS-a, a u drugom slučaju moguće je koristit poseban alat nazvan SceneBuilder koji omogućava lakše dizajniranje aplikacije. JavaFX je integriran od JDK(Java Development Kit) 7 verzije, stoga je on kompatibilan sa naprednijim verzijama JDK-a i upotrebljiv na različitim platformama koje sadrže zadovoljavajuću verziju JDK-a.



Slika 2.1: JavaFX logo

**2.2 Povijest JavaFX-a**

JavaFX je zamisao čovjeka zvan Chris Oliver. Chris je napravio JavaFX, koji se u početku zvao *F3*(Form Follows Function) dok je Chris radio za kompaniju SeeBeyond koja se bavila razvojem poslovnih integracijskih softvera. Chris je htio biti uključen u razvoju softvera od samog početka radi čega je došlo do nesuglasica s direktorom što će poslje predhoditi njegovom razlazu sa firmom. On je počeo proučavati Adobe-ov *Flash. Adobe Flash* je bio baziran na *XML-*u i *JavaScriptu.* Smatraoje dase *XML* ne bi trebao koristiti kao programski jezik i da se programski jezik mora sastojati od varijabli i funkcija. Šest mjeseci poslje u rujnu 2005. F3 je rođen. Nedugo zatim kompanija Sun kupuje kompaniju SeeBeyond i pri preuzimanju ne pokazuju interes za njegov projekt. U to vrijeme Sun je tjerao svoje inžinjere da počnu pisati blogove. Chris je napisao blog o sebi i svoj projektu što je postalo hit, te je s time počeo graditi poznato ime u svijetu programiranja. Dodatan uspjeh će biti zapažanje izumitelja Jave programskog jezika James Gosling-a nakon čega se 2007. *F3*  preimenovao u *JavaFX Script. Službena verzija JavaFX Script* ili JavaFX 1.0 izdata je u prosincu 2008. I ona se razvijala do listopada 2011. Kada službeno izlazi JavaFX 2.0 ili JavaFX koju danas poznajemo.[1]

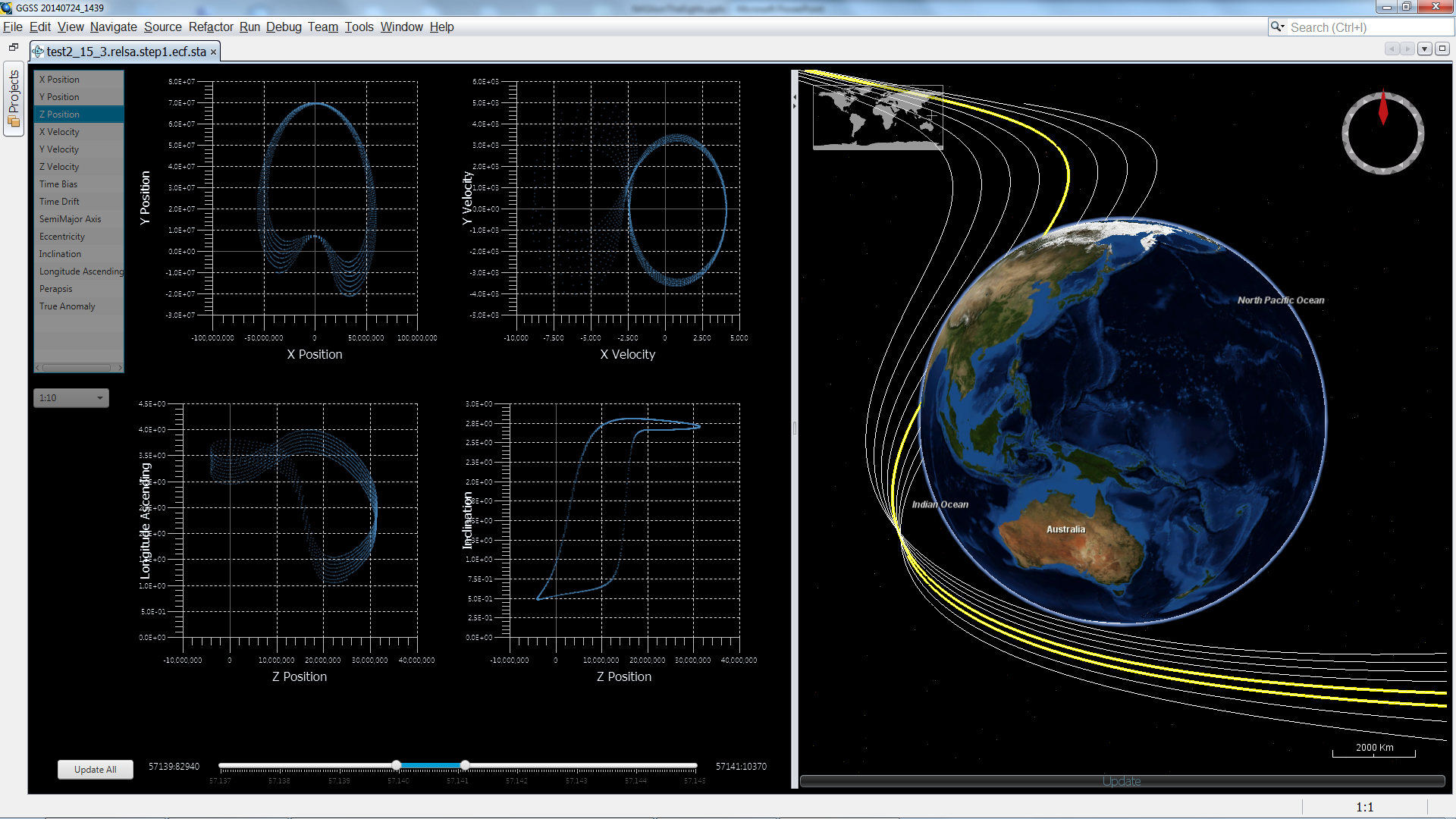
Novosti koje je uvela JavaFX 2.0 su :

* JavaFX postaje „*open-source“* što znači da se originalni kod može slobodno redistribuirati i modificirati.
* Podrška za mobilne telefone(JavaFX Mobile).
* Novi set *Java API-a.*
* Podrška za Linux OS.
* Korištenje *FXML-a* umjesto *XML-a*
* *GUI(Graphical User Interface)* komponente

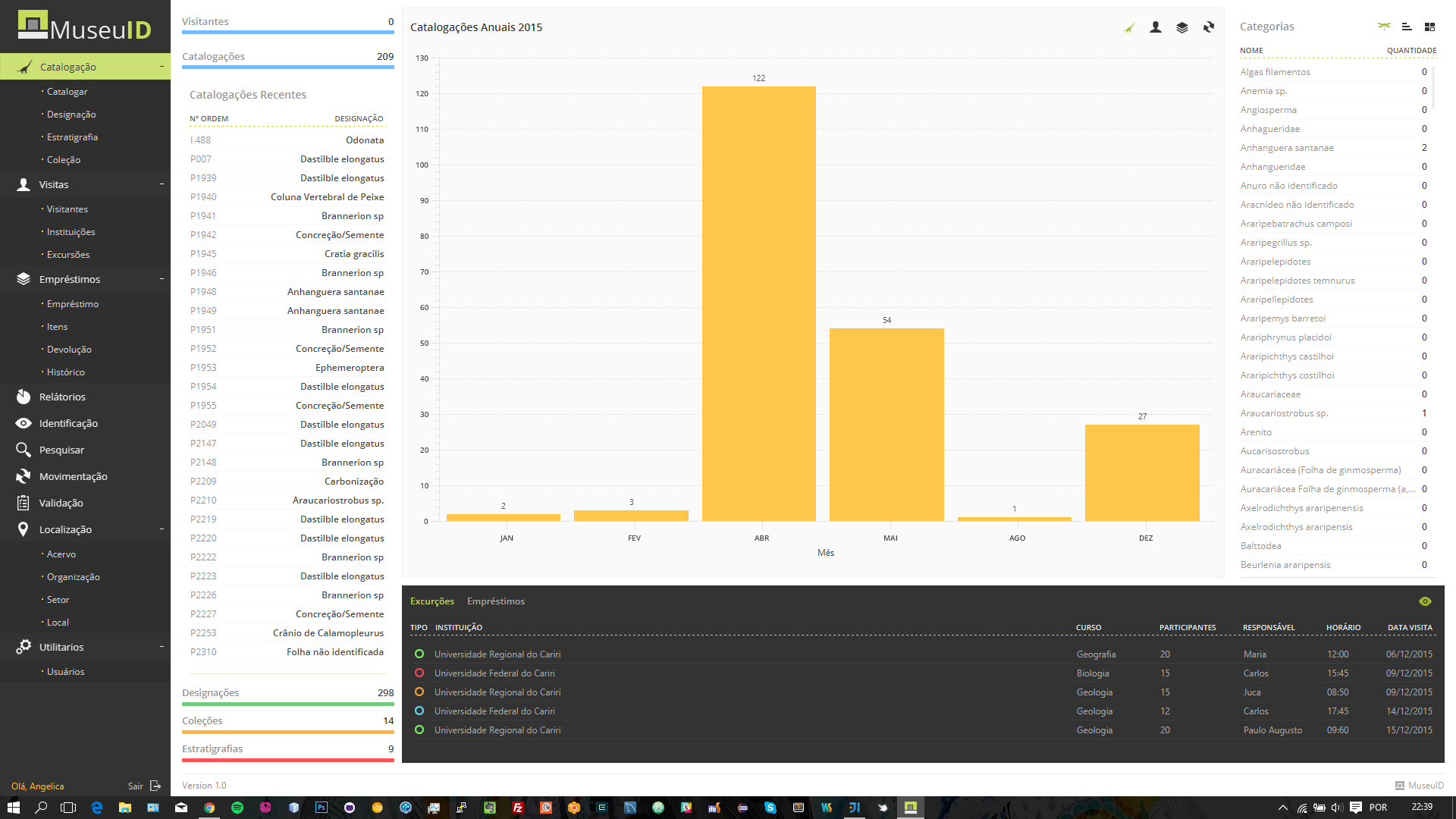
**2.3 JavaFX danas**

Danas imamo verziju JavaFX 11 povodom izlaska novog Java 11 JDK-a. Od izlaska Java 8 JDK-a nije više potrebno posebno implementirati JavaFX u naše alate za rad s JavaFX-om jer Java 8 JDK u sebi sadrži JavaFX. Današnja verzija pruža potporu za 3D grafiku što omogućuje da aplikacije pravljene u JavaFX izgledaju bolje i inovativnije. Konkurencija JavaFX-u zove se *Java Swing* koja je već zastarjela inačica za pravljenje desktop aplikacija u Java programskom jeziku i možemo reći kako je JavaFX poptuno istisnuo Swing iz upotrebe.[2] Jedan od većih razloga je lakši i bolji pristup programiranja jer upotrebom *FXML-a* jasnije se dočarava *MVC*(Model-View-Controller) sustav po kojem JavaFX radi.

Aplikacije poput : AIDA,MuseoId, Deep Space Trajectory Explorer NASA, Geons NASA su vrhunske aplikacije rađene u JavaFX-u koje služe za poslovne,zdravstvene i istraživačke svrhe.



Slika 2.2: Geons Nasa aplikacija



Slika 2.3: MuseoId aplikacija

**3 ARHITEKTURA JAVAFX-A**

**3.1 Pozornica-scena-čvor**

Pozornica-scena-čvor ili „Stage-scene-node“ je vizualni pojam koji označava način na koji JavaFX generira GUI, tj. ono što vidimo. Pozornica ili Stage predstavlja dio koda koji je zadužen za prikaz i generiranje određene scene. To je ono što vidimo u programskoj traci našeg operativnog sustava. Pokretanjem JavaFX aplikacije može se generirat više od jedne pozornice, ali je opća praksa da se koristi jedna na kojoj ćemo izmjenjivati naše scene.[3]

Scena ili Scene je početna točka za konstruiranje JavaFX aplikacije. Ona je hijerarhijsko stablo čvorova koje prestavlja sve vizualne elemente GUI-a aplikacije. Scena se mjenja kodom s obzirom na interakciju korisnika s aplikacijom, npr. korisnik se želi registrirati na aplikaciji pri čemu mu se otvara na prozoru scena s poljima za ispunjavanje. Scena također može biti „izrenderirana“. JavaFX API javafx.scene dopušta kreiranje različitih tipova sadržaja, kao npr. :

* Čvorovi: 2-D i 3-D oblici,slike,media,web pretraživači,tekst,upravljači događajima.
* Stanja: vizualni efekti i transformacije(pozicioniranje i orijentacija čvorova).
* Efekti: sjene,prilagođavanje boje, zamagljenja i sve što utječe na izgled scene.

Čvor ili Node je zaseban element na sceni. Svaki čvor ima svoj ID, klasu i svojstva. Svaki čvor u sceni ima svog roditelja, osim glavnog „root“ čvora. Čvor također može imati sljedeće:

* Efekte,npr. sjene i maglu (Blur)
* Neprozirnost (Opacity)
* Transformacije
* Upravljače događajima(Event handlers)

|  |
| --- |
| public class Main extends Application {  @Override  public void start(Stage primaryStage) {  try {  Parent root = FXMLLoader.load(getClass().getResource(“/application/Login.fxml“)  Scene scene = new Scene(root);  scene.getStylesheets().add(Main.class.getResource("login.css").toExternalForm());  primaryStage.setScene(scene);  primaryStage.setResizable(false);  primaryStage.show();  } catch(Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  launch(args);  }  } |

Kod 3.1: Postavljanje pozornice,scene i čvorova u projektu

U prikazanom kodu predstavljen je način postavljanja pozornice,scene i čvorova u projektu oko kojeg je baziran ovaj rad. public void start(Stage primaryStage) je nadjačana metoda koja se prva pokreće prilikom paljenja aplikacije i u njoj se poziva početna pozornica i scena.

U dijelu koda:

Parent root = FXMLLoader.load(getClass().getResource(“/application/Login.fxml“)

Scene scene = new Scene(root);

scene.getStylesheets().add(Main.class.getResource("login.css").toExternalForm());

primaryStage.setScene(scene);

primaryStage.setResizable(false);

primaryStage.show();

}

Parent root je klasa koja učitava FXML datoteku koja se nalazi u paketu application i koja služi da se postavi na scenu Scene scene = new Scene(root);. FXML datoteka sadrži sve elemente koji će biti prikazani na sceni. Sceni se dodaje CSS dizajn metodom .getStylesheets().add(). Zatim je tu scenu potrebno postaviti koristeći metodu na pozornici primaryStage.setScene(scene). Kada se scena postavi ona se mora i prikazati metodom .show(). Metoda .setResizable(false) onemogućuje korisniku da proizvoljno razvlači ili smanjuje prozorčić pozornice kako bi elementi koji će biti prikazani na sceni ostali pozicionirani onakvim kakvi su postavljeni u FXML datoteci,te duljina i širina pozornice također ovisi o FXML datoteci.



Slika 3.1: Prikaz aplikacije nakon izvršavanja navedenog koda

**3.2 FXML**

FXML je skriptirani XML baziran jezik za konstruiranje Java objektnih grafova, u ovom slučaju scena u JavaFX. Omogućuje zgodnu alternativu konstruiranja grafa u proceduralnomkodiranju i savršeno je pogodan za pravljenje korisničkog sučelja u JavaFX-u pošto koristi XML hijerarhijsk ustrukturu zatvarnja elemenata.[4]

U FXML, XML element predstavlja jedno od sljedećeg:

* Instancu klase
* Svojstvo instance klase
* Statičko svojstvo
* Definicijski blok
* Blok skriptiranog koda

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <?import javafx.geometry.Insets?>  <?import javafx.scene.control.Button?>  <?import javafx.scene.control.Label?>  <?import javafx.scene.control.PasswordField?>  <?import javafx.scene.control.TextField?>  <?import javafx.scene.layout.VBox?>  <?import javafx.scene.text.Font?>  <VBox fx:id="vb" alignment="TOP\_CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="350.0" prefWidth="500.0" style="-fx-background-color: gray;" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.LoginController">  <children>  <Label fx:id="welcome" alignment="CENTER" contentDisplay="TOP" prefHeight="38.0" prefWidth="190.0" text="WELCOME" textAlignment="CENTER" textFill="#a10c0c">  <font>  <Font size="24.0" />  </font>  </Label>  <TextField fx:id="username" prefHeight="54.0" prefWidth="394.0" promptText="Username">  <VBox.margin>  <Insets left="100.0" right="100.0" top="50.0" />  </VBox.margin>  </TextField>  <PasswordField fx:id="password" prefHeight="54.0" prefWidth="394.0" promptText="Password">  <VBox.margin>  <Insets left="100.0" right="100.0" top="10.0" />  </VBox.margin>  </PasswordField>  <Button mnemonicParsing="false" onAction="#login" prefHeight="37.0" prefWidth="79.0" text="Login">  <VBox.margin>  <Insets top="5.0" />  </VBox.margin>  </Button>  <Button mnemonicParsing="false" onAction="#register" prefHeight="37.0" prefWidth="79.0" text="Register">  <VBox.margin>  <Insets top="10.0" />  </VBox.margin>  </Button>  <Label fx:id="info">  <VBox.margin>  <Insets right="350.0" top="20.0" />  </VBox.margin>  </Label>  </children>  </VBox> |

Kod 3.2: Prikaz Login.fxml FXML datoteke

U navedenom kodu vidimo elemente kao što su Label,Button,TextField i da oni mogu sadržavati svojstva kao prefHeight,prefWidth,id,alignment što podsjeća na XML i HTML dijelove koda s obzirom da je FXML baziran na XML-u. Najbitnije svojstvo je id jer on omogućuje da kontroler koji upravlja tim FXML-om može preopoznati određeni element. Svojstvo fx:controller="application.LoginController" nam definira kontroler.

Funkcije koja postavlja FXML datoteku je :

FXMLLoader.load(getClass().getResource())

Njena upotreba se može pronaći na kodu Kod 3.1 u public void start(Stage primaryStage) funkciji.

**3.3 CSS**

CSS je kratica od Cascading Style Sheets. Radi se o stilskom jeziku, koji se rabi za opis prezentacije dokumenta napisanog pomoću HTML jezika. Kako se web razvijao, prvotno su u HTML ubacivani elementi za definiciju prezentacije (npr. tag <font>), ali je dovoljno brzo uočena potreba za stilskim jezikom koji će HTML osloboditi potrebe prikazivanja sadržaja (što je prvenstvena namjena HTML-a) i njegovog oblikovanja (čemu danas služi CSS). Drugim riječima, stil definira kako prikazati HTML elemente. CSS-om se uređuje sam izgled i raspored stranice.[5]

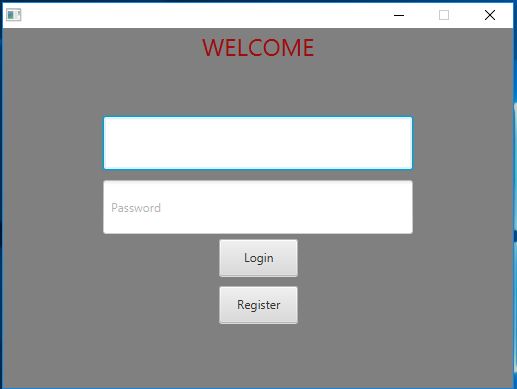
CSS je moguće koristiti u JavaFX-u. Koristi se za stiliziranje FXML dokumenta.

|  |
| --- |
| #vb {  -fx-background-image: url("/slike/background.jpg.jpg");  }  .label {  -fx-font-size: 12px;  -fx-font-weight: bold;  -fx-text-fill: #333333;  -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );  }  #welcome {  -fx-font-size: 32px;  -fx-font-family: "Arial Black";  -fx-fill: #818181;  -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6, 0.0 , 0 , 2 );  }  .button {  -fx-text-fill: white;  -fx-font-family: "Arial Narrow";  -fx-font-weight: bold;  -fx-background-color: linear-gradient(#61a2b1, #2A5058);  -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );  }  .button:hover {  -fx-background-color: linear-gradient(#2A5058, #61a2b1);  }  #info {  -fx-fill:#FF0000;  -fx-text-fill:#FF0000;  -fx-font-weight: bold;  -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );  } |

Kod 3.3: Prikaz CSS datoteke za Login.fxml

Na navedenom kodu vidimo CSS dizajn login prozora aplikacije. Po svim karakteristikama vidimo da se on koristi potpuno isto koa u HTML-u. ID selektori, selektori s točkom,selektori po imenu se koriste kako bi se dizajnirali sami elementi u FXML datoteci i upravo davanje id i name svojstva nam omogućuju za komunikacija s određenim elementima u FXML datoteci.

Dio CSS koda .button postavlja svim gumbovima na sceni zadano css svojstvo, #welcome postavlja samo labeli welcome jer se radi o ID selektoru.



Slika 3.2: Izgled Login prozora bez CSS-a



Slika 3.3: Izgled Login prozora sa CSS-om

**3.4 Kontroler događajem**

Događaji su, poput svega u Javi, predstavljeni objektima. Postoji nekoliko različitih vrsta događaja, a svaka vrsta zasebno predstavljena je klasom. Sve klase događaja korištene u AWT-u su podklase apstraktne klase java.awt.AWTEvent. Klase događaja koje opisuju točno određene vrste događaja, poput MouseEvent i ActionEvent, nalaze se u paketu java.awt.event. Da bi događaj imao utjecaj, potrebno je da program otkrije događaj i odgovori na njega. Za svaku klasu događaja postoji sučelje koji određuje jednu ili više metoda za odgovor na događaje te klase. Na primjer uz klasu ActionEvent vezano je sučelje koje se zove ActionListener. To sučelje definira metodu, "public void actionPerfromed(ActionEvent evt)".

Primjenom ActionListener sučelja, objekt postaje sposoban "slušati" događaje. Ali prije nego može stvarno "čuti" događaje, mora biti prijavljen kao slušač događaja (action listener). Događaji stvaraju komponente. Tako, na primjer, kad korisnik pritisne tipku, generira se jedan događaj. Svaka komponenta koja može stvarati događaje ima definiranu metodu addActionListener, koja služi za prijavljivanje slušača događaja kod komponente. Kad komponenta stvori događaj, obavještava sve prijavljene slušače događaja pozivanjem njihovih actionPerformed() metoda.[6]

|  |
| --- |
| @FXML Label changepw;  @FXML PasswordField pw;  @FXML PasswordField reenterpw;  @FXML Button confirm;  @FXML private ChoiceBox<String> cb\_menu;  public static MediaPlayer mediaPlayer = new MediaPlayer(media);  public static boolean running = false;  public static int tezina;  public static String tezinaS;  public void changepwClick(MouseEvent e) {  if (pw.isVisible()) {  pw.setVisible(false);  reenterpw.setVisible(false);  confirm.setVisible(false);  pw.setText("");  reenterpw.setText("");  } else {  pw.setVisible(true);  reenterpw.setVisible(true);  confirm.setVisible(true);  }  }  public void NewGameClick(ActionEvent event) {  tezina = cb\_menu.getSelectionModel().getSelectedIndex() + 1;  tezinaS = cb\_menu.getSelectionModel().getSelectedItem();  try {  mediaPlayer.stop();  running = false;  Stage stage = (Stage) btn\_options.getScene().getWindow();  Parent root;  root = FXMLLoader.load(getClass().getResource("Game.fxml"));  Scene scene = new Scene(root);  scene.getStylesheets().add(Main.class.getResource("game.css").toExternalForm());  stage.setScene(scene);  } catch (IOException e) {  }  } |

Kod 3.4: Primjer ActionEventa i MouseEventa u projektu u MenuController

U navedenom kodu imamo primjer funkcija changepwClick koja predstavlja MouseEvent i NewGameClick koja predstavlja ActionEvent. changepwClick funkcija će na labelu changepw klikom miša otkriti PasswordField-ove pw,reenterpw i Button confirm ako su nevidljivi, u suprotnom će ih sakriti.

NewGameClick funkcija će na pritisak Button-a NewGame izvršiti dio koda u kojem će okrenuti na scenu FXML datoteke Game.fxml pri čemu koristi statičku varijablu tezina kako bi se na netočne odgovore skidao određeni broj bodova s obzirom na težinu i još jednu statičku varijablu tezinaS kako bi se iz baze povukla pitanja određene težine koju je korisnik odabrao pomoću ChoiceBox-a cb\_menu.

Važno je napomenuti da se prilikom inicijaliziranja bilo kojeg događaja tj. funkcije koja će se izvršiti prilikom interakcije s određenim elementom nije potrebno inicijalizirati taj element kao što vidimo u slučaju Button-a NewGame.

Događaj se može postavljati u FXML datoteci ručno koristeći svojstvo onAction=“#ime\_funkcije\_u\_kontroleru“, a može i vizualno koristeći jako dobar i važan alat SceneBuilder čije će korištenje biti objašnjeno u idućem poglavlju.

**4 SCENE BUILDER**

**4.1 Što je i čemu služi Scene Builder**

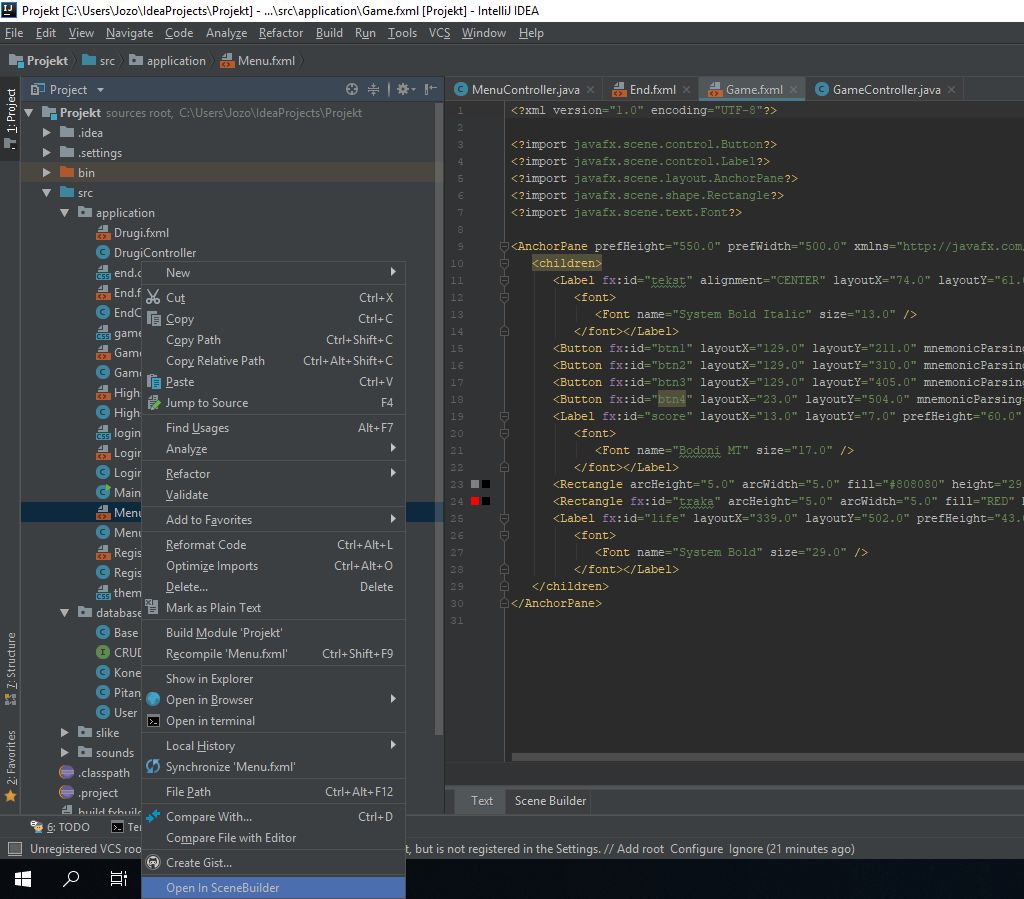
JavaFX Scene Builder je vizualni alat koji omogućava dizajniranje JavaFX aplikacija bez upotrebe kodiranja. Omogućuje jednostavnu „drag-and-drop“ metodu pozicioniranja GUI elemenata na JavaFX scenu. Kako gradimo izgled za naše korisničko sučelje Scene Builder automatski generira FXML kod za određeni okvir. Scene Builder također omogućuje ljudima koji su slabi s programiranjem direktno povezivanje GUI komponenata s logikom u aplikaciji(id,kontroleri...). Najčešće je upotrebljavan Gluon-ov Scene Builder.[7]



Slika 4.1: Gluon Scene Builder logo

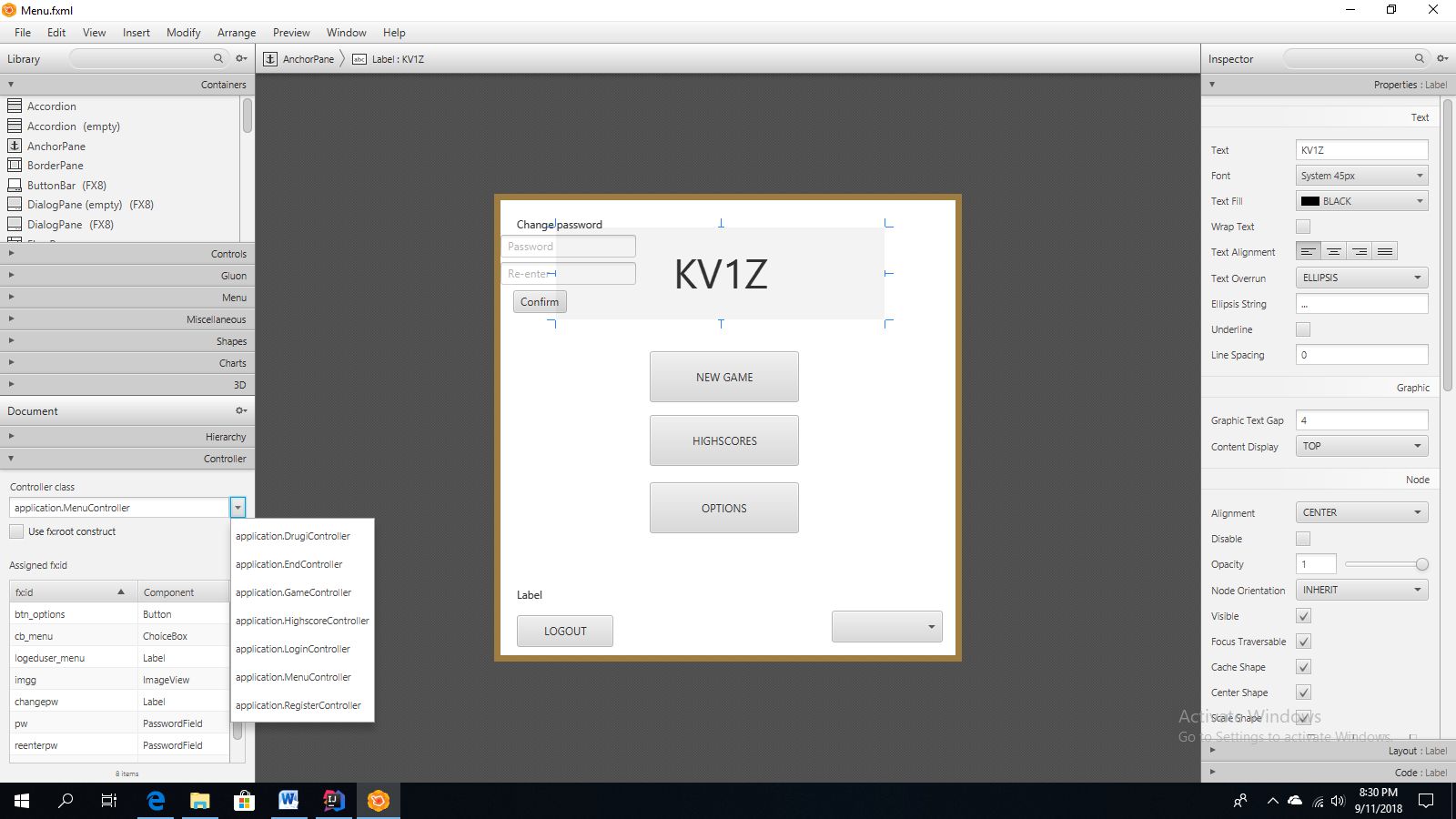
**4.2 Povezivanje GUI-a aplikacije pomoću Scene Builder-a**

Da bi se Scene Builder mogao koristiti potrebno ga je prvo skinuti s Gluon-ove web stranice <https://gluonhq.com> ,a zatim odabrati odgovarajući operativni sustav i instalirati.



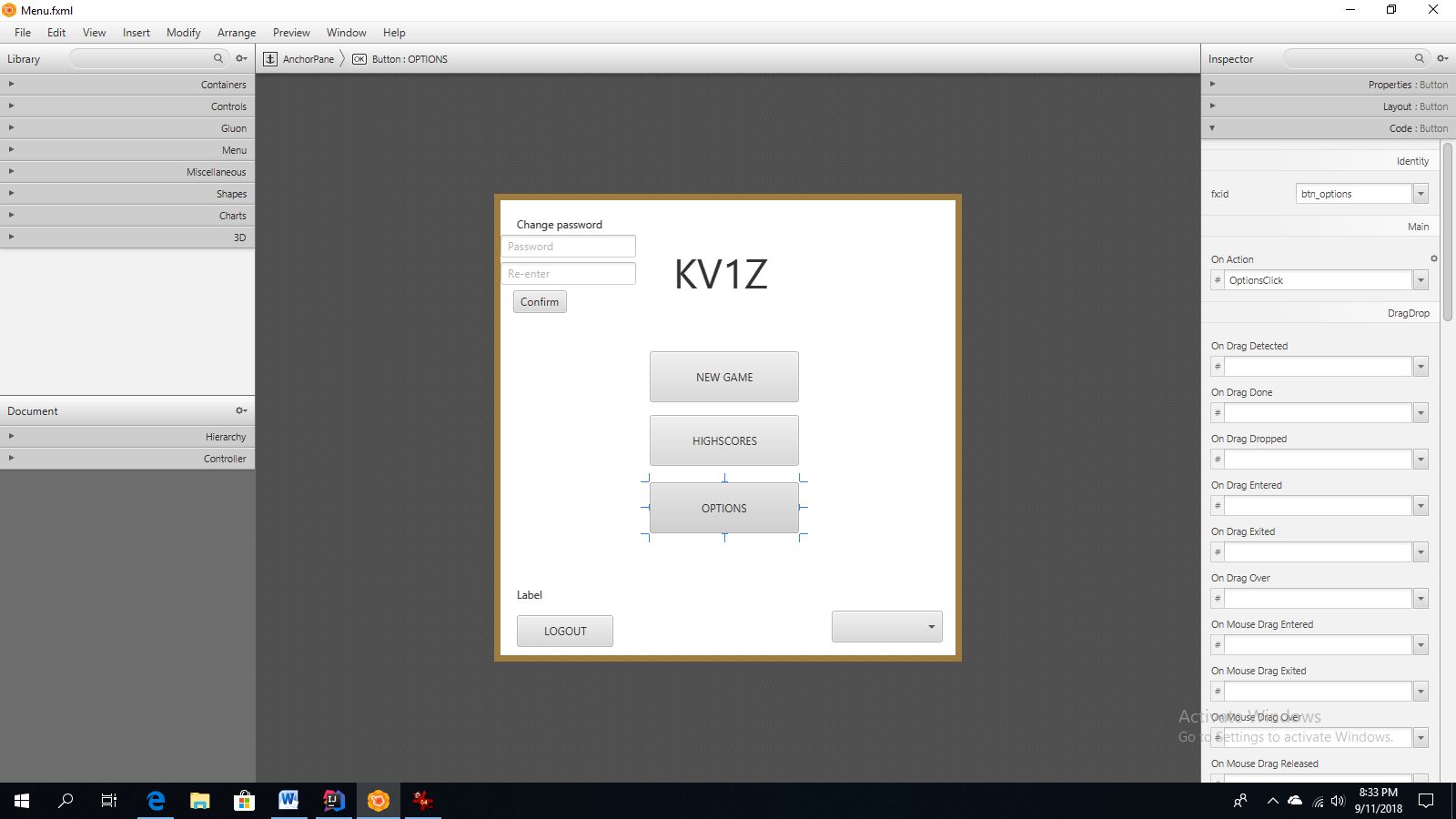
Slika 4.2: Otvaranje Scene Builder-a

Zatim je potrebno desnim klikom na *FXML* datoteku iz projekta odabrati opciju „*Open in SceneBuilder*“.



Slika 4.3: Postavljanje kontrolera preko Scene Buildera

Kada se Scene Builder otvori potrebno je u donjem desnom kutu u prozoru „Controller Class“ odabrati određeni kontroler za našu *FXML* datoteku. Ona će tim postupkom automatski stvoriti u sebi kod za povezivanje sa kontrolerom.

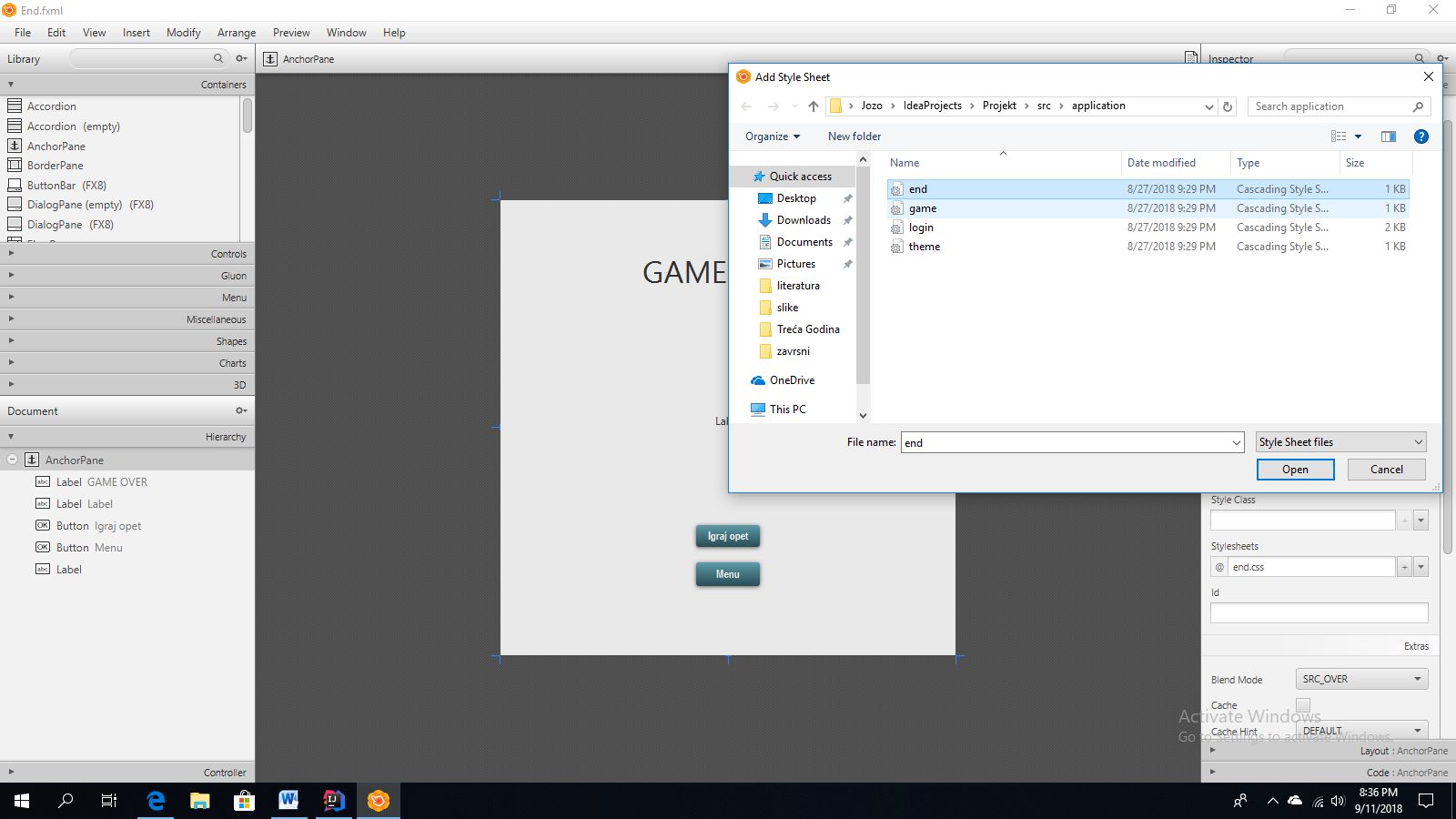


Slika 4.4: Postavljanje ID-a i funkcije za određeni GUI element

Kada je kontroler postavljen, potrebno je povezati elemente koji su definirani u *Menu.controller* kontroleru sa *FXML* datotekom. Pomoću Scene Buildera to se radi tako što se odabere „Code“ prozor i u polje „fx:id“ se upiše ID elementa u ovom slučaju *Button-a* *Options* i u polje „On Action“ se odabere funkcija iz kontrolera koja je predviđena da se izvrši na korisnikovu interakciju s *Options Button-om.*

**4.3 Umetanje CSS-a pomoću Scene Builder-a**

Scene Builder nudi mogućnost umetanja *CSS* koda u *FXML* datoteku. Da bi se to uradilo potrebno je imati *CSS* datoteku u projektu.



Slika 4.5: Postavljane CSS-a pomoću Scene Builder-a

Klikom na element roditelja „*AnchorPane*“ s desne strane u prozoru „*Properties“* otvara se mogućnost ubacivanja *CSS-a*. Pritiskom na znak “+“ u polju Stylesheets otvara se izbornik koji navigira na zadani projekt , te se odabere CSS datoteka koja je potrebna.

**5 MULTIMEDIJA**

**5.1 Java Media**

U ovom podpoglavlju predstavljeni su osnovni pojmovi o Java Media. Formati koje podržava Java Media su: AAC,AIFF,WAV,MP3. Podrška za specifične formate ovisi o platformi, neke značajke i formati ne treba posebno instalirati dok neki zahtjevaju dodatne programske podrške. Projekt na kojem se prikazuje upotreba Java Media koristi MP3 i WAV formate.

Java Media ili skraćeno rečeno *Media API* sastoji se od sljedećih klasa:

* AudioClip
* Media
* MediaPlayer
* MediaView
* MediaErrorEvent
* MediaException

*AudioClip* se koristi za reprodociranje zvučnih efekata, *Media* predstavlja služi samo za čitanje datoteke koju želimo reproducirati pomoću *MediaPlayer-a* za zvuk i *MediaView-a* za video. *MediaErrorEvent* je događaj koji se odradi ako dođe do greške u *Media API-u* npr. nedostaje datoteka za reprodukciju, *MediaException* je *Exception* kojeg baca *Media API*.[8]

U projektu se koriste *AudioClip*,*MediaPlayer* i *Media*.

**5.2 Upotreba AudioClip-a**

*AudioClip* služi za reproduciranje kraćih zvučnih datoteka npr. korištenje zvučnih efekata u interakciji s događajem JavaFX elementa. Poželjno je koristi *WAV* format datoteke radi njene izuzetno male veličine. Da bi se AudioClip mogao koristiti, potrebno je koristiti pomoćnu klasu Applet.

|  |
| --- |
| public void btn1Click(ActionEvent e) {  if (btn1.getText().equals(current.getTocanodgovor())) {  clipY.play();  rez += MenuController.tezina;  score.setText("Score:\n" + rez + "");  zamjeni();  } else {  zivot--;  life.setText("Zivot : " + zivot);  clipN.play();  if (zivot < 1) {  timer.cancel();  Stage stage = (Stage) ((Button) e.getSource()).getScene().getWindow();  try {  Parent root = FXMLLoader.load(getClass().getResource("/application/End.fxml"));  Scene scene = new Scene(root);  scene.getStylesheets().add(Main.class.getResource("end.css").toExternalForm());  stage.setScene(scene);  stage.show();  return;  } catch (IOException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }  }  zamjeni();  }  }  @Override  public void initialize(URL location, ResourceBundle resources) {  rez = 0;  zivot = 3;  vrijeme = true;  tekst.setWrapText(true);  score.setText("Score:\n" + rez + "");  life.setText("Zivot : " + zivot);  pitanja.addAll(Pitanje.getByDifficulty(MenuController.tezinaS));  postavi();  zamjeni();  clipY = Applet.newAudioClip(GameController.class.getResource("/sounds/musical002.wav"));  clipN = Applet.newAudioClip(GameController.class.getResource("/sounds/cartoon006.wav"));  setTimer();  } |

Kod 5.1: Upotreba AudioClip-a u GameController.java

U navedenom kodu vidimo uporabu AudioClip-a u kontroleru GameController. Da bi se AudioClip mogao koristiti potrebno je koristiti statičku funkciju klase Applet .newAudioClip() koja prima parametre putanje koja vodi na mjesto u kojem je smještena *WAV* datotekau ovom slučaju "/sounds/musical002.wav" i "/sounds/cartoon006.wav". Instanca klase clipY služit će za zvučni efekt prilikom korisnikovog točnog odgovora, a clipN prilikom netočnog.

Kada su datoteke postavljenje potrebno je složiti kodsku logiku kojom će se one reproducirati. Na primjeru događaja na klik jednog od *Button-a* s ponuđenim odgovorom public void btn1Click(ActionEvent e) u slučaju točnog odgovora reproducira se AudioClip clipY metodom .play(), te se logika koda nastavlja izvršavati (uvećavanje rezultata rez za određeni broj s obzirom na težinu MenuController.tezina i promjena pitanja funkcijom zamjeni() ). Kada korisnik odgovori pogrešno reporducira se clipN istom metodom .play(), te se nastavlja izvršavati logika(umanjivanje zivot za jedan,provjeravanje da li je korisnik izgubio i promjena pitanja ).

**5.3 Upotreba MediaPlayer-a**

*MediaPlayer* je dio Java *Media API-a* koji služi za reproduciranje glazbe duljeg trajanja. *MediaPlayer* sadrži veliku zbriku metoda za njeno upravljanje.Da bi se *MediaPlayer* mogla koristiti potrebno je imati pomoćnu klasu *Media* koja služi za učitavanje datoteke u *MediaPlayer*. To se može najlakše shvatiti kao ubacivanje *CD-a* u radio. U projektu se koristi *MP3* format za demonstraciju *MediaPlayer-a*.

|  |
| --- |
| @FXML  private ImageView imgg;  @FXML  private ChoiceBox<String> cb\_menu;  public static Media media = new Media(new File("chocobo\_song.mp3").toURI().toString());  public static MediaPlayer mediaPlayer = new MediaPlayer(media);  public static boolean running = false;  public void promjeni(MouseEvent e) {  if (mediaPlayer.isMute()) {  mediaPlayer.setMute(false);  Image image = new Image(getClass().getResource("/slike/sazvukom.png").toExternalForm());  imgg.setImage(image);  } else {  mediaPlayer.setMute(true);  Image image = new Image(getClass().getResource("/slike/bezzvuka.png").toExternalForm());  imgg.setImage(image);  }  }  @Override  public void initialize(URL location, ResourceBundle resources) {  .  .  .  logeduser\_menu.setText("Welcome " + LoginController.logeduser.getUsername());  if (running == false) {  mediaPlayer.play();  running = true;  }  mediaPlayer.setOnEndOfMedia(new Runnable() {  public void run() {  mediaPlayer.stop();  mediaPlayer.play();  running = true;  }  });  if (mediaPlayer.isMute()) {  Image image = new Image(getClass().getResource("/slike/bezzvuka.png").toExternalForm());  imgg.setImage(image);  } else {  Image image = new Image(getClass().getResource("/slike/sazvukom.png").toExternalForm());  imgg.setImage(image);  }  } |

Kod 5.2: Postavljanje MediaPlayer-a u MenuController.java

U navedenom kodu prikazane su varijable i funkcije koje sudjeluju u reprodukciji MediaPlayer-a. Već u samom inicijaliziranju varijabla mediaPlayer i media prikazano je postavljanje datoteke "chocobo\_song.mp3". Nakon toga u funkciji initalize() ostvarena je početna interakcija mediaPlayer i ImageView-a imgg koja se nastavlja kroz događaj koji se poziva klikom miša na imgg funkcijom promjeni(). Ako je mediaPlayer stišan imgg prikazuje sliku precrtanog zvučnika, a ako svira prikazuje sliku zvučnika. Manipulacija MediaPlayer-a odvija se metodama .stop() i .play(). mediaPlayer treba biti statička varijabla kako bi mogli s njome manipulirati kroz ostale kontrolere gdje želimo da glazba svira. Statičko svojstvo mediaPlayer najbolje je prikazano pri povratku na Menu.fxml , ukoliko varijabla running ima vrijednost true mediaPlayer će svirati, u suprotnom bit će zaustavljena.

|  |
| --- |
| Public void initialize(){  .  .  MenuController.mediaPlayer.setOnEndOfMedia(new Runnable() {  public void run() {  MenuController.mediaPlayer.stop();  MenuController.mediaPlayer.play();  MenuController.running = true;  }  });  .  .  . |

Kod 5.3: Upravljanje MediaPlayer-om iz drugog kontrolera

Postavljanje statičkog MediaPlayer-a dobiva se mogućnost njenim upravljanjem kroz druge kontrolere. Na navedenom kodu prikazan je dio koda iz *DrugiController* na čijem primjeru vidimo pozivanje jednog „slušača“ .setOnEndOfMedia() koji se aktivira kad mediaPlayer završi u čijem se kodu odvija ponovno pokretanje mediaPlayer. Isti postupak se nalazi i u *HighscoreConroller*, te samim time statičko postavljanje MediaPlayer-a pokazuje još svojih prednosti.

**6 DRETVE**

**6.1 Što su dretve i čemu služe**

Izvršni program je niz instrukcija mikroprocesoru smještenih u datoteku. Pored instrukcija izvršni program obično sadrži i podatke (inicijalizirane globalne varijable). Proces je instanca izvršnog programa koja se izvršava u memoriji računala. Govorimo o instanci stoga što svako pokretanje programa kreira novi proces: ako deset korisnika pokrenu *VI Editor* bit će kreirano deset različitih procesa u memoriji. Svaki proces ima svoj zaseban memorijski prostor u kojem se nalazi kod procesa i podaci (varijable, dinamički alocirana polja itd.) koje on koristi.

Različiti procesi izvršavaju se prividno paralelno tako što operacijski sustav alocira hardwareske resurse svakom pojedinom procesu tek jedan kratak vremenski interval. Nakon što je proces iskoristio svoj interval, prestaje se izvršavati i prelazi u pasivno stanje u kojem čeka da mu operacijski sustav ponovo dozvoli izvršavanje. Budući da je vremenski interval koji se procesima dodjeljuje vrlo kratak stvara se privid paralelnosti. Na višeprocesorskim računalima paralelizam može biti realan.

Procesi koji se (prividno) paraleleno izvršavaju u memoriji računala posve su neovisni jedan od drugoga. To je posljedica činjenice da ne dijele zajednički memorijski prostor i da svaki izvršava svoj skup instrukcija, neovisno o drugim procesima. Dva se procesa mogu dovesti u komunikaciju tek posebnim tehnikama u kojim se alocira dio memorije koju će moći zajednički koristiti.

Dretve predstavljaju oblik paralelizacije na razini procesa. Svaki se proces može podijeliti u jedan broj dretvi koje se izvršavaju prividno paralelno, na isti način kao i procesi. To znači da je kod procesa podijeljen u više podskupova koji se izvršavaju prividno paralelno. Za razliku od procesa, sve dretve unutar jednog procesa dijele zajedničku memoriju što omogućava efikasnu komunikaciju između različitih dretvi.

Sposobnost kreiranja dretvi unutar procesa svojstvo je operacijskog sustava. Da bi se u Java-programu mogle koristiti dretve operacijski sustav ih mora podržavati. Danas većina popularnih operacijskih danas ima podršku za višedretvene programe (*multithreading*).

Dretve su vrlo korisne u primjenama. Na primjer, one omogućavaju web pregledniku da istovremeno vrši komunikaciju putem mreže i prikazuje svoj sadržaj na zaslonu; web server mora imati mogućnost paralelnog procesiranja različitih zahtjeva, itd. U Javinim programima koji imaju grafičko sučelje sakupljanje događaja vrši se u posebnoj drevi. Isto tako, garbage collector radi u zasebnoj dretvi.

Uporeba dretvi učestala je u industriji računalnih igara,stoga kompleksnije igre zahtjevaju jače i brze procesore kako bi dretve bolje obavljale zadane zadatke. Projekt na kojem se bazira ovaj rad sama po sebi je računalna igrica stoga i ona sadrži dretve.[9]

**6.2 Uporaba i rad s dretvama**

U ovom podpoglavlju bit će prikazano kako se koriste dretve u zadanom projektu, kakva je njihova korist i kako se odvija interakcija s multimedijskim komponentama JavaFX-a(slika).

|  |
| --- |
| @Override  public void initialize(URL location, ResourceBundle resources) {  rez = 0;  zivot = 3;  vrijeme = true;  tekst.setWrapText(true);  score.setText("Score:\n" + rez + "");  life.setText("Zivot : " + zivot);  pitanja.addAll(Pitanje.getByDifficulty(MenuController.tezinaS));  postavi();  zamjeni();  clipY = Applet.newAudioClip(GameController.class.getResource("/sounds/musical002.wav"));  clipN = Applet.newAudioClip(GameController.class.getResource("/sounds/cartoon006.wav"));  setTimer();  }  public void setTimer() {  timer.scheduleAtFixedRate(new TimerTask() {  @Override  public void run() {  if (traka.getWidth() <= 0) {  timer.cancel();  vrijeme = false;  Stage stage = (Stage) tekst.getScene().getWindow();  try {  Parent root = FXMLLoader.load(getClass().getResource("/application/End.fxml"));  Scene scene = new Scene(root);  scene.getStylesheets().add(Main.class.getResource("end.css").toExternalForm());  Platform.runLater(() -> stage.setScene(scene));  } catch (IOException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }  }  Platform.runLater(() -> traka.setWidth(traka.getWidth() - 0.6));  }  }, 0, 100);  } |

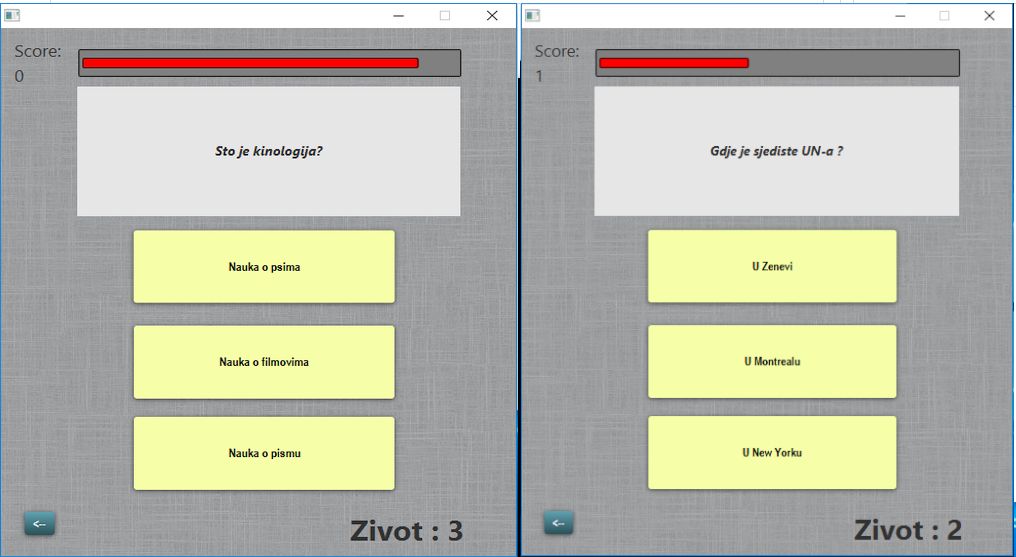
Kod 6.1: Postavljanje i pozivanje dretve

Na navedenom kodu prikazana je funkcija setTimer() koja služi za postavljanje dretve i implementaciju logike koda.Varijabla timer klase *Timer* predstavlja dretvu. *Setter-e* nije moguće koristiti u dretvi samo njihovim pozivom jer se ne može samostalno direktno utjecati na instance klasa Jave iz dretve već tome služi klasa *Platform* i njena statička funkcija .runLater() kao posrednik između dretve i glavnog procesa. Zadatak setTimer() funkcije je umanjivati traku za određeno vrijeme timer koje s metodom .scheduleAtFixedRate(logika(),zastoj,period) gdje logika() predstavlja kod koji će on izvršavati, zastoj pauzu u milisekundama i period nakon koliko milisekundi će timer odraditi logika(). Kada duljina traka dođe na 0 koristi se metoda .cancel() koja će ugasiti dretvu timer i vodi nas na kraj igre.

|  |
| --- |
| public void btn2Click(ActionEvent e) {  if (btn2.getText().equals(current.getTocanodgovor())) {  clipY.play();  rez += MenuController.tezina;  score.setText("Score:\n" + rez + "");  zamjeni();  } else {  zivot--;  life.setText("Zivot : " + zivot);  clipN.play();  if (zivot < 1) {  timer.cancel();  Stage stage = (Stage) ((Button) e.getSource()).getScene().getWindow();  try {  Parent root = FXMLLoader.load(getClass().getResource("/application/End.fxml"));  Scene scene = new Scene(root);  scene.getStylesheets().add(Main.class.getResource("end.css").toExternalForm());  stage.setScene(scene);  stage.show();  return;  } catch (IOException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }  }  zamjeni();  } |

Kod 6.2: Zatvaranje dretve u slučaju gubitka života

Na funkciji btn2Click(ActionEvent e) koja služi da se logika odvije kada korisnik pritisne na njezin Button u slučaju propadanja poziva se .cancel() na timer iz tog razloga jer dretva mora biti ugašena u svakom slučaju kako ona ne bi ostala raditi u pozadini i opterećivala procesor. Zbog bolje optimizacije poželjno je ugasiti dretvu u svakom slučaju kad ona više nije potrebna.



Slika 6.1: Isticanje vremenske trake primjenom dretve

**7 ZAKLJUČAK**

Iz svega navedenog vidimo da za napraviti jednu ovakvu igricu zahtjeva prošierno znanje ne samo na području poznavanja Jave i JavaFX-a, već i dodatnih alata multimedije kao što je Scene Builder za lakši dizajn i sliku,Media API za zvuk, te dretve za interakciju navedenih komponenata.

Poznavanje jednog , a ne znati drugo nikako ne može proći. Sreća je što postoji alat kao Scene Builder koji olakšava dizajn i ubacivanje slika na JavaFX pa korisnicima koji se prvi put susreću s JavaFX okruženjem je olakšan posao. Naravno bez elementarnog poznavanja Jave i objektno orijentiranog programiranja ovakav jedan projekt nije moguće napraviti.

JavaFX je okruženje koje omogućuje laganu upotrebu multimedije, te bi za razvoj desktop aplikacije s multimedijskim zahtjevima trebao biti prvi i najbolji izbor. Što se tiče same kvalitete i uporabe multimedije zaključak je da se uz malo mašte,truda i optimizacije može izgraditi jedna kompletna aplikacija koja zahtjeva jednostavne i kompleksne multimedijske zahtjeve.

**8 LITERATURA**

[1] Codelate, “History of JavaFX.”

[2] Wikipedia, “Povijest JavaFX.” [Online]. Available: https://en.wikipedia.org/wiki/JavaFX.

[3] J. Potts, N. Hildebrandt, J. Gordon, and C. Castillo, “Getting Started with JavaFX Release 8,” no. August, p. 68, 2014.

[4] Oracle, “Introduction to FXML.” [Online]. Available: https://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction\_to\_fxml.html.

[5] Wikipedia, “CSS.” [Online]. Available: https://hr.wikipedia.org/wiki/CSS.

[6] F. Maja Čić, “Event Controllers.” [Online]. Available: http://laris.fesb.hr/java/mouse.htm.

[7] Oracle, “Get Started with Scene Builder.”

[8] A. Pomarolli, “JavaFX Programming Cookbook,” p. 219, 2016.

[9] PMF Zagreb, “Dretve.” [Online]. Available: https://web.math.pmf.unizg.hr/nastava/rp2/pred8/pred8.html.

**8 POPIS SLIKA**

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2

Type chapter title (level 2)2