UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Vranešič

JAVAFX

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor:
prof. dr. Saša Divjak

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00481/2009
Datum: 15.10.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALEŠ VRANEŠIČ**

Naslov: **JAVAFX**

JAVAFX

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Delo naj predstavi JavaFX, ki sodi v sklop tehnologij za razvoj bogatih spletnih aplikacij. Opišite obstoječe alternativne tehnologije. Opišite arhitekturo, platformo JavaFX in razložite sintakso JavaFX Script. Preskusite navedeno tehnologijo s programom, ki kaže možnosti uporabe za pripravo animacij in interakcije z uporabnikom, kakršne zasledimo pri simulacijah in igrah. Podajte tudi vizijo prihodnosti tehnologije JavaFX.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00481/2009
Datum: 15.10.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALEŠ VRANEŠIČ**

Naslov: **JAVAFX**

JAVAFX

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Delo naj predstavi JavaFX, ki sodi v sklop tehnologij za razvoj bogatih spletnih aplikacij. Opišite obstoječe alternativne tehnologije. Opišite arhitekturo, platformo JavaFX in razložite sintakso JavaFX Script. Preskusite navedeno tehnologijo s programom, ki kaže možnosti uporabe za pripravo animacij in interakcije z uporabnikom, kakršne zasledimo pri simulacijah in igrah. Podajte tudi vizijo prihodnosti tehnologije JavaFX.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a **Aleš Vranešič**,

z vpisno številko **63040348**,

sem avtor/-ica diplomskega dela z naslovom:

JAVAFX

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  prof. dr. Saše Divjaka

  in somentorstvom (naziv, ime in priimek)

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne **10.03.2010** Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju prof. dr. Saši Divjaku za mentorstvo ter za nasvete, ki mi jih je nudil pri izdelavi diplomskega dela.

Zahvaljujem se svoji družini za podporo in potrpežljivost v času študija.

Zahvaljujem se tudi svojim prijateljem, ker so me spodbujali v času pisanja diplome.

Kazalo

Seznam kratic

AJAX	*(angl.) Asynchronous JavaScript and XML; Asinhroni JavaScript in Xml.*
DOM	*(angl.) Document Object Model; Specifikacija, kako so objekti na spletni strani (besedilo, slike, povezave ...) predstavljeni*
PHP	*(angl.) Personal Home Page Tools; Skriptni jezik uporaben za razvoj spletnih strani in se ga lahko vključi v HTML.*
RIA	*(angl.) Rich Internet applications; Bogate spletne aplikacije.*
CSS	*(angl.) Cascading Style Sheets; Prekrivne predloge.*
HTML	*(angl.) HyperText Markup Language; Označevalni jezik za oblikovanje večpredstavnostnih dokumentov.*
GPS	*(angl.) Global Positioning System; Sistem za pozicioniranje.*
XML	*(angl.) Extensible Markup Language; Razširljivi označevalni jezik.*
JDK	*(angl.) Java Development Kit; Orodja za razvoj v Javi.*
SDK	*(angl.) Software development kit; Orodja za razvoj aplikacij.*
IDE	*(angl.) Integrated development environment; Integrirano razvojno okolje.*
JPG	*(angl.) Joint Photographic Experts Group; Rastrski slikovni format.*
PNG	*(angl.) Portable Network Graphics; Rastrski slikovni format.*
MP3	*(angl.) MPEG-1 Audio Layer 3; Oblika zapisa glasbe z izgubami.*
WMA	*(angl.) Windows Media Audio; Oblika zapisa glasbe z izgubami.*
WMV	*(angl.) Windows Media Video; Stisnjen video format*
720p	*(angl.) High-definition television; 720p spada v kategorijo video formata visoke ločljivosti.*

Povzetek

V uvodnem poglavju sem predstavil naraščujoči trend uporabe RIA ter široko uporabo te tehnologije na tržišču.

Drugo poglavje govori splošno o RIA, njihovih lastnostih, delovanju le-teh ter prednostih in slabostih, ki jih imajo z vidika navadnih spletnih aplikacij.

V tretjem poglavju sem naštel in opisal nekaj glavnih tehnologij za razvoj RIA, ki so danes na trgu. Opisal sem tehnologije AdobeFlex, Microsoft Silverlight, AJAX ter JavaFX. Vsako sem predstavil, opisal nekaj glavnih značilnosti ter njihovo delovanje.

Četrto poglavje podrobno opisuje tehnologijo JavaFX, ki predstavlja osrednje sporočilo diplomske naloge. Opisal sem arhitekturo, platformo JavaFX, glavne značilnost ter sintakso jezika JavaFX. Sintaksa JavaFX Script je razložena skozi kratke primere.

V sklopu diplomske naloge sem v petem poglavju predstavil preprosto 2D igro, ki sem jo razvil. Z njo sem hotel prikazati, kaj lahko s tehnologijo JavaFX razvijemo v relativno kratkem času.

Napoved prihodnosti tehnologije JavaFX sem predstavil v šestem poglavju, ki je za zdaj negotova in ogrožena z veliko konkurence, hkrati pa je še neuveljavljena ter v začetni fazi razvoja.

V zadnjem poglavju sem podal ugotovitve in sklep, do katerih sem prišel med izdelavo diplomskega dela.

Ključne besede:

bogata spletna aplikacija
tehnologije za razvoj RIA
sintaksa
animiranje
igra

Abstract

In the first chapter will I present growing trend of using RIA and its ability for using this tehnology in a lot different devices that are available in the market.

The second chapter is about RIA in general, their properties, advantages and disadvantages from the point of view of ordinary internet applications.

In chapter three I listed and described some of the main technologies for RIA development, which are on the market today. I described AdobeFlex, Microsoft Silverlight, AJAX and JavoFX technology. Every technology for developing RIAs that I listed, is presented and shown some of the main characteristics and performances.

The fourth chapter describes in detail JavaFX technology for developing RIAs which represents the common thread in my graduation thesis. I described the architecture platform JaveFX, the main feature of the language and syntax of JavaFX, which is explained through short examples.

Within the diploma thesis I presented in the fifth section a simple 2D game that I developed. With it I wanted to show what can be developed with JavaFX technology in a relatively short time.

In the sixth chapter I presented the future of JaveFX technology which is at present time still uncertain and threatened with plenty of competition. JavaFX is very new technology therefore is still unestablished and in the early stages of development.

Final chapter includes my findings and conclusion based on observations and practical work during this dissertation.

Keywords:

rich internet application
technologies for developing RIA
syntax
animation
game

1. Uvod

Bogate spletne aplikacije (RIA) se danes že uveljavljajo v vsakdanjem življenju. Trend nakazuje, da se bo uporaba le-teh samo še povečevala. K temu bo pripomogla čedalje večja uporaba ne samo internetnih in namiznih aplikacij ampak številnih mobilnih naprav, ki jih je na tržišču vedno več. RIA nudijo preformanse in hitrost namiznih aplikacij. Tehnologije za razvoj RIA so Adobe Flex, Ajax, Microsoft Silverlight ... Med njimi je tudi JavaFX, ki je novejša ter še neuveljavljena. Ravno zaradi tega je zanimiva, saj je trg nasičen s tehnologijami za razvoj RIA. Kljub veliki izbiri razvijalcev pri uporabi tehnologije za razvoj RIA obstaja za JavaFX upanje, da obstane na tržišču oziroma vanj tudi prodre. V zaledju ima močno razširjeno ter uveljavljeno Javo, kar predstavlja prednost tehnologije JavaFX pred ostalimi tehnologijami za razvoj spletnih aplikacij. Ponuja dokaj preprosto sintakso in je intuitivna pri razvoju programske kode. Ravno preprostost je tisto, kar razvijalcem omogoča krajši čas razvoja aplikacij ter povečuje preglednost nad ustvarjeno kodo. Kljub svoji preprostosti je mogoče razviti vizualno zelo všečne aplikacije, ki so tudi tehnično zadovoljive, gledano z vidika hitrosti delovanja. Tehnologija JavaFX tudi omogoča prenosljivost aplikacije, saj se lahko enaka aplikacija izvaja kot namizna, spletna ali pa kot mobilna aplikacija.

2. Bogate spletne aplikacije

Spletni razvijalci imajo danes na voljo široko paleto tehnologij za razvoj bogatih spletnih aplikacij. “Te tehnologije se med seboj razlikujejo po bogatosti uporabniške izkušnje, dosegu, integriranosti v samo razvijalsko platformo in ne nazadnje po razvijalski izkušnji [1].”

2. 1. Splošno o RIA

Termin bogata spletna aplikacija (Rich Internet Application, RIA v nadaljevanju) je bil prvič predstavljen pri predstavitvi programskega orodja Macromedia Flash MX, podjetja Macromedia. RIA ponuja podobne funkcionalnosti, kot jih imajo namizne aplikacije. Nekatere RIA lahko delujejo znotraj brskalnikov, druge pa samostojno. RIA poskušajo biti neodvisne od brskalnikov pa tudi sistemsko. Ponuja obilico interakcijskih rešitev, ki so bolj “bogate” kot pri običajnih spletnih straneh [2].

Nudijo bogate ter intuitativne grafične vmesnike, na podlagi česar postanejo računalniške aplikacije dostopnejše in bolj vabljive, hkrati pa ponujajo bogat nabor interakcij za učinkovito rokovanje s podatki. Omogočajo preformanse in hitrost namiznih aplikacij ter sinhrono in asinhrono povezljivost med ljudmi in podatki. Pri izvedbi čim boljše interakcije pomagajo grafični elementi, zvok in video ter dobro zasnovani vmesniki, ki naj bi bili jasni in natančni.

V primerjavi s tradicionalnimi spletnimi aplikacijami je mogoče s pomočjo te tehnologije razviti spletno aplikacijo, ki na eni strani razbremeni strežnik, na drugi pa zagotavlja bogat uporabniški vmesnik na odjemalcu. Tehnologije za razvoj RIA so: Flex, Ajax, Microsoft Silverlight, JavaFX.

Primeri strani v stilu RIA:

- Google Maps (spletni zemljevidi)
- GAP (spletno nakupovanje)
- Odeo (spletni audio/video kanali)
- Flickr (organiziranje in souporaba (sharing) fotografij)

Koliko “bogastva” (v nadaljevanju bogastva) naj dodamo:

- Ne dodajajmo preveč bogastva.
- Večini ljudi obstoječi model, temelječ na straneh in z omejeno interaktivnostjo, zadošča.
- Prilagajanje novim pristopom terja čas.
- Tudi dogovori se počasi pojavljajo in uveljavljajo.
- Bogastvo dodajajmo počasi, le tam, kjer bistveno izboljša uporabnost.
- Raziskave uporabnikov in redno preverjanje uporabnosti nam pomaga pri določanju, koliko bogastva je primernega v danem času. [3]

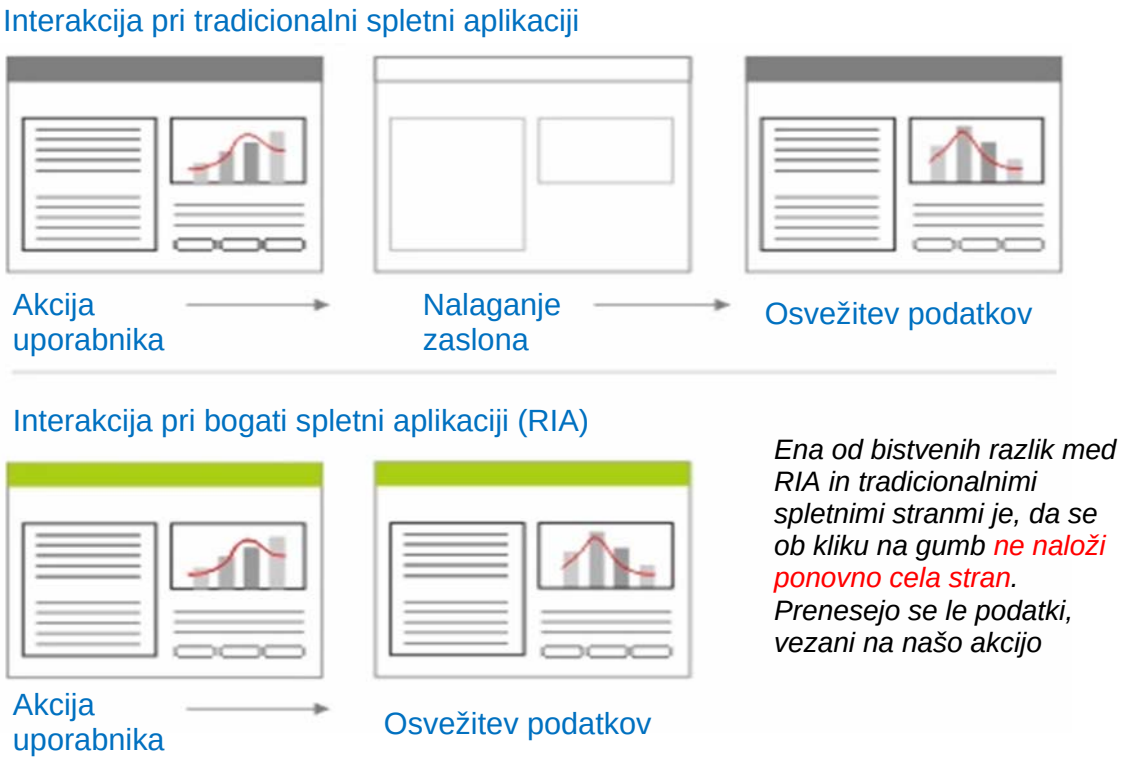
Uporabniki želijo, da so spletne aplikacije čim bolj podobne namiznim, torej več interakcije, večje možnosti ter večjo odzivnost. Bogastvo lahko dodajamo tako, da uporabniki direktno interaktirajo z elementi na strani npr. urejanje teksta, vlečenje in spuščanje grafičnih elementov,

premikanje zemljevida. Ljudje morajo biti sposobni odkrivanja kontrol in njihovega načina uporabe. Uporabnik uporablja zaslonski prikaz bolj učinkovito, če se osnovni strukturni elementi prikaza ne spreminjajo prepogosto [4].

Bogate spletne aplikacije omogočajo sinhrone izmenjave podatkov v realnem času, kar je uporabno pri avdio, video konferencah, tekstovnih klepetalnicah, nadzoru kritičnih podatkov v realnem času … "Večina spletnega prometa je doslej temeljila na modelu zahtevek-odgovor. Podatki so shranjeni pasivno, dokler jih ne zahtevamo."

"Tradicionalne HTML strani vsebujejo podatke kot predstavitveno informacijo, RIA tipično izmenjujejo s strežnikom le pakete podatkov. Skupaj s procesno močjo odjemalca je tako splošen odziv hitrejši [5]."

2. 2. Stalnost in odziv



Slika 1: Razlike interakcije med tradicionalno spletno aplikacijo in RIA

Z uporabo interneta smo si ljudje razvili svoj miselni model o delovanju le-tega. Povprečen uporabnik interneta si ga predstavlja nekako takole: vsak klik nas pripelje do nove strani, gumb »back« pa nas vrne na prejšnjo stran. RIA omogoča pri izdelavi internetnih aplikacij mnogo več, kot je naš miselni model. Pri razvoju RIA je potrebno biti previden, saj lahko nepremišljena uporaba vseh možnosti, ki jih ponuja RIA, spletno stran naredi nepregledno. Uporabnik uporablja zaslonski prikaz bolj učinkovito, če se osnovni strukturni elementi prikaza ne spreminjajo prepogosto. Zato ne poskušajmo vsega dodati na eno stran samo zato, ker je to mogoče.

Pomembno pa je tudi, da ljudem ne otežujemo uporabe interneta, kot je npr. onemogočitev gumba “back” [3].

2. 3. Prednosti in slabosti RIA

Prednosti:

- delujejo lahko na različnih napravah (npr. osebni računalniki, mobilni telefoni, dlančniki …)
- niso vezani na brskalnike (za delovanje ne potrebujejo brsklalnikov)
- RIA lahko delajo brez brskalnika, kot namizna aplikacija. RIA potrebujejo plug in, sandbox (pekkovnik) ali inštalacijo virual machine (virtualnega stroja) na uporabnikovem računalniku. Ker so RIA aplikacije ponavadi manjše kot tradicionalne namizne aplikacije, je možno (share) med uporabnikom in serverjem. Omogoča tudi, da je aplikacija dostopna, čeprav uporabnik ni prijavljen (offline).
- bogatost namizja
- možnost sinhrone izmenjave podatkov v realnem času
- dostopnost interneta
- radikalna povezljivost
- ne rabi uporabljati brskalnika
- bogata vsebina
- visoka inrteraktivnost
- gibanje, zvok, video
- osveževanje dela strani
- validacija in fiter že pri klientu

[2]

Slabosti:

- slabša varnost kot pri tradicionalnih spletnih straneh

- bolj kompleksne od tradicionalnih spletnih aplikacij

- Največji problem je varnost. Čeprav so RIA aplikacije bolj varne od tradicionalnih aplikacij, so že po svoji naravi manj varne od spletnih aplikacij.

- Velika verjetnost pri razvijalcih je, da uporabijo preveč interaktivnosti, bogastva spletne strani, saj je tako dodajane upravičljivo le, če se s tem bistvno izboljša uporabnost. Povprečni uporabnik je namreč povsem zadovoljen z današnjimi spletnimi stranmi z omejeno interaktivnostjo, zato je pri vsaki uporabi novosti potrebno upoštevati želje in navade uporabnikov. Težave lahko nastopijo tudi takrat, ko uporabnik ne ve, kako uporabiti nove kontrole, zato morajo biti ljudje sposobni odkrivanja njihovega načina uporabe. Uporabnik uporablja zaslonski prikaz bolj učinkovito, če se osnovni strukturni elementi prikaza ne speminjajo prepogosto.

Glede na to, da se stran osvežuje delno, je potrbno določeno spremembo, ki jo je uporabnik naredil, prikazati, da bo to opazil (npr. to lahko dosežemo s povratno informacijo, ki uporabniku sporoči spremembo – tekstovno sporočilo, drugačna barva gumba …)[5].

Slika 2: Povratna informacija, ki uporabnika obvesti o določeni spremembi

- Razvijanje RIA aplikacije je ponavadi obsežnejše kot razvijanje tradicionalnih spletnih strani.

[2]

3. Tehnologije za razvoj RIA

Obdobje RIA se je pričelo s pojavom Java-apletov, ki so uporabniku nudili interakcijo na strani uporabnika, ter spodobno uporabniško izkušnjo. Vendar sta čas ter boljša tehnologija, kot so Flash, Ajax ..., kmalu zamenjala široko uporabo Java-apletov. RIA aplikacije imajo značilnosti namiznih aplikacij in so večinoma predstavljene s pomočjo AJAX ogrodja. Danes je spekter tehnologij za izdelavo RIA dokaj pester. V nadaljevanju bom predstavil le nekaj od teh, ki so najbolj uporabljene ter prespektivne.

3. 1. AdobeFlex

Adobe Flex je brezplačno, odprto kodno orodje za razvoj bogatih spletnih aplikacij. Adobe Flex je prva od tehnologij za razvoj bogatih spletnih aplikacij podjetja Adobe Systems. Flex se je pojavil na trgu leta 2004, takrat še pod okriljem Macromedie.

Definiranje, kaj Flex je, je dokaj zapleteno, saj Flex v bistvu vsebuje konbinacije različnih tehnologij. Flex ni somostojen programski produkt, ampak je sestavljen iz 4 glavnih delov:

- ActionScript 3 in MXML
- Flex SDK
- Flex Builder
- Flash Player

[6]

Flex uporablja MXML deklarativni jezik, ki je podoben XML, namenjen gradnji uporabniškega vmesnika ter objektno orientirani programski jezik ActionScript za izvedbo logičnega vmesnika (za samo interaktivnost spletne aplikacije). ActionScript je tudi glavni skriptni jezik pri Adobe Flash platformi. Za lažje upravljanje izgleda se lahko uporablja tudi CSS, ki močno olajša delo programerja pri morebitnih spremembah izgleda. Programe Flex je mogoče razvijati z brezplačnim kompletom za razvoj programske opreme Flex SDK. Razvijalci pa lahko s programsko opremo Adobe Flex Builder™ 3, ki temelji na poznanem Eclipse ID-u (Integrated Development Environment), bistveno pospešijo razvoj. Omogoča interaktivno razhroščevanje in grafično oblikovanje uporabniškega vmesnika ter programiranje v ozadju. Hitro odjemalčevo izvajanje temelji na vseprisotnem Adobe Flash Player-ju. Adobe Flex temelji na platformi Adobe Flash, kar pomeni, da je danes prisoten na večini osebnih računalnikov (Flash ima okrog 99 % prisotnost) [7, 8, 9].

Flex omogoča razvoj visoko interaktivnih ter grafično bogatih aplikacij, ki izboljšujejo uporabniško izkušnjo. Flex aplikacije se lahko izvajajo v spletnem brskalniku s pomočjo izvajalnega okolja Flash Player. Izvajalno okolje AIR (Apollo Integraded Runtime) pa omogoča izvajanje Flex aplikacij kot namizne aplikacije in s tem lahko Flex aplikacije dostopajo do lokalnih podatkov in sistemskih sredstev na namizju. Za delovanje je potrebno le namestiti AIR izvajalno okolje na osebni računalnik.

Flex Mobile pa po novem omogoča poganjanje na mobilnih in drugih napravah [10].

Glavni format platforme Flash je FLV (Flash Video), ki je vsebniški format, kar pomeni, da sam ni video format, ampak za zapis slike in zvoka vsebuje druge formate in jih tako povezuje. Flex ponuja bogato podporo video, avdio in slikovnih formatov v primerjavi z MPEG-4 do MP4, #GP in MOV. Format FLV podpira za zapis zvoka format MP#, za zapis slike pa H.263 ali VP6 [11].

Flex trenutno podpira naslednje operacijske sisteme:

- Microsoft Windows 2000, Windows XP, Windows Vista, Windows 7
- Mac OS X
- GNU/Linux

Flex je razvojno ogrodje (framework) in orodje, ki ga lahko uporabimo za razvoj RIA. Te aplikacije se izvajajo preko odjemalskega zagonskega okolja Adobe Flash Player.

Nekaj napomembnejših razlik med Flex in Flash je:

- Flex framework: Flash ima svojo zbirko komponent, ki imajo podobne funkcionalnosti kot Flex SDK, vendar ne vključuje toliko komponent in ne vsebuje grafikonov, razporejevalnikov (layout container) ter ostalega, kar je uporabno pri razvoju večjih aplikacij.

- MXML: Možna uporaba značkovnega jezika MXML z uporabo jezika Flex v jeziku Flash ni možna, Flash uporablja isti Action Script 3 skriptni jezik kot Flex.

- Močno integrirano razvojno okolje (IDE): Flex Builder je bil razvit za razvojno orodje za razvoj aplikacij, Flash Authoring tool pa je prvotno razvit za izdelavo animacij. Za razvoj

RIA se lahko uporabi oba, vendar ima Flex Builder zmožnost skrivanja kode, močan razhroščevalnik ter na splošno močnejše razvojno orodje.

- Kot je bilo že napisano, se lahko za razvoj RIA uporablja tudi Flash, vendar potem ne moremo računati na prednosti orodja Flex Builder (skrivanje kode), uporabe značkovnega jezika MXML. Flash je zelo uporaben pri izdelavi animiranih videov, Flex Builder pa je zasnovan za izdelavo aplikacij in ne animacij, saj ne podpira risalnih orodij ter nima »časovnega traka« (timeline).

[6]

Preprosta Flex aplikacija: Hello World

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
        <mx:Label x="10" y="10" text="Output" id="output"/>
        <mx:Button x="10" y="36" label="Click Me" click="{output.text = 'Hello
        World '}"/>
</mx:Application>
```

3. 2. SilverLight

Microsoft je tehnologijo za razvoj bogatih spletnih aplikacij poslal na trg leta 2007, in sicer z izidom Microsoft Silverlight-a, ki podpira delovanje na različnih brskalnikih.

Microsoft Silverlight 3 omogoča enkraten razvoj kode ter možnost poganjanja tako v operacijskem sistemu Windows kakor tudi v Mac OS. V začetku leta 2009 pa so pri Microsoftu ponudili Moonlight okolje ter tako omogočili izvajanje Silverligth aplikacij na operacijskem sistemu Linux [12, 13].

Silverlight temelji na tehnologiji WPF (Windows Presentation Foundation) ter tako nudi vektorsko grafiko, animacije, multimedijo in interaktivnost v enem samem izvajalnem okolju. Predstavlja neposredno konkurenco Adobe Flashu, vendar je slednji bolj uveljavljen ter prav tako prisoten na vsakem osebnem računalniku. Grafični uporabniški vmesniki so deklarirani v XAML (aXtensible Application Markup Language) jeziku, samo interaktivnost pa je mogoče programirati v kateremkoli izmed .Net programskih jezikov.

Za poganjanje Silverlight aplikacij skrbi Silverlight CoreCLR (Core Common Language Runtime), ki ga moramo namestiti kot Silverlight dodateko ali vtičnik oziroma se pri operacijskem sistemu Windows ta vtičnik namesti samodejno preko storitve Windows Update. Silverlight ponuja podporo multimedijskim formatom JPG, PNG, WMAm MP3, WMV, 720p HD, H.264 ter VC-1 video [14].

Med najpomembnejšimi novostmi Silverlight 3 je omogočanje izvajanja Silverlight aplikacij kot namiznih aplikacij. Silverlight 3 naj bi bil tudi boljša platforma za igre, kar omogoča podpora za

3D grafiko, saj ima podporo za strojno pospeševanje s koriščenjem visokih zmogljivosti najnovejših grafičnih jeder. Pri Silverlight 3 so tudi povečali število gradnikov in izboljšali povezave s podatkovnimi viri, s čimer so izboljšali osnovo za pripravo grafično prijaznih interaktivnih aplikacij. Omogoča predvajanje filmskih posnetkov na svetovnem spletu (**Smooth Streamin**).

Poleg predstavitve Silverlight 3 je Microsoft predstavil Expression 3, ki je orodje za izdelavo programov v Silverlightu pa tudi za pripravo drugih spletnih vsebin. Program Expression SketchFlow, ki je prav tako novost, pa omogoča izdelavo prototipov aplikacij v Silverlightu. Pri zadnji verziji Silverlight je tudi izboljšana povezava z razvojnim orodjem Visual Studio, ki je prav tako uprabljen pri programih Silverlight. V kombinaciji z inovativnimi orodji v aplikacijah Expression Studio 3, ki nadgrajuje okolje Visual Studio in se povezuje s strežnikom Windows Server, imajo tako razvijalci na voljo vse za razvoj bogatih spletnih aplikacij [15, 16].

Konec leta 2009 je bila predstavljena Beta verzija Silverlight 4, ki prinaša številne novosti, kar delo dizajnerjev ter razvijalcev še bolj poenostavi [14, 17, 18, 19].
Je projekt odprtokodnega operacijskega sistema za ultraprenosne računalnike, dlančnike, pametne telefone, avtomobilske računalnike ... Arhitektura Moblin omogoča podporo večim platformam, osrednji del arhitekture. Z uporabo ogrodja Silverlight bodo razvijalci lahko aplikacije napisali le enkrat, izvajale pa se bodo lahko tako na operacijskem sistemu Windows kot na napravah Moblin, s čimer bodo aplikacije Silverlight dostopne večjemu številu uporabnikov in na več napravah, kot so osebni računalniki, televizijski sprejemniki in telefoni [20].

Osrednji del arhitekture je splošna plast, ki se imenuje Moblin jedro, strojna in uporabniška plast, ki je neodvisna in zagotavlja enotno pot za razvoj takih naprav. Pod Moblin jedrom se nahaja Linux kernel in gonilniki naprav strojne opreme, nad jedrom Moblin pa je uporabniški vmesnik in uporabniški interakcijski model za ciljno napravo [21].

3. 3. Ajax

Običajna spletna aplikacija za vsako spremembo na strani pošlje HTTP zahtevo, strežnik pa kot odgovor pošlje celotno stran. Tako mora brskalnik za vsako poizvedbo osvežiti celotno stran, zato prihaja do motečega obnavljanja spletne strani. Aplikacija je posledično počasna ter neprijazna do uporabnika.

Ajax (asinhroni JavaScript in XML) je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij, ki so podobne namiznim aplikacijam. Gledano s tehnološkega vidika je pristop Ajax tisti, ki je povzročil preskok pri prehodu v novo obdobje svetovnega spleta. Kot nov koncept pri razvoju spletnih aplikacij temelji na tehnologijah, poznanih že v preteklosti. Med seboj povezuje naslednje spletne tehnologije, ki so nameščene skoraj na vsakem računalniku, to so JavaScript, XHTML, CSS ... Problem obstaja s kompatibilnostjo na različnih brskalnikih, saj kar dela na primer na Firefox, morda ne deluje na Internet Explorer in obratno. AJAX ni nova tehnologija, je namreč skupek tehnik, kot so:

- XML izmenjava podatkov

- posredovanje metod Javascript odjemalcu
- DHTML widgets
- XML in XSLT

Ajax je kreiranje zahteve, pošiljanje zahteve na določeno stran in nato pridobivanje informacij, ki jih stran (strežnik) poda nazaj. Zahteva http je videti tako, da klient zahtevo pošlje strežniku vsakič, ko želi naložiti novo stran. Dinamične spletne strani pa utegnejo zahtevati, da strežnik pošlje kake podatke tudi medtem, ko uporabnik dela z eno samo stranjo (in ostaja na njej). Da bi skripta v JavaScriptu lahko pridobila podatke od strežnika, ne da bi brskalnik osvežil celotno stran, brskalnik ponuja objekt XMLHttpRequest, ki mu povemo URL zahteve in argumente, nakar sprožimo zahtevo, ki je lahko sinhrona ali asinhrona. Ob sinhroni zahtevi skripta in z njo celotna stran obstoji, dokler strežnik ne vrne odgovora. Ob asinhroni zahtevi pa skripta teče naprej, objektu XMLHttpRequest pa moramo predpisati funkcijo, ki naj jo pokliče, ko dobimo strežnikov odgovor. Ko strežnik odgovori, se torej pokliče določena funkcija, ki od XMLHttpRequesta izve vsebino strežnikovega odgovora, s katero potem stori, kar je potrebno.

Spletne aplikacije si izmenjujejo podatke s strežnikom asinhrono v ozadju, ne da bi ponovno naložile strani. Medtem ko aplikacija čaka podatke iz strežnika, lahko uporabnik nemoteno uporablja spletno stran. Ko so podatki pripravljeni, določena JavaScript funkcija prikaže podatke na strani brez potrebe po ponovnem nalaganju.

Asinhrona komunikacija s tem omogoča tekoče in hitrejše spremljanje ter spreminjanje vsebine na spletni strani. Poleg tega se zmanjša obremenitev spletnega strežnika, ker se veliko obdelave lahko naredi na strani odjemalca. Ajax aplikacije dajejo vtis, kot da v celoti tečejo na računalniku uporabnika.

Izmenjava podatkov je izvedena s pomočjo XMLHttp objektov ali s pomočjo Remote Scripting (v starejših brskalnikih, ki ne podpirajo Ajax tehnologije). Uporaba AJAX tehnologij je tipična za Web 2.0. Kljub imenu Ajax (asinhroni JavaScript + XML) uporaba tehnologij Javascript in XML ni pogoj za izvajanje Ajax-a. Pojem Ajax je Jesse James Garrett prvič uporabil leta 2005, nanaša se na naslednje tehnologije:

- HTML ali XHTML ter CSS za predstavitev,
- DOM (Document Object Model) za dinamično prikazovanje vsebine in interakcijo s podatki,
- XML (angl. Extensible Markup Language) in XSLT za delo s podatki,

  XML je razširljiv označevalni jezik podoben HTML, ki določa format za opisovanje strukturiranih podatkov ali arhitekturo za prenos oziroma njihovo izmenjavo podatkov med več omrežji. XML spreminja mnogo aspektov računalništva, še posebej na področju komuniciranja aplikacij in strežnikov. Da pa se ga tudi razširiti, saj si lahko sami izmislimo imena značk. Zelo je uporaben pri komunikaciji, saj ima zelo preprosto in pregledno zgradbo.

XML je razdeljen na 3 dele:

- podatkovni (vanj shranimo podatke v predpisani obliki z želenimi značkami),

- deklarativni (skrbi za to, da lahko pri dodajanju novih podatkov vidimo, kaj kakšna značka predstavlja),

- predstavitveni (z njim oblikujemo izpis podatkov).

- XMLHttpRequest objekt za asinhrono komunikacijo ter
- JavaScript, ki povezuje vse te tehnologije (JavaScript služi tudi kot vmesnik med posameznimi komponentami).

Zaradi napredka v razvoju in tehnologijah je sedaj del Ajax-a tudi:

- Poleg jezika JavaScript kot skriptni jezik na strani odjemalca nastopa tudi VBScript.
- XML in XSLT nista več obvezna za izmenjavo in obdelavo podatkov. Pogosto se kot alternativa uporablja JSON (JavaScript Object Notation), čeprav je možno uporabiti tudi druge načine predstavitve podatkov na primer HTML ali pa celo golo besedilo (Plain Text).

Prednosti:

- Možnost izdelave hitrejših, boljših in uporabniku bolj prijaznih spletnih aplikacij.
- Prenašanje samo določenih podatkov, ne pa celotne strani, kar drastično zmanjša količino prometa med strežnikom in odjemalcem.
- Za pravilno delovanje ni potrebe po vnaprejšnji namestitvi vtičev (ang. plug-in).

Slabosti:

- Podatki se osvežujejo samo znotraj strani, zato je navigacija po strani z brskalnikom (naprej, nazaj ...) otežena.
- Brskalniki, ki ne uporabljajo skriptnih jezikov JavaScript ali podobnih, imajo težave s prikazom strani.
- S programerskega vidika neenotno delovanje na različnih brskalnikih, kar otežuje programiranje aplikacij.
- Izdelava takih aplikacij zahteva znanje:
  - enega izmed strežniških jezikov (v našem primeru PHP)
  - enega izmed odjemalčevih jezikov (v našem primeru JS)
  - enega od označevalnih jezikov (v našem primeru HTML)
  - poznavanje osnove tehnologije DOM (ang. Document Object Model)

[22, 23]

3. 4. JavaFX

JavaFX temelji na Javinem ogrodju, je skupek izdelkov in tehnologij, ki znatno olajšajo razvoj aplikacij za številne naprave. Je programska platforma za razvoj RIA, kjer razvijalci za razvoj uporabljajo deklarativen jezik JavaFX Script. Velika uporabnost JaveFX je ta, da se lahko v programih JavaFX vključi tudi kodo programskega jezika Java ter njenih knjižnic. Izvaja se lahko na veliko različnih načinov tako od namizne pa do spletne aplikacije, prav tako pa na različnih napravah (mobilne naprave, TV in podobno). Glede na to, da sem v diplomski nalogi največ prostora namenil tehnologiji JavaFX, je naslednje poglavje namenjeno tej tehnologiji [24].

4. Pregled tehnologije JavaFX

JavaFX je družina proizvodov za izdelavo RIA, ki temeljijo na računalniškem okolju Java. Aplikacije razvite s tehnologijo JavaFX se lahko uporabljajo v mobilnih napravah,
namiznih aplikacijah kot bogate internetne aplikacije ... JavaFX s svojo preprosto sintakso in uporabo olajšuje razvoj aplikacij za te naprave in tehnologije. V družino proizvodov, ki sestavljajo izdelek JavaFX, spadata proizvoda JavaFX Script in JavaFX Mobile. Omogoča hiter in preprost razvoj internetnih aplikacij, ki združujejo 2D- in 3D-grafiko, zvok in video visoke kakovosti ter animacijo.

Ker JavaFX script programski jezik temelji na Java platformi, je potrebno imeti na sistemu nameščeno Javo JDK 5 oziroma novejšo. Seveda je potrebno imeti nameščen tudi JavaFX prevajalnik in izvajalno okolje. Vsaka napisana izvorna koda v JaviFX mora biti pretvorjena v bitno kodo (jezik virtualnega stroja), preden se lahko izvaja na sistemu [25, 26].

JavaFX je skupek izdelkov in tehnologij, ki naj bi znatno olajšale razvoj aplikacij za številne naprave, ki podpirajo mobilno ali standardno različico jave. Prva predstavljena člana družine sta skriptni jezik JavaFX Script in programska infrastruktura JavaFX Mobile, ki je namenjena ponudnikom mobilnih naprav in omogoča hiter razvoj naprednih mobilnih storitev.

JavaFX Mobile je na voljo proizvajalcem mobilnih in namenskih naprav in ponuja tako temeljni sloj operacijskega sistema (trenutno GNU/Linux) kot programsko podlago s ključnimi tehnologijami. Izdelek temelji na tehnologiji, ki jo je Sun pridobil z nakupom podjetja SavaJe. JavaFX je platforma za razvoj aplikacij za mobilne naprave, ki je del JavaFX platforme. JavaFX Mobile aplikacije so lahko razvite v JavaFX Script jeziku, kot so razvite namizne ali pa internetne aplikacije, ki „tečejo” na brskalniku ter uporabljajo enaka orodja JavaFX SDK in JavaFX Production Suite. Z integracijo JavaME lahko JavaFX aplikacije dostopajo do datotečnih sistemov, kamere, GPS, bluetooth [27]…
JavaFX Script je deklarativni programski jezik, omogoča kreiranje kompleksnih grafičnih elementov in prijaznejšo uporabno komponent Swing. JavaFX Script uporablja sintakso za specifikacijo GUI-komponent, omogoča povezovanje podatkov iz aplikacije in uporabniškim vmesnikom, za razvoj le-teh pa ga je mogoče vključiti v različna razvojna okolja, kot so Eclipse, NetBeans ... V primeru namestitve NetBeans IDE za JavoFX ali JavaFX plugin za Eclipse sta JavaFX compiler (prevajalnik) in runtime nameščena avtomatsko. V primeru uporabe drugega

razvojnega okolja je najlažji način, da se posname celoten JavaFX SDK, ki nudi runtime, prevajalnik in veliko ostalih orodij.

JavaFX Script ponuja hitro opisno definiranje uporabniških vmesnikov in statične podatkovne tipe, zaradi tega kombinira relativno preprostost uporabe in neomejeno izrabo javine podlage oziroma njenih knjižnic. Mogoče je uporabiti poljuben javin API, hkrati pa ponuja pakete, dedovanje, ločeno prevajanje in izvajanje, kakor smo tega vajeni pri "običajni" Javi. JavaFX Script ni naslednik Groovya, ki je nastal zaradi drugačnih razlogov in se bo razvijal ločeno [28].

JavaFX Preview SDK nudi prevajalnik in izvajalno okolje, knjižnice za 2d grafiko in medije, navodila, dokumentacijo API in vzorčno kodo.

NetBeans 6.1 IDE z integriranim vtičnikom JavaFX je razvojno okolje za razvoj, pregled in razhroščevanje aplikacij JavaFX. Zelo zmogljivo okolje **Java Runtime Environment 6 Update 10 Beta** z novim vtičnikom za spletne brskalnike omogoča prestavitev aplikacije iz brskalnika na namizje med njenim delovanjem.

4. 1. Razlogi za uporabo tehnologije JavaFX

Na trgu obstaja ogromno programskih jezikov. Zakaj se bi kot pri razvoju določene aplikacije nekdo sploh odločil za JavaFX Script? JavaFX omogoča razvoj bogatih internetnih aplikacij, mobilnih ter namiznih aplikacij. JavaFX temelji na platformi Java, ki jo imajo nameščeno od mobilnih telefonov pa vse do superračunalnikov. Ker temelji na Javi, je mogoča tudi uporaba vseh njenih knjižnic. Pomembna lastnost pa je tudi, da omogoča zelo hiter razvoj grafičnih aplikacij. JavaFX Script je zasnovan tako, da lahko razvijalec brez težav uresniči svoje želje. Večina programskih jezikov zahteva od razvijalca, da se točno izrazi, kako izvesti določen učinek. V nasprotnem primeru je pri JaviFX opis želja omogočen deklarativno, kar pomeni, da povemo, kaj hočemo, ta pa ugotovi, kako naj to uresniči. Deklarativno programiranje je še posebej močno v izražanju interaktivnih razmerij med grafičnimi elementi, kar poenostavi in pospeši razvoj [29].

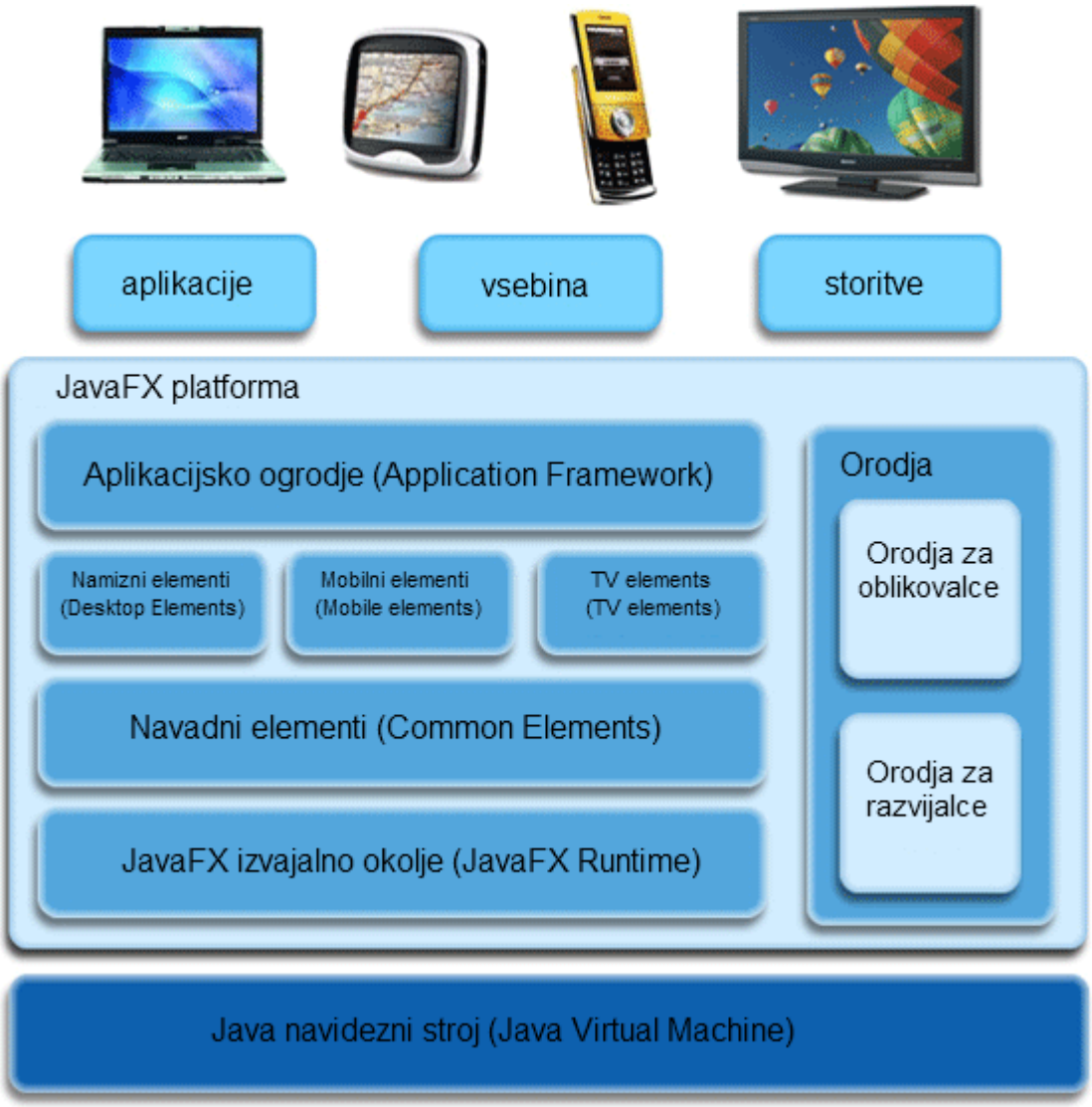
4. 2. Arhitektura tehnologije JavaFX

Jezik Java je močan programski jezik, vendar so strukturni in objektno usmerjeni programski jeziki manj primerni glede na današnje standarde, trend hitrega in učinkovitega razvoja aplikacij. Namreč današnji uporabniki ne pričakujejo od mobilne ali namizne aplikacije le, da bo funkcionalna, ampak tudi, da bo grafično všečna – torej da bo uporabnik imel bogato uprabniško izkušnjo. To pomeni, da programski jeziki, ki so bili zasnovani za potrebe prejšnejga stoletja, dandanes niso najbolj primerni za trend bogatih uporabniških vmesnikov.

Zaradi novih potreb na trgu je bila razvita povsem nova platforma in programski jezik JavaFX. JavaFX je bil zasnovan prav za potrebe razvoja RIA z vsemi gradniki in možnostmi, ki jih RIA danes imajo in tudi z deklarativno sintakso za razvoj grafičnih vmesnikov [30].

Arhitektura JavaFX platforme vsebuje medplatformna in platformno specifična izvajalna okolja ter podporne knjižnice. Vsebuje deklarativen programski jezik JavaFX Script prav tako pa tudi razvojna in oblikovalska orodja. Vse to omogoča, da aplikacije obdržijo konsistenten videz in občutek delovanja na različnih napravah z različnimi procesorskimi in grafičnimi sposobnostmi.

Platforma JaveFX vsebuje nabor orodij in tehnologij, ki omogočajo razvijalcem in oblikovalcem sodelovanje, izdelavo in (deploy applications with expressive content to browsers and desktops). Razvijalci mobilnih aplikacij lahko uporabljajo JavaFX Mobile emulator za predpogled aplikacij za mobilne naprave, ki uporabljajo platformo JavaFX [26].



Slika 3: Arhitektura platforme JaveFX

3. 2. 1. Glavne značilnosti platforme JavaFX

Vse na enem mestu: Platforma JavaFX omogoča, da z enotnim modelom razvoja razvijemo aplikacijo, ki bo delovala na brskalniku; kot namizna in tudi kot mobilna.

Širok tržni doseg: Naprave lahko odpirajo najširši portfelj vsebin in storitev v industriji. Aplikacije lahko delajo na številnih napravah prav zaradi prenosljive lastnosti JaveFX.

Nižji stroški implementacije: Razvijalci aplikacij JavaFX lahko uporabljajo kakršno koli javino knjižnico za gradnjo vmesnikov. Na mobilnih napravah se lahko JavaFX uporablja že na obstoječi JavaME in Mobile Service Arhitecture (MSA) platformi zaradi zmanjšanja implementacijskih stroškov proizvajalcev naprav.

Učinkovito sodelovanje med razvijalci in oblikovalci: Skupek orodij in vtičnikov (plug-in) omogoča kooperativeno sodelovanje razvijalcev in oblikovalcev z uporabo JavaFX Production Suite, kar zmanjšuje produkcijski cikel načrtovanja in izdelave aplikacij.

Zmogljivo izvajalno okolje (JavaFX Runtime): Izvajalno okolje JaveFX je izjemno razširljivo, močno, učinkovito in varno.

Možnost delovanja brez brsklalnika: Z verzijo Java SE 6 desete posodobitve ter novejše lahko aplikacijo, ki deluje preko brskalnika, povlečemo na namizje, tako da tam deluje kot samostojna aplikacija. Taka inovativna novost je edinstvena na tem področju [26].

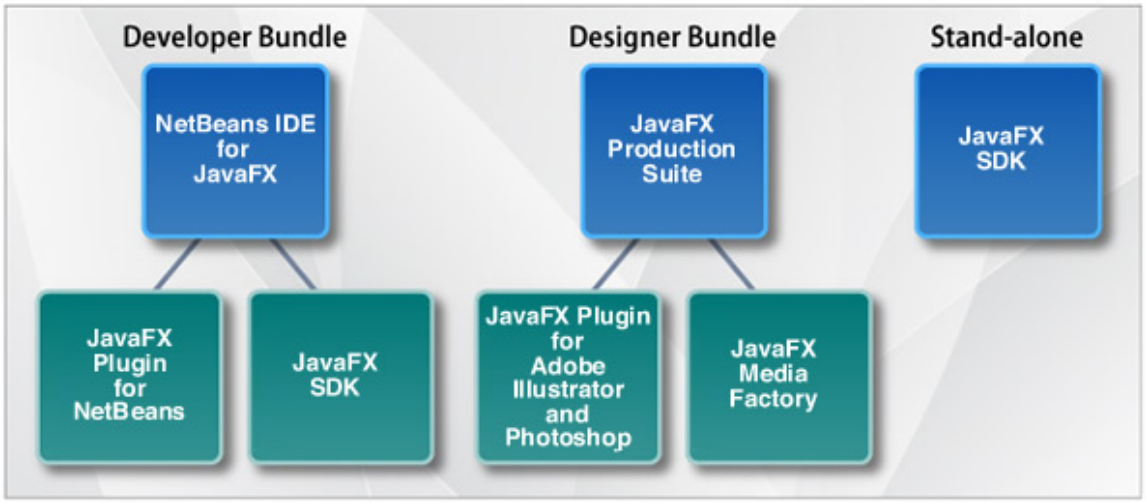
4. 2. 2. Komponente platforme

- **Java navidezni stroj** (Java Virtual Machine) - Izvajalno okolje vseh javinih aplikacij.
- **JavaFX platforma:**
    - **JavaFX izvajalno okolje:** Izvajalno okolje JavaFX se namesti samodejno pri namestitvi JRE, ki je potrebna za izvajanje javinih apikacij.
    - **Navadni elementi:** JavaFX API-ji z uporabo navadnih elementov delujejo konsistentno na različnih platformah.
    - **Namizni elementi:** JavaFx API-ji, ki so značnilni za namizne platforme.
    - **Mobilni elementi:** JavaFx API-ji, ki so značnilni za mobilne platforme.
    - **TV elmenti:** JavaFx API-ji, ki so značnilni za TV platforme.
    - **Aplikacijsko ogrodje:** Gradniki aplikacij.

- **Orodja:**
  - **razvijalska orodja:** Razvijalcem pomagajo razvoj JavaFX aplikacij, vsebin in storitev.

  - **oblikovalska orodja:** Oblikovalcem pomagajo pri pretvarjanu medijskih formatov v JavaFX format in ogled JavaFX medijev.

JavaFX platforma vsebuje medplatformna in platformno specifična izvajalna okolja ter podporne knjižnice. Vsebuje deklarativen jezik JavaFX Script kot tudi nabor orodij za razvoj in oblikovanje. To zagotavlja da imajo aplikacije konsistenten videz in občutek delovanja z na različnih napravah različnimi zmogljivostmi [26].



Slika 4: Dostopna orodja za platformo JavaFX

Razvijalski paket vsebuje naslednja orodja (Developer Bundle):

- **Netbeans IDE za JavaFX:**
  JavaFX tehnologija je integrirana z Netbeans IDE, ki je zrelo in močno razvojno orodje, ki omogoča preprosto izgradnjo, predpogled in poganjanje (debug) JavaFX aplikacij.

  Instance objektov, ki se privzeto ustvarijo ob kreaciji projekta:

Tabla 1: Object Literals Created by Default

Object Literal	**Description**
`Stage`	Je vsebovalnik (container) in je osnovno okno, ki ima funkcijo prikazati vse vizualne objekte v JavaFX in lahko vsebuje druge komponente grafičnega vmesnika (gumbe, tekstovna polja …). Privzete osnovne spremenjivke objekta *Stage* so *title (ime), width (širina) in height (višina)*.
`Scene`	Spremenjivka *scene* definira instanco od *Scene* objekta (literal), ki nastavi površino, kjer se lahko vključijo JavaFX objekti. Podobno risalni površini za grafično vsebino aplikacije. Spremenjivka *scene* je instančna spremenjivka, ki vsebuje *content* spremenjivko, ki se uporablja za vsebovanje JavaFX grafičnih elementov in definiranjem grafične vsebine od aplikacije. Instančne spremenljivke (spremenljivke objekta), *width* in *height* definirajo višino in širino, kjer bodo vsi grafični elementi. Razredu *Scene* lahko dodamo različne ostale osnovne in naprednejše oblike, in sicer kot polje teh vozlišč pod atribut *content*.
`Text`	Definira grafični element, ki prikazuje tekst (v objektu Scene).
`Font`	Definira velikost pisave, ki bo prikazana (v objektu Scene).

Tabela 1: Objekti, ki se privzeto kreirajo pri razvoju JavaFX aplikacije

- **JavaFX vtičnik (plugin) za NetBeans:**
  V že namešče Netbeans IDE brez JaveFX se lahko vključi vtičnik za možnost razvoja aplikacij JavaFX.

Oblikovalski paket vsebuje naslednja orodja (Developer Bundle):

- **JavaFX Production Suite vsebuje orodja, ki omogočajo oblikovalcem izmenjavo vizualnih zamisli z razvijalci:**

  - JavaFx vtičnik za Adobe Photoshot in JavaFX vtičnik za Adobe Illustrator izvozi grafično zamisel iz oblikovalskega orodja v format JavaFX.

- JavaFX Media Factory vsebuje dve ločeni orodji:
  - SVG Converter(pretvornik): Pretvori SVG grafiko v JavaFX format.

  - JavaFX Graphic Viewer: Omogoča predpogled grafične zamisli, kakor se bo pokazalo na namizni ali pa mobilni aplikaciji.

4. 3. Samostojna (Stand-alone) SDK

V primeru uporabe drugih orodij ali razvoj direktno preko ukazne vrstice je možen prenos samostojnega SDK.

Java SDK vsebuje naslednje komponente:

- JavaFX Desktop Runtime
- JavaFX Mobile Emulator (for Windows)
- JavaFX APIs
- JavaFX Compiler
- JavaFX API documentation
- Samples (primere)

[26]

4. 4. JavaFX Script sintaksa in jezik

V nadaljevanju bom predstavil glavne značilnosti jezika JavaFX Script in jih prikazal s primeri. Prikazan pa bo tudi daljši primer preproste računalniške igre, napisan v tem jeziku.

4. 4. 1. Spremenljivke

Ob deklaraciji spremenljivk je potrebno navesti:

- način dostopa
- določitev možnost spreminjanja vrednosti spremenjivke
- ime spremenjivke
- dvopičje
- podatkovni tip
- enačaj
- začetno vrednost (neobvezno)
- podpičje

Primer deklaracije spremenjivke:

```
Način_dostopa var_ali_def ime_spremenjivke: podatkovni_tip = vrednost;
```

Spremenljivke se lahko deklarirajo z dvema predznakoma **var** ali **def**.
Določilo **var** dopušča spreminjanje vrednosti po inicializaciji, **def** pa spremenljivko določi kot konstanto. Spremenljivko, ki je določena kot konstanta, je možno po inicializaciji spremeniti, le če ji vrednost dodelimo z določilom **bind**. Definiranje podatkovnega tipa je opcijsko, saj prevajalnik sam ugotovi, katerega podatkovnega tipa je spremenljivka. Uporaba spremenljivk je podobna kot v Javi, s tem da JavaFX ne dopušča seštevanja String-ov z operatorjem +, vendar se to izvede na naslednji način:

```
def niz1 =  ”Aleš”;
var niz2 =  “Vranešič”;
var niz3: String = “oba niza: {niz1} {niz2}”

public var spremenljivka: String = ”Hello world”;
def spremenjivka = 1234;
def s;
s = bind 1234;
```

Poznamo: - skriptne (Script Variables)
- instančne (Instance Variables)
- lokalne (Local Variables) spremenljivke

Skriptne spremenljivke so deklarirane na vrhu skripte. Skriptne spremenljivke so vidne znotraj skripte brez potrebnega dodajanja dostopnega določila. Če se dodajo dostopna določila (**public**, **protected**, **package**, **public-read**, and **public-init**), je spremenljivka vidna tudi zunaj skripte. V Javi bi jim rekli globalne spremenljivke. Življenjska doba skriptne spremenljivke je od zagona skripte pa do njenega konca izvajanja.

Primer:

```
var thing = "Thing";
class A {
  function getThing() : String { thing }
}
```

Primer:

```
public def bohr = 0.529177e-10
```

Znotraj razreda (in njegovih podrazredih) se v **instančne spremenljivke** dostopa samo z imenom spremenljivke. Drugače pa se do njih dostopa preko objekta, kateremu pripadajo. Dostopna

določila (**public**, **protected**, **package**, **public-read**, and **public-init**) določajo vidnost instančne spremenljivke. Če ni podanega dostopnega določila, je instančna spremenljivka vidna samo znotraj skripte. Življenska doba instančne spremenljivke traja toliko časa kot živ objekt razreda, na katerega se ta spremenljivka nanaša.

Primer:

```
def anA = A{ rat: true };
println(anA.rat);
class A {
  var rat : Boolean;
  function isIt() { rat }
}
class B {
  function wellisIt() { anA.rat }
}
```

Lokalne spremenljivke so deklarirane znotraj blokov, tudi blokov, ki so telesa funkcij. Vidljivost teh spremenljivk je samo znotraj bloka, kjer je le-ta deklarirana. Dostopna določila se pri lokalnih spremenljivkah ne uporabljajo [31].

4. 4. 2. Dostopna določila (Access Modifiers)

Dostopna določila se uporabljajo za določitev omejitev pri dostopanju do spremenljivk, razredov ...

- package

  Dostopno določilo package določa dostop do razredov, vsebovanih v paketu.

- protected

  Dostopno določilo protected omogoča dostop, iz katerega koli razreda v istem paketu in iz kateregakoli podrazreda, ne glede na paket.

- public

  Dostopno določilo public omogoča dostop iz kateregakoli razreda ne glede na paket.

- public-read

  Spremenljivka se lahko bere kjerkoli.

- public-init

  Spremenljivka se lahko inicializira ali bere kjerkoli.

[32]

4. 4. 3. Primitivni podatkovni tipi JavaFX

Programski jezik JavaFX Script vsebuje naslednje primitivne podatkovne tipe:

JavaFX	vrednost
String	“besedilo”
Boolean	true oz. false
Number	cela števila [0, 2, 5...)
Integer	decimalna št. (-1.3,0.2...)

Tabela 2: Primitivni podatkovni tipi v programskem jeziku JavaFX Script

4. 4. 4. Funkcije

Namesto imena metoda, ki je v Javi, se za isto stvar v JaviFX imenuje funkcija. Funkcije predstavljajo določen blok programske kode (zaporedje stavkov), ki ga rabimo večkrat oziroma ga kličemo z različnih mest v programu.

Deklaracija funkcije:

vrsta_dostopanja function ime_funkcije (argumenti : podatkovni_tip_argumenta) :

```
podatkovni_tip_kar_funkcija_vrne{

        //telo funkcije
        //funkcija lahko vrača vrednosti: ruturn spremenljivka
}
```

Če funkcija vrača vrednost, lahko, ni pa nujno, navedemo kakšen podatkovni tip bo funkcija vračala. Return stavek z vrednostjo, ki ga bo funkcija vrnila, je enak kot v programskem jeziku Java.

Skriptne funkcije (funkcije razreda v Javi) so deklarirane na vrhu skripte, torej zunaj bloka razreda. Te funkcije so vidne v vsej skripti.

Če se funkciji doda dostopno določilo, postane funkcija vidna tudi zunaj skripte in se do nje dostopa preko imena skripte. Primer: funkcija kvadrat je vsebovana v skripti Geometrija.fx:

```
public function ploscina_kvadrata (x : Number) : Number { x*x }
```

dostop do funkcije preko imena skripte Geometrija.fx:

```
println(Geometrija.ploscina_kvadrata(5));
```

[33]

Instančne funkcije (funkcije objekta v Javi) so deklarirane znotraj bloka razreda. Te funkcije so vidne v skripti, zunaj razreda je dostop do funkcije mogoč preko objekta tega razreda. Znotraj razreda in njegovih podrazredov se dostopa do funkcije samo z imenom funkcije. Primer:

```
class Scale {
  var factor : Number;
  function transform(x : Number) : Number { factor * x }
}
var tf = Scale { factor: 25.0 }
println(tf.transform(10.0));
```

Primer klica funkcije:

```
println(zdruziBesedi(¨ales¨,¨vranesic¨));
```

Primer funkcije, ki vrača določeno vrednost:

```
function zdruziBesedi (niz1 : String, niz2 : String) : String {
        return "oba niza: {niz1} {niz2}"
}
```

4. 4. 5. Zaporedja oz. polja (sequences)

JavaFX za podatkovno strukturo ponuja zaporedje (sequence), ki predstavlja urejen seznam objektov. Zaporednje elementov je postavljeno med oglati oklepaj in oglati zaklepaj, vejica pa ločuje vsak element med seboj. Prvi element zaporedja je na poziciji 0, zadnji pa na n-1.

Primer: zaporedje sodih števil

```
var sodo = [0,2,4,6,8,10,12];
sodo[4]; // vrednost sodo[4] = 8
```

Z zaporedjem lahko tudi manipuliramo, in sicer z uporabo besed: before, after, into, delete, reverse.

Primer: manipulacije z zaporedjem

```
insert 11 before sodo[3];//doda 11 pred 4. element zaporedja
                         //[0,2,4,11,6,8,10,12]

insert 9 after sodo[1]; //doda 9 za 2. elementom zaporedja
                        //[0,2,9,4,11,6,8,10,12]

insert 1 into sodo; //na zadnje mesto zaporedja doda 1
                    //[0,2,9,4,11,6,8,10,12,1]

delete sodo[2];//izbriše element na 3. mestu zaporedja
                //[0,2,4,11,6,8,10,12,1]
```

```
reverse sodo; //vrne obrnjeno zaporedje
              //[1,12,10,8,6,11,4,2,0]
```

4. 4. 5. Operatorji in izrazi

Operatorji

Opreatorji so posebni simboli, ki opravijo specifično operacijo nad enim ali dvema operantoma ter vrnejo rezultat.

Operatoji v JaviFX so povsem enaki kot v Javi. Poznamo več vrst operatorjev, kot so prireditveni, primerjalni, aritmetični, unarni ter pogojni operatorji.

Prireditveni operatorji(assignment operators)

Prireditveni operator, ki se uporablja v JavaFX, je znak enačaj (=), ki priredi vrednost spremenjivke, ki je na levi strani enačaja z vrednostjo na desni strani enačaja.

```
def niz= niz;
st1=3;
st2=5;
rezultat=st1+st2;
//rezultat ima vrednost 8
```

Aritmetični operatorji (arithmetic operators)

Aritmetični operatorji omogočajo seštevanje, odštevenje, množenje in deljenje. Operator mod deli operand z drugim ter vrne ostanek kot rezultat.

Ti operatorji so:

- operator za seštevanje (+)

```
var result = 1 + 2; // result is now 3
println(result);
```

- operator za odštevanje (-)

```
result = result - 1; // result is now 2
println(result);
```

- operator za množenje (*)

```
result = result * 2; // result is now 4
```

```
println(result);
```

- operator za deljenje (/)

```
result = result / 2; // result is now 2
println(result);

result = result + 8; // result is now 10
println(result);
```

- operator ostanka pri deljenju (mod)

```
result = result mod 7; // result is now 3
println(result);
```

Pri aritmetičnih operatorjih je prav tako kot v Javi možno združevanje aritmetičnih operatorjev s prireditvenimi operatorji, kot je razvidno v naslednjem primeru:

```
rezultat+=1;
rezultat=rezultat+1;
```

V obeh primerih se rezultat poveča za 1.

Unarni operatorji(Unary Operators)

Unarni operatorji uporabljajo samo en operand za izvajanje operacij.

- => unarni operator minus negira vrednost spremenjlivke
++ => operatorja za seštevanje povečata vrednost za ena
-- => operatorja za odštevanje zmanjšata vrednost za ena
not => logični operator spremeni vrednost boolean

Primer:

```
var rezultat = 1; // rezultat je 1

rezultat--;  // rezultat je 0
println(rezultat);

rezultat++; // rezultat je 1
println(rezultat);

rezultat = -rezultat; // rezultat je -1
println(rezultat);

var uspeh = false;
```

```
println(uspeh); // false
println(not uspeh); // true

var rezultat = 3;
rezultat++;
println(rezultat); // rezultat je 4
++rezultat;
println(rezultat); // rezultat je 5
println(++rezultat); // rezultat 6
println(rezultat++); // še vedno izpiše rezultat 6!
println(result); // ampak sedaj pa je rezultat 7
```

Primerjalni operatorji(Equality and Relational Operators)

Poznamo več vrst primerjalnih operatorjev:

je enako ==
ni enako !=
večje kot >
manjše kot <
večje ali enako >=
manjše ali enako <=

Primer:

```
def num1=1;
def num2=2;

println(num1 == num2); //false
println(num1 != num2); //true
println(num1 < num2); //true
```

Pogojni operatorji (Conditional Operators)

Pogojni **in (and)** ter pogojni **ali (or)** operatorja imata enak pomen kot v Javi, vendar, kot kaže spodnji primer, se ju drugače navede.

oprator	JavaFX	Java
in	and	&&
ali	or	||

Primer:

```
def username = "foo";
def password = "bar";

if ((username == "foo") and (password == "bar")) {
    println("Test 1: username AND password are correct");
}

if ((username == "") and (password == "bar")) {
    println("Test 2: username AND password is correct");
}

if ((username == "foo") or (password == "bar")) {
    println("Test 3: username OR password is correct");
}

if ((username == "") or (password == "bar")) {
    println("Test 4: username OR password is correct");
}

Test 1: username AND password are correct
Test 3: username OR password is correct
Test 4: username OR password is correct
```

[34]

Izrazi

Izrazi so deli programske kode, ki se preslikajo v vrednost. Programski jezik JavaFX Script je izrazni jezik, kar pomeni, da so vse – vključno z zankami, pogojnimi stavki, bloki kode – izrazi [35].

Bločni izrazi

Bločni izraz je sestavljen iz več deklaracij spremenljivk oziroma drugih izrazov.
Vrednost bločnega izraza je enaka vrednosti zadnjega izraza.

Primer:

```
var stevila = [5, 8, 3, 1];
var skupaj = {
    var vsota = 0;
    for (foo in stevila) { vsota += foo };
    vsota;
}
println("vsota je {skupaj}.");//vsota je 17
```

Izraz if

Z izrazom if je mogoče usmerjati tok programa z izvajanjem določenih blokov kode, ki ustrezajo pogojem, ki jih določa if stavek.

Primer 1:

```
If (pogoj) {
        //v primeru, da je pogoj izpolnjen, se izvrši ta blok
}
Else if (pogoj2) {
//v primeru, da ni bil izpolnjen prejšnji pogoj in je izpolnjen ta pogoj, se izvrši ta //blok
} else{
        // v primeru, da ni noben od zgornjih pogojev izpolnjen, se izvrši ta blok
}
```

Primer 2:

```
var foo=10;

if (foo < 5 ) {
    println("foo je manj kot 5");
} else if (foo < 20 or foo >= 10) {
    Println("foo je manjši od 20 ter večji ali enak 10");
} else {
    Println("foo ni nič od navedenega");
}
//izpiše: foo je manjši od 20 ter večji ali enak 10
```

Izraz for

Izraz for predstavlja koncept zanke (ponavljanja) za izpis elementov zaporedja.

```
for(i in zaporedje){
        //v vsakem koraku izbiše naslednji člen zaporedja
}
```

Primer:

```
var stevila=[1,2,3,4,5];

for (stevilo in stevila){
        println(stevilo);
}

var  teden=["pon","tor","sre","čet","pet","sob","ned"];

for (dan in teden) {
    println(dan);
}
//izpis
pon
tor
sre
čet
pet
sob
ned
```

Izraz while

Izraz while je tudi mehanizem zanke (ponavljanja), ki se ga uporabi nekoliko drugače kot izraz for. Pogoj za ponavljanje je na začetku, ponavljanje traja toliko časa, dokler je pogoj v oklepajih true, torej dokler je pogoj znotraj oklepajev izpolnjen. Možno je, da se jedro zanke ne izvrši niti enkrat.

Primer:

```
while(pogoj){
//dokler je pogoj izpolnjen, se zanka ponavlja in izvršuje programska koda znotraj //bloka
}

ver st = 0;       //vrednost spremenljivke st je 0
while (st<4){  //while se izvaja, dokler je vrednost st  < 10
        println("stevec = {st}");       //izpisemo vrednost spremenjivke st
        st++;  //povečamo vrednost spremenjivke st
}
Izpis:

stevec = 0
stevec = 1
stevec = 2
stevec = 3
```

Izraz za krajše definiranje aritmetičnih vrst (izrazi obsega==sem zasledil)

Primer:

```
var stevila = [0..5];
println(stevila);
//izpis: [ 0, 1, 2, 3, 4, 5 ]
```

Primer:

```
var stevila = [1..10 step 2];
println(stevila);
//izpis: [ 1, 3, 5, 7, 9 ]
```

Izraz break in continue

Izraza break in continue vplivata na število itracij zanke. V primeru, da se izvede izraz break, se zanka nemudoma zaključi. Če pa se izvede izraz continue, konča trenutno iteracijo ter nadaljuje z naslednjo. Oba sta sintaktična izraza, tako da sta tipa Void ter ne vračata nobene vrednosti [35].

Primer:

```
for (i in [0..25]) {
    if (i > 10) {//ko je i večji od 10 se zanka konča
        break;
    }

    if (i mod 2 == 0) {
   /*v primeru da je (i mod 2 = 0), se iteracija zaključi in ne izpiše trenutnega i-ja, ampak skoči
na vrh zank ter nadaljuje z novim i-jem.*/
        continue;
    }

    println(i);
}

Izpis:
1
3
5
7
9
```

Izrazi try, catch in finally

V delovanju aplikacij se pojavljajo tudi dogodki, ki zmotijo normalen tok izvajanja skripte. Dober program mora obvladovati tudi izjeme, zmožen je obvladati tudi izjemne dogodke (Exceptions), ki lahko v primeru neobvladovanja le-teh povzročijo škodo npr. izgubo podatkov. Izjeme (Exceptions) so objekti, njihovi tipi so poimenovani glede na kakšno izjemno situacijo, predstavljajo npr. objekt FileNotFoundException je namenjen situaciji, ko določna datoteka ne obstaja. Objekt Exception je nadrazred vseh razredov za preverjanje izjem, kot je npr. razred FileNotFoundException. Če je verjetnost, da lahko v delu kode prihaja do izjem, se uporabi naslednje konstrukte: try, catch in finally.

Primer nepričakovanega dogodka:

- skripta hoče brati datoteko, ki ne obstaja,
- poskus dostopa do elementa izven meja nekega polja,
- poskus delitve z vrednostjo 0,
- poskus dostopa do nekega URL z napačnim protokolom .

[35, 36]

Primer:

```
try {
    foo();

/*ce se kaj nenavadnega zgodi v funkciji foo, to izjemo ulovi try. Ta prekine izvajane kode v
funkciji foo in nadaljuje v bloku ki ga zaobjema izraz catch*/

} catch (e: Exception) {
    println("{e.getMessage()} (izjema je ujeta)");
} finally {
    /*
    Blok finally se zgodi vedno neglede ali je nastopila kakšna izjema
    ali ne. Običajno obstajajo operacije, ki se morajo izvesti ne  glede na to da je prišlo do izjeme
ali ne. Ta del kode spada pod blok finally. Lovljenje izjem je mogoče izvesti tudi brez bloka
finally.
    */
    println("sledi še koda v finylly bloku...");
}
```

4. 4. 6. Objekti in razredi

Deklaracija razreda ter koncept objektno usmerjenega programiranja v JavaFX Script je podoben kot v Javi. Tako lahko razredi dedujejo, se gnezdijo, ne morejo pa implementirati vmesnikov (interface), saj v JavaFX Script slednji ne obstajajo.

Razred se deklarira z določilom class, ki je obvezen. Znotraj razreda se doda atribute razreda, funkcije …

Primer deklaracija razreda:

```
class Avto {
  var barva : String;
  var moc: Number;
  var znamka : String;
  var cena : Number;

  function izpisiPodatke(): Void{
        println(znamka);
  }
}
```

Primer kreacije objekta oz. primerka razreda Avto:

```
def a4= Avto {
  barva: "bela"
  moc: 140
  znamka: "Audi"
  cena: 14000
}
A4.izpisiPodatke();//Audi
```

4. 4. 7. Vezanje podatkov in sprožilci

Mehanizem vezanja podatkov oz. data binding je eno od pomembnejših stvari, ki jih JavaFX Script programski jezik ponuja. V bistvu je bind mehanizem za zagotavljanje, da sta dve spremnljivki povezani, kar pomeni, da se ob morebitni spremembi vezane spremenljivke takoj spremeni vrednost druge povezane spremenljivke. To v praksi pomeni, da ob spremembi vezane spremenljivke povzroči takojšen odziv funkcionalnosti aplikacije ali uporabniškega vmesnika (Cause and effect). Vezanje (binding) lahko uporabimo pri spremenljivkah, funkcijah, objektih, zaporedjih [37].

Primer vezanja podatkov (binding to a variable):

```
var x=0;

//vezanje spremenljivke x s spremenljivko y. Vrednost y se spremeni vsakič, ko se //spremeni
vrednost spremenljivk x.

def y= bind x;
x=1;
println(y);//y je enak 1
x=20;

/*
Ker sta spremenljivki vezani, se je vrednost spremenljivke y ob
spremembi sprmenljivke x avtomatično spremenila na novo vrednost.
*/

println(y);//y je enak 20
```

Primer vezanja podatkov:

```
var a = "zdravo";
var b = bind a;
println("a:{a} b:{b}");
a = "nasvidenje";
println("a:{a} b:{b}");
/*rezultat:
a: zdravo b: zdravo
a: nasvidenje b:nasvidenje
*/
```

Primer vezanja izrazov:

```
var x=0;

//spremenljivka y je vezana na preprost izraz x+5
var y=bind x+5;

x=2;
println(y );//y ima vrednost 7

x=10;
println(y);//y ima vrednost 15
```

Primer vezanja podatkov v objektih:

```
var znamka_avta=«Audi«
var tip_avta=«a6«
var moc_avta=170
var cena_avta=24000

//spremnljivka avto je vezana z Avto objektom

def avto =  bind Avto{
        znamka: znamka_avta;
        tip: tip_avta;
        moc: moc_avta
        cena: cena_avta
}

Println(»avto.znamka=={avto.znamka}«);//avto.znamka==Audi
```

Če sedaj spremenim vrednost znamka_avta, se bo spremenljivka znamka v objektu avto spremenila. Spreminjanje vrednosti znamka_avto pravzaprav povzroči kreacijo novega objekta tipa Avto, ki se mu ponovno dodeli naslov spremenljivke.

```
znamka_avta=«BMW«

println(»avto.znamka=={avto.znamka}«);//avto.znamka==BMW
```

V naslednjem primeru bo primer vezanja podatkov, ne da bi se ustvaril nov objekt, ampak se vežejo spremenljivke že obstoječega objekta. V drugem primeru se predpostavlja, da ima objekt avto enake začetne vrednosti kot prejšni primer.

```
def avto =  bind Avto{//bind v tej vrstici se lahko tudi izpusti
        znamka:bind znamka_avta;
        tip:bind tip_avta;
        moc:bind moc_avta
        cena:bind cena_avta
}
Println(»avto.znamka=={avto.znamka}«);//avto.znamka==Audi
znamka_avta=«BMW«
println(»avto.znamka=={avto.znamka}«);//avto.znamka==BMW
```

Primer vezanja funkcij:

```
class Tocka {
  var x : Number;
  var y : Number;
}
```

```
var skala = 1.0;
function narediTocko(x0 : Number, y0 : Number) : Tocka {
  Point {
    x: x0 * skala
    y: y0 * skala
  }
}

var mojX = 3.0;
var mojY = 3.0;
def tocka = bind narediTocko(mojX, mojY);
println(tocka.x);//3.0
mojX = 10.0;
/*
Sprememba vrednosti argumaenta povzroči ponoven klic funkcije narediTocko.
*/
println(tocka.x);//10.0
skala = 2.0;
/*
Ker se funkcija narediTocko obnaša kot "črna škatla", sprememba vrednosti spremenljivke skala
nebo povzročila klica funkcije.
*/
println(tocka.x);//10.0
```

Da bi se izvedla funkcija, naredi Točko ob spremembi spremnljivka scale, potrebno je dodati pred funkcijo bound, kot kaže spodnji primer.

```
bound function narediTocko(x0 : Number, y0 : Number) : Tocka {
...
```

Sprožilci(triggers)

Uporabnost JaveFX omogočajo tudi sprožilci (triggers), ki omogočajo izvedbo funkcije, ko se podani spremenljivki spremeni vrednost. Prožilci zamenjave so deli programske kode, ki se "pripnejo" na določeno spremenljivko. Ko se vrednost "pripete" spremenljivke spremeni, se prožilci sprožijo. V spodnjem primeru je podana programska koda delovanja prožilca zamenjave [38, 39].

Primer:

```
var ime: String = "Ales Vranesic" on replace oldText{
        println("ime se je spremenilo");
        /*tekst se izpiše prvič pri deklaraciji ter vsakič, ko se v spremenljivko ime vpiše drugačna
vrednost.*/
};
```

Primer:

```
var x = 0 on replace staraVrednost { println("x je bil {staraVrednost} in je sedaj: {x}") }

x=1;
x=3;

Izpis:

x je bil 0 in je sedaj: 0
x je bil 0 in je sedaj: 1
x je bil 1 in je sedaj: 3
```

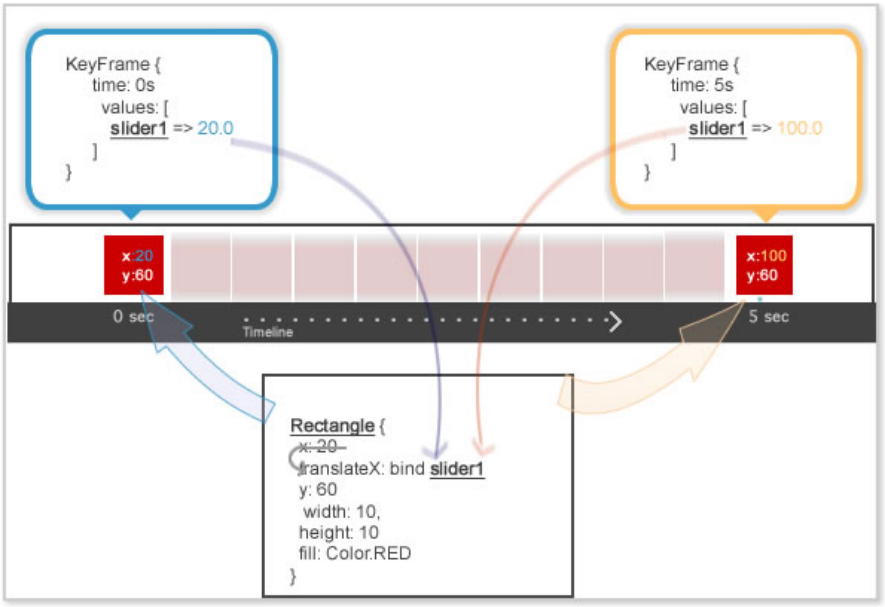
4. 4. 8. Seznam rezerviranih besed

Rezervirane besede v JavaFX Script so:

abstract, after, and, as, assert, at, attribute, before, bind, bound, break, catch, class, continue, def, delete, else, exclusive, extends, false, finally, first, for, from, function, if, import, indexof, in, init, insert, instanceof, into, inverse, last, lazy, mixin, mod, new, not, null, on, or, override, package, postinit, private, protected, public-init, public, public-read, replace, return, reverse, sizeof, static, step, super, then, this, throw, trigger, true, try, tween, typeof, var, where, while, with.

4. 4. 9. Animiranje grafičnih objektov

JavaFX knjižnice omogočajo učinkovito podporo za animiranje objektov, in sicer prehajanje (transition) stanj npr. rotacija, večanje oziroma manjšanje objekta, enostven premik, pojemanje … Animiranju objektov so namenjene knjižnice, ki se nahajajo v paketu javafx.animation, mogoče je animirati vsako vozlišče (Node). Animacije objektov so v osnovi manipulacije z enim ali več atributi objekta skozi čas. V razredu Scene definiramo objekte, torej vsebino, ki bo prikazana na zaslonu. Tako je možno vsak objekt tudi animirati, saj lahko atribute objektov vežemo (binding) z atributom, ki neprestano spreminja vrednost definirano v razredu KeyFrame, kot je prikazano na spodnji sliki. Kreiranje lastnih animacij omogoča razred Timeline, ki deluje kot zanka od sprožitve le-te (play();) pa do ustavitve (stop();). V instanci razreda Timeline definiramo potek animacije z definiranjem instanc razredov KeyFrame. Z vsako instanco razredov KeyFrame programer opiše stanja animiranega objekta skozi čas oziroma prelomne točke v poteku animacij (instance razredov KeyFrame). Te predstavljajo spremembo izvajanja animacije v določenem času [40].



Slika 5: Prikaz delovnja vezanja podatkov

Primer:

```
var kvadratX=0;//začetna  pozicija kvadrata

var tl=Timeline {
 repeatCount: Timeline.INDFINITE
   keyFrames: [
      KeyFrame {  //začetna točka animacije
         time: 0s
         canSkip: true
         values: [
            kvadratX => 10.0
         ]
      }
      KeyFrame {  //končna točka animacije
         time: 5s
         canSkip: true
         values: [
            kvadratX => 80.0 tween Interpolator.LINEAR
         ]
      }
   ]
}
tl.play();

Stage {
...
}
```

Opis primera:

Atribut repeatCount ima vrednost INDEFINATE, kar pomeni nedoločeno tranjane animacije. Obstajo pa seveda klici funkcij za nadzor nad potekom animacije, tako je možno animacijo tudi ustaviti. Seznam teh funkcij za nadzor poteka animacij:

```
play();//začetni zagon animacije oz. nadaljvanje animacije(premika)
playFromStart();//Začetek premika
stop();//ustavitev premika
pause();//zamrznitev premika
```

Atribut keyFrames ima dve prelomni točki izvajanja animacije. Prva se proži pri 0. sekundi ter traja do 5. sekunde, druga pa nastopi pri 5. sekundi. Atribut canScip pomeni, če je možno animacijo tudi prekiniti. Atribut values vsebuje podatek (lahko tudi več), katerega vrednost se v danem času spreminja. Tako v tem primeru začne kvadrat v x osi, kjer je x=10, ter po določenem času naredi pot do točke, kjer je x=80. Spreminjanje vrednosti spremenljivke kvadrat je v 0. sekundi do 5. sekunde ralizirano s transformiranjem vrednosti spremenljivke, ki se enakomerno spreminja od vrednosti 0 pa do vrednosti 10. V 5. sekundi nastopi neenakomerno spreminjanje vrednosti spremenljivke. Beseda twen pomeni, na kakšen način oziroma s kakšno hitrostjo se bo vrednost spremenljivke spreminjala. V tem primeru je bil uporabljen Interpolator.LINEAR, ki vrednost spremenljivke spreminja linearno.
Ostali so še:

- EASEIN: Uporablja vrednost faktorja 0.2 za pospeševanje.
- EASEOUT: Uprablja vrednost faktorja 0.2 za pojemanje.
- EASEBOTH: Uporablja privzete vrednosti 0.2 in 0.2 za pospeševanje in pojemanje.
- DISCRETE: Objekt ustavi , šele ko se začetna vrednost izenači s končno jo prestavi na končno lokacijo.

[41]

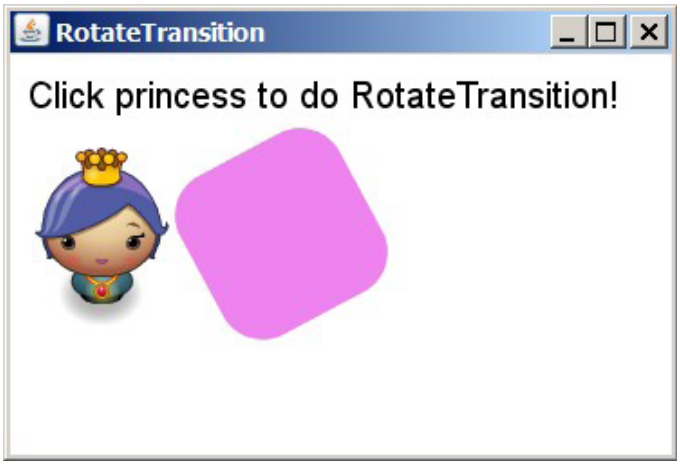
Animacija se začne izvajati z klicem funkcije play();. Definiranje časa, ki je ključnega pomena pri animacijah, je z razredom Duration.

Obstajajo tudi že v naprej določene animacije, ki ne temeljijo na KeyFrame:

```
RotateTranstion – rotacija
FadeTransition – motnost
TranslateTransition – premik objekta v ravni črti
PathTransition – premik objekta po že vnaprej določeni poti
ScaleTranstion – povečanje ali zmanjševanje dimenzije objekta
```

Primer:

```
var mynode = Rectangle {
//...
}
var rotTransition = RotateTransition {
  duration: 3s
  node: mynode
  byAngle: 180
  repeatCount:4
  autoReverse: true
}
var princess:ImageView = ImageView {
  image: Image {
  url: "{__DIR__}princess.png"
}
onMouseClicked: function( e: MouseEvent ):Void {
   rotTransition.play();
}
```



Slika 6: Preprosta aplikacija

4. 5. JavaFX in 3D

Java 3D API je vmesnik za razvoj in prikaz tridimenzionalne grafike. API zagotavlja kolekcijo visoko razvitih konstruktov za razvoj in manipulacijo 3D geometrijskih objektov. Ti objekti so postavljeni v virtulanem prostoru (virtual universe), ki je kasneje dodano. API je kreiran s fleksibilnostjo za izdelavo natančnega virtulanega prostora ne glede na velikost oziroma majhnost le-tega. Kljub široki funkcionalnosti je Java 3D API enostaven za uporabo [42].

3D Canvas

3D Canvas je orodje, ki uporablja intituativen povleci in spusti (drag-and-drop) pristop za razvoj 3D objektov. Zapleteni 3D modeli so sestavljeni iz številnih preprostih objektov oziroma z uporabo 3D Canvas Object Building orodja. To orodje omogoča preoblikovanje, skulpuriranje ter barvanje 3D objektov [43].

V javiFX je mogoče ustvarjenim objektom dodati 3D efekt z uporabo javafx.scene.effect paketa.

Primer:

```
package javafx3deffect;
import javafx.scene.paint.*;
import javafx.application.*;
import javafx.scene.effect.*;
import javafx.scene.geometry.*;
import javafx.application.Stage;
import javafx.scene.geometry.Circle;
import javafx.scene.paint.Color;
import javafx.animation.Timeline;
import javafx.animation.Interpolator;
import javafx.animation.KeyFrame;
import javafx.ext.swing.Button;
import javafx.input.MouseEvent;
import javafx.scene.text.Text;
import javafx.scene.Font;
import javafx.scene.FontStyle;

var val:Integer;
var t=Timeline {
  repeatCount: Timeline.INDEFINITE
  autoReverse:true
  keyFrames : [
    KeyFrame {
      time : 0s
      values:val=>0
    },KeyFrame {
      time : 4s
      values:val=>400 tween Interpolator.EASEBOTH
    }
  ]
}
Frame {
  title: "Painting Variations"
  width: 500
  height: 400
  closeAction: function() {
  java.lang.System.exit( 0 );
}
visible: true
stage: Stage {
  content: [
  Text {
    font: Font {
      size: 24
      style: FontStyle.PLAIN
    }
  x: 0, y: 20
  content: "HelloWorld"
  effect:GaussianBlur{radius:10}
  scaleX:3.4
  scaleY:3.4
},
 Circle {
```

```
 centerX: bind val, centerY: 40
 radius: 60
 opacity:0.6
 fill: RadialGradient{
 centerX:30
 centerY:10
 radius:30
 proportional:false
 stops:[
  Stop {offset: 0.3 color: Color.GREEN},
  Stop {offset: 1.0 color:Color.DARKGREEN }
 ]
}
onMouseMoved: function( e: MouseEvent ):Void {
 t.start();
}
},Text {
 font: Font {
  size: 24
  style: FontStyle.PLAIN
 }
 x: 10, y: 30
 content: "HelloWorld"
 fill:RadialGradient{
 centerX:60
 centerY:30
 radius:60
 proportional:false
 stops:[
  Stop {offset: 0.3 color: Color.GREEN},
  Stop {offset: 1.0 color:Color.LIGHTGREEN }
 ]
}
scaleX:3
scaleY:3
}
]
}
}
```

4. 6. JavaFX in ogrodje Swing

Namizna aplikacija, zgrajena s pomočjo tehnologije Swing:

```
public class Okno{
        public static void main(String[] args){
                JFrame okno=new JFrame("okno Swing");
                okno.setSize(250,250);
                okno.setResizable(false);

                JPanel panel=new JPanel();
                Panel.setBackground(Color.RED);

                okno.getRootPane().setContentPane(panel);
                okno.setVisible(true);
        }
}
```

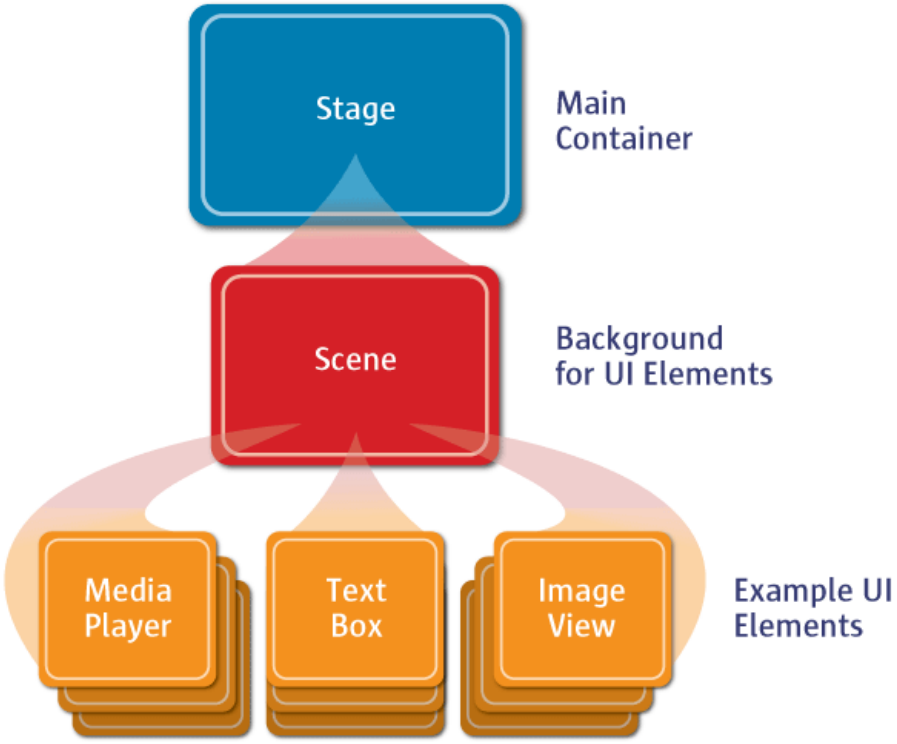
4. 7. Razvoj GUI z JavaFX

Primer okna, zgrajenega s tehnologijo JavaFX skriptnega programskega jezika:

```
Stage{
        title : "okno JavaFX"
        width: 250, height: 250
        resizable: false

        scene: Scene{
                fill: Color.BLUE
        }
}
```

4. 8. Gradnja grafičnega vmesnika



Slika 7: Sestava grafičnega vmesnika

Nov koncept za JavaFX uporabniške vmesnike je, da so vsi objekti prikazani na Stage (oder), ki je kontejner za vse elemente grafičnega vmesnika aplikacije. Z uporabo paketa javafx.stage.Stage kreiramo glavni kontejner za prikaz uporabniškega vmesnika. Na Stage (oder) se postavi Scene (scena) z uporabo paketa javafx.scene. JavaFX Scene je kot nekakšna risarska površina, na katero lahko po želji dodajamo grafične elemente uporabniškega vmesnika [44].

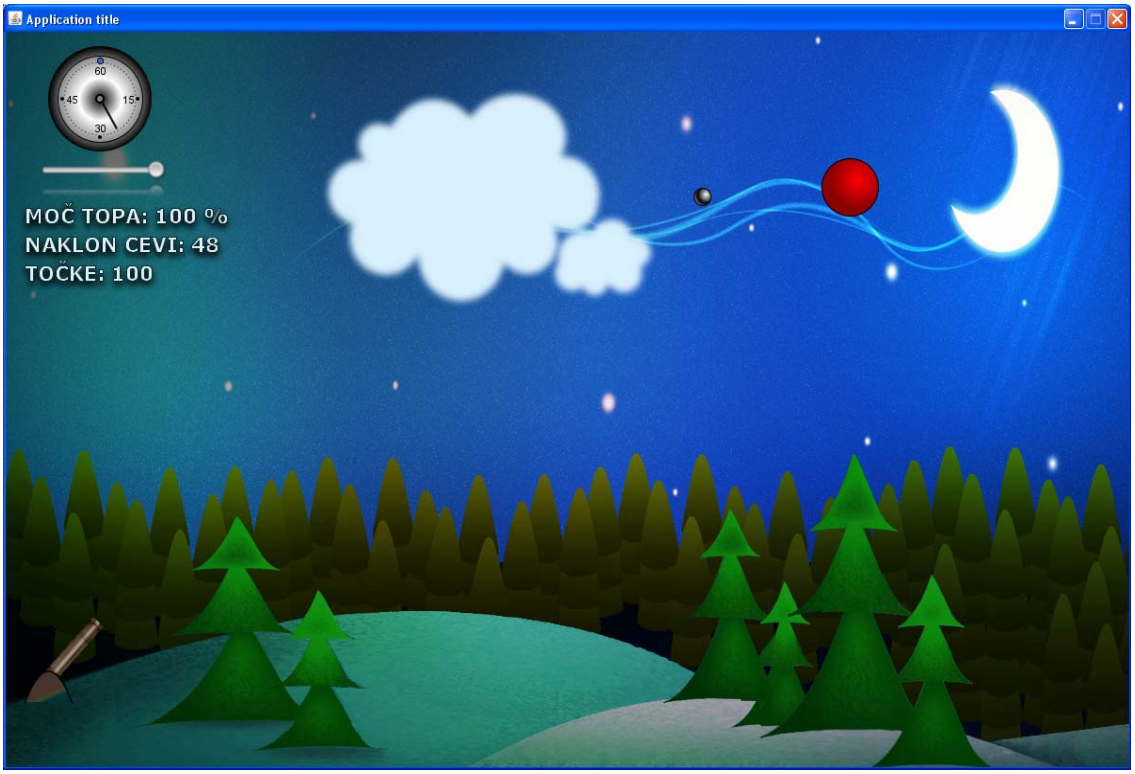
Primer:

```
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
...

Stage {
 scene: Scene {
       content: [
          Text {
             font: Font {
                 size: 16
             }
             x: 10
             y: 20
             content:
                    //Spinning Faces Example
                    //Can be ImageViews, TextBoxes, MediaPlayer, your own Custom Nodes...
                    ...
                    }
         }
  ...
  }
...
 }
```

5. Primer 2D igre

V sklopu diplomske naloge sem izdelal 2D igro, s katero sem želel predstaviti tehnologijo JavaFX na konkretnem primeru, ki bi bil nekoliko obsežnejši od tistih, ki sem jih podajal v prejšnjem poglavju. Pri igri sem hotel predstaviti čim več animiranja objektov, kot so na primer premikajoči se oblak, kazalec na štoparici, ki prikazuje dani čas za igranje igre, cev topa, izstrelek, izpis točk na poziciji zadete tarče, prikaz nove tarče na zaslonu. Možno bi bilo podati še veliko več animiranja, vendar sem zaradi hitrosti delovanja igre izbral kompromis med animiranjem in hitrostjo.



slika 8: Primer 2D igre

Poleg tega sem se pri razvoju igre nagibal k temu, da bi bila vizualno všečna z uporabo senčenja, prehajanja med barvami, odsevanja, dodajanja slik v igro ... Želel sem prdstaviti, da je z JavoFX mogoče razviti uporabniku všečno, vizualno bogato in učinkovito aplikacijo.

Na začetju so kratka navodila za igranje. Ob pritisku na tipko enter se igra prične. S tipkami gor, dol uravnavamo naklon cevi topa. Z drsnikom pod štoparico je omogočena regulacija moči topa, torej izstrelne hitrosti, ki je privzeto nastavljena na maksimum. Poleg že omenjene štoparice in drsnika so v levem zgornjem kotu tudi izpisi parametrov, s katerimi smo seznanjeni z osnovnimi parametri med igranjen igre. Prvi je moč topa, ki procentualno prikazuje začetno hitrost izstrelka. Pod izpisom moči topa se nahaja naklon cevi, ki prikazuje trenutni naklon cevi v stopinjah. Izpis točke pa prikazuje vsoto trenutnih točk, ki so bile zbrane med trenutno igro. Posamezna igra traja

60 sekund, v katerih je potrebno zadeti čim več tarč. Namreč, med pojavljanjem nove tarče se ta povečuje do določenega radija kroga in čim manjši radij zadanemo med povečevanjem tarče, več točk dobimo. Krogla, ki jo izstrelimo, potuje fizikalno gledano kot poševni met, kar tudi nekoliko otežuje igralcu uspešnost pri zadetkih. Kot že rečeno, je začetno hitrost mogoče regulirati in s tem spreminjati pot letenja krogle pri enakem naklonu cevi topa. Ob zadetku tarče s kroglo ta izgine in na njenem mestu se pojavi izpis trenutnih točk, ki jih je igralec dobil za zadetek tarče. Trenutne točke animirano pojenjajo in po 2 sekundah popolnoma izginejo. Spet pa se pojavijo, ko zadenemo novo tarčo. Ko je igre konec, sledi še izpis točk, ki smo jih dosegli med igro. Po koncu igre je omogočena možnost za ponovno igranje.

6. Prihodnost JavaFX

Ponudba programskih jezikov za razvoj bogatih spletnih aplikacij je danes pestra. Smiselno se je vprašati, ali je med tehnologijami, kot so *AdobeFlex, Microsoft Silverligth, Ajax ...* še kaj prostora za *JavoFX.*

JavaFX je nova tehnologija, zato še ni tako razširjena na tržišču, kot so ostale. To pomeni, da bo potrebno za obstanek vložiti še mnogo truda in denarja ter da bo JavaFX igrala ključno vlogo pri razvoju bogatih spletnih aplikacij. Ravno tu se kaže še ena negotovost, saj je leta 2008 podjetje Oracle kupilo Sun Microsystems. To pomeni, da bo podjetje Oracle moralo imeti vizijo ter videti smiselnost za vlaganje virov v razvoj te tehnologije. V nasprotnem primeru bi bila edina smiselna možnost, da bi razvoj prepustili širšim množicam kot odprto kodno tehnologijo. Java zagonsko okolje (runtime) je nameščeno na različnih napravah kot JavaSE oziroma JavaME. Zato ima JavaFX velik potencial, saj lahko neposredno komunicira z Javo, pa tudi pri razvoju aplikacije uporablja Javine knjižnice. Flex in Silvelight sta sedaj v prednosti, saj sta bolj uveljavljena na tržišču, zato JavoFX čaka težka pot. Dobra novica za JavoFX je, da je danes Java široko uporabljena in JVM zastopan na vseh računalniških platformah. Tako lahko izkorišča ves potencial, ki ga nudi Java. Ponuja tudi intuativno sintakso, ki je enostavna za razumevanje ter se jo da naučiti. Dejstvo je, da JavaFX vstopa relativno pozno na trg kot thnologija za razvoj RIA. Z investiranjem v JavoFX bi se povečala možnost za prodor na nova tržišča (prenosni telefoni, televizije, Blue-ray predvajalniki …) in se ustvarila kunkurenca med ponudniki podobnih tehnologij. Zato bosta morala tudi Adobe in Microsoft še dodatno izbolješevati tehnologije za razvoj bogatih spletnih aplikacij [45, 46, 47].

7. Sklepne ugotovitve

V diplomski nalogi sem naredil pregled pomembnejših tehnologij za razvoj RIA, predstavil naraščujoči trend uporabe RIA ter široko uporabo te tehnologije na tržišču. Opisal sem, kaj so RIA, kakšne so njihove lastnosti, delovanje le-teh ter prednosti in slabosti, ki jih imajo z vidika navadnih spletnih aplikacij. Vsako tehnologijo za razvoj RIA sem predstavil, opisal glavne značilnosti ter jo skušal prikazati čim bolj objektivno.

Najbolj zanimiva sposobnost RIA aplikacij, ki sem jo opazil pri izdelavi diplomskega dela, je konvergenca spleta in namizja. Torej aplikacije, ki se razvijejo za splet, pričenjajo prodirati na namizja. Te aplikacije je možno tudi »osvoboditi« iz brsklalnika in spletno aplikacijo interaktivno prestaviti na namizje. Pri razvoju RIA tehnologij prihaja do konfliktov interesov, saj organizacije, ki vplivajo na razvoj standardov(Adobe, Microsoft, Google …), same vlagajo veliko sredstev in energije v razvoj lastnih RIA tehnologij.

Pri izdelavi diplomskega dela sem se osredotočil na novo tehnologijo za razvoj RIA, JavaFX. JavaFX je tehnologija, ki ima velik potencial in možnosti za uspeh. Predstavil sem arhitekturo tehnologije ter novosti, ki jih prinaša. Skozi kratke primere in pri izdelavi 2D igre sem se naučil sintakse JavaFX Script, ki je različna od Jave. JavaFX Script je skriptni jezik, ki omogoča deklarativno programiranje, kar pomeni, da povemo, kaj hočemo, slednja pa ugotovi, kako naj želeno uresniči. Deklarativno programiranje je še posebej močno v izražanju interaktivnih razmerij med grafičnimi elementi, kar poenostavi in pospeši razvoj. Pomembna lastnost, ki jo prinaša JavaFX, je tudi vezanje podatkov (data binding). JavaFX Script je zasnovan tako, da lahko razvijalec brez težav realizira svoje zahteve. Večina programskih jezikov zahteva od razvijalca, da se točno izrazi, kako izvesti določen učinek. Ta lastnost je izrazito vidna pri tem, da omogoča zelo hiter razvoj grafičnih aplikacij. Omogoča razvoj tako bogatih internetnih aplikacij kot mobilnih ter namiznih aplikacij. JavaFX temelji na platformi Java, ki je nameščena od mobilnih telefonov pa vse do superračunalnikov. Ker temelji na Javi, je mogoča tudi uporaba vseh njenih knjižnic. To je tudi priložnost, da se uporabi že obstoječa programska koda, napisana v Javi, s kodo, napisano v JavaFX Script jeziku.

JavaFX ima močno konkurenco, vendar ravno zaradi Jave, ki je danes široko uporabljena, vidim prihodnost in razvoj tudi za tehnologijo JavaFX. Za njeno večjo uveljavitev in prodor bo potrebno slediti razvoju ostalim – konkurenčnim – tehnologijam za razvoj bogatih aplikacij. Vsakrša konkurenca je dobrodošla, saj imamo tako razvijalci na voljo kvalitetnejše in učinkovitejše tehnologije za razvoj bogatih spletnih aplikacij.

Slike

Tabele

Viri

[1] (2009) Srečanje MSDN in TechNet. Dostopno na: http://www.microsoft.com/slovenija/dogodki/msdntech/srecanje_msdntech_nov07.mspx#Zupancic

[2] (2009) What is RIA? Dostopno na: http://www.ehow.com/about_5376400_ria.html

[3] (2009) Bogate spletne aplikacije v izobraževanju. Dostopno na: http://www.sirikt.si/slo/sirikt2009/plenarna_predavanja/sasa_divjak.html

[4] (2009) Microsoft - Silverlight 3 in Expression3. Dostopno na: http://www.buyitc.si/novice/2971/Microsoft-Silverlight-3-in-Expression-3-prinasata-vrsto-novosti-z-zanimanjem-jih-pricakujejo-tudi-slovenski-partnerji

[5] (2008) SUMMER SCHOOL 2008. Dostopno na: http://colos.fri.uni-lj.si/SUMMER_SCHOOL_2008/GRADIVA/ria.ppt

[6] Adobe Flex 3.0 For Dummies, 2008

[7] (2009) Adobe Flex 3. Dostopno na: http://www.adobe.com/products/flex/owerview

[8] (2009) Adobe Flex 3. Dostopno na: www.adobe.com/products/flex/faq/

[9] (2010) Adobe Flash. Dostopno na: http://en.wikipedia.org/wiki/Adobe_Flash

[10] (2009) What is Adobe® AIR®? Dostopno na: http://www.adobe.com/products/air/faq/

[11] Foundation ActionScript 3.0 with Flash CS3 and Flex, 2008

[12] (2009) Microsoft Silverlight to run on Moblin devices. Dostopno na: http://www.infoworld.com/t/application-development/microsoft-silverlight-run-moblin-devices-989

[13] (2010) Moonlight (runtime). Dostopno na: http://en.wikipedia.org/wiki/Moonlight_(runtime)

[14] Microsoft SilverLight. Dostopno na: http://silverlight.net/GetStarted/overview.aspx

[15] (2009) Izšel je SilverLight 3. Dostopno na:
http://www.monitor.si/novica/izsel-je-silverlight-3/

[16] (2009) Microsoft SilverLight 3. Dostopno na:
http://www.racunalniske-novice.com/novice/programska-oprema/novice-in-dogodki/microsoft-silverlight-3.html

[17] (2009) Silverlight 4 Beta Information. Dostopno na:
http://www.silverlight.net/getstarted/silverlight-4-beta

[18] (2009) Silverlight for Mobile. Dostopno na:
http://silverlight.net/learn/mobile/

[19] (2009) Intel's Moblin 2.1 Linux Operating System to Support AIR, Silverlight. Dostopno na:
http://www.streamingmedia.com/article.asp?id=11387

[20] (2009) Microsoft Silverlight in Mobli. Dostopno na:
http://www.racunalniske-novice.com/novice/dogodki-in-obvestila/microsoft-silverlight-in-moblin.html

[21] (2009) MOBLIN OVERVIEW. Dostopno na:
http://moblin.org/documentation/moblin-overview

[22] (2010) AJAX (programiranje). Dostopno na:
http://sl.wikipedia.org/wiki/AJAX_(programiranje)

[23] AJAX And PHP – Building Responsive Web Applications. Založba PACKT, 2006

[24] (2010) JavaFX. Dostopno na:
http://en.wikipedia.org/wiki/JavaFX

[25] (2010) JavaFX getStarted. Dostopno na:
http://java.sun.com/javafx/1/tutorials/core/getStarted/

[26] (2010) JavaFX overview. Dostopno na:
http://javafx.com/about/overview/

[27] (2010) JavaFX Mobile. Dostopno na:
http://en.wikipedia.org/wiki/JavaFX_Mobile

[28] (2007) KONFERENCA JAVAONE 2007. Dostopno na:
http://home.izum.si/COBISS/OZ/2007_3/html/clanek_07.html

[29] (2010) Why JavaFX Script? Dostopno na:
http://openjfx.java.sun.com/current-build/doc/reference/ch01s01.html

[30] (2010) Three Reasons Why Your Next Java ME Mobile Application Should Include JavaFX Mobile. Dostopno na:
http://java.sun.com/developer/technicalArticles/javame/javafxmobile-javame/

[31] (2009) Kinds of Variables. Dostopno na:
http://openjfx.java.sun.com/current-build/doc/reference/ch03s02.html

[32] (2009) Access Modifiers. Dostopno na:
http://openjfx.java.sun.com/current-build/doc/reference/Modifiers.html

[33] (2009) Kinds of Functions. Dostopno na:
http://openjfx.java.sun.com/current-build/doc/reference/ch04s02.html

[34] (2010) Learning the JavaFX Script Programming Language - Tutorial Overview. Dostopno na:
http://java.sun.com/javafx/1/tutorials/core/operators/

[35] (2010) Expressions. Dostopno na:
http://java.sun.com/javafx/1/tutorials/core/expressions/index.html

[36] (2009)Obravnava izjem. Dostopno na:
http://209.85.129.132/search?q=cache:eVJri9BDEQcJ:lgm.fri.uni-lj.si/PA/IZJEME/izjeme.ppt+izjeme+v+Javi&cd=6&hl=sl&ct=clnk&gl=si

[37] (2009) Binding. Dostopno na:
http://openjfx.java.sun.com/current-build/doc/reference/ch07s01.html

[38] (2009) JavaFX Script Language: Data binding & Triggers. Dostopno na:
http://209.85.135.132/search?q=cache:py-FAxaokIsJ:www.javapassion.com/javafx/javafx_binding.pdf+vezanje+podatkov+javaFX&cd=8&hl=sl&ct=clnk&gl=si

[39] (2009) Data Binding and Trigger Basics. Dostopno na:
http://java.sun.com/javafx/1/tutorials/core/dataBinding/

[40] (2010) animiranje grafičnih objektov. Dostopno na:
http://javafx.com/docs/articles/animation_basics/

[41] (2010) JavaFX – Animation. Dostopno na:
http://java.sun.com/javafx/1/docs/api/javafx.animation/javafx.animation.Interpolator.html

[42] (2010) Java 3D API Tutorial. Dostopno na:
http://java.sun.com/developer/onlineTraining/java3d/index.html

[43] (2009) 25 (Free) 3D Modeling Applications You Should Not Miss. Dostopno na:
http://www.hongkiat.com/blog/25-free-3d-modelling-applications-you-should-not-miss/

[44] (2010) UI Built on Stage, Scene and Content. Dostopno na:http://javafx.com/docs/articles/midlet/

[45] (2010) The Future of JavaFX? Dostopno na: http://piliq.com/javafx/?p=872

[46] (2010) The Future of JavaFX? Dostopno na: http://www.compare-review-information.com/future-of-javafx/

[47] (2008) JavaFX and its future. Dostopno na: http://blog.cjtech.co.uk/index.php/2008/12/08/javafx-future/

Izjava o samostojnosti dela

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorja prof. dr. Saše Divjaka.

Ljubljana, 9.3.2010 Aleš Vranešič