SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

ELLA GRACIN

IZRADA APLIKACIJE ZA GENERIRANJE VEKTORSKIH OBLIKA S INTERPRETACIJOM U POSTSCRIPT PROGRAMSKI KOD

DIPLOMSKI RAD

Zagreb, 2019.

Sveučilište u Zagrebu
Grafički fakultet

ELLA GRACIN

IZRADA APLIKACIJE ZA GENERIRANJE VEKTORSKIH OBLIKA S INTERPRETACIJOM U POSTSCRIPT PROGRAMSKI KOD

DIPLOMSKI RAD

Mentor:
Doc.dr.sc. Tibor Skala

Student:
Ella Gracin

Zagreb, 2019.

ZAHVALA

Ovaj je rad od ideje do realizacije pratio neposredni voditelj Vladimir Cviljušac, kojemu želim zahvaliti na pomoći, savjetima, strpljenju, uloženom vremenu i riječima motivacije. Hvala na svemu!

Hvala i svim prijateljima i dobrim ljudima s Grafičkog fakulteta koji su mi uljepšali godine studiranja.

SAŽETAK

Razvoj tehnologije, a time i raznih aplikacija i programa, značajno pridonosi napretku u grafičkoj industriji, posebno u granama grafičkog dizajna. Početak većeg napretka u grafičkom dizajnu obilježio je nastanak grafičkih programskih jezika namijenjenih programiranju vektorske grafike. Dizajnerima je danas posao stvaranja grafike koristeći prigodne programske alate znatno olakšan, a to mogu zahvaliti nizu aplikacija namijenjenih stvaranju vektorske grafike kao što su Illustrator, InkScape, Sketch i slično, koji omogućuju vizualno manipuliranje oblicima. Ipak, iako je sigurno reći da se radi o velikom pomaku u odnosu na korištenje programskih jezika za izradu grafike, niti jedan od postojećih aplikacija za vizualno generiranje vektorskih oblika ne nudi sve opcije koje je moguće izvesti koristeći PostScript, koji je poznat kao jedan od prvih grafičkih programskih jezika. Jedna od velikih prednosti PostScripta je u tome što omogućuje manipulaciju grafičkim elementima programskim kodom (korištenje petlji, uvjeta grananja, visoka preciznost...) uz moguće promjene svih dostupnih parametara (boje, debljine linije, veličine stranice i slično). Mogućnosti PostScripta u izradi grafike nisu ograničene, ali njegovo korištenje, pogotovo u izradi kompleksnijih oblika, zahtjeva znatno više vremena potrebnog za učenje, ali i za samu izradu u odnosu na korištenje nekih od ranije spomenutih aplikacija. Cilj ovog rada je izrada aplikacije koja omogućuje stvaranje vektorskih oblika uz vizualnu interpretaciju u stvarnom vremenu, a u pozadini dobivene oblike prevodi u PostScript programski kod. Pritom se u izradi aplikacije važnost osim funkcionalnosti daje i izradi oku ugodnog i intuitivnog grafičkog korisničkog sučelja, u čemu pomaže analiza postojećih programa namijenjenih kreiranju vektorskih oblika. Na taj način korisniku je omogućeno da na mnogo brži i lakši način dođe do potrebnog koda te se lakše može upoznati s PostScriptom kao programskim jezikom i načinom na koji funkcionira. Sve to korisniku znatno olakšava učenje kroz olakšano razumijevanje povezanosti programskog koda i računalne grafike.

Ključne riječi: PostScript, Aplikacija, Računalna grafika, Korisničko sučelje

ABSTRACT

The development of technology, and thus various applications and programs, significantly yields progress in the graphics industry, especially in graphic design branches. The start of noticeable progress in graphic design has marked the origin of graphic language programs used for programming vector graphics. The job of making graphics using the appropriate programming tools is considerably easier for modern designers thanks to various apps designed to create vector graphics like Illustrator, InkScape, Sketch and such, which enable visually manipulating shapes. Still, although it is safe to say that it is a major upgrade compared to the use of programming languages to make graphics, not one of the existing applications used for making vector shapes offers all options that are possible using PostScript, which is known as one of the first graphics programming languages. One of the great advantages of the Postscript is that it allows users to manipulate the graphical elements with the programming code (use of loops, branched conditions, high precision...) with possible changes to all available parameters (color, line thickness, size and so on). Postscript options in graphic design are not limited, but its use, especially in the design of more complex forms, requires considerably more time for learning, but also for the creation of some of the aforementioned applications. The aim of this work is to create an application that enables the creation of vector shapes with a real-time visual interpretation and simultaneously translates resulting shapes into PostScript code. At the same time, besides functionality, the application also gives importance to a pleasant and intuitive graphical user interface, which was accomplished by analyzing some of the existing programs used for the creation of vector shape. In this way, the user is able to get the necessary code in a much faster and easier way, and it is easier to get acquainted with Postscript as a programming language and the way it works. All this makes it much easier for the user to learn thanks to a facilitated understanding of a connection between the programming code and computer graphics.

Keywords: PostScript, App, Computer graphics, User interface

SADRŽAJ

1. UVOD

Jedno od bitnih obilježja razdoblja u kojem živimo je ubrzan razvoj tehnologije. Izrada aplikacija prilagođenih raznim uređajima postaje sve brža i lakša, a time dolazi i do popularnosti zanimanja koja se bave takvim poslovima, čemu u prilog svakako ide činjenica da su korisnici neprestano u potrazi za novim i boljim programima koji će im olakšati i ubrzati izvršavanje poslovnih procesa. Desktop aplikacija „PostScript Builder“ čiji se razvoj od ideje do realizacije prati u ovom radu nastala je s ciljem da olakša učenje PostScripta kao grafičkog programskog jezika koji se danas još rijetko može pronaći u široj komercijalnoj upotrebi. Razlog tome je nedostatak grafičkog sučelja za upravljanje PostScript mogućnostima. PostScript Builder omogućuje izradu jednostavnih vektorskih oblika uz korištenje grafičkog korisničkog sučelja putem kojega će korisnik, nakon što vizualno generira oblike, dobivenu grafiku prevesti u PostScript programski kod. Iako se čini kako upravo zahvaljujući novim aplikacijama i programima koji su s godinama i novim verzijama uveli gotovo sve mogućnosti koje su potrebne za brzo i lako stvaranje računalne grafike, neke su od mogućnosti PostScripta do današnjeg dana ostale nezamjenjive novim verzijama modernih aplikacija. Osim što će olakšati učenje grafičkog programskog jezika, aplikacija razvijena u ovom radu bi, iako ne nudi sve mogućnosti PostScripta, daljnjim razvojem mogla doprinijeti povećanju njegove popularnosti.

2. TEORIJSKI DIO

2.1. Objektno orijentirano programiranje

U objektno orijentiranom programiranju sve se operacije odvijaju kroz objekte i njihove metode. Takav je način programiranja inspiriran predmetima iz stvarnog života koji se lako mogu opisati kao objekti, od kojih je svaki definiran nekim metodama. [1] Objekt je naziv za skup svojstava koje možemo objediniti u smislenu cjelinu. Klasa (*Class*) je definicija objekta, odnosno pravila koja opisuju njegova svojstva. Klasa je, dakle, samo opis objekta, dok je objekt konkretna realizacija napravljena na temelju razreda. Klasa može naslijediti svojstva prethodno definiranih klasa, sve njihove metode i atribute te dodati neke nove. [2] Pojam klase uveden je 1967. godine kada se pojavljuje Simula, prvi programski jezik sa svojstvima objektno orijentiranih jezika, dok se prvi pravi objektno orijentirani programski jezik pod nazivom Smalltalk pojavljuje 1972. godine. Njihova je pojava dovela do revolucije u programiranju jer objektno orijentirani jezici osim što omogućuju ponovno korištenje gotovih programskih komponenti usmjeravaju programere na rad u okvirima standarda. Takav kod lakše je mijenjati i ponovno koristiti što olakšava daljnji razvoj. [3]

2.1.1. Model View Controller (MVC)

Kako bi se lakše shvatio princip nastanka aplikacije čiji se razvoj prati u ovom radu, bitno je objasniti *Model View Controller* (*MVC*) kao obrazac softverske arhitekture koji se koristi za odvajanje dijelova aplikacije u komponente ovisno o namjeni. *MVC* odvaja aplikaciju na tri dijela – model, grafički prikaz i upravitelj. Model se sastoji od podataka, pravila, logike i funkcija ugrađenih u programsku logiku. Objekti modela vraćaju i spremaju stanje modela u bazu podataka. *View* (grafički prikaz) opisuje korisničko sučelje te omogućuje prikaz prethodno moderiranih podataka poput obrazaca, tablica ili slično. Prikazuje podatke koristeći model i korisniku dopušta da iste podatke promijeni. Upravitelj upravlja korisničkim zahtjevima, prihvaća ulazne naputke i pretvara ih u naloge upućene

modelu ili grafičkom prikazu te povezuje model i grafički prikaz određujući način na koji će reagirati na korisničke zahtjeve. [4]

Gotovo svi programski jezici koriste *MVC* u nekom obliku uz blage varijacije, a takva arhitektura olakšava nezavisan razvoj, testiranje i održavanje aplikacije. Takav je obrazac koristan kod planiranja aplikacije jer olakšava organizaciju i realizaciju ideje u programski kod. Taj princip rada stoji iza ideje da bi se programski kod trebao razvrstavati prema svojoj funkciji, a zahvaljujući popularnosti u primjeni, omogućuje i drugim programerima lako snalaženje u kodu te olakšava suradnju više programera na jednom projektu.

Prilikom objektno orijentiranog programiranja bitna je redukcija ovisnosti elemenata *MVC* obrasca, što znači da model mora funkcionirati bez greške neovisno o stanju u kojem se grafički prikaz u tom trenutku nalazi, ali isto je tako važno da i u prikazu nema smetnji koje bi se mogle pojaviti ovisno o pojedinim vrijednostima modela

2.1.2. ActionScript 3.0

Programski jezik ActionScript 3.0 koji je korišten u ovom radu objektno je orijentiran jezik, a osim objekata i klasa definiranih prilikom programiranju aplikacije, u izradi su poslužile i prethodno definirane klase sadržane u korištenom integriranom razvojnom okruženju. Kao i svaki programski jezik ActionScript je određen nizom pravila koje je potrebno pratiti, a pravila sadrže načine na koje se pomoću niza riječi i znakova strukturira kod za postizanje željenih rezultata. [5] ActionScript prepoznaje razliku između velikih i malih slova te je bitno razlikovati ih u programskom kodu. Osim toga, bitno je prilikom programiranja važno je pravilno koristiti interpunkcijske znakove i pravilno označavati i puniti varijable. Lakšem snalaženju u ovom programskom jeziku doprinosi mogućnost ostavljanja komentara, a to je moguće napraviti na dva načina. Prvi od načina je dodavanje dvije kose linije ispred reda teksta i koristi se samo u slučaju da se radi o jednoj liniji koda. Drugi način odnosi se na veće odlomke i omogućuje komentiranja većih dijelova teksta dodavanjem kose crte i

zvjezdice prije teksta te zvjezdice i kose crte nakon njega, a u tom slučaju komentar nije ograničen brojem redova. [6]

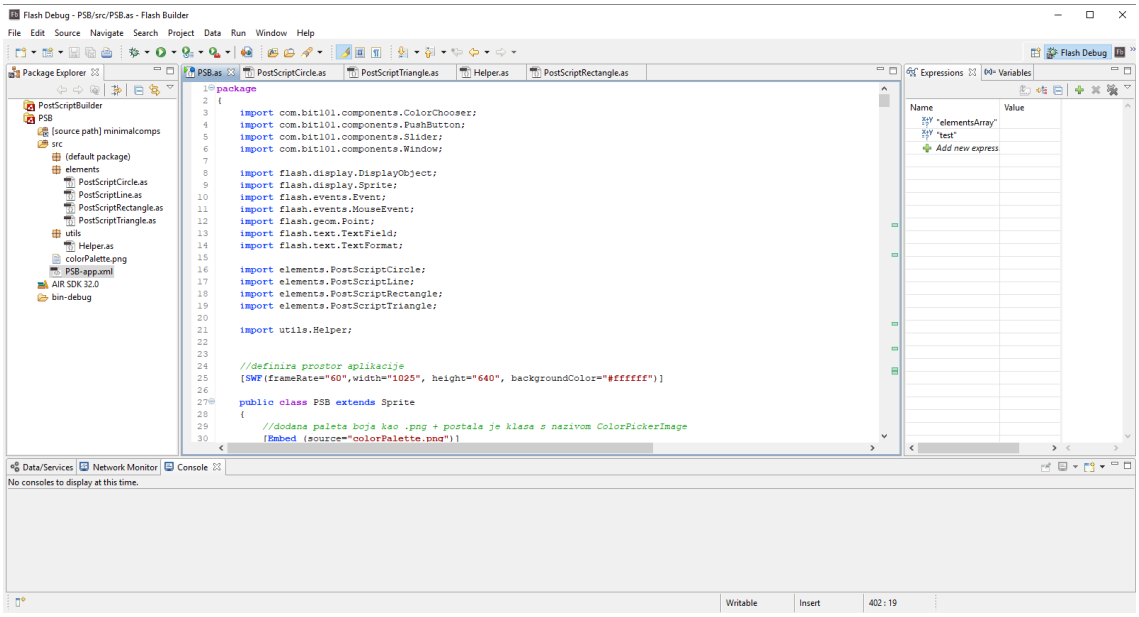
Kroz ovaj će se rad u eksperimentalnom dijelu pobliže objasniti neke od funkcija korištenih za dobivanje aplikacije i principi njihovog korištenja u ActionScript programskom jeziku.

2.2. Razvojno okruženje

U programerskom smislu, razvojno okruženje (*framework*) označava slojevitu strukturu koja se sastoji od gotovih kodova i programa te korisniku omogućuje selektivne izmjene u svrhu razvoja aplikacije. Često sadrži prethodno definirane klase i funkcije koje mogu biti korištene za procesiranje ulaznih informacija, upravljanje uređajima i interakciju sa sistemskim softverom, a u sebi ima sadržan i *Application Programming Interface (API)*. [7]

S druge strane, razlikujemo integrirano razvojno okruženje (*eng. Integrated Development Environment - IDE*), čiji je naziv u engleskom jeziku znatno različit od ranije spomenutog „*frameworka*“, dok u hrvatskome jeziku čini malu, ali značajnu razliku. Integrirano razvojno okruženje čini specijalizirani softverski paket koji sadrži uređivač teksta prilagođen pisanju programa, prevoditelj programa, alate za testiranje programa i alate za lociranje pogrešaka. Integrirano razvojno okruženje, kao i *framework*, u sebi sadrži knjižnice gotovih programskih kodova, a ključna je razlika u tome što korisniku pruža i predodređene elemente korisničkog sučelja koje možemo koristiti prilikom razvoja aplikacija. [8]

U ovom je radu za izradu „PostScript Builder“ aplikacije odabran objektno orijentiran programski jezik, ActionScript 3.0, a integrirano razvojno okruženje kojim je olakšano njegovo korištenje je Flash Builder 4.7 (slika 1). *Adobe Flash* Builder namijenjen je izradi igara i aplikacija uz korištenje ActionScript programskog jezika i Flex *frameworka*.



Slika 1. Prikaz integriranog razvojnog okruženja Flash Builder 4.7

2.3. PostScript (grafički programski jezik)

Računalna grafika definirana je kao vid grafike u kojoj se za generiranje, memoriranje, obradu i prezentiranje slikovnih sadržaja koristi računalo. Iako se računalna grafika u pravom smislu riječi pojavljuje 60-ih godina prošloga stoljeća, kada nastaju prvi računalno animirani film i prva video igra, neki njezine začetke pronalaze već u 19. stoljeću, kada je James Joseph Sylvester osmislio matričnu notaciju, često korištenu u algoritmima računalne grafike.

PostScript je grafički programski jezik kojega Adobe Systems predstavlja tek 1984. godine kao svoj prvi komercijalni proizvod. Nudi mogućnost iscrtavanja vektorskih oblika korištenjem jednostavnih naredbi. Te su naredbe najčešće kratice ili skupovi engleskih riječi, a one mogu stajati same ili uz vrijednost koja određuje na koji način će se izvršavati. [9] Takve vrijednosti stoje ispred naredbi, a ovisno o naredbi mogu označavati koordinate na kojoj se odvija, vrijednosti za debljinu linije, radijus, boju i slično. Primjerice, linija zelene boje i debljine 3px povučena od A(100,200) do A(200,250) po koordinatnoj osi prikazat će se korištenjem sljedeće naredbe:

```
100 200 moveto
3 setlinewidth
1 0 1 setcmykcolor
200 250 lineto
```

Isječak koda 1. Prikaz koda iz kojeg nastaje zelena linija

Naredbe se u PostScriptu mogu unositi putem tekst editora poput Notepad-a, a dovoljno ih je odvajati razmacima iako je zbog bolje preglednosti i lakšeg snalaženja u kodu poželjno svaku pisati u novom redu. Lakšem snalaženju pomaže i dodavanje komentara, što je se radi dodavanjem znaka „%“ ispred teksta. Ono što se želi prikazati upisujući naredbe može se vidjeti u preglednicima poput GS View-a, iz čega se kasnije može prebaciti u PDF format. Tako prikazane oblike moguće je prikazati na odvojenim stranicama, a prelazak na sljedeću stranicu omogućuje *showpage* naredba.

Ono što PostScript čini naprednim u odnosu na programe koji koriste grafičko korisničko sučelje pri definiranju oblika je mogućnost da se željeni elementi programski odrede s promjenama kroz petlju i to uz savršenu preciznost. Svakom je elementu moguće promijeniti parametre poput boje, veličine ili debljine linije kroz određeni broj ponavljanja. Takve mogućnosti svoju primjenu i danas nalaze u zaštitnom tisku. Jedan od načina zaštite korištenjem PostScripta je generiranje oblika čije linije mogu sadržavati jedinstveni isprekidani uzorak. Intervali razmaka između isprekidanih linija koje čine putanju može se izvaditi iz polja, a može činiti i nasumične brojeve. Takav je uzorak vrlo teško ponoviti. PostScriptu ne nedostaje mogućnosti niti u odabiru boja te je boju moguće u željenoj vrijednosti. PostScript podržava RGB, CMYK i HSB sustave boja, a svakom je elementu uz boju moguće promijeniti i opacitet. [10]

2.4. Povijest aplikacija za izradu računalne grafike

Kada je riječ o izumima značajnima za razvoj programa namijenjenih stvaranju računalne grafike kakvu poznajemo danas, bitno je spomenuti Ivana Shutehrlanda, koji je već 1963. godine predstavio Sketchpad, revolucionarni

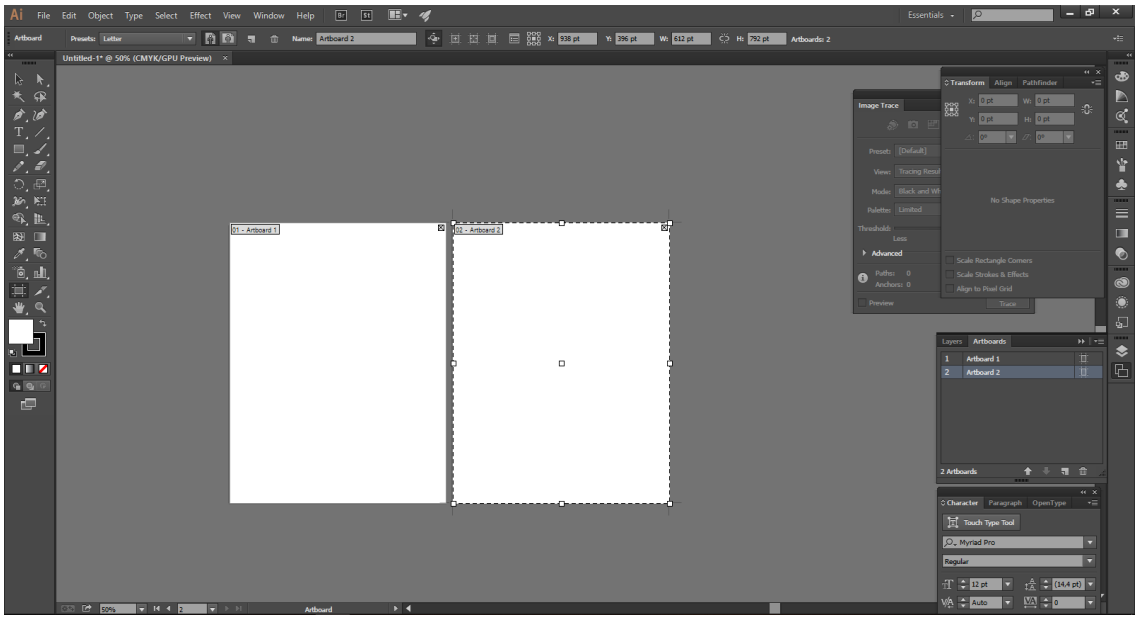
program kojega danas smatramo pretečom modernog računarski potpomognutog crtanja (*eng. Computer-Aided Design - CAD*). Sketchpad je bio prvi program koji je koristio grafičko korisničko sučelje. Korisnici su imali mogućnost pomoću elektroničke olovke povezane s računalom na ekranu iscrtavati željene oblike. [11] Iako je pojava PostScripta označavala napredak u odnosu na Shutherlandov izum, koji je u svojoj prvoj verziji bio ograničen samo na oblike iscrtane pripadajućom olovkom, napredak u tehnologiji doveo je do razvoja programa čiji se korisnici, poput korisnika Sketchpada, služe korisničkim sučeljem te su oni danas u svojoj primjeni zasjenili grafičke programske jezike.

2.5. Pregled današnjih aplikacija za izradu računalne grafike

Kada govorimo o programima namijenjenima izradi grafike, teško je izdvojiti samo jedan od takvih programa koji bi se danas mogao smatrati najpopularnijim. Različite potrebe, ali i osobne preferencije korisnika, kao i razlika u razini znanja i iskustva, dovode do velikog broja programa koji se u određenim situacijama mogu smatrati idealnim. U nastavku će biti predstavljeni neki od grafičkih alata dostupnih u današnje vrijeme:

2.5.1. Illustrator

Kada je 1984. godine Adobe predstavio PostScript, koji je kao grafički alat odgovarao potrebama toga vremena, istodobno započinje razvoj Adobe Illustratora, koji će biti predstavljen 1987. godine kao program namijenjen izradi logotipa i grafičkom dizajnu. Prva je verzija izdana pod kodnim imenom „Picasso“, a već sljedeće godine izlazi nova verzija s velikim brojem novih alata. Adobe već od 1993. nadogradnjom unosi mogućnost razvrstavanja elemenata u slojevima, 3 godine kasnije uvodi mogućnost odabira boje s ekrana, ispune boje i korištenje gradijenta, a tek se 2008. godine pojavljuje mogućnost stvaranja većeg broja površina za crtanje. [12] Brojni su osnovni alati uvedeni već u prvoj verziji, a kasnije nadogradnje većinom donose inovacije koje popravljaju ranije nastale probleme, olakšavaju korištenje postojećih i ubrzavaju rad.



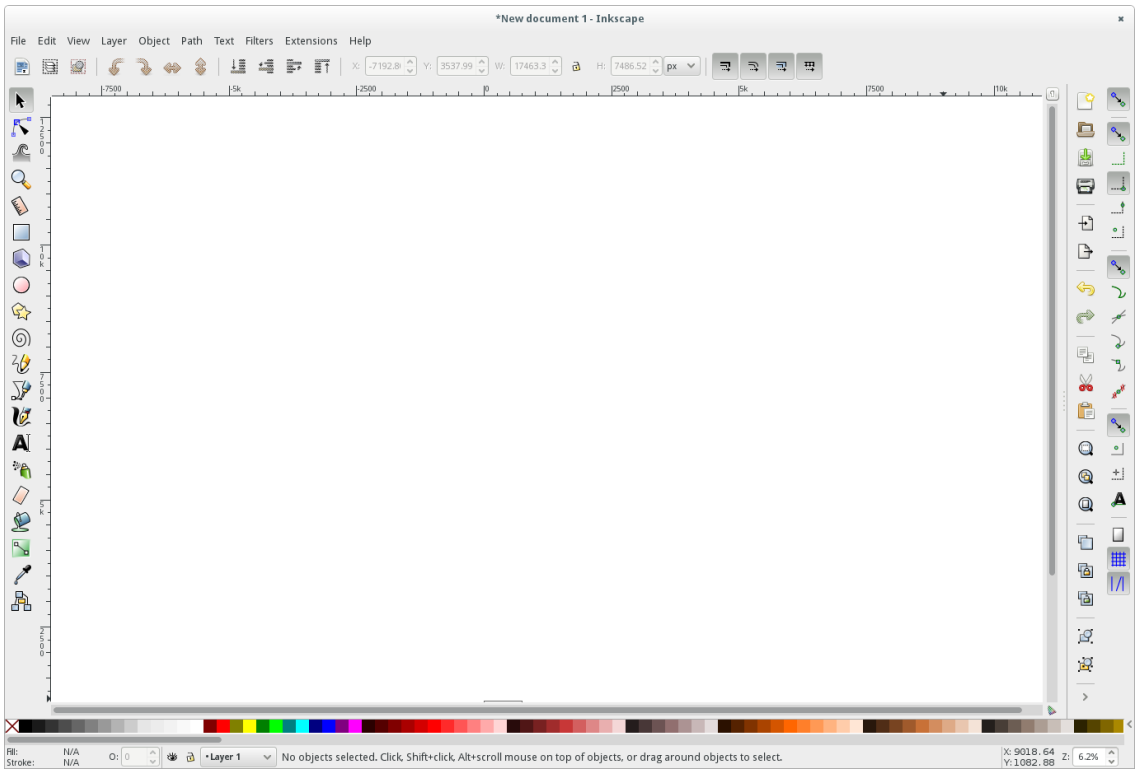
Slika 2. Illustrator - grafičko korisničko sučelje

Najnovija verzija Adobe Illustratora, CC 2019 (23.1.0) sadrži brojne alate posložene na sučelju na kojemu se dugogodišnji korisnici Adobe paketa brzo i lako snalaze. Ipak, veliki broj mogućnosti zahtjeva i određeni broj sati koje je potrebno provesti kako bi se novi korisnik s njima naučio snalaziti ako nije otprije upoznat s dizajnom korisničkog sučelja koji je tipičan za većinu Adobe aplikacija.

Iako je među najpoznatijim alatima koji se koriste u dizajnu upravo zbog niza mogućnosti koje omogućuju kvalitetan rad, veliki nedostatak Illustratora je u visokoj cijeni, što ga čini nedostupnim početnicima koji bi ga koristili u svrhu učenja.

2.5.2. InkScape

InkScape je besplatan program za izradu grafike koji omogućuje lako kreiranje vektorskih elemenata, primarno namijenjen spremanju u SVG (*Scalable Vector Graphics*) formatu iako može biti spremljen i u drugim formatima. Nastao je 2005. godine, a od tada, kao i Illustrator, prati nastanak niza novih mogućnosti i promjena. [13]

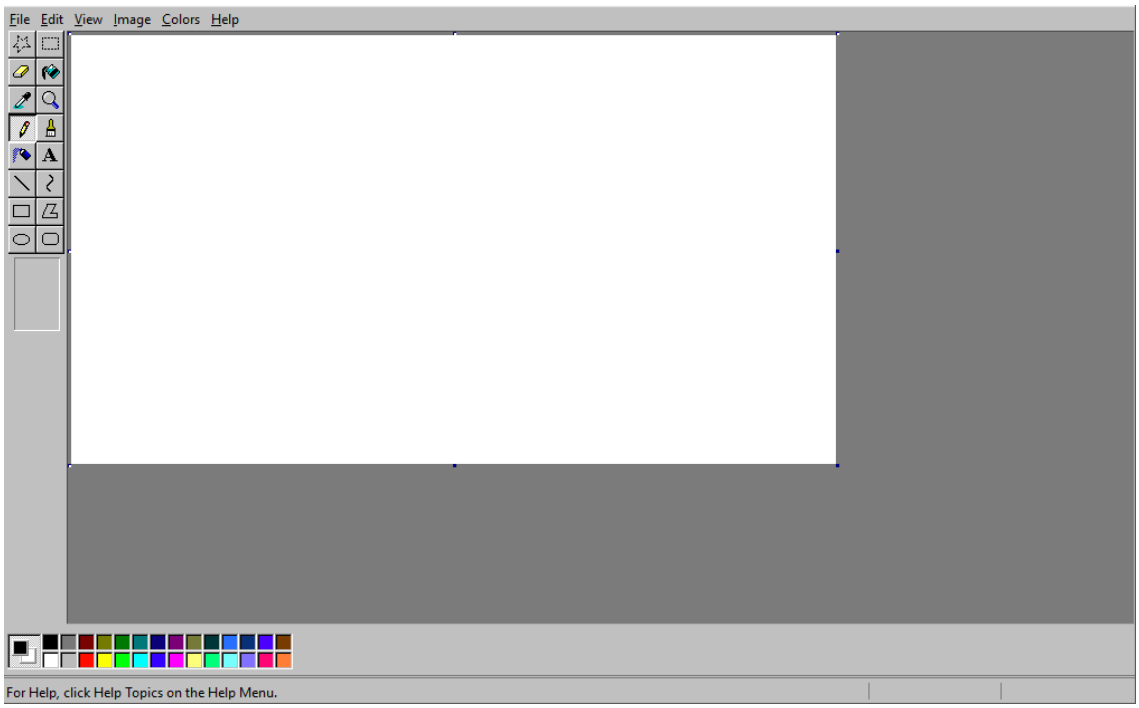


Slika 3. InkScape – grafičko korisničko sučelje

Iako mogućnostima ne zaostaje mnogo za svojim skupim konkurentima na tržištu te sadrži sve osnovne funkcije koje početnicima mogu osigurati kvalitetu i brzinu u radu, InkScape se ne može pohvaliti brzinom rada kojom bi mogao obrađivati grafiku većeg formata. Ipak, nešto smanjen broj mogućnosti dovodi do lakšeg snalaženja na sučelju, što ga čini prigodnim za početnike.

2.5.3. JS Paint

JS Paint je web aplikacija nastala po uzoru na zastarjelu verziju Microsoft Painta, rekreirana u JavaScriptu. Nakon što je microsoftova besplatna verzija povučena i više ne nudi mogućnost preuzimanja, nostalgični korisnici mogu se poslužiti ovom također besplatnom verzijom dostupnom svima koji imaju pristup internetu.



Slika 4. JS Paint – grafičko korisničko sučelje

Ipak, ova aplikacija ne posjeduje mnogo naprednih mogućnosti te jedinu prednost može pronaći u svojoj jednostavnosti i lakoći snalaženja. Najveća bi zamjerka ovakvoj aplikaciji, osim nedostatka brojnih potrebnih funkcija mogla biti i u činjenici da ne nudi mogućnost iscrtavanja vektorskih oblika. Naime, iako će prividno ispisati ravne linije i bezierove krivulje, približavanjem je lako uočiti da se na virtualnom platnu nalaze pikseli, a ne vektorska grafika.

2.6. Korisničko iskustvo

Pojam korisničkog iskustva (*eng. User Experience - UX*) pojavljuje se krajem 20. stoljeća, a označava ponašanja, stavove i emocije koje korisnik doživljava tijekom uporabe određenog proizvoda ili usluge. Kada je riječ o računalnim programima, obično se usko veže uz korisničko sučelje (*eng. User Interface - UI*), prostor u kojemu se odvija komunikacija između računala i čovjeka. Korisničko iskustvo u kontekstu izrade aplikacija čini raspored elemenata na stranici, ali i način na koji su same stranice povezane. [14] Cilj korisničkog iskustva je korisniku omogućiti lako i intuitivno snalaženje čak i bez prethodnog svjesnog učenja. Zahvaljujući sve većoj popularnosti razvoja aplikacija koje su u današnje vrijeme svima lako

dostupne, dolazi do velike konkurencije na tržištu, zbog čega *UX* i *UI* danas imaju ključnu ulogu u njihovoj popularnosti.

2.7. Grafičko korisničko sučelje

Najveća razlika između korisničkog sučelja i korisničkog iskustva u razvoju aplikacija je u tome što je korisničko sučelje vidljivo korisniku. Čini ga niz elemenata koje će korisnik uočiti prilikom korištenja, a s nekima od njih može i „komunicirati“ dodirom ekrana ili klikom miša. [15] Za kreiranje oku ugodnog korisničkog sučelja poželjno je imati dovoljno znanja o dizajnu i pripadajuće vještine. [16] Pomoć u izradi grafičkog korisničkog sučelja nude brojne biblioteke gotovog koda koje sadrže gotove klase i u kojima je već opisan dizajn elemenata. Zahvaljujući detaljnom opisu i dokumentaciji koje su lako dostupne na internetu potrebno ih je samo uklopiti u kod pozivajući naziv klase. Jedna od takvih biblioteka korištena je i u ovom radu. Riječ je o Minimalcomps biblioteci koja je dizajnirana u minimalističkom stilu, bez puno boja, uz jednostavne oblike elemenata i fontove. Budući da u ovom radu naglasak nije bio u dizajnu već u funkcionalnosti, Minimalcomps savršeno odgovara idejnom konceptu aplikacije.

Korisničko iskustvo i grafičko korisničko sučelje bitni su za izradu kvalitetne aplikacije te će prethodna, ranije spomenuta analiza postojećih programa za izradu vektorskih oblika dovesti do odabira prigodnog izgleda grafičkog korisničkog sučelja desktop aplikacije čiji se proces izrade prati u ovom radu.

3. PRAKTIČNA IZVEDBA

3.1. Idejna razrada

Ideja o kreiranju aplikacije PostScript Builder potaknuta je stalnom potrebom za napretkom u razvoju alata korištenih u grafičkom dizajnu. Za svakog je modernog dizajnera u poslu nužno poznavanje te vješto i brzo snalaženje u programima koje koriste, a izbor programa, srećom, nije ograničen. Ipak, iako sadrže brojne napredne mogućnosti u odnosu na aplikacije koje nemaju korisničko sučelje, grafički programski jezici u današnje vrijeme rijetko ostvaruju sve potrebe korisnika, kao što su lakoća u snalaženju i brzina izvođenja zadataka te su kao takvi zamijenjeni brojnim aplikacijama koje omogućuju vizualno kreiranje i manipulaciju vektorskim oblicima. U prilog programskim jezicima ne ide ni činjenica da je za korištenje istih potrebno više vremena provedenih učeći te zahtijevaju korištenje logičkog razmišljanja i povezivanja, dok su aplikacije koje sadrže grafičko korisničko sučelje znatno intuitivnije za korištenje korisnicima bez prethodnog iskustva. Krenuvši od pretpostavke da će olakšano učenje korištenja programskog koda u kreiranju grafike rezultirati njegovom češćom primjenom, razvila se ideja o kreiranju aplikacije koja će to i omogućiti. PostScript Buider korisnicima omogućuje prevođenje vizualno generiranih oblika u PostScript programski kod, što korisnicima olakšava razumijevanje tog programskog jezika. Iako verzija aplikacije razvijena u ovom radu ne nudi napredne mogućnosti, korisnici aplikacije dobit će uvid u osnove programiranja vektorske grafike te će se lakše zainteresirati za učenje, a daljnjim proučavanjem prednosti PostScripta i njegovih principa rada moći će se, ako to požele, lakše prilagoditi njegovom korištenju bez pomoći korisničkog sučelja.

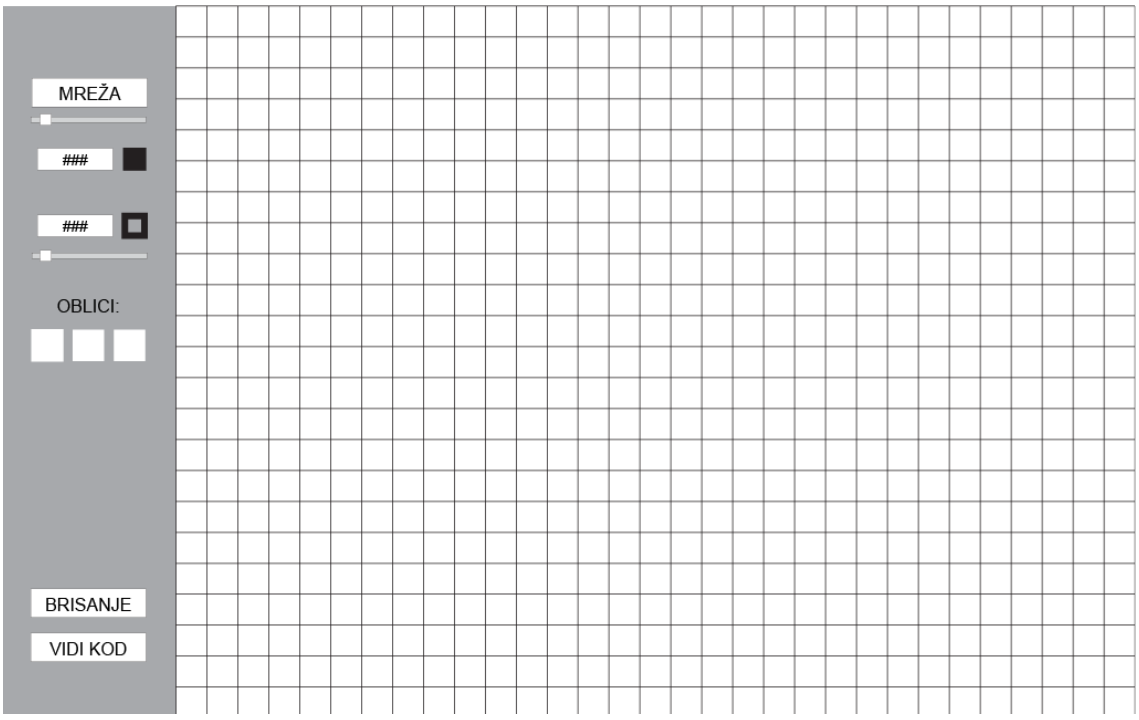
3.2. Ciljana publika

Desktop aplikacija PostScript Builder predviđena je za mlade dizajnere, ali i sve ljude koji žele na lakši način naučiti osnove programiranja vektorske grafike koristeći PostScript. Ona omogućuje prikaz koda vizualno generiranih vektorskih oblika, na temelju kojega će korisnici dobiti uvid u način primjene tog

programskog jezika. Ono što ciljanoj publici posebno olakšava korištenje je intuitivno korisničko sučelje čiji je dizajn inspiriran postojećim aplikacijama kako bi se korisnici lakše snašli zahvaljujući već naučenim principima rad. U današnje vrijeme rijetko se može pronaći mlada osoba koja nije informatički obrazovana barem na osnovnoj razini, stoga se može reći da je aplikacija čiji se razvoj prati u ovom radu u odgovarajućoj mjeri prilagođena ciljanoj publici.

3.3. Blue print / wireframe

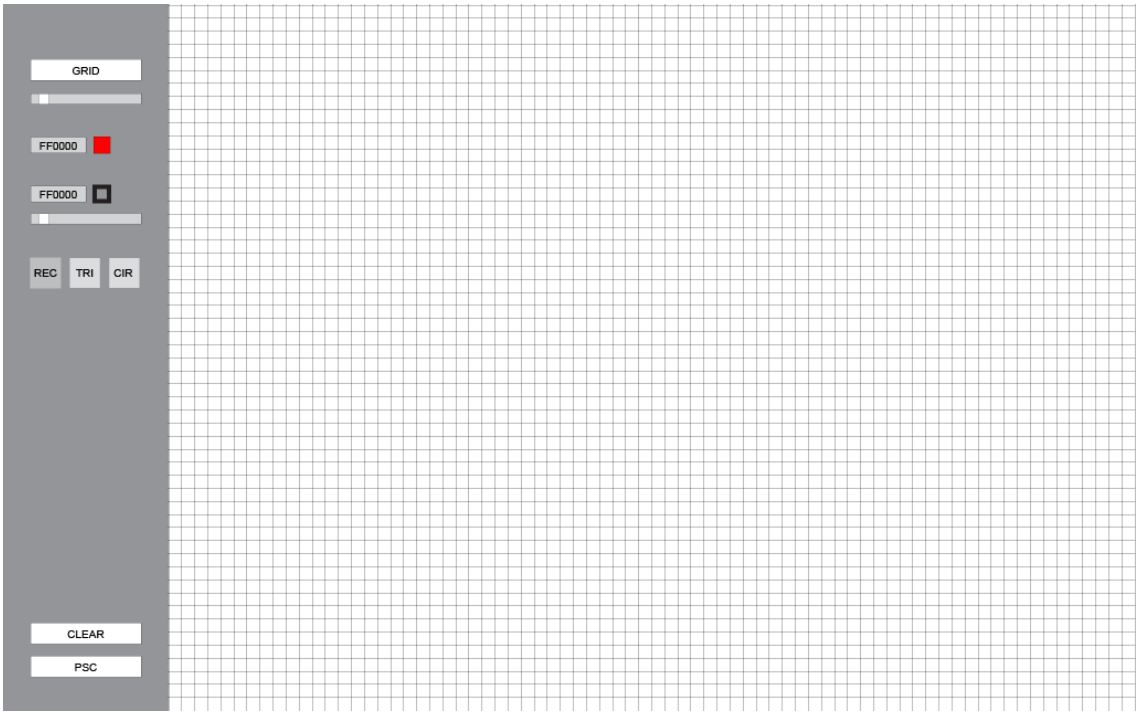
Aplikacija se sastoji od samo jednoga ekrana podijeljenoga na dva dijela, a raspored elemenata prikazan je na slici 5. Prvi dio čini alatna traka na kojoj su smještene sve opcije dostupne korisniku pomoću kojih je moguće definirati što će se i na koji način moći iscrtati na drugom dijelu ekrana. Drugi dio ekrana digitalno je platno na kojem korisnik može dodavati elemente prethodno definirane u alatnoj traci i upravljati njima.



Slika 5. *Blue print* aplikacije PostScript Builder

3.4. Dizajn grafičkog korisničkog sučelja

Prethodna analiza postojećih aplikacija za vizualno upravljanje vektorskim oblicima dovela je do olakšanog shvaćanja potreba korisnika i kreiranja korisničkog iskustva, kao i dizajna grafičkog korisničkog sučelja (slika 6). Korisnici se lakše snalaze u poznatome te je zbog velikog broja aplikacija s jednakom ili sličnom namjenom njihov dizajn korišten kao predložak kako bi se korisnicima olakšalo učenje i razumijevanje aplikacije. Sam dizajn napravljen je minimalistički, bez puno boja, kako bi se naglasak stavio na funkcije aplikacije, a ne na sam izgled, a pritom je korištena gotova biblioteka programskog koda s postojećim klasama - Minimalcomps čime je olakšana izrada aplikacije.



Slika 6. Dizajn korisničkog sučelja aplikacije PostScript Builder

Osim dizajna korisničkog sučelja, izrada aplikacije zahtjeva i dizajn ikone putem koje korisnik pristup aplikaciji. Ikonu je potrebno pirpremiti u nekoliko različitih dimenzija kako bi se zadovoljile sve potrebe prilikom instalacije na različite operativne sustave.

Slika 7. Dizajn logotipa aplikacije

Kao i sama aplikacija, logo je izrađen u minimalističkom stilu, bez korištenja boja (slika 7). Za kraticu naziva aplikacije korišten je jednostavan font bez serifa, a u pozadini su prikazani oblici koje je moguće izraditi korištenjem aplikacije.

3.5. Programiranje grafičkog korisničkog sučelja

Razvoj PostScript Builder aplikacije olakšalo je korištenje Flash Builder razvojnog okruženja. Flash Builder omogućuje kreiranje predloška prazne aplikacije s osnovnim klasama i funkcijama odmah spremnima za korištenje. Prilikom kreiranja nove aplikacije, osim pozivanja svih klasa koje će se koristiti, u glavnom dijelu se definira prostor aplikacije, odnosno veličina ekrana koji će se prikazati. To će se definirati na sljedeći način:

```
[SWF(frameRate="60",width="1025",height="640",
backgroundColor="#ffffff")]
```

Isječak koda 2. Definiranje prostora aplikacije

Osim veličine ekrana pritom se definira i boja pozadine (*background color*) te broj puta koliko će se u sekundi aplikacija izvršavati (*frame rate*) o čemu ovisi fluidnost pokreta. Definirani prostor aplikacije potrebno je detaljnije opisati dodajući vizualne elemente na stranici, odnosno opisujući grafičko korisničko sučelje.

Grafičko korisničko sučelje podijeljeno je na tri dijela. Dva su dijela vidljiva na početku, a to su alatna traka, gdje se nalaze svi gumbi koji korisniku omogućuju

odabir oblika i definiranje varijabli za svaki od elemenata koji se dodaju i radni dio na kojem je moguće dodavati elemente, ali i brisati ih s ekrana. Treći dio dostupan je odabirom „PSC“ gumba s alatne trake, a zamjenjuje radni dio prikazujući PostScript programski kod elemenata prikazanih u radnom dijelu. Izradu korisničkog sučelja znatno je olakšala Minimalcomps biblioteka programskog koda koja sadrži već opisane elemente koji su se u ovom radu dodali na predviđena mjesta. Elementi opisani u toj biblioteci definirani su bojom i oblikom, ali im je naknadno bilo potrebno prilagoditi veličinu i smještaj te definirati funkcije povezane s pojedinim elementima.

Prostor alatne trake opisan je unutar „drawToolbar“ funkcije kao pravokutnik kojemu su definirane dimenzije i boja, a preko kojega su dodani svi gumbi i pomočni alati zaduženi za dodavanje oblika i promjene njihovih vrijednosti.

```
toolbar = new Sprite();

toolbar.graphics.beginFill(0x949494);
toolbar.graphics.drawRect(0,0,150,stage.stageHeight);
toolbar.graphics.endFill();

addChild(toolbar);
```

Isječak koda 3. Definiranje prostora alatne trake

U aplikaciji su unutar iste funkcije dodani gumbi (*PushButton*), alati za odabir boje (*ColorChooser)* i klizne trake za promjenu vrijednosi varijabli (*Slider*) koji su preuzeti iz Minimalcomps biblioteke koda i prilagođeni potrebama aplikacije.

Kao primjer korištenja spomenute biblioteke može se izdvojiti „ColorChooser“, grafički oblikovana funkcija pomoću koje se na grafičkom sučelju korisniku omogućuje odabir boje. [17] Kako bi se ova funkcija mogla koristiti u aplikaciji, uvozi se pomoću sljedeće naredbe:

```
//ColorChooser se dodaje (import) iz minimalcomps
import com.bit101.components.ColorChooser;
```

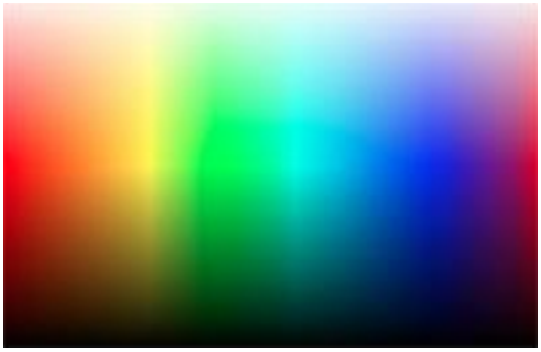
Isječak koda 4. Uvoz komponente iz biblioteke

Uvezena komponenta definira se unutar klase kao nova varijabla:

```
//unutar klase se importanoj varijabli ColorChooser
da ime colorPicker
private var colorPicker:ColorChooser;
```

Isječak koda 5. Deklariranje varijable

Odabir boje odvija se po principu očitavanja vrijednosti za boju sa slike koja se prikazuje nakon odabira gumba za promjenu boje (Slika 8).



Slika 8. Slika putem koje se odabire boja, colorPalette.png

Ta je slika dodana u aplikaciju kao PNG datoteka, uvezena (*embed*) u kod aplikacije te definirana kao nova klasa, kao što je prikazano u sljedećem primjeru:

```
//dodana paleta boja kao .png + postala je klasa
s nazivom ColorPickerImage
  [Embed (source="colorPalette.png")]
  private var ColorPickerImage:Class;
```

Isječak koda 6. Uvoz palete boja i definiranje nove klase

Nakon deklariranja varijable i dodavanja slike putem koje se odabire željena boja slijedi inicijalizacija te dodatno definiranje komponente i načina na koji će se ponašati u aplikacija:

```
//unutar funkcije drawToolbar se doda:
            colorPicker = new ColorChooser(this, 0,
0,0xff0000, onColorChange);
            colorPicker.x = 20;
            colorPicker.y = 110;
            colorPicker.usePopup = true;
            colorPicker.model = new ColorPickerImage();
            colorPicker.popupAlign = ColorChooser.BOTTOM;
            toolbar.addChild(colorPicker);
```

Isječak koda 7. Inicijalizacija varijable i opis komponente

Na sličan način definirana je i klizna traka za promjenu debljine linije [18], kao što je vidljivo u sljedećem primjeru:

```
   strokeSizeSlider = new Slider();
   strokeSizeSlider.maximum = 15;
   strokeSizeSlider.minimum = 0;
   strokeSizeSlider.width = 100;
   strokeSizeSlider.tick = 1;
   strokeSizeSlider.x = 25;
   strokeSizeSlider.y = 175;
   strokeSizeSlider.addEventListener(Event.CHANGE,
changeLineWidth);
   strokeSizeSlider.value = 1;
   toolbar.addChild(strokeSizeSlider);
```

Isječak koda 8. Definiranje vrijednosti klizne trake za promjenu debljine linije

Alatna traka nudi odabir jednog od tri oblika koje je moguće iscrtavati pomoću aplikacije, a odabir se vrši putem pripadajućeg gumba čiji je izgled u aplikaciji definiran na sljedeći način:

```
rectangleButton = new PushButton(toolbar,25,208,"REC");
   rectangleButton.setSize(30,30);
   rectangleButton.toggle = true;
   rectangleButton.selected = true;
   rectangleButton.draw();


   triangleButton = new PushButton(toolbar,60,208,"TRI");
   triangleButton.setSize(30,30);
   triangleButton.toggle = true;
   triangleButton.draw();


   circleButton = new PushButton(toolbar,95,208,"CIR");
   circleButton.setSize(30,30);
   circleButton.toggle = true;
   circleButton.draw();
```

Isječak koda 9. Definiranje gumba za odabir elemenata

Kako ne bi došlo do pogreške, aplikacija ne smije dopustiti istodobni odabir više od jednoga gumba, zbog čega je u kodu potrebno dodati „osluškivače“ povezane s funkcijom „objectButtonClicked“ koja provjerava je li jedan od tri spomenuta gumba odabran i u tom slučaju ostale označava kao neodabrane. Ista se funkcija brine i za to da je pravokutnik odabran odmah pri otvaranju aplikacije i da korisnik u slučaju da niti jedan gumb nije odabran ne može iscrtavati elemente na korisničkom sučelju. U aplikaciji je to definirano na sljedeći način:

```
rectangleButton.addEventListener(MouseEvent.CLICK, objectButtonClicked);
triangleButton.addEventListener(MouseEvent.CLICK, objectButtonClicked);
circleButton.addEventListener(MouseEvent.CLICK, objectButtonClicked);

function objectButtonClicked(evt:MouseEvent):void {
    if (evt.currentTarget == rectangleButton) {
     triangleButton.selected = false;
     circleButton.selected = false;
    }
  else if (evt.currentTarget == triangleButton) {
     rectangleButton.selected = false;
     circleButton.selected = false;
    }
  else if (evt.currentTarget == circleButton)
    {
     triangleButton.selected = false;
     rectangleButton.selected = false;
    }
  else if (evt.currentTarget != rectangleButton || evt.currentTarget !=
    triangleButton || evt.currentTarget != circleButton)
    {
    }
  else {rectangleButton.selected = true;}
```

Isječak koda 10. Dio koda koji onemogućuje istovremeni odabir više elemenata

Lakše snalaženje korisniku u dodavanju elemenata koristeći aplikaciju omogućeno je dodavanjem mreže napravljene koristeći niz vodoravnih i okomitih linija međusobno odmaknutih u jednakim razmacima. U aplikaciji je takva mreža definirana na sljedeći način:

```
//podloga
mash.graphics.beginFill(backgroundColor);
            mash.graphics.drawRect(startX,star  ',endX,endY);
            mash.graphics.endFill();
//okomite linije
for(var i:int=startX; i<endX; i+=gap)
        {
           mash.graphics.lineStyle(1,0x808080,0.2);
           mash.graphics.moveTo(i,0);
           mash.graphics.lineTo(i,endY);
        }
//vodoravne linije
for(var j:int=startY; j<endY; j+=gap)
        {
           mash.graphics.lineStyle(1,0x808080,0.2);
           mash.graphics.moveTo(0,j);
           mash.graphics.lineTo(endX,j);
       }
```

Isječak koda 11. Prikaz mreže

Razmaci između linija pritom su definirani kao varijable kako bi se korisniku omogućila manipulacija veličinom mreže pomoću *slidera*. [18] Taj je *slider* kao i ostali vizualni elementi uvezen iz Minimalcomps biblioteke koda sa svojim prethodno definiranim vrijednostima, a u ovoj mu je aplikaciji pridodana funkcija kojom se vrijednost varijabli definiranih u prikazu mreže mijenjaju. U kodu aplikacije definiran je na sljedeći način:

```
//definiranje slidera za promjenu veličine mreže
   mashSizeSlider = new Slider();
   mashSizeSlider.maximum = 100;
   mashSizeSlider.minimum = 5;
   mashSizeSlider.width = 100;
   mashSizeSlider.tick = 5;
   mashSizeSlider.x = 25;
   mashSizeSlider.y = 80;
   mashSizeSlider.addEventListener(Event.CHANGE,
changeMashSize); //odnosi se na funkciju changeMeshSize
(kasnije dodanu)
   mashSizeSlider.value = 15;
   toolbar.addChild(mashSizeSlider);
```

Isječak koda 12. *Slider* za manipulaciju veličinom mreže

Funkcija koja mijenja vrijednost varijabli za razmake između linija u kodu aplikacije definirana je na sljedeći način:

```
//odnosi se na ranije definirani slider za promjenu
veličine mreže
  private function changeMashSize(e:Event):void
  {
   var slider:Slider = e.target as Slider;
   drawMash(150,0,1025,640,slider.value,16448250);
  }
```

Isječak koda 13. Funkcija koja mijenja veličinu mreže

Svakim pomakom *slidera* na sučelju se iscrtava nova mreža. Kako bi se spriječilo gomilanje nepotrebnih linija na ekranu aplikacije, potrebno je izbrisati staru mrežu prije nego što se promjenom vrijednosti varijabli doda i nova mreža, a to je postignuto dodavanjem sljedećeg uvjeta:

```
if(mash != null && contains(mash))
            {
                mash.graphics.clear();
            }
            else
            {
                mash = new Sprite();
            }
```

Isječak koda 14. Uvjet koji sprečava gomilanje linija mreže

Iako mreža olakšava precizno smještanje elemenata zbog čega je vidljiva pri otvaranju aplikacije, korisniku je osim promjene gustoće omogućeno i uklanjanje mreže odabirom gumba na alatnoj traci. Gumb je povezan s funkcijom „showHideGrid“ i definiran na sljedeći način:

```
showGridButton = new PushButton(toolbar,25,50,"GRID",showHideGrid);
   showGridButton.toggle = true;
```

Isječak koda 15. Gumb povezan s funkcijom za prikaz i uklanjanje mreže

Funkcija koja će odabirom gumba prikazuje i uklanja mrežu u kodu izgleda ovako:

```
private function showHideGrid(e:MouseEvent):void

  {

   if(mash != null && contains(mash))
   {
    removeChild(mash);
    mashSizeSlider.enabled = false;
   }
   else
   {
    addChildAt(mash,0);
    mashSizeSlider.enabled = true;
   }
  }
```

Isječak koda 16. Funkcija odgovorna za prikaz i uklanjanje mreže

Treći dio grafičkog korisničkog sučelja, gdje se prikazuje PostScript programski kod, još je jedna od Minimalcomps komponenti. Radi se o prozoru (*Window*) do kojega se dolazi klikom na „PSC“ gumb na alatnoj traci, a sam prozor definiran je unutar „showPostScriptCode“ funkcije unutar koje je definiran i font teksta koda koji se prikazuje unutar prozora.

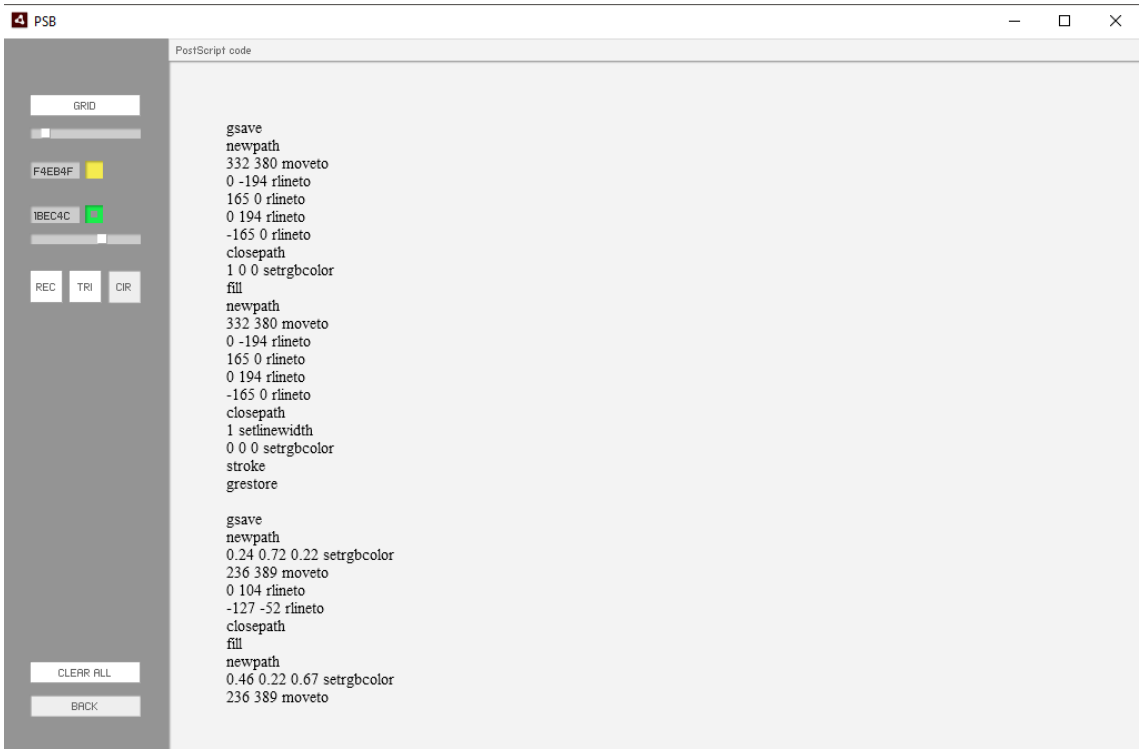
```
pscWindow = new Window(this, toolbar.width, 0, "PostScript code");
   pscWindow.width = stage.stageWidth - toolbar.width;
   pscWindow.height = stage.stageHeight;
   pscWindow.hasMinimizeButton = false;
   pscWindow.draggable = false;
   var textLabelFormat:TextFormat = new TextFormat("Roboto-Light" ,
14 , 0,null,null,null,null,null,"left");

   var textLabel:TextField = new TextField();
   textLabel.x = 50;
   textLabel.y = 50;
   textLabel.width = pscWindow.width - 100;
   textLabel.height = pscWindow.height - 100;
   textLabel.defaultTextFormat = textLabelFormat;
   textLabel.text = code;
   pscWindow.content.addChild(textLabel);
```

Isječak koda 17. Definiranje izgleda prozora za prikaz PostScript programskog koda

3.6. Funkcionalnosti

Kao što je već ranije navedeno, primarna zadaća aplikacije PostScript Builder je pomoći novim korisnicima PostScripta i olakšati njegovo učenje. U prvoj verziji aplikacije ona ne sadrži sve napredne mogućnosti koje nudi PostScript, ali omogućuje da se kroz intuitivno korisničko sučelje minimalističkog dizajna korisnik lako snađe u dodavanju osnovnih vektorskih oblika na digitalno platno te iz željenog dizajna lako dođe do njegove interpretacije u programski kod. PostScript Builder prilikom kreiranja svakog objekta korisniku omogućuje odabir željene boje i debljine linije te pomak svakoga elementa odnosno promjenu njegove pozicije. Objekti koje korisnik može dodati mogu biti u obliku pravokutnika, kruga ili jednakokračnog trokuta, a osim vrste objekta, korisnik sam može povlačenjem miša odabrati i njegovu veličinu. Dodane je oblike moguće izbrisati pojedinačno, a moguće je klikom na gumb izbrisati i sve dodane

elemente odjednom. Kako bi se korisniku omogućilo precizno dodavanje elemenata, aplikacija sadrži i mrežu kojoj je moguće promijeniti veličinu, a moguće ju je i ukloniti. Najbitnija značajka aplikacije koja ju čini različitom od ostalih trenutno popularnih aplikacija upravo je mogućnost prevođenja željenih oblika kreiranih i prikazanih na ekranu u PostScript programski kod (Slika 9), što je također moguće ostvariti odabirom gumba koji se nalazi na alatnoj traci. Korisnici na taj način mogu provjeriti kako bi željeni dizajn mogli dobiti koristeći samo taj programski jezik, a njegovi bolji poznavatelji dobiveni kod mogu iskoristiti kako bi ga uredili ili nadopunili. Ponovnim odabirom istoga gumba korisnik se vraća na dio predviđen za dodavanje elemenata gdje može nastaviti svoj rad na željenom dizajnu.



Slika 9. Prikaz PostScript programskog koda oblika generiranih putem aplikacije

Sve navedene funkcionalnosti i način na koji su ostvarene u programskom kodu PostScript Builder aplikacije bit će pobliže objašnjene u nastavku teksta.

3.6.1. Izrada osnovnih oblika

PostScript Builder korisniku omogućuje dodavanje tri osnovna elementa i njihovu manipulaciju. Svaki od tih elemenata ima nekoliko varijabli kojima je moguće

upravljati kroz grafičko korisničko sučelje, a to su boja ispune, boja linije, debljina linije i pozicija elementa na stranici. Osim toga, svaki je od njih definiran i odgovarajućim varijablama koje su potrebne da bi se definirala njihova veličina. Obzirom da se radi o različitim elementima, različit je i način na koji se oni opisuju kroz ActionScript i PostScript, a koriste i različit broj varijabli kojima su određeni.

3.6.1.1. Pravokutnik

PostScript pravokutnik jedan je od tri elementa koje je moguće dodati u aplikaciji PostScript Builder. Varijable kojima je pravokutnik određen odnose se na poziciju njegove početne točke, njegovu širinu i visinu. U glavnom dijelu aplikacije definirano je da se pravokutnici iscrtavaju ako je gumb za iscrtavanje pravokutnika označen kao aktivan. U zagradama su redom nabrojane varijable koje funkcija treba primiti, a njihove vrijednosti određene su odabirom boje elementa i linije, kao i debljine linije pomoću grafičkog korisničkog sučelja te označavanjem dijela na kojemu će se željeni pravokutnik prikazati. Varijable „startPoint.x“ i „startPoint.y“ preuzimaju vrijednosti za udaljenost miša od početne točke aplikacije u trenutku kad korisnik klikom miša označi početak iscrtavanja pravokutnika, a „mouseX“ i „mouseY“ bilježe vrijednost za udaljenost miša u trenutku kad korisnik otpusti klik. Zato su vrijednosti koje će funkcija primiti za visinu i širinu pravokutnika prikazane kao „mouseX - startPoint.x“ i „mouseY - startPoint.y“.

```
if (rectangleButton.selected == true)
   {
    newElement = new PostScriptRectangle(startPoint.x - 150,
startPoint.y,fillColorPicker.value,mouseX - startPoint.x,
mouseY - startPoint.y, getRect(workingContainerBackground),
strokeSizeSlider.value, strokeColorPicker.value);
    }
```

Isječak koda 18. Dodavanje novog pravokutnika uz pripadajuće varijable

Koje će varijable funkcija PostScriptRectangle primiti potrebno je definirati u klasi PostScriptRectangle koja sadrži sve informacije potrebne za definiranje načina na koji će se pravokutnici ponašati u aplikaciji:

```
public function PostScriptRectangle(_x:int, _y:int,
_color:int, _width:Number, _height:Number, _rect:Rectangle,
_lineWidth:Number, _lineColor:int)
```

Isječak koda 19. Funkcija koja prima varijable za pravokutnik

ActionScript u sebi sadrži gotovu funkciju kojom je moguće iscrtavati pravokutnike. To je u klasi „PostScriptRectangle“ unutar funkcije „drawShape“ postignuto dodavanjem sljedeće naredbe u kojoj su već sadržane prethodno definirane varijable za visinu i širinu.

```
rectangle.graphics.drawRect(0, 0, width,height);
```

Isječak koda 20. Gotova ActionScript funkcija koja iscrtava pravokutnike

U PostScript programskom kodu iscrtavanje pravokutnika postiže se na drugačiji način. U dijelu koji je odgovoran za ispisivanje PostScript koda to izgleda ovako:

```
resultCode += "newpath" + " \n";
   resultCode += String(rectangle.x) + " " +
String(Helper.calculateYCoordinate(rectangle.y))
+ " " + "moveto" + " \n";
   resultCode += "0" + " " + String(0 - height)
+ " " + "rlineto" + " \n";
   resultCode += String(width) + " " + "0" + " "
+ "rlineto" + " \n";
   resultCode += "0" + " " + String(height) + "
" + "rlineto" + " \n";
   resultCode += String(0 - width) + " " + "0" +
" " + "rlineto" + " \n";
   resultCode += "closepath" + " \n";
   resultCode += String(int(rgbObj.R*100)/100) +
" " + String(int(rgbObj.G*100)/100) + " " +
String(int(rgbObj.B*100)/100) + " " +
"setrgbcolor" + " \n";
   resultCode += "fill" + " \n";
```

Isječak koda 21. Dio ActionScript koda zadužen za prikaz PostScript koda

Ovaj dio određuje koji će se tekst prikazivati na stranici klikom na PSC gumb. Vrijednosti varijabli preuzete su od iscrtanog pravokutnika i prikazane kao skup znakova, odnosno broj čijom su vrijednošću određene u tom trenutku, a popraćene su nizovima znakova koji u PostScript kodu određuju što će se događati i na koji način. Naredba *moveto* određuje početnu točku, a *rlineto* se odnosi na udaljenost od prethodne točke na kojoj će se iscrtati linije elementa.

3.6.1.2. Krug

U klasi PostScriptCircle glavna funkcija, osim vrijednosti za boju linije, veličinu linije i boju ispune, prima vrijednosti x i y koordinata središta kružnice i polumjera kružnice. Kao i u slučaju pravokutnika, te će vrijednosti primiti preko položaja miša u trenutcima kad korisnik spusti i otpusti klik miša. Kao i za pravokutnik, ActionScript sadrži metodu kojom je moguće iscrtati krug, a prikazuje se ovako:

```
circle.graphics.drawCircle(0, 0, radius);
```

Isječak koda 22. Prikaz ActionScript koda koji opisuje krug

Način prikaza istog elementa se u PostScript kodu također razlikuje te će ga korisnici PostScript programskog jezika prikazati na sljedeći način:

```
x y radius 0 360 arc
```

Isječak koda 23. Prikaz PostScript koda kojim se opisuje krug

U tom dijelu koda x i y označavaju koordinate središta kružnice, a „radius“ označava njegov polumjer. Brojevi 0 i 360 odnose se na kut početka i završetka ispisivanja kružnice, a u ovom slučaju označavaju da se radi o punom krugu. Arc je oznaka za kružnicu, a nakon nje bi za prikaz iste na ekranu uslijedila naredba *fill* za ispunu ili *stroke* za obrub definirane kružnice.

U ovom radu taj je dio koda, kako bi se ispisao na ekranu aplikacije napravljene koristeći ActionScript, definiran ovako:

```
resultCode += String(int(rgbObj.R*100)/100)
+ " " + String(int(rgbObj.G*100)/100) + " "
+ String(int(rgbObj.B*100)/100) + " " +
"setrgbcolor" + " \n";
   resultCode += String(circle.x) + " " +
String(Helper.calculateYCoordinate(cir-
cle.y)) + " " + String(radius) + " " + "0"
+ " " + "360" + " " + "arc" + " \n";
   resultCode += "fill" + " \n";
```

Isječak koda 24. Dio ActionScript koda zadužen za prikaz PostScript koda

3.6.1.3. Trokut

Za razliku od ranije opisanih elemenata i njihovih klasa, definiranje klase „PostScriptTriangle“ nešto je kompliciranije jer ActionScript ne sadrži gotovu metodu kojom je moguće prikazati trokut. U aplikaciji PostScriptBuilder omogućena je izrada jednakokračnih trokuta. Glavna funkcija u klasi prima vrijednosti koordinate početne točke i koordinate kojima su određene širina i visina trokuta, a putem njih se određuju koordinate linija koje opisuju trokut. Osim što je u kodu potrebno definirati koordinate svake od linija koje će iscrtavati trokut, potrebno je dodati i uvjete koji će provjeravati odnos početne i krajnje točke trokuta budući da o njima ovisi orijentacija vrha trokuta. Oblik trokuta definiran je redoslijedom povlačenja linija čije su vrijednosti određene spomenutim varijablama te je u „drawShape“ funkciji opisan na ovaj način:

```
if(endY < 0 && endX > 0){
 triangle.graphics.moveTo(0,0);
 triangle.graphics.lineTo(endX,0);
 triangle.graphics.lineTo(endX/2,endY);
 triangle.graphics.endFill();
 triangle.x = pX;
 triangle.y = pY;
}
else if(endY < 0 && endX < 0){
 triangle.graphics.moveTo(0,0);
 triangle.graphics.lineTo(0,endY);
 triangle.graphics.lineTo(endX,endY/2);
 triangle.graphics.endFill();
 triangle.x = pX;
 triangle.y = pY;
}
else if(endY > 0 && endX < 0){
 triangle.graphics.moveTo(0,0);
 triangle.graphics.lineTo(endX,0);
 triangle.graphics.lineTo(endX/2,endY);
 triangle.graphics.endFill();
 triangle.x = pX;
 triangle.y = pY;
}
else
{
 triangle.graphics.moveTo(0,0);
 triangle.graphics.lineTo(0,endY);
 triangle.graphics.lineTo(endX,endY/2);
 triangle.graphics.endFill();
 triangle.x = pX;
 triangle.y = pY;
}
```

Isječak koda 25. Iscrtavanje trokuta kroz ActionScript

Zbog nedostatka gotove metode za iscrtavanje trokuta u ActionScriptu, kod njegovog iscrtavanja u PostScriptu postoji veća sličnost jer se također definira povlačenjem linija. U ovom je slučaju također potrebno dodati uvjete koji provjeravaju odnos početne i krajnje točke trokuta.

```
if (endY < 0 && endX > 0)
   {
   resultCode += String(pX) + " " + String(Helper.calculateYCoordinate(pY))
+ " " + "moveto" + " \n";
   resultCode += String(endX) + " " +  String(0) + " " + "rlineto" + " \n";
   resultCode += String(0-endX/2) + " " + String(0-endY) + " " + "rlineto"
+ " \n";
   }
   else if (endY > 0 && endX > 0)
   {
   resultCode += String(pX) + " " + String(Helper.calculateYCoordinate(pY))
+ " " + "moveto" + " \n";
   resultCode += String(0) + " " +  String(0-endY) + " " + "rlineto" + "
\n";
   resultCode += String(endX) + " " + String(endY/2) + " " + "rlineto" + "
\n";
   }
   else if (endY > 0 && endX < 0)
   {
   resultCode += String(pX) + " " + String(Helper.calculateYCoordinate(pY))
+ " " + "moveto" + " \n";
   resultCode += String(endX) + " " +  String(0) + " " + "rlineto" + " \n";
   resultCode += String(0-endX/2) + " " + String(0-endY) + " " + "rlineto"
+ " \n";
   }
   else
   {
   resultCode += String(pX) + " " + String(Helper.calculateYCoordinate(pY))
+ " " + "moveto" + " \n";
   resultCode += String(0) + " " +  String(0-endY) + " " + "rlineto" + "
\n";
   resultCode += String(endX) + " " + String(endY/2) + " " + "rlineto" + "
\n";
   }

   resultCode += "closepath" + " \n";
   resultCode += "fill" + " \n";
```

Isječak koda 26. Dio ActionScript koda zadužen za prikaz trokuta kroz PostScript

3.6.2. Označavanje elemenata

Nakon što korisnik iscrta željeni oblik, PostScript Builder omogućuje mu naknadnu promjenu debljine linije, boje ispune i boje linije označenih elemenata, ali i promjena pozicije elemenata. Kako bi se mogle izvršiti promjene, elemente je prvo potrebno označiti klikom miša. U programskom kodu aplikacije potrebno je definirati funkciju u kojoj će se provjeravati je li prilikom klika obuhvaćena površina jednog od iscrtanih elemenata. Korisnik će označeni element prepoznati zahvaljujući jednostavnoj neprekidnoj animaciji promjene opaciteta. Ta je animacija dodana iz GreenSock biblioteke koda [19] te ju je potrebno uvesti prije pozivanja. U ovom je radu animacija dodana unutar funkcije koja se pokreće u

slučaju da je jedan od elemenata označen. Prethodno dodani uvjet potreban je kako bi ugasio animaciju ako je ona prethodno već pokrenuta i opacitet elementa vratio na 1, kao što je prikazano.

```
if(animation)
   {
    animation.kill();
    this.alpha = 1;
    animation = null;
   }

animation = new TweenMax(this, 0.5, {al-
pha:0.5, repeat:-1, yoyo:true});
```

Isječak koda 27. Animacija opaciteta za označene elemente

3.6.3. Brisanje elemenata sa stagea

Kako bi korisniku bila omogućena što veća fleksibilnost prilikom izrade željenog dizajna koristeći aplikaciju, ali i pravo na pogrešku ili promjenu ideje u procesu, osim pomicanja elemenata omogućeno je i njihovo uklanjanje. Označeni elementi uz sebe prikazuju i kvadratić s oznakom x klikom na koji će se element ukloniti. Ti su kvadratići definirani kao gumbi koji se prikazuju u označenom stanju. Svaki od elemenata u svojoj klasi sadrži funkciju „removeFromStage“ koja to omogućuje.

```
private function removeFromStage(e:Event):void
  {
   removeListeners();
   while (this.numChildren > 0) {
    this.removeChildAt(0);
   }
   dispatchEvent(new Event("deleteElement", true));
  }
```

Isječak koda 28. Funkcija koja uklanja označeni element s ekrana

Korisnik može odjednom ukloniti sve elemente klikom na za to predviđeni gumb na alatnoj traci. Klikom na taj gumb pokreće se funkcija „clearEverything“ koja u kodu izgleda ovako:

```
private function clearEverything(e:Event):void
  {
    workingContainer.removeChildren(1);
    elementsArray.length = 0;
  }
```

Isječak koda 29. Funkcija koja uklanja sve elemente s ekrana

3.6.4. Polje elemenata

Kako bi se lakše pratio redoslijed dodavanja elemenata korišteno je polje. Smještanje elemenata u polje omogućuje manipulaciju indeksa elemenata odnosno njihove pozicije u odnosu na druge elemente. Dodavanjem novih oblika svaki od njih dobiva svoj novi položaj u polju koji se nastavlja na onaj prethodni. Ali ako se na ekranu iscrtaju dva ili više elemenata koji se djelomično ili potpuno preklapaju, element koji se označi klikom miša postat će zadnji u polju te će se prikazati preko svih ostalih elemenata na ekranu. Kao i u vizualnom prikazu, takvu je promjenu mjesta u polju potrebno izvršiti i u PostScript programskom kodu tako da se dio koda odgovoran za prikaz označenog oblika ispiše poslijednji. Navedene su promjene u kodu aplikacije smještene u funkciji selectElement koja provjerava je li jedan od elemenata označen te ga u tom slučaju premješta na zadnje mjesto u polju. Isto je potrebno napraviti i za dio PostScript koda koji se odnosi na označeni oblik. Ta funkcija definirana je na sljedeći način:

```
  private function selectElement(e:MouseEvent):void
  {
   var mousePoint:Point = this.localToGlobal(new Point(mouseX,mouseY));
   var len:int = elementsArray.length;

   var lastIndex:int = -1;

   for(var i:int = 0; i < len; i++)
   {
    //provjerava je li jedan od elemenata u polju označen mišem
    if( elementsArray[i].hitTestPoint(mousePoint.x,mousePoint.y,true))
    {

     if(lastIndex < i){
      lastIndex = i;
     }

    }
    else
    {
     elementsArray[i].Selected = false;
    }
   }

   if(lastIndex != -1)
   {

    elementsArray[lastIndex].Selected = true;
    workingContainer.addChild(elementsArray[lastIndex]);
    setSelectedValues(elementsArray[lastIndex]);

    //ovo će vratiti izvući element iz polja i vratiti ga na zadnje
mjesto u PS kodu
    var newElement:DisplayObject = elementsArray[lastIndex];
    elementsArray.removeAt(lastIndex);
    elementsArray.push(newElement);
   }
  }
```

Isječak koda 30. Funkcija koja mijenja redoslijed označenih elemenata u polju

Prije nego što korisnik doda elemente u aplikaciji ili nakon što sve ukloni, polje je prazno. U funkciji showPostScriptCode nalazi se petlja koja provjerava je li polje prazno te u tom slučaju umjesto PostScript koda ispisuje poruku „Add an object to see PostScript code." kao što se može vidjeti u sljedećem isječku koda:

```
if(elementsArray == null || elementsArray.length == 0)
   {
    code = "Add an object to see PostScript code.";
   }
   else
   {
    var len:int = elementsArray.length;


    for(var i:int = 0; i < len; i++)
    {
     code += elementsArray[i].toPostScript();
    }

   }
```

Isječak koda 31. Petlja koja provjerava je li polje prazno

3.6.5. Generiranje postscript koda iz vizulanih elemenata

ActionScript programski jezik, iako je kao i PostScript produkt Adobe Systems-a, s PostScriptom nema puno dodirnih točaka niti način automatske preobrazbe u PostScript. Elementi su u aplikaciji opisani kao klase, a za svaki je element potrebno definirati i pripadajuće PostScript naredbe kako bi se oblici dodani i iscrtani na korisničkom sučelju mogli prevesti u PostScript programski kod. Pritom ih je potrebno povezati s varijablama koje određuju njihov položaj, boju i debljinu linije. Izazov izrade ove aplikacije bio je u brojnim razlikama u načinu prikaza vektorskih oblika, odnosno njihovom dobivanju u različitim programskim jezicima. PostScript je namijenjen programiranju vektorske grafike, dok je u ActionScriptu dodavanje vektorskih oblika tek jedna od sporednih mogućnosti.

Osim u samom načinu pisanja koda za pojedine elemente, razlika je i u koordinatnom sustavu, ali i sustavu boja koji je bilo potrebno preračunati kako bi se mogao dobiti točan kod koji je moguće koristiti u PostScriptu. Tako je bitno naglasiti kako je prilikom programiranja grafike kroz PostScript ishodište smješteno u donjem lijevom kutu, dok je početna točka koordinatnog sustava ActionScripta u gornjem lijevom kutu. Kako bi svi elementi bili poredani prema

željama korisnika aplikacije bez neželjenih pomaka, bilo je potrebno u kodu odrediti pomak koji čini razliku u ishodištu, odnosno preračunati y koordinatu.

Za potrebe preračunavanja vrijednosti y koordinate u PostScript kodu dodana je nova klasa, nazvana „Helper“. Kako bi se vrijednost y koordinate prikazanih elemenata prikazala u PostScript programskom kodu, gdje je ishodište smješteno u donjem lijevom kutu, potrebno je od visine prozora aplikacije oduzeti vrijednost y koordinate koja je zabilježena u ActionScriptu, a ishodište ima u gornjem lijevom kutu tog prozora. Taj je preračun u novoj klasi s nazivom „Helper“ definiran na sljedeći način:

```
public static function calculateYCoordinate(y:int):int
  {
   return stageHeight - y;
  }
```

Isječak koda 32. Funkcija koja preračunava vrijednost y osi za PostScript

U kodu je bilo potrebno odrediti i način na koji će se preračunati boje budući da se kroz Actionscript boje određuju u heksadecimalnom obliku, dok ih je u PostScript programskom kodu potrebno prikazati kao vrijednosti za pojedinu boju u RGB, HSB ili CMYK sustavu boja te su u ovom radu prikazane kroz RGB sustav boja. To je postignuto pomoću sljedeće funkcije koja se, kao i preračun vrijednosti y koodinate, odvija unutar klase „Helper“:

```
public static function convertRGBforPS(color:int):Object
  {
   var result:Object = new Object();

   result.R = int(color/(Math.pow(256,2)))/255;
   result.G = int((color/256)%256)/255;
   result.B = int(color%256)/255;

   return result;
  }
```

Isječak koda 33. Funkcija koja preračunava vrijednosti boja u PostScript kod

Dobiveni je rezultat konačno za svaku od vrijednosti iz RGB sustava boja izašao kao decimalni broj s vrlo velikim brojem decimala. Zbog preglednosti je broj decimala u prikazu zaokružen na dvije decimale, a način na koji se to postiglo objašnjen je u sljedećem primjeru:

```
resultCode += String(int(rgbObj.R*100)/100)
+ " " + String(int(rgbObj.G*100)/100) + " "
+ String(int(rgbObj.B*100)/100) + " " +
"setrgbcolor" + " \n";
```

Isječak koda 34. Pozivanje funkcije za preračunavanje vrijednosti boja

U tom prikazu riječ „String“ označava da se sve navedeno nakon nje prikazuje kao skup znakova. Varijabla „rgbObj.R“ u sebi sadrži brojčanu vrijednost koja će se prikazati u znakovnom obliku. Prije nego što se u tom obliku ispiše na ekranu, od spomenute vrijednosti pomnožene sa 100 je uzet samo cijeli broj, a potom se taj broj podijelio sa 100 kako bi se dobio rezultat s 2 decimale.

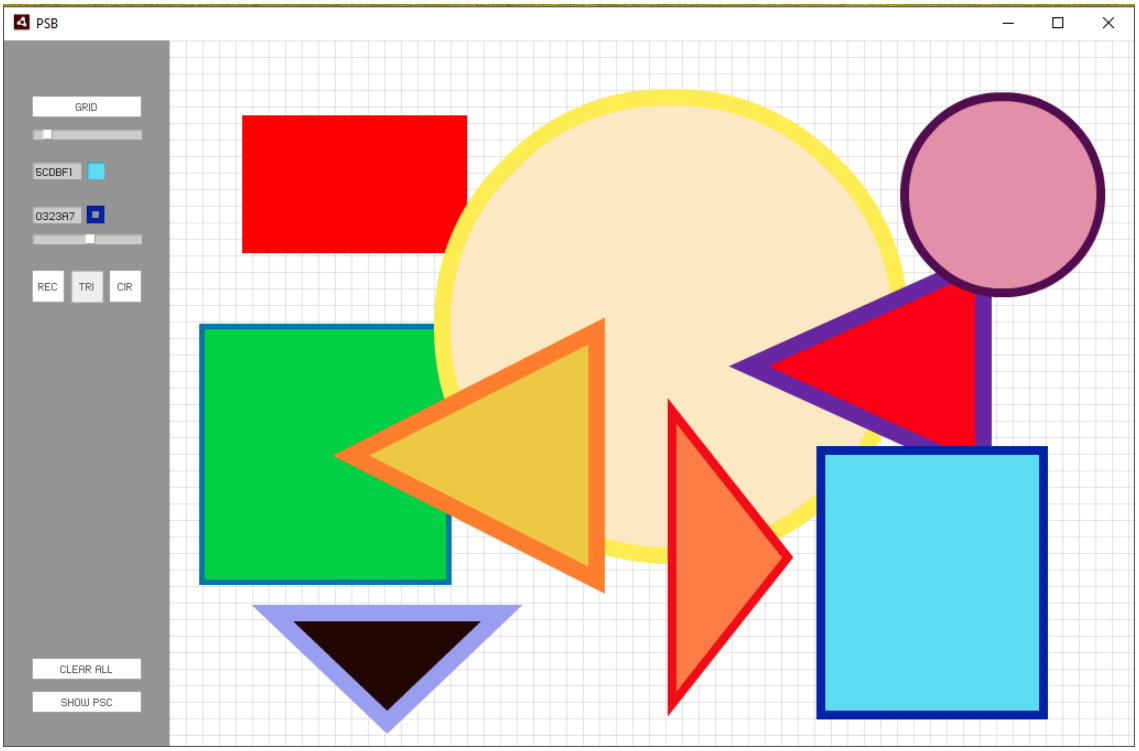
Svaki od elemenata u svojoj klasi sadrži javnu funkciju „toPostScript“ koja se za svaki element sadržajno razlikuje ovisno o načinu pozivanja PostScript koda za pojedini element, a njezin se sadržaj poziva u glavnom dijelu aplikacije unutar funkcije „showPostScriptCode“. Ta će funkcija iz polja dohvaćati elemente i upisivati kod određen toPostScript funkcijom. Ako korisnik na sučelju još nije dodao niti jedan element, umjesto PostScript programskog koda funkcija će ispisati "Add an object to see PostScript code." kako bi korisnik znao što mu je činiti ako želi dobiti programski kod i na taj se način olakšalo shvaćanje aplikacije.

```
  private function showPostScriptCode(e:Event):void
  {
     if(pscWindow != null  && contains(pscWindow))
   {
    closePSCWindow(null);
    return;
   }

   var code:String = "";

   if(elementsArray == null || elementsArray.length == 0)
   {
    code = "Add an object to see PostScript code.";
   }

   else
   {
    var len:int = elementsArray.length;

    for(var i:int = 0; i < len; i++)
    {
     code += elementsArray[i].toPostScript();
    }

   }

   pscWindow = new Window(this, toolbar.width, 0, "PostScript code");
   pscWindow.width = stage.stageWidth - toolbar.width;
   pscWindow.height = stage.stageHeight;
   pscWindow.hasMinimizeButton = false;
   pscWindow.draggable = false;

   var textLabelFormat:TextFormat = new TextFormat("Roboto-Light", 14,
0,null,null,null,null,null,"left");

   var textLabel:TextField = new TextField();
   textLabel.x = 50;
   textLabel.y = 50;
   textLabel.width = pscWindow.width - 100;
   textLabel.height = pscWindow.height - 100;
   textLabel.defaultTextFormat = textLabelFormat;
   textLabel.text = code;
   pscWindow.content.addChild(textLabel);
  }
```

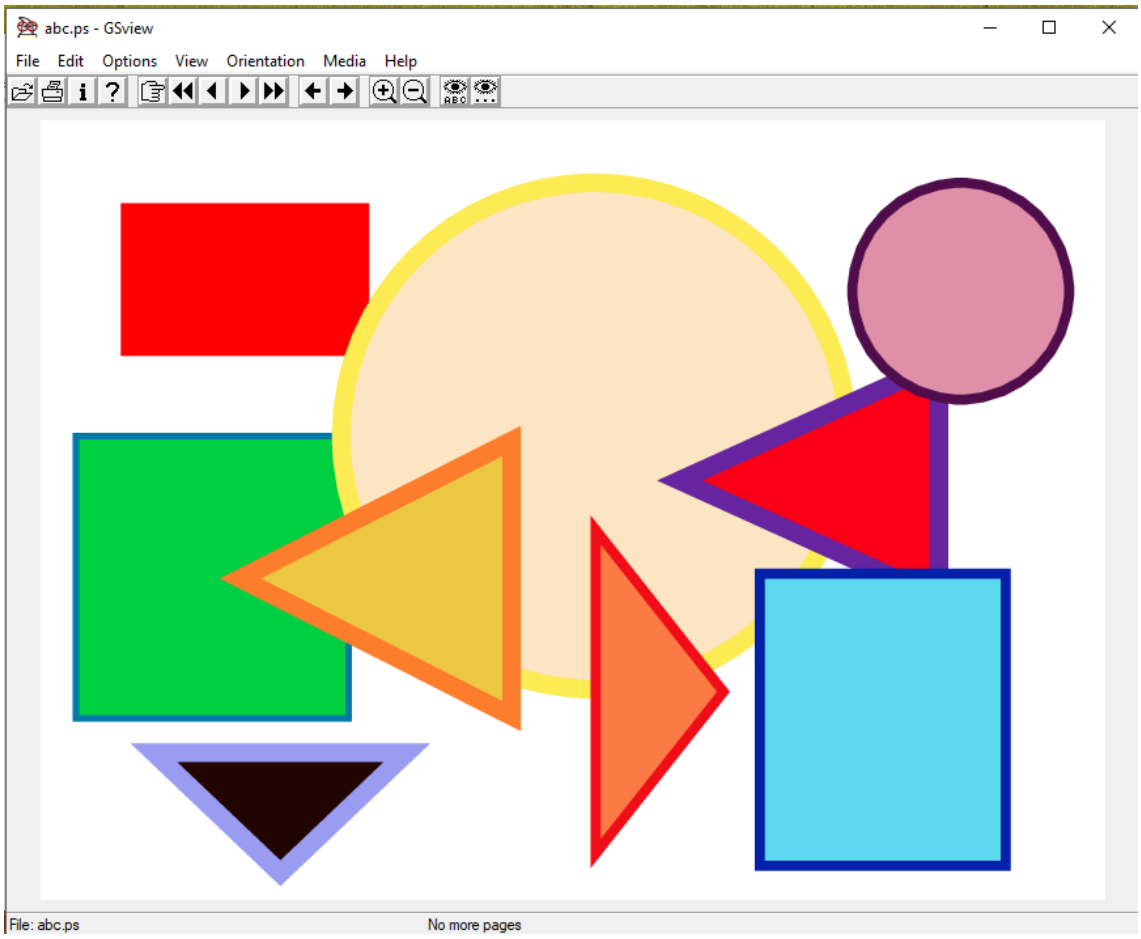
Isječak koda 35. Dio zadužen za otvaranje prozora s prikazom PostScript koda

4. REZULTAT I RASPRAVA

Kao što je i ranije navedeno, cilj izrade aplikacije PostScript Builder je omogućiti korisniku da na jednostavan način dođe do PostScript programskog koda željenih oblika bez prethodnog poznavanja tog programskog jezika. Kako bismo konačno mogli zaključiti koliko je aplikacija uspješno izvedena, potrebno je usporediti grafiku kreiranu kroz grafičko korisničko sučelje aplikacije s grafikom dobivenom iz generiranog programskog koda. Usporedba je vidljiva na odnosima slika 10 i 11. Slika 10 prikazuje ekran aplikacije PostScript Builder u kojoj je kreirana grafika od nasumično dodanih elemenata različitih pozicija, boja i debljina linija. Na slici 11 vidljiv je prikaz iste te grafike dobivene iz generiranog PostScript koda koji je učitan putem GS View preglednika. Zbog bolje usporedbe, u pregledniku su za format stranice odabrane dimenzije aplikacije na kojoj je grafika kreirana.



Slika 10. Prikaz grafike kreirane u PostScript Builder aplikaciji



Slika 11. Usporedba grafike dobivene iz generiranog koda kroz GSView

Kao što je vidljivo iz priloženih primjera, grafika dobivena iz PostScript programskog koda generiranog pomoću PostScript Builder aplikacije prikazuje oblike jednake onima koji su u toj aplikaciji kreirani. Ta preciznost korisniku omogućuje pouzdano dobivanje potrebnog koda i mogućnost učenja primjene PostScripta. Iako aplikacija razvijena u ovom radu nudi tek izradu osnovnih oblika, ovim je radom dokazana mogućnost njezine izrade, ali i njezina korisnost.

4.1. Prednosti aplikacije

Osim navedenih prednosti i funkcionalnosti, bitno je napomenuti da aplikacija korisniku nudi veliku fleksibilnost u pozicioniranju dodanih oblika, a u tome mu značajno pomaže mreža čiju veličinu brzo i lako može promijeniti, a može ju po želji i ukloniti. Upravo zahvaljujući skromnom broju funkcija, početnik se vrlo lako može snaći na jednostavnom korisničkom sučelju, čemu dodatno pridonosi

minimalističko dizajn te odsustvo boja koje bi mogle odvratiti pozornost. Iako danas postoje brojni programi za izradu grafike uz pomoć grafičkog korisničkog sučelja ili programskog koda, još uvijek na tržištu ne postoji aplikacija koja bi omogućila dobivanje programskog koda iz elemenata dodanih putem korisničkog sučelja, zbog čega je takva aplikacija unatoč ograničenim funkcionalnostima jedinstvena, što joj daje veliku prednost.

4.2. Ograničenja

Unatoč brojnim prednostima, korisnici aplikacije suočit će se i s nekim ograničenjima. Osim nedostatka mogućnosti slobodnog povlačenja običnih linija i kreiranja željenih oblika koji nisu među ponuđenima, korisnik nije u mogućnosti mijenjati opacitet kreiranih oblika niti ih opisati isprekidanim linijama. PostScript Builder ne omogućuje izradu petlji u kojima bi se oblici mogli nekoliko puta ponoviti uz promjenu varijabli. Nadalje, korisnik nije u mogućnosti istodobno označiti više elemenata. Osim toga, kroz grafičko korisničko sučelje nisu ostvarene ni brojne mogućnosti PostScripta koje ga čine naprednim u odnosu na moderne aplikacije koje koriste grafičko korisničko sučelje pa tako aplikacija razvijena u radu ne sadrži ni mogućnost promjene varijabli elemenata kroz petlju. Kada govorimo o potrebi korisnika da nauče osnove korištenja PostScripta, bitno je napomenuti i da postoji nekoliko različitih načina za opisivanje određenih elemenata te kod dobiven kroz PostScript Builder možda nije uvijek najjednostavniji, već je prilagođen standardnim slučajevima kako ne bi dolazilo do greške. Za potrebe ovoga rada u prvoj verziji aplikacije korisniku je omogućeno upoznavanje s osnovama grafičkog programskog jezika PostScript te je cilj izrade aplikacije uistinu ostvaren. Ipak, daljnjim razvojem moguće je postići mnogo napredniju aplikaciju koja bi mogla ući u širu upotrebu među ciljanom publikom.

4.3. Mogućnost daljnjeg razvoja

U budućnosti bi se razvojem aplikacije mogla riješiti sva ranije spomenuta ograničenja. Jedna od korisnih funkcionalnosti koja bi mogla doprinijeti široj

uporabi je mogućnost mijenjanja PostScript koda dobivenog na sučelju putem kojega bi se u aplikaciji mogli dodati i mijenjati elementi. Na taj način korisnicima bi se dodatno olakšalo učenje jer za eksperimentiranje s promjenama u kodu ne bi bilo potrebno spremati kod i mijenjati ga u drugoj aplikaciji kako bi se ponovno mogao otvoriti u drugom pregledniku. Korisno bi bilo i dodavanje mogućnosti spremanja tekstualne datoteke s PostScript kodom koji bi se automatski mogao učitati u PostScript pregledniku. Veliki bi napredak činila i mogućnost manipulacije veličinom elemenata. Ipak, prva verzija ove aplikacije zadovoljava cilj i osnovne potrebe ciljane publike te će daljnji razvoj i dodavanje naprednih funkcionalnosti biti naknadno razmotreno.

5. ZAKLJUČAK

U ovom radu prati se nastanak desktop aplikacije PostScript Builder od ideje do konačne izvedbe. Proces nastanka aplikacije sastoji se od definiranja ideje, analize potreba ciljane publike, analize sličnih aplikacija, dizajna korisničkog sučelja i konačno, izrade aplikacije. Analiza aplikacija namijenjenih izradi vektorske grafike pokazala je da na tržištu danas postoje brojne aplikacije s naprednim mogućnostima koje se ne mogu pronaći u aplikaciji PostScript Builder, no na tržištu trenutno ne postoji aplikacija koja iz vizualno kreiranih oblika može generirati programski kod, stoga je aplikacija čiji se razvoj opisuje u ovom radu danas nezamjenjiva. Zahvaljujući mogućnosti prikaza programskog koda grafike dobivene pomoću korisničkog sučelja korisnik se lako može upoznati s osnovama programiranja vektorske grafike i na taj način približiti brojnim mogućnostima koje PostScript nudi. Upravo to je cilj izrade aplikacije koja iako nudi tek ograničene mogućnosti izrade grafike, daljnjim razvojem može konkurirati funkcionalnostima danas popularnih grafičkih alata. Konačno, analiza danas popularnih aplikacija za izradu grafike koje koriste grafičko korisničko sučelje pokazala je kako su i one najnaprednije poput Adobe Illustratora u svojoj prošlosti imale mnogo skromniji početak.

6. LITERATURA

1. Weisfeld M. (2013.) "**The Object-Oriented Thought Process (4th Edition)**", Pearson Education, Inc., London, ISBN-13: 978-0321861276
2. *http://www.moje-instrukcije.com/index.php?option=com_content&view=article&id=1518:osnove-objektno-orijentiranog-programiranja&catid=178&Itemid=184*
   Osnove objektno orijentiranog programiranja, pristup: 4.8.2019.
3. *https://www.fer.unizg.hr/_download/repository/2.0_Zasto_OOP_i_neke_smjernice[2].pdf*
   Predavanja – Objektno orijentirano programiranje, pristup: 2.9.2019.
4. *https://www.codecademy.com/articles/mvc*
   MVC: Model, View, Controller | Codecademy, pristup: 4.8.2019.
5. Roger B. (2010.) „**ActionScript 3.0 Bible**“, Wiley Publishing, Inc., Indianapolis, ISBN: 978-0-470-52523-4
6. Shupe R., Rosser Z. (2008.) "**Learning ActionScript 3.0: A Beginner's Guide**", O'Reilly Media, Inc., Sebastopol, ISBN-13: 978-0596527877
7. *https://whatis.techtarget.com/definition/framework*
   What is framework? - Definition from WhatIs.com, pristup: 4.8.2019.
8. *https://searchsoftwarequality.techtarget.com/definition/integrated-development-environment*
   What is integrated development environment? - Definition from WhatIs.com, pristup: 4.8.2019.
9. Žiljak V., Pap K. (1998.) „***Postscript: Programiranje grafike***“, FS d.o.o., Zagreb, ISBN: 953 - 199 – 000
10. Adobe Systems Inc. (1999.) „**PostScript Language Reference Manual**”, Addison-Wesley, Reading, ISBN: 0-201-37922-8
11. *https://computeranimationhistory-cgi.jimdo.com/computer-sketchpad-1963/*
    Computer Sketchpad (1963), pristup: 4.8.2019.

12. *https://www.vecteezy.com/blog/2015/5/24/the-history-of-adobe-illustrator* **The History of Adobe Illustrator**, pristup: 4.8.2019.
13. Tavmjong B. (2011.) „**Inkscape: Guide to a Vector Drawing Program**“, Prentice Hall, Upper Saddle River, ISBN-13: 978-0-13-276414-8
14. *https://uxplanet.org/what-is-ui-vs-ux-design-and-the-difference-d9113f6612de* **What is UI design? What is UX design? UI vs UX: What's the difference),** pristup: 4.8.2019.
15. Levy J. (2015.) „**UX Strategy: How to Devise Innovative Digital Products that People Want**“, O'Reilly Media Inc., Sebastopol, ISBN-13: 978-1449372866
16. Johnson J. (2014.) "**Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines**", 2010 Elsevier Inc., Waltham, ISBN: 978-0-12-407914-4
17. *https://github.com/minimalcomps/minimalcomps/blob/master/src/com/bit101/components/ColorChooser.as* **minimalcomps/ColorChooser.as at master · minimalcomps/minimalcomps · GitHub**, pristup: 4.8.2019.
18. *https://github.com/minimalcomps/minimalcomps/blob/master/src/com/bit101/components/Slider.as* **minimalcomps/Slider.as at master · minimalcomps/minimalcomps · GitHub,** pristup: 4.9.2019.
19. *https://greensock.com/tweenmax* **TweenMax - Products – GreenSock,** pristup: 4.9.2019

7. POPIS MANJE POZNATIH RIJEČI I AKRONIMA

PostScript Builder - aplikacija čiji je razvoj opisan u ovom radu

PostScript - grafički programski jezik namijenjen programiranju vektorske grafike

Objektno orijentirano programiranje - način programiranja inspiriran predmetima iz stvarnog života koji se mogu opisati kao objekti, a od kojih je svaki definiran metodama

Objekt – u objektno orijentiranom programiranju objekt je naziv koji se odnosi na skup svojstava koje možemo objediniti u smislenu cjelinu

Klasa (Class) - u objektno orijentiranom programiranju definira pravila koja opisuju svojstva objekta

Model View Controller (MVC) - obrazac softverske arhitekture koji se koristi za odvajanje dijelova aplikacije u komponente ovisno o namjeni

ActionScript - objektno orijentiran programski jezik korišten u ovom radu

Razvojno okruženje (framework) - označava slojevitu strukturu koja se sastoji od gotovih kodova i programa te korisniku omogućuje selektivne izmjene u svrhu razvoja aplikacije

Application Programming Interface (API) - sučelje za programiranje aplikacija, skup određenih pravila i specifikacija koje programeri slijede da bi se mogli služiti uslugama ili resursima operacijskog sustava ili nekog drugog složenog programa

Integrated Development Environment (IDE) - integrirano razvojno okruženje, specijalizirani softverski paket koji sadrži uređivač teksta prilagođen pisanju programa, prevoditelj programa, alate za testiranje programa i alate za lociranje pogrešaka

Flash Builder - integrirano razvojno okruženje korišteno u ovom radu

Computer-Aided Design (CAD) - računarski potpomognuto crtanje

UX - User Experience - označava ponašanja, stavove i emocije koje korisnik doživljava tijekom uporabe određenog proizvoda ili usluge

UI - User Interface - vizualno opisan prostor u kojemu se odvija komunikacija između računala i čovjeka

Slider – klizna traka na grafičkom korisničkom sučelju putem koje se odabire željena vrijednost varijable