Osnove računalne grafike u Pythonu

Uljević, Ivo

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

Osnove grafike u Pythonu

Ivo Uljević

Split, rujan 2021.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

Osnove računalne grafike u Pythonu

Ivo Uljević

Sažetak

Tema ovoga rada je osnove računalne grafike u Pythonu. U ovome radu su prikazane osnovne funkcije koje prikazuje grafički prikaz u Pythonu. Kornjačina grafika je modul koji se koristi za grafiku u Pythonu i preko njegovih funkcija će se objasniti osnove računalne grafike u Pythonu. Osim kornjačine grafike u radu je objašnjen modul koji se koristi za izradu grafičkog korisničkog sučelja, a to je tkinter. Osim ova dva još je spomenut graphics.py. Za projekt je korištena kombinacija kornjačine grafike i tkintera.

Ključne riječi: Grafika, Kornjačina grafika, Python, tkinter, graphics.py, moduli

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 39 stranica, 0 grafičkih prikaza, 3 tablica i 5 literaturnih navoda.

Izvornik je na hrvatskom jeziku

Mentor: Dr.sc. Ani Grubišić, viši predavač Prirodoslovno-matematičkog fakulteta Sveučilišta u Splitu

Ocjenjivači: izv.prof.dr.sc. Ani Grubišić, izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

izv.prof.dr.sc. Branko Žitko, izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Ines Šarić-Grgić, mag.ing.rač. , asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Split

Basic documentation card

Thesis

University of Split

Faculty of Science

Department of Computer Science

Ruđera Boškovića 33, 21000 Split, Croatia

Basic of graphics in Python

Ivo Uljević

The topic of this paper is the basics of computer graphics in Python. This paper presents the basic functions displayed by a graphical representation in Python. Turtle Graphics is a module used for graphics in Python and its functions will explain the basics of computer graphics in Python. In addition to turtle graphics, the paper explains the module used to create a graphical user interface, which is tkinter. In addition to these two, graphics.py is also mentioned. A combination of turtle graphics and tkinter was used for the project.

Key words: Graphics, Turtle Graphics, Python, tkinter, graphics.py, modules

Supervisor: Ani Grubišić, Ph.D. , Senior Lecturer of Faculty of Science, University of Split

Reviewers: Ani Grubišić, PhD, Associate Professor, Faculty of Science, University of Split

Branko Žitko, PhD, Associate Professor, Faculty of Science, University of Split

Ines Šarić-Grgić, mag.ing.comp. Assistant, Faculty of Science, University of Split

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom OSNOVE GRAFIKE U PYTHONU izradio samostalno pod voditeljstvom dr.sc. Ani Grubišić. U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u završnom radu na uobičajen, standardan način citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Student

Ivo Uljević

SADRŽAJ

1. UVOD

U današnje vrijeme računalna grafika je svuda oko nas i ona se koristi u mnogim industrijama poput medicine, filma, brodarstva i mnogih drugih grana. Računalna grafika je dio informacijske tehnologije koji se bavi dobivanjem grafičkih prikaza uz pomoć računala. Prikazi se mogu generirati, obrađivati i pohranjivati u vektorskom (vektorska slika) ili rasterskom zapisu (rasterska slika ili bitmapa), te se prikazati na zaslonu računala, ispisati pisačem i nacrtati crtalom. [1] Za početke računalne grafike možemo zahvaliti Euklidu koji je formulirao osnove aksioma geometrije koji se koriste danas u grafici. Rene Descartés je razvio analitičku geometriju i koordinatni sustav koji koristimo u modulima kako bi nešto pozicionirali. Prvi znaci računalne grafike koje poznajemo se tek pojavljuju 60-ih godina, a najvažniji od njih je bio prvi računalni animirani film *Two-Gyro Gravity-Gradient Attitude Control System*. Prvi interaktivni program koji je nastao od strane Ivana Sutherlanda je *Sketchpad*. Bitno je pomogao u načinu rukovanja s računalom, tako da se *Sketchpad* smatra pretkom modernog računalnog potpomognutog crtanja (CAD) kao i glavni pomak u razvoju uopće u rračunalnoj grafici. [2] U ovome završnom radu su opisane osnove računalne grafike kroz tri modula.

Na početku rada ćemo se upoznati s kornjačinom grafikom koja se danas koristi u školama i to zbog svoje jednostavnosti. Približno će se objasniti njene osnovne funkcije kretanja, kako koristiti boje i kako crtati s kornjačom. Nakon kornjačine grafike objašnjen je *tkinter* koji je jako zanimljiv zbog izrade grafičkog korisničkog sučelja, a u radu su opisane botun, labela i *textbox*. U projektu je objašnjena primjena modula *tkinter* i kornjačine grafike preko igrice „Vješalo“. Osim ova dva modula opisan je *graphics.py* koji je sličan kornjači po kodiranju i sličnim rezultatima, a zbog toga je uzet kao dodatni modul.

2. KORNJAČINA GRAFIKA U PYTHONU

Program u kojem će se prikazati osnove grafike zove se Python. On je programski jezik opće namjene kojeg je napravio Guido van Rossum 1990. godine. Python je jedan od popularnijih programskih jezika jer dopušta programeru da se koristi različitim stilovima poput objektno orijentiranog, strukturnog i aspektno orijentiranog programiranja. U ovome radu koristit će se grafički dio Pythona.

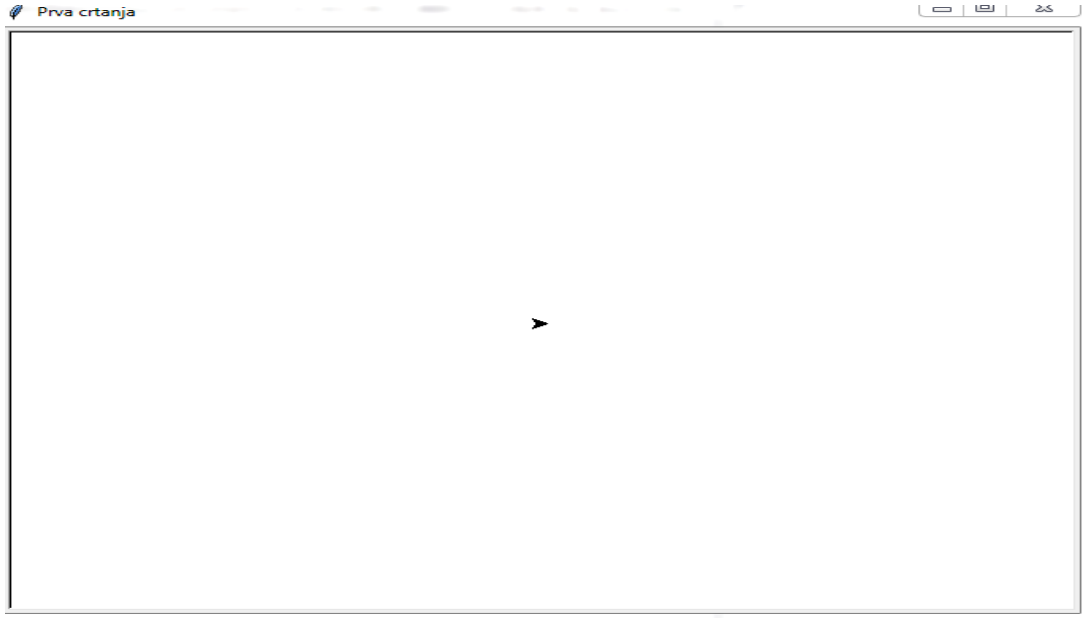
U programskoj biblioteci Pythona postoje mnogi moduli, ali postoje i razni moduli koji se mogu skinuti s interneta za grafički prikaz u Pythonu. Može se reći da je Logo bio prva inačica kornjačine grafike.

Kornjačina grafika je stvorena za obrazovnu uporabu, ponajprije za konstruktivističko učenje, od strane Daniela G. Bobrowa, Wallyja Feurzeiga i Seymoura Paperta. Danas je najviše poznat po svojoj takozvanoj *turtle* grafici, ali jezik također posjeduje značajne mogućnosti za baratanje listama, datotekama i U/I. [3]

Kornjačina grafika je stvorena kako bi se djeca lakše naučila programirati. Za njih je ovakav način programiranja zanimljiviji zbog toga što se na kraju dobiva grafički prikaz rezultata.

2.1. Kornjačin prozor

Prije nego što krenemo programirati treba uvesti *turtle* u Python, a to se radi tako što se na početku svakog algoritma upiše *import turtle*. Ova naredba će omogućiti da koristimo sve moguće funkcije koje nam nudi *turtle*. Nakon što se *turtle* uvede iz biblioteke, za otvaranje prozora je samo potrebno staviti title i unutar zagrada staviti naziv prozora. Ako se ne postavi naziv unutar naredbe on poprima univerzalni naziv, a to je *Python Turtle Graphics*. Nakon toga se otvara prozor koji je veličine 600x600 piksela, a to znači da prozor ima ukupno 3600 piksela. [3]. Piksel je najmanji element na zaslonu ili na nekoj bitmap slici. Za svaki piksel se može odrediti koordinata, a prema tome, prozor se može zamisliti kao koordinatni sustav koji ima ishodište u sredini. Ako napišemo naredbu *reset()* pojaviti će se kornjača koja se nalazi na ishodištu koordinatnog sustav i koja gleda prema pozitivnoj osi x, koja je prikazana na slici sljedećoj slici.



Slika 2.1. Kornjačin prozor

2.2. Osnovne funkcije za gibanje kornjače

Razmak između piksela na zaslonu ovisi o broju stupaca i redaka, a kako je broj stupaca i redaka promjenjiv, tako je i razmak između piksela promjenjiv. Taj razmak nazivamo rasterskom jedinicom.

U sljedećoj tablici su prikazane funkcije za relativno gibanje koju još nazivamo gibanjem po vektorskoj grafici. Vektorska grafika je prikazivanje slike pomoću geometrijskih oblika poput linije, točke, krivulje i tako dalje.

Tablica 2-1 Osnovne funkcije za relativno gibanje

Fukcija	Alternativni naziv	Opis funkcije
Forward(y)	Fd(y)	Pomiče kornjaču za y jedinica naprijed
Backward(y)	Bk(y)	Pomiče kornjaču za y jedinica unazad
Left(y)	Lt(y)	Pomiče kornjaču za y stupnjeva ulijevo
Right(y)	Rt(y)	Pomiče kornjaču za y stupnjeva udesno

Osim što ima funkcije za relativno gibanje modul ima i funkciju za gibanje kornjače po apsolutnim koordinatama ili gibanje po rasterskoj grafici koje su opisane u sljedećoj tablici.

Tablica 2-2 Osnovne funkcije za apsolutno gibanje

Funkcija	Alternativni naziv	Opis funkcije
Goto(x,y)	Setpos(x,y)	Pomiče kornjaču na točku s koordinatama (x,y)
Goto(t)	Setpos(t)	Pomiče kornjaču na točku s koordinatama t, gdje je t par brojeva
Setx(x)		Postavlja prvu koordinatu na x, a druga ostaje nepromijenjena
Sety(y)		Postavlja drugu koordinatu na y, a prva ostaje nepromijenjena
Setheading(kut)	Seth(kut)	Usmjerava kornjaču tako da pokazuje smjer kuta kut

2.3. Upravljanjem kornjačom i crtežom

U sljedećoj tablici se nalaze funkcije koje nam pomažu pri crtanju neke slike.

Tablica 2-3 Osnovne funkcije za kornjačino crtanje

Funkcija	Alternativni naziv	Opis funkcije
Pendown()	Pd(), down()	Kornjača ostavlja trag
Penup()	Pu(), up()	Kornjača ne ostavlja trag
Pensize(d)	Width(d)	Kornjača ostavlja trag debljine d
Showturtle()	St()	Kornjača postaje vidljiva
Hideturtle()	Ht()	Kornjača postaje nevidljiva
Home()		Kornjača se vraća na početni položaj
Undo()		Poništava se posljedna naredba

Osim ovih naredbi za upravljanjem kornjače postoji još jedna važna funkcija pomoću koje brišemo sve crteže koje smo napravili u grafičkom prikazu, a to je funkcija koja se zove *clear().*

2.4. Speed() i tracer()

Za brzinu kretanje kornjače u grafičkom prozoru koristimo funkciju *speed*(brzina), a brzina nam je vrijednost koju će kornjača poprimiti za svoje kretanje. *Tracer()* je funkcija koja nadzire brzinu kretanja kornjače, a ona zapravo usporava odvijanje programa kako bismo mogli vidjeti promjene u grafičkom prozoru. Ako ne želimo usporiti odvijanje programa onda se to obavlja naredbom *tracer(False),* a ponovno uključivanje naredbom *tracer(True).* Ako su pozvane *tracer()* i *speed()* funkcije, onda brzina može poprimiti vrijednost iz intervala [1,10], pri čemu je 1 najmanja brzina, a 10 najveća brzina. U slučaju da ostavimo prazne zagrade unutar funkcije *speed()*ona će poprimiti brzinu vrijednosti 3, a *tracer()* će poprimiti vrijednost 1 ili *true*.

3. BOJANJE U PYTHONU

U modulu postoje funkcije pomoću kojih se mogu izrađivati crteži u boji. Pomoću modula možemo obojati pozadinu grafičkog prozora, trag kornjače i ispune likova omeđenih zatvorenom crtom.

Python koristi za bojanje RGB(*red,green,blue*) model i svaka boja sadrži udjel u intervalu od 0 do 255. U Pythonu je definirano sveukupno 147 boja, a neke od osnovnih boja prikazane su na sljedećoj slici.

Boja			r	g	b	r	g	b
	String		colormode(255)			colormode(1.0)		
crna	'black'	'#000000'	0	0	0	0.0	0.0	0.0
crvena	'red'	'#ff0000'	255	0	0	1.0	0.0	0.0
zelena	'green'	'#00ff00'	0	255	0	0.0	1.0	0.0
plava	'blue'	'#0000ff'	0	0	255	0.0	0.0	1.0
žuta	'yellow'	'#ffff00'	255	255	0	1.0	1.0	0.0
magenta	'magenta'	'#ff00ff'	255	0	255	1.0	0.0	1.0
tirkizna	'cyan'	'#00ffff'	0	255	255	0.0	1.0	1.0
bijela	'white'	'#ffffff'	255	255	255	1.0	1.0	1.0
smeđa	'brown'	'#99661f'	153	102	31	0.60	0.40	0.12
narančasta	'orange'	'#faa01f'	250	160	31	0.98	0.625	0.12
ružičasta	'pink'	'#fa0ab2'	250	10	178	0.98	0.04	0.7
ljubičasta	'violet'	'#9966b2'	153	102	178	0.60	0.4	0.7

Slika 3.1. Neke od boja u Pythonu

3.1. Bojanje pozadine grafičkog prozora

Za postavljanje boje u grafičkom prozor koristimo funkciju *bgcolor(boja)*, gdje je boja *string* koji odabire neku od boja, a ako se ostavi prazna zagrada funkcija odabire trenutačnu boju pozadine.

3.2. Bojanje traga kornjače

Nakon pokretanja programa boja traga kornjače je crna, a da bismo to promijenili koristimo funkciju *pencolor(boja).* Boja traga će se promijenit nakon što se pozove funkcija i nakon što se kornjača pokrene. *Pencolor()*bez postavljanja boje će ispisati trenutačnu boju traga.

3.3. Bojanje likova

Za ispunjavanje likova koji su nastali zatvorenom crtom koristimo funkciju *fillcolor().* Također, imamo naredbu *begin_fill()* koja označava početak bojanja, te funkciju *end_fill()* koja označava završetak bojanja. Za funkciju *fillcolor()* je važno napomenuti da će ona ispuniti lik jedino ako se nalazimo s kornjačom unutar njega.

Funkcija *color()* se može pozvati na tri načina. Prvi je bez parametara gdje funkcija vraća dvije boje, a to su boja traga i boja ispune. Drugi je način s jednim parametrom gdje funkcija mijenja boju ispune i traga tako da budu iste boje. Treći način je s dva parametra gdje funkcija *color(boja1,boja2)* mijenja boju traga u *boja1*, a boja ispune u *boja2*.

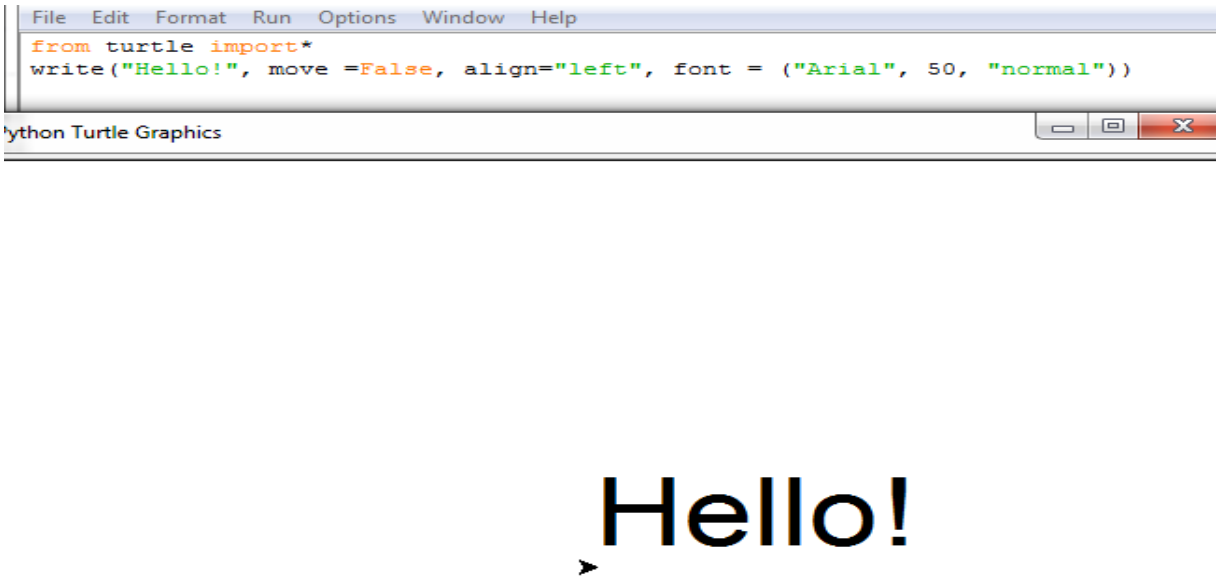
3.4. Pisanje u kornjačinom prozoru

U modulu *turtle* postoje funkcije koju mogu ispisivati riječi, brojeve ili znakove, a jedna od tig funkcija je i funkcija *write()*.

```
Write(arg, move =False, allign="left", font = („Arial", 8,
„normal"))
```

Iznad je prikazan primjer jednog ispisa teksta. *Arg* je string koji će se ispisivati kao i funkcija *print()*. Ona će ispisivati i brojeve tipa *int* u predviđenom obliku. *Move* se koristi za pomicanje kornjače. Ako je *move = False* kornjača se pri ispisivanju teksta neće pomicati, a ako je *move = True* ona će se nakon ispisa pomaknuti na kraj ispisanog teksta.

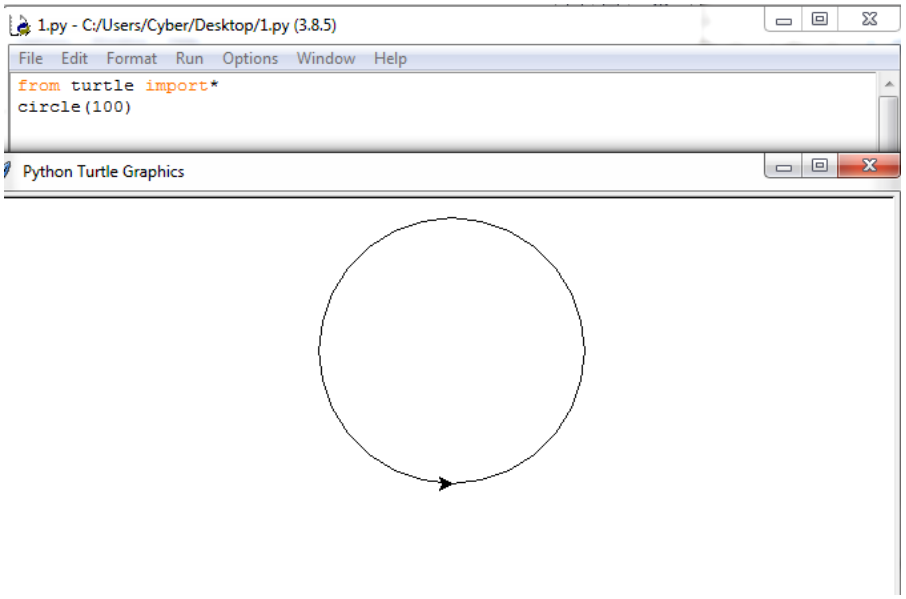
Allign određuje smještaj teksta u odnosu na položaj kornjače, a to može biti *left, right* i *center*. Ako *allign* postavimo na *left*, tekst će se ispisati tako da se njegov početak nalazi na mjestu položaja kornjače. S druge strane, ako ga postavimo na *right*, tekst će se ispisati tako da se kraj ispisanog teksta nalazi na mjestu kornjače. Zadnja opcija je *center*, kod kojega se tekst nalazi na mjestu položaja kornjače. Font koristimo za izgled teksta i veličinu teksta.



Slika 3.2. Pisanje u Pythonu

3.5. Crtanje kružnice

Za crtanje geometrijskih oblika kao što je kocka, kut i slično, koristimo osnovne funkcije gibanja, ali te iste funkcije ne mogu nacrtati kružnicu. Zato unutar modula postoji ugrađena naredba *circle()* koja crta kružnicu. Unutar zagrada postavlja se veličinu polumjera, a kad nacrta kružnicu, ona se vrati u početni položaj.



Slika 3.3. Crtanje kružnice

4. TKINTER

Jedan od modula koji se nalazi unutar Pythona za izradu grafičkih korisničkih sučelja je *Tkinter*. Dostupan je na svim Unix platformama i Windows sustavima, a poziva se tako što se uvede Tkinter u Pythonu.[5] Kada se pozove funkcija *Tk()* otvorit će se prozor u kojem možemo dodati labele, textboxove, botune i ostalo. *Tkinter* je jako zanimljiv za izradu aplikacija u kojima se dodaju razni tekstovi pri kojem će se nešto ispisivati na prozor.

4.1. Labele

Postoje funkcije koje se mogu koristiti kako bi se stvorili elementi grafičkog sučelja.

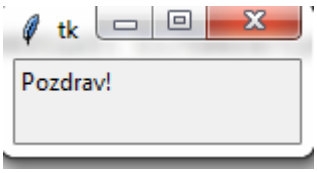
Za dodavanje labele koriste se sljedeće funkcije:

```
from tkinter import *

prozor = Tk()

lbl = Label(prozor, text="Pozdrav!")
lbl.grid()
```

Kao rezultat dobije se prikaz koji se nalazi na sljedećoj slici.



Slika 4.1. Primjer labele u Pythonu

Unutar funkcije *Label* može se promijeniti font, te se može povećati ili smanjiti veličina slova, a to se radi na način da se postavi *font = „Željeni stil slova“, veličina*. Visina i širina labele mijenja se pomoću varijabli *width* i *height*. Boja slova i boja labele se također može promijeniti, tako što se stavi *foreground* ili *fg* koji će promijeniti boju slova nakon znaka jednakosti, a potrebno je i postaviti željenu boju. *Background* ili *bg* se koristi za promjenu boje labele. Kao što se mijenjaju slova, na isti način se mijenja i boja pozadine. U labeli se može mijenjati tekst preko neke druge prije definirane varijable, a to se postiže tako što se

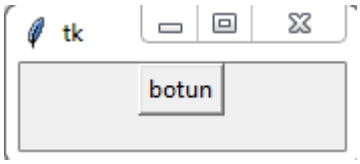
stavlja *textvariable = ime varijable* i onda samo se postavi ime *varijable.set()*. Preko *get* funkcije se dobiva ono što se nalazi unutar labele.

4.2. Button

Funkcija *button* se koristi kako bi se dodali botuni u prozoru.

```
import tkinter
prozor = tkinter.Tk()
B = tkinter.Button(prozor, text ="botun")
B.pack()
```

Kao rezultat se dobiva botun prikazan na sljedećoj slici.



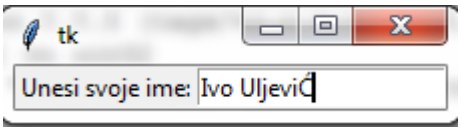
Slika 4.2. Primjer botuna u Pythonu

Da bi se boja pozadine promijenila, koristi se *background* ili *bg*, a da bi se promijenila boja slova koristi se *foreground* ili *fg*. Preko *command* fukncije može se pozvati neka druga funkcija kada se pritisne botun. Za širinu i visinu koristi se *width* i *height*. Također se može mijenjati font na isti način kao što se mijenja u labeli.

4.3. Textbox

Textbox se koristi za unos podataka, odnosno tekstualnih podataka od strane korisnika aplikacije. Ako korisnik unese duži tekst od raspoloživog prostora sadržaj će se pomicati ulijevo.

```
from tkinter import *
prozor = Tk()
l1=Label(prozor, text="Unesi svoje ime:").grid(row=0)
e1 = Entry(prozor)
e1.grid(row=0, column=1)
```

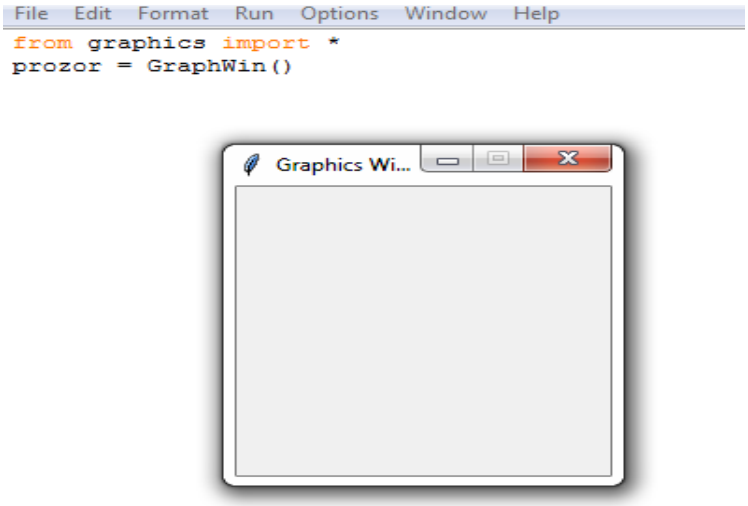


Slika 4.3. Primjer unosa teksta u Pythonu

Kao i u prethodnim primjerima, na isti način mijenjaju se boja, visina i širina. *Textbox* sadrži metode koje pomažu pri rješavanju nekih problema. Metoda *delete* se koristi za brisanje nekog znaka ili niza znakova. *Get* se koristi kako bi se dobilo ono što se nalazi u *textboxu*, ali prije toga se mora definirati određena varijabla, a potom u *entry* unijeti *textvariable= ime* te varijable.

5. GRAPHICS.PY

Najbolji i najlakši modul za shvatiti osnove grafike u Pyhtonu je graphics.py kojeg je razvio John Zelle. Zelleov modul nije dio standardne Python distribucije, a da bi ga prepoznao mora ga se dohvatiti. Nakon toga je potreban *GraphWin* koji će otvoriti grafički prozor. *GraphWin* je vrsta objekta iz Zelleovog grafičkog paketa koji automatski prikazuje prozor kada je kreiran. Nakon pokretanja Pythona otvara se prozor kao što vidimo na slici ispod.

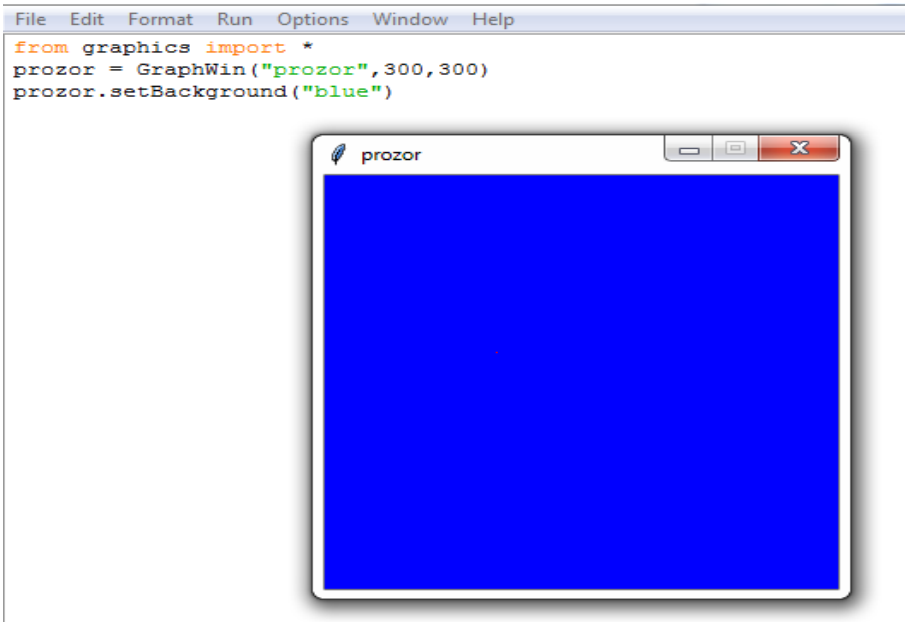


Slika 5.1. Graphics Win

5.1. GraphWin

GraphWin predstavlja prozor na ekranu na kojem se mogu crtati različite grafičke slike. On koristi metode poput *GraphWin* (naziv, visina, dužina) koja konstruira prozor određene visine i dužine, a ako ona nije definirana program je sam postavlja na 200x200 piksela.

Također, može se postaviti boja pozadine preko naredbe *setBackground*. Za zatvaranje koristi se naredba *close*.

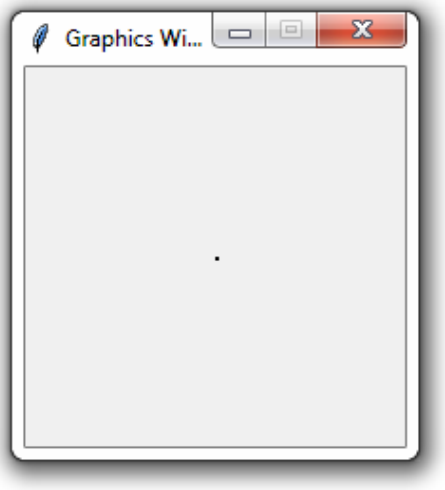


Slika 5.2. Bojanje Graphics Win

5.2. Point metoda

Point objekt će stvorit točku na određenoj koordinati, što je prikazano na sljedećoj slici i ona će biti veličine jednog piksela. *Point* se definira kao točka te se mogu odrediti njene koordinate koje se mogu nabaviti preko metoda *getx* i *gety*. *Point* objekt se koristi i za crtanje drugih objekata kao što se vidi u sljedećem primjeru.

```
from graphics import *
prozor = GraphWin()
točka = Point(100, 100)
točka.draw(prozor)
```

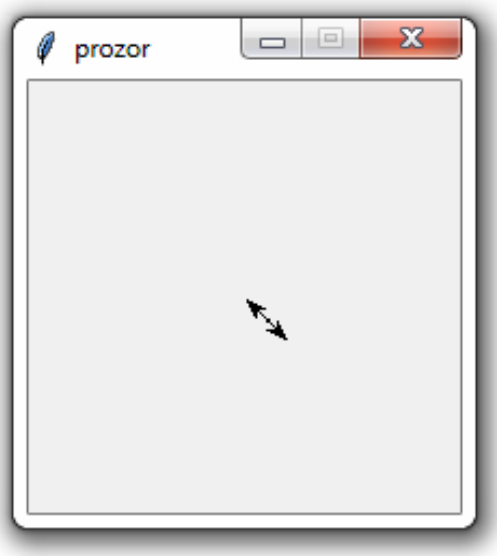


Slika 5.3. Stvaranje točke u Graphics Win

5.3. Line metoda

Line naredba se koristi za kreiranje linije koja ima dvije točke, a koja se definira unutar naredbe *line*. *SetArrow* se koristi kako bi se stavile strelice na liniju. Ona ima 4 opcije: *both*, *first*, *last* i *none*. Prema programu, ako se ne stavi opcija on će sam stavit *none* opciju. *GetCenter* postavlja liniju u sredini i podebljava točku.

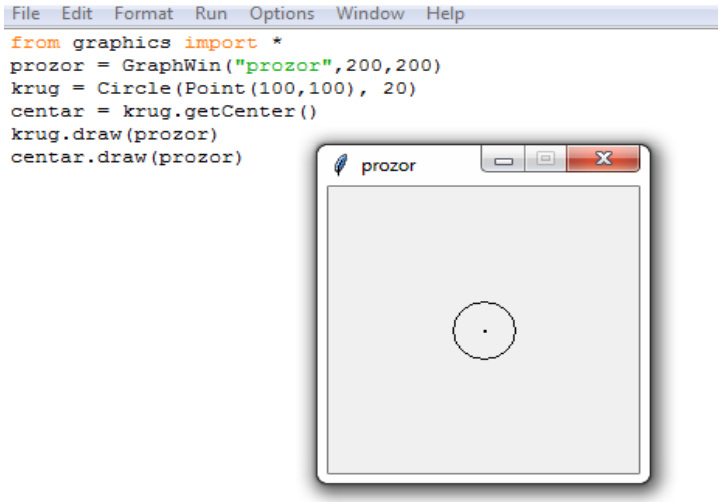
```
from graphics import *
prozor = GraphWin("prozor",200,200)
linija = Line(Point(100,100),Point(120,120))
linija.setArrow("both")
linija.draw(prozor)
lin = linija.getCenter()
lin.draw(prozor)
```



Slika.5.4. Linija u Grapchis Win

5.4. Circle metoda

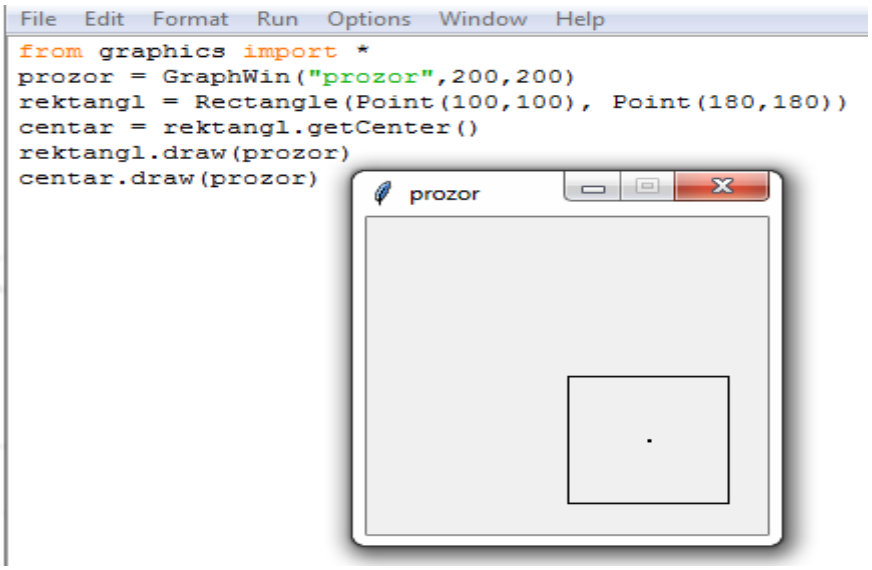
Za kreiranje kružnice koristi se naredba *circle* u kojoj se prvo napiše središte kruga, a zatim se definira njezin radijus. *GetCenter* stvara u sredini kružnice točku kao njezinog središta, a za dobivanje radijusa koristi se naredba *getRadius*.



Slika 5.5. Stvaranje kruga u Graphics Win

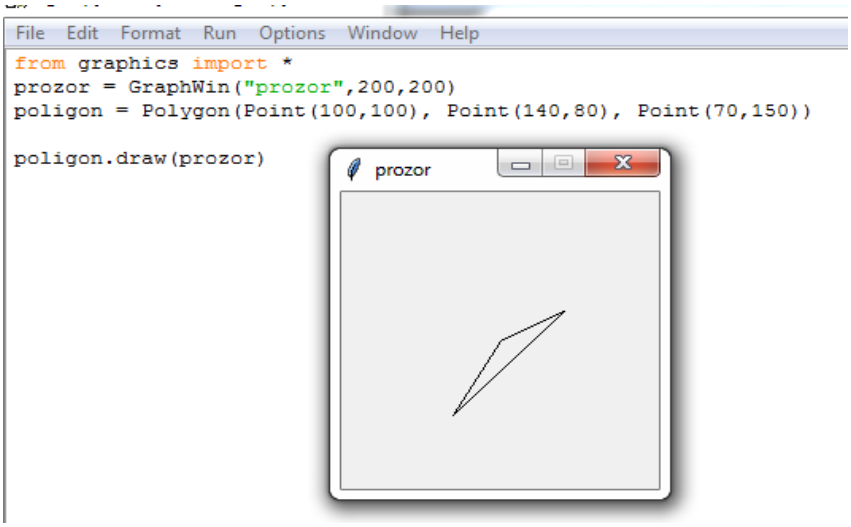
6. OSTALE METODE

Za crtanje pravokutnika koristi se metoda *Rectangle*. Ona isto kao i *line* metoda uzima 2 točke pomoću koje će se nacrtati pravokutnik. Za određivanje središta također se koristi *getCenter*.



Slika 6.1. Stvaranje kocke u Graphics Win

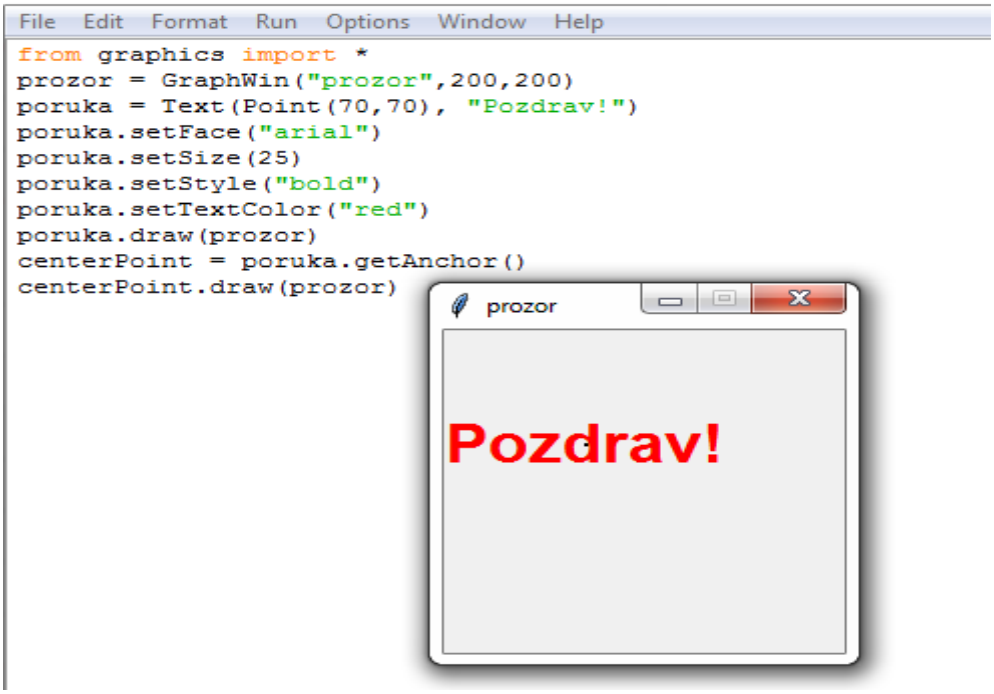
Za crtanje poligona koristi se *Polygon* naredba. Na sljedećoj slici je nacrtana sa 3 točke, ali ona se može crtati i s više točaka.



Slika 6.2. Stvaranje poligona u Graphics Win

6.1. Tekst metode

Osim što graphics.py može crtati objekte u prozoru, on može i ispisivati tekstualne poruke na grafičkom prozoru. Naredba pomoću koje se to radi je *Text* i ona u sebi sadrži koordinate na koje će tekst ispisivati. Ostale naredbe su: *setText* pomoću koje se pretvara objekt u string, *getText* koja vraća string natrag, *getAnchor* stavlja točku u sredini teksta, *setFace* postavlja font koji želimo za poruku, *setSize* mijenja veličinu slova u tekstu, *setStyle* mijenja tekst u određeni stil kao *italic*, *bold*, *normal* i *italic bold*, i *setTextColor* za boju teksta.



Slika 6.3. Prikaz teksta unutar Graphics Win

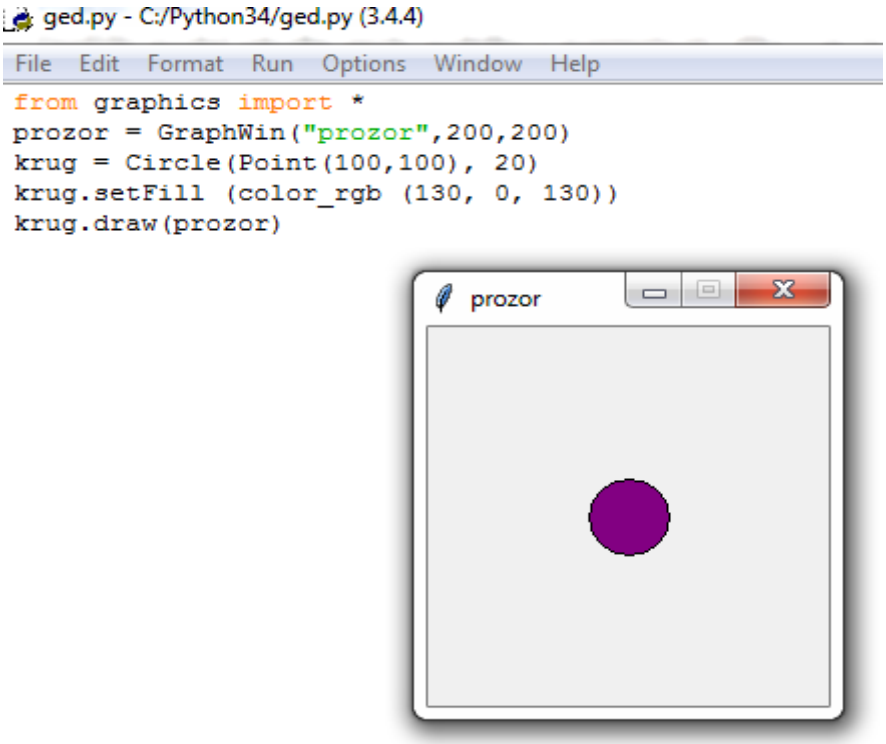
6.2. Prikaz slika

Grafički modul također pruža podršku u manipulaciji slika i on će najmanje podržavati GIF i PPM. Slike se mogu pomicati po x i y kordinatama i također se može po njim crtati. Metode koje se koriste za manipulaciju slika su *Image*, koji koristi *point* za postavljenje slike i naziv slike, *getAnchor* koja postavlja točku u sredinu slike, *getWIdth* i *getHeight* se koriste za širinu i visinu, *getPixel* ispisuje RGB na određenoj koordinati, *setPixel* na koordinati postavlja točku boje i na kraju naredba *save* koja sprema sliku na desktopu.

6.3. Generiranje boja

Mnoge boje dolaze u raznim nijansama, kao što su "*red1*", "*red2*", "*red3*","*red4*", koje su sve tamnije crvene boje. Grafički modul također pruža funkciju za miješanje vlastitih boja numerički.

Naredba za miješanje boja je *color rgb(red, green, blue)* od intenziteta crvene, zelene i plave boje i one su u rasponu od 0 do 255.

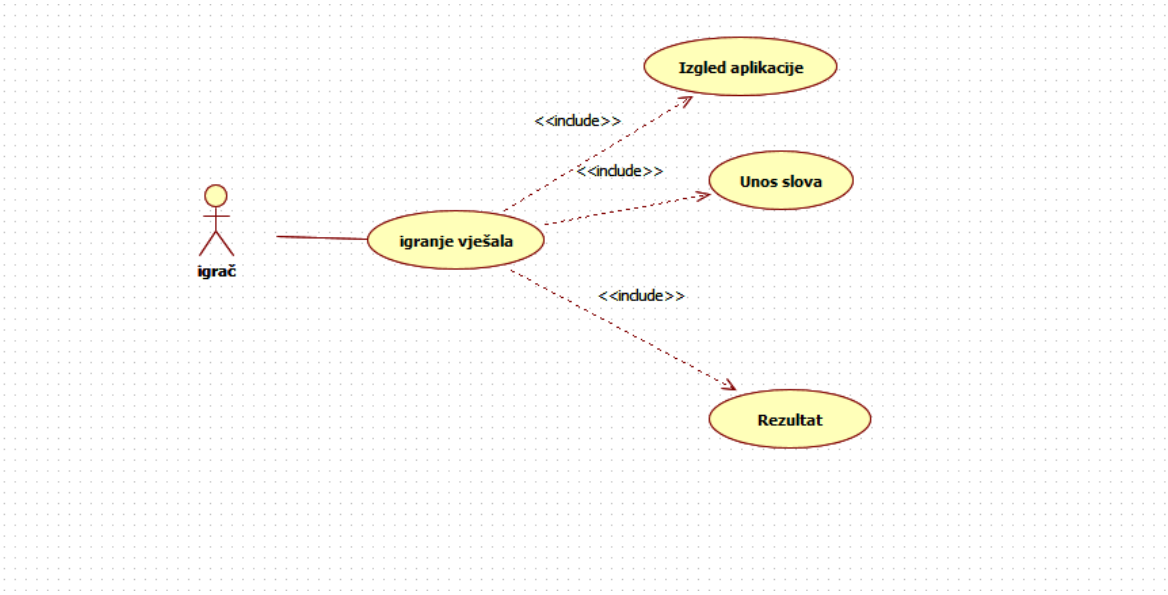


Slika 6.4. Bojanje u Graphics Win

7. PRIMJENA ELEMENATA RAČUNALNE GRAFIKE U IGRICI VJEŠALO

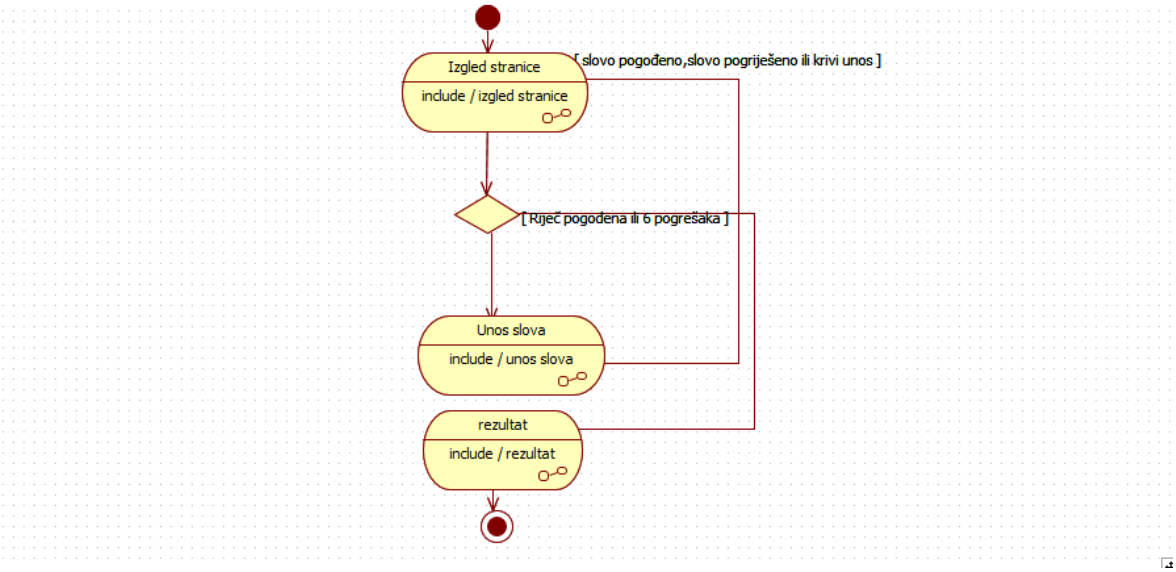
Cilj je izraditi igricu „Vješalo“ koja će dočarati osnove grafike u Pythonu s modulima *tkinter* i *turtle*. Vješalo je igrica koja se igra između dva ili više igrača gdje jedan od igrača pogađa što se skriva iza crtica, a crtice predstavljaju slova koja predstavljaju jednu riječ. U igrici se nalazi vješalo koje za svako pogrešno slovo dobiva jedan dio tijela. Vješalo dobiva glavu, tijelo, dvije ruke i dvije noge što znači da se može imati ukupno šest grešaka. Za svako pogođeno slovo otvara se crtica i nastavlja se dalje. Igra se završava tek kad su sva slova pogođena ili kada se pogriješi šest puta.

Kod dijagrama korištenja igrač je povezan s igranjem vješala dvosmjernom asocijacijom jer igrač daje informacije igri i od igrice prima informaciju. Igranje vješala opisano je u tri koraka, a to su izgled aplikacije, unos slova i rezultat.



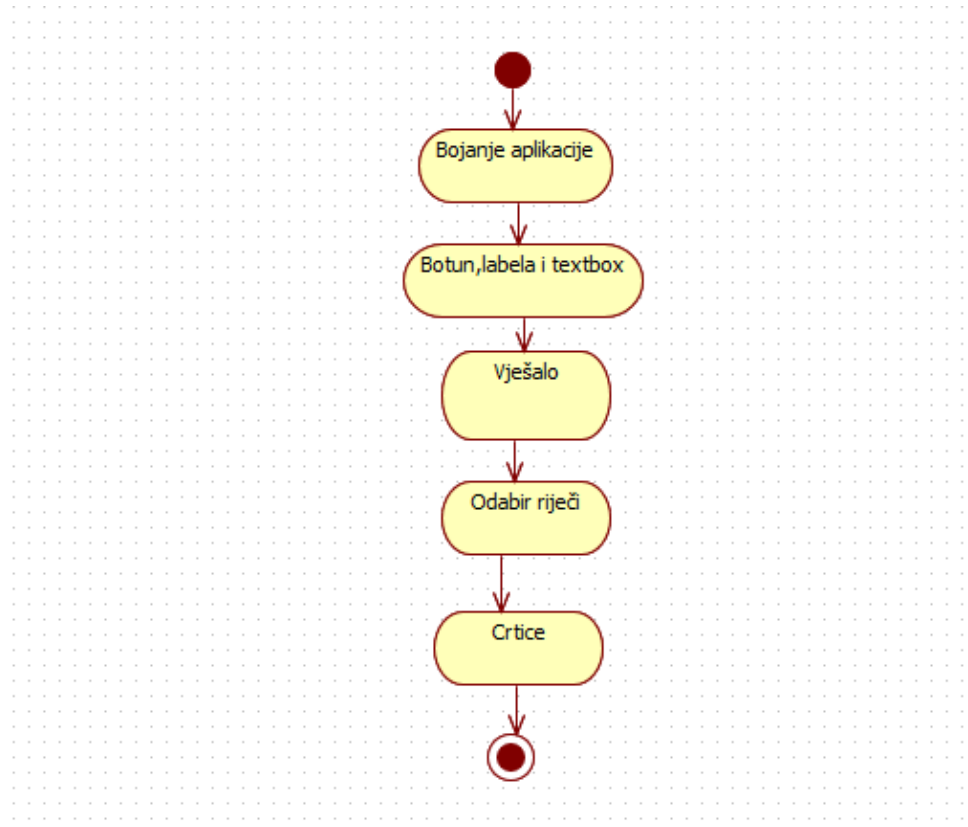
Slika 7.1 Dijagram korištenja

U sljedećem dijagramu opisano je kako ide tijek igranja od samog izgleda stranice, zatim unosa slova u kojoj ide provjera hoće li se vratiti na izgled stranice ili će ići na sljedeću aktivnost, a to je rezultat.



Slika 7.2. Dijagram aktivnosti

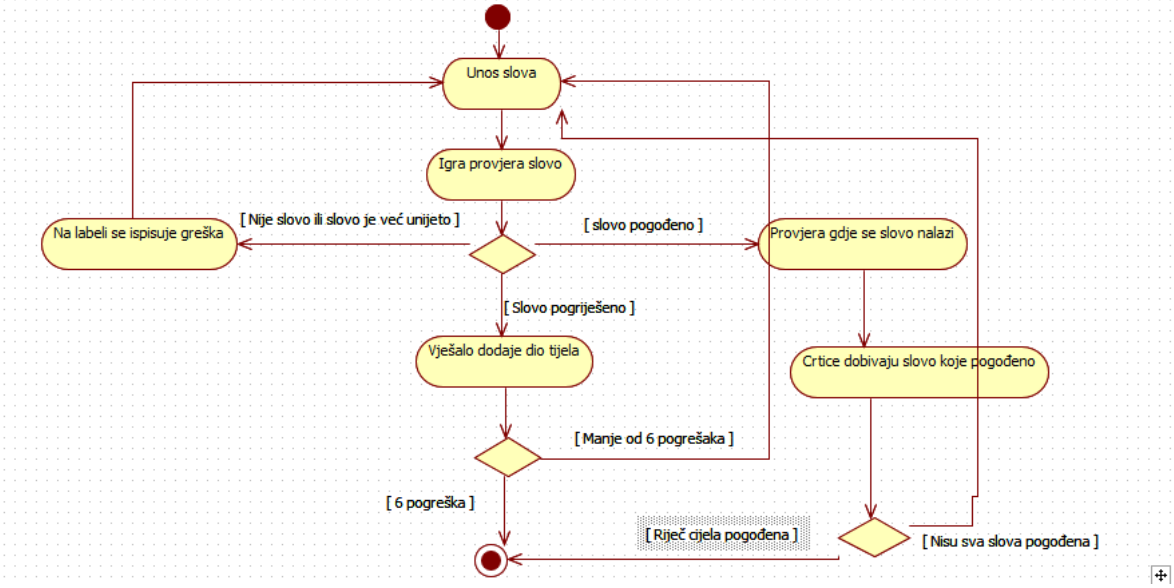
Na sljedećem dijagramu je prikazan tok igrice kada se igrica pokrene .



.

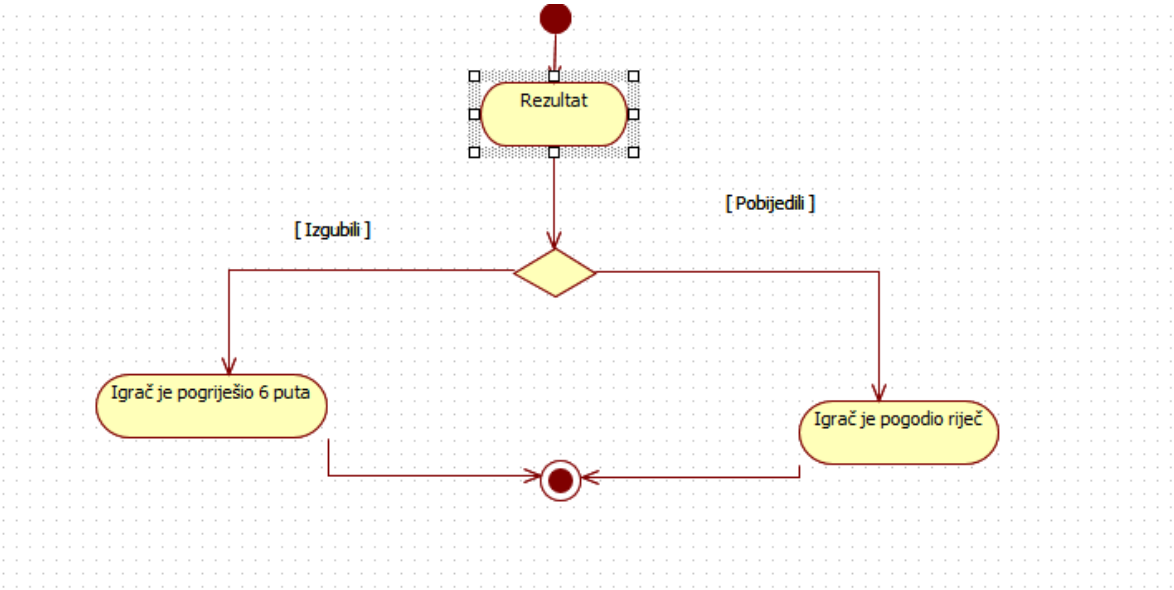
Slika 7.3.Dijagram podaktivnosti za izgled stranice

Nakon unosa slova igra provjera slovo. Rezultat provjere može biti da je unijeto slovo nije slovo ili da je to slovo već unijeto prije i taj rezultat vraća na unos i ispisuje se greška. Ako je slovo pogođeno provjerava se mjesto gdje se nalazi to slovo i zatim ide provjera je li riječ pogođena. U slučaju da nije, vraća se na unos slova, a ako je, završava se unos slova. Zadnja provjera slova je kada se pogriješi i u tom slučaja provjera je li to 6 pogreška, ako je onda se završava unos, a ako nije onda se vraća na unos.



Slika 7.4. Dijagram podaktivnosti za unos slova

Kod rezultata se samo gleda je li igrač izgubio ili pobijedio.



Slika 7.5. Dijagram podaktivnosti za rezultat

7.1. Izgled

Da bi se moglo koristiti *tkinter* i *turtle* skupa, koristi se *Canvas* .

```
canvas = tkinter.Canvas(width = 800, height = 800 )
canvas.grid(padx=2, pady=2, row=0, column=0, rowspan=10,
columnspan=10)
```

Za opcije se postavljaju visina i širina jer inače ne bi moglo stati sve što je potrebno za igricu. Grid se koristi samo da bi postavili *canvas* u grafički prikaz. Da bi se kornjača prikazala u *canvasu* potrebna je sljedeća naredba:

```
t = turtle.RawTurtle(canvas)
```

Da bismo obojali pozadinu canvasa koristi se sljedeći algoritam.

```
canvas.configure(bg="LightSalmon")
```

„***Vješalo***“ i „unesi slovo“ koriste se labele i unutar zagrada se uređuje po želji.

```
l1 = Label(window, text="***Vješalo***",font="times 50",
fg="Darkred", bg="LightSalmon")
l1.grid(row=0,column=1)
l2 = Label(window, text="Unesi slovo",font="times 20",
bg="LightSalmon")
l2.grid(row=1,column =1)
```

Za unos slova koristimo textbox.

```
slovo = StringVar()
text= Entry(window, textvariable=slovo, bd=5)
text.grid(row=1,column=2)
```

Za crtanje crtica koje predstavljaju riječ potrebno je prvo definirati niz koji će zadržavati određene riječi. Potom se odabire riječ koja će biti skrivena, pa se tek onda crtaju crtice. Potrebno je postaviti kornjaču na koordinate na kojima želimo da se crtice počnu crtati.

```
rijec= niz[random.randint(0,len(niz))-1]
t.pu()
t.setx(-300)
t.sety(-200)
t.speed(10)
for i in range(len(rijec)+1):
        t.fd(20)
```

```
t.pu()
t.fd(20)
t.pd()
```

Sljedeći algoritam prikazuje kako nacrtati vješalo, ali bez tijela, pomoću kornjače.

```
t.pu()
t.setx(100)
t.sety(-50)
t.pd()
t.fd(100)
t.rt(180)
t.fd(50)
t.rt(90)
t.fd(170)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(50)
```

Također, potreban je botun jer se pomoću njega pozivaju funkcije, nakon što se unese slovo, koje pomažu da igra poštuje sva pravila.

```
button1 = Button(window, text="Pogodi", font='Times 20 bold',
bg='Darkred', fg='white', height=3, width=8, command=
lambda:btnClick())
button1.grid(row=2, column=1)
```

Napravljena je još jedna *labela* koja se ne vidi dok se ne unese prvo slovo i dok se ne pritisne botun.

```
krivo= StringVar()
l3 = Label(window, text="",font="times 20",
textvariable=krivo, bg="LightSalmon")
l3.grid(row=4,column=1)
```

Slika 7.6. Izgled programa

7.2. btnClick

B*tnClick* se poziva kad se pritisne botun i onda ona izvodi sljedeće algoritme.Prvo se dohvaća slovo iz *textboxa* pomoću *get* naredbe.

Potom se provjera nalazili se tekst unutar *textboxa* i ako nije, onda se ispisuje na *labelu* koja je prethodno ostavljena prazna te se postavlja *textbox* da bude prazan.

```
if len(znak) ==0:
        krivo.set("Niste ništa unili ")
        slovo.set("")
```

Sljedeće što se provjerava je li unijeto slovo, odnosno da nije neki znak ili više slova odjednom.

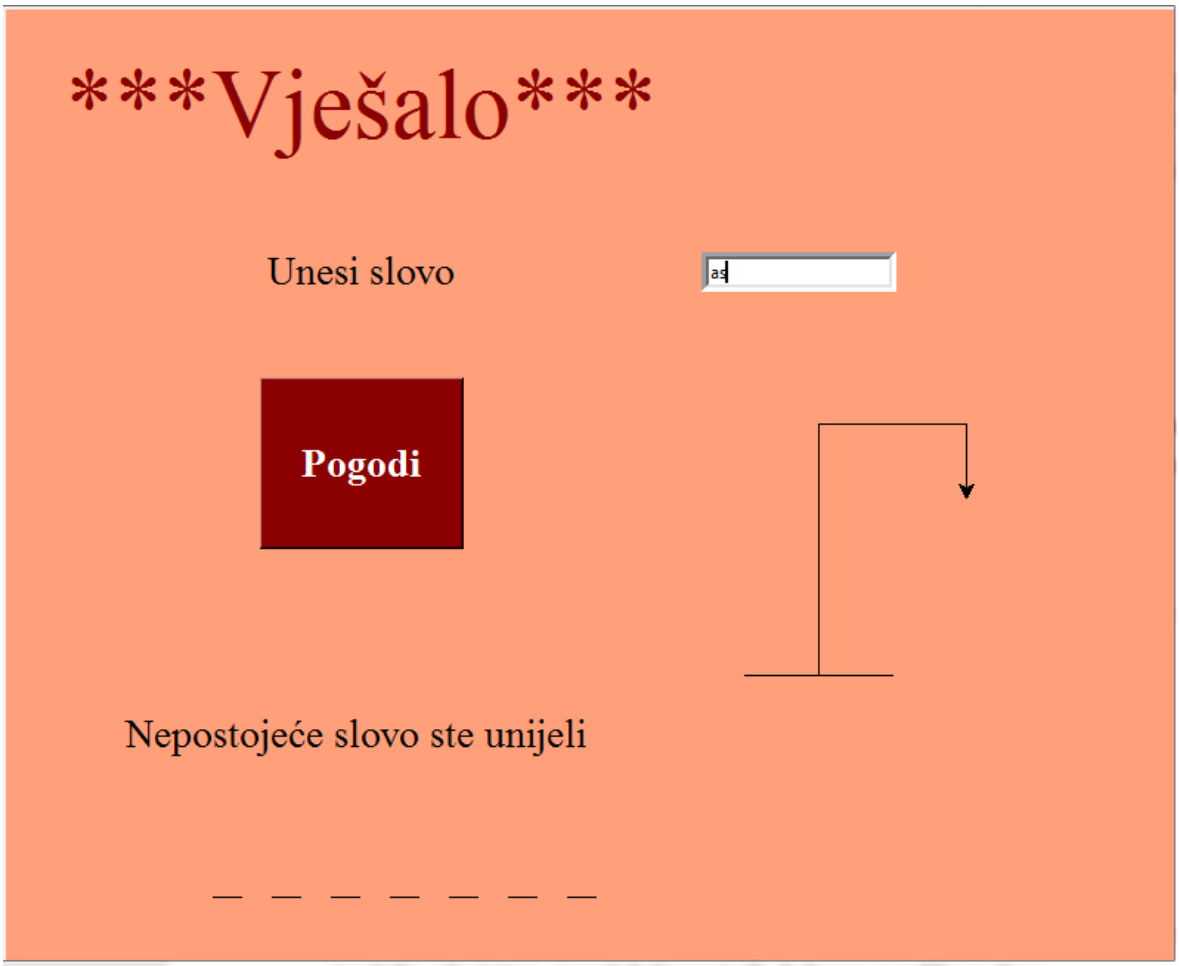
```
elif len(znak)>1 or znak.islower()==False:
        krivo.set("Nepostojeće slovo ste unijeli ")
        slovo.set("")
```

Nakon toga provjerava se je li slovo već jednom uneseno.

```
elif znak in pokusaji:
        krivo.set(znak +" već ste pokušali sa ovim slovom")
        slovo.set("")
```

Kad se sve ovo provjerilo, tek onda se provjerava nalazi li se slovo u toj riječi i ako se nalazi, poziva se sljedeća funkcija, a to je pogodak.

```
elif znak in rijec:
        pokusaji.append(znak)
        pog.append(znak)
        t.clear()
        krivo.set("Slovo "+znak+" se nalazi")
        pogodak(znak,brojac)
        slovo.set("")
```



Slika 7.7. Pogreška pri unosu

Ako nije niti jedan od uvjeta zadovoljen, onda se smatra da se pogriješilo i da bi se trebao nacrtati sljedeći dio vješala. Ako se pogriješilo šest puta, onda se poziva funkcija kraj i završava se igra.

```
elif slovo.get() not in rijec:
        pokusaji.append(znak)
        krivo.set("slovo " + znak+ " se ne nalazi u riječi")
```

```
brojac= brojac+1
slovo.set("")
crtVješalo(brojac)
if brojac == 6:
    kraj()
```

Slika 7.8. Unijeli ste slovo koje se ne nalazi u riječi

7.3. Funkcija pogodak

Kada se pogodi slovo, prvo se sve briše što je kornjača dosad nacrtala i počinje se ispočetka. Koristi se naredba *tocno* da bi se provjerilo jesu li pogođena sva slova.

```
t.pu()
t.home()
t.setx(-300)
t.sety(-200)
tocno=0
```

Prvo se radi *for* petlja koja provjerava svako slovo i gleda gdje se nalazi pogodak. Ako je pogođeno na crticu se stavlja slovo, a ako nije, stavlja se crtica.

```
for i in range(len(rijec)):
        if rijec[i] in pog:
            t.color("red")
            t.write(rijec[i],font="Times 50")
            t.pu()
            t.fd(50)
            t.color("black")
            tocno+=1
            if tocno == len(rijec):
                gotovo()
        else:
            t.pd()
            t.fd(20)
            t.pu()
            t.fd(20)
```

Slika 7.9. Slovo je pogođeno

Kako je prethodno sve obrisano, potrebno je nacrtati vješalo i potrebno je pozvati funkciju za svaki put kada je bila greška. Za svaku pogrešku koristi se *for* petlja za crtanje svakog dijela tijela.

```
t.pu()
t.setx(100)
t.sety(-50)
t.pd()
t.fd(100)
t.rt(180)
t.fd(50)
t.rt(90)
t.fd(170)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(50)
for i in range(br):
    crtVjesalo(i+1)
```

7.4. Funkcija crtVjesalo

Ova funkcija se koristi kada se unese pogrešno slovo da se nacrta sljedeći dio tijela. U prvoj greški se crta glava, pa se crta tijelo, lijeva noga, desna noga, lijeva ruka i na kraju desna ruka. Prima jedan parametar, a to je koja se greška po redu nalazi.

```
if incorrect == 1:
    t.pu()
    t.setx(230)
    t.sety(50)
    t.pd()
    t.circle(20)
    t.ht()
  if incorrect == 2:
    t.pu()
    t.setx(250)
    t.sety(30)
    t.pd()
    t.fd(50)
    t.ht()
  elif incorrect == 3:
    t.rt(45)
    t.fd(30)
    t.ht()
  elif incorrect == 4:
    t.pu()
    t.bk(30)
    t.lt(90)
    t.pd()
    t.fd(30)
    t.ht()
  elif incorrect == 5:
    t.pu()
    t.bk(30)
    t.rt(45)
    t.bk(40)
    t.pd()
    t.rt(45)
    t.fd(30)
    t.ht()
```

```
elif incorrect == 6:
  t.pu()
  t.bk(30)
  t.lt(90)
  t.pd()
  t.fd(30)
  t.pu()
  t.ht()
```

Slika 7.10. Crtanje vješala

7.5. Funkcije gotovo i kraj

Funkcije *gotovo* i *kraj* su dvije funkcije koje se koriste kada je kraj igre. Funkcija *gotovo* se koristi kada su sva slova pogođena, a *kraj* kad je šest puta pogriješeno. U oba slučaja na kraju se stavlja botun koji onemogućava daljnje igranje.

```
def gotovo():
    krivo.set("Winner")
    button1.config(state=DISABLED)
def kraj():
    krivo.set("Game over")
    button1.config(state=DISABLED)
```

8. ZAKLJUČAK

U današnje vrijeme sve od igrica pa sve do raznih animacija napravljeno je pomoću računalne grafike. Svakim danom računalna grafika se unaprjeđuje te dolazi novi i bolji programi za razvoj grafike. Cilj ovoga rada je bio prikazati osnove grafike u Pythonu preko igrice „Vješalo“. Za izradu projekta su se koristili moduli *tkinter* i kornjačina grafika.

Kornjačina grafika je opisana u ovom radu jer profesori informatike sve više koriste ovu grafiku za rad s djecom jer je djeci lakše savladavati gradivo i zanimljivije programirati preko njega. Razlog tome su vizualni rezultati koji se ne nalaze u običnim Pythonu.

Za rad na ova dva modula je potrebno znati osnovne funkcije za izračunavanje, kako funkcionira *string* i *integer* i kako se koristiti petljama. Sve ovo se možete naučiti bez većih problema bez prethodnog znanja.

Za *tkinter* se može reći da također nije kompliciran jer je većinu stvari potrebno pozvati samo naredbom. *Graphics.py* je dosta sličan kornjači, ali jedina mu je mana što je potrebno instalirati dodatak, što je nekima kompliciranо i nepotrebno te je zbog toga bolje koristiti kornjačinu grafiku. Iako su u radu nabrojana samo tri modula, postoje još mnogi drugi moduli koji se mogu lako skinuti i instalirati.

POPIS FUNKCIJA

Funkcija	Opis funkcije
Canvas()	Canvas se koristi kako bi se povezale kornjača i tkinter.
grid()	Grid se koristi da pokrene prozor u kojem kombiniraju kornjača i tkinter. Koristi se još i za pozivanje labela, textboxa i botuna.
Turtle.RawTurtle()	Da bi se kornjača prikazala u canvasu potrebna je ta naredba.
Canvas.configure()	Ova naredba se koristi za bojanje pozadine.
Label()	Label stvara novu labelu.
Entry()	Stvara textbox.
pu() i pd()	Pu() se koristi kako bi se moglo crtati pomoću kornjače. Pd() se koristi kako bi kornjača prestala crtati.
Setx() i sety()	Naredba se koristila u programu kako bi se kornjača postavila na točnu poziciju koordinatnom sustavu gdje bi se započelo crtanje.
Speed()	Naredba se koristi kako bi kornjača brže ili sporije crtala
Fd() i bk()	Naredbe se koriste kako bi se kornjača kretala put naprijed ili nazad.
Rt() i lt()	Naredbe se koriste kako bi se kornjača rotirala u desno ili lijevo.
Button()	Naredba se koristi kako bi se napravio botun
Home()	Kornjača se vraća u središte koordinatnog sustava.
Circle()	Koristi se za crtanje kruga u ovome primjeru kosrti se za crtanje glave.

POPIS SLIKA

POPIS TABLICA

LITERATURA

1) https://www.enciklopedija.hr/natuknica.aspx?ID=68673
2) https://bs.wikipedia.org/wiki/Sketchpad
3) Leo Budin, Predrag Brođanac, Zlato Markučič, Smiljana Perić, Dejan Škvorc, Magdalena Babić: Računalno razmišljanje i programiranje u Pythonu;2017
4) http://ipaq.petagimnazija.hr/wp-content/uploads/2014/12/Kornjacina-grafika.pdf
5) https://docs.python.org/3/library/tkinter.html