Dokumentacija svih važnijih dijelova koda igre „Zmajko“

```
class Button_Slika():
    def __init__(self, x, y, image, scale):
        width = image.get_width()
        height = image.get_height()
        self.image = pygame.transform.scale(image, (int(width * scale),
int(height * scale)))
        self.rect = self.image.get_rect()
        self.rect.topleft = (x, y)
        self.clicked = False

    def draw(self, surface):
        action = False
        pos = pygame.mouse.get_pos()

        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
                self.clicked = True
                action = True

        if pygame.mouse.get_pressed()[0] == 0:
            self.clicked = False
        surface.blit(self.image, (self.rect.x, self.rect.y))
        return action
```

Klasa Button_Slika() služi za proizvodnju gumba koji je ujedno slika. Sadrži dvije funkcije. Funkcija __init__() prikuplja sve potrebne podatke o gumbu kao što su koordinate na kojima će se gumb nalaziti, slika koja će se prikazivati i faktor kojim će slika biti uvećana ili umanjena. Druga funkcija je draw() ona crta gumb na zadanim koordinatama te provjerava je li gumb pritisnut ili nije.

```
class Button:
    def __init__(self, text_input, text_size, text_color,
rectangle_width_and_height, rectangle_color, rectangle_hovering_color,
position):
        self.text_input = text_input
        #rectangle ispod teksta
        self.rectangle = pygame.Rect((position[0]-
(rectangle_width_and_height[0]/2), position[1]-
(rectangle_width_and_height[1]/2)), rectangle_width_and_height)
        self.rectangle_color, self.rectangle_hovering_color = rectangle_color,
rectangle_hovering_color
        #tekst u gumbu
```

```
        self.font = pygame.font.Font(None, text_size)
        self.text_surface = self.font.render(text_input, False, text_color)
        self.text_rectangle = self.text_surface.get_rect(center =
self.rectangle.center)
    def update(self, screen):
        pygame.draw.rect(screen, self.rectangle_color, self.rectangle)
        screen.blit(self.text_surface, self.text_rectangle)
    def checkForCollision(self, mouse_position):
        if mouse_position[0] in range(self.rectangle.left,
self.rectangle.right) and mouse_position[1] in range(self.rectangle.top,
self.rectangle.bottom):
            return True
        return False
    def changeButtonColor(self):
        self.rectangle_color = self.rectangle_hovering_color
    def changeTextInput(self, new_text):
        self.text_input = new_text
        self.text_surface = self.font.render(self.text_input, False, (255,
255, 255))
        self.text_rectangle =
self.text_surface.get_rect(center=self.rectangle.center)
```

Klasa Button služi za proizvodnju gumba koji nema sliku već prikazuje tekst. Funkcija __init__() prikuplja podatke o gumbu kao što su tekst koji će se prikazivati, veličina teksta, boja teksta, širina i visina gumba, boja gumba te boja gumba kada se preko njega prijeđe i pozicija gumba na ekranu. Funkcija update() kao argument prima mjesto gdje će se gumb prikazivati te prikazuje gumb na dobivenom mjestu. Funkcija checkForCollision() provjerava nalazi li se pokazivač miša na gumbu. Funckija changeButtonColor() mijenja boju gumba. Funkcija changeTextInput mijenja tekst koji gumb prikazuje.

```
def draw_text(text,font,text_col,x,y):
        img = font.render(text,True,text_col)
        screen.blit(img,(x,y))
```

Funkcija draw_text() kao argumente prima tekst, font teksta, boju teksta i koordinate na kojima će se tekst prikazivati, a svrha funkcije je da se željeni tekst može prikazati bilo gdje na ekranu.

```
def main():
    global screen, text_font, text_font2, run
    pygame.init()

    #font i tekst
    text_font = pygame.font.SysFont("Arial",50)
    text_font2 = pygame.font.SysFont("Arial",80)
    screen = None
```

```
    os.environ["SDL_VIDEO_CENTERED"]="1"
    info=pygame.display.Info()
    screen_width, screen_height=info.current_w,info.current_h
    screen = pygame.display.set_mode((screen_width-
int(0.005*screen_width),screen_height-int(0.06*screen_height)),
pygame.FULLSCREEN)
    pygame.display.set_caption("Zmajevi")

    run = False
    while not run:
        run = main_menu()
```

Funkcija main() inicijalizira Pygame, postavlja prozor za igru s određenim parametrima i pokreće glavnu petlju koja će pozivati funkciju main_menu() dok se varijabla run ne postavi na True.

```
def main_menu():
 ...
while run == True:
        screen.blit(zmajko_pozadina, (0,0))
        MENU_MOUSE_POS = pygame.mouse.get_pos()

        #gumbovi na desnoj strani
        IGRAJ_GUMB = Button("Kampanja", 70, "Black", (screen.get_width()/3.5,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.8,
screen.get_height()*0.3))
        LEVEL_GUMB = Button("Nemoguć level", 70, "Black",
(screen.get_width()/3.5, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.8, screen.get_height()*0.45))
        BOZIC_GUMB = Button("Božićni level", 70, "Black",
(screen.get_width()/3.5, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.8, screen.get_height()*0.6))
        ACHIEVEMENTS_GUMB = Button("Postignuća", 70, "Black",
(screen.get_width()/3.5, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.8, screen.get_height()*0.75))
        QUIT_BUTTON = Button("Izađi", 70, "Black", (screen.get_width()/3.5,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.8,
screen.get_height()*0.9))
        USERS_BUTTON = Button("Promijeni igrača", 50, "Blue",
(screen.get_width()/5.5, screen.get_height()/20), "#AE7A60", "darksalmon",
(screen.get_width()*0.172, screen.get_height()/3.2))
        TUTORIAL_BUTTON = Button("Kako igrati?", 50, "Black",
(screen.get_width()/5.5, screen.get_height()/20), "#AE7A60", "darksalmon",
(screen.get_width()*0.45, screen.get_height()/2))
        for gumb in [IGRAJ_GUMB, QUIT_BUTTON, LEVEL_GUMB, ACHIEVEMENTS_GUMB,
USERS_BUTTON, BOZIC_GUMB, TUTORIAL_BUTTON]:
            if gumb.checkForCollision(MENU_MOUSE_POS):
```

```
                gumb.changeButtonColor()
            gumb.update(screen)
...
```

Funkcija main_menu() služi za proizvodnju glavnog izbornika igre. U izborniku su prisutni različiti gumbi koji imaju neke svoje funkcije(game menu, achievement menu, božićni level, nemoguć level i izlaz iz igrice), mogućnost za promjenu korisnika, odabir skina za zmaja.

```
def brisanje_usera():
    global ime,
level_state,postignuce1,postignuce2,postignuce3,postignuce4,postignuce5,postig
nuce6,uništeniProtivnici
    ime =""
    level_state = "Level 1-1"
    postignuce1 = "ne"
    postignuce2 = "ne"
    postignuce3 = "ne"
    postignuce4 = "ne"
    postignuce5 = "ne"
    postignuce6 = "ne"
    uništeniProtivnici = "0"
    spremi_igru()
```

Funkcija brisanje_usera() služi za vraćanje svih bitnih varijabli povezanih s korisnikom na početne te se pokreće funkcija spremi_igru(). Podaci o igraču spremaju se u txt file „igraci“ prikazan na slici ispod.

```
1//Level 1-1/ne/ne/ne/ne/ne/ne/0
2//Level 1-1/ne/ne/ne/ne/ne/ne/0
3//Level 1-1/ne/ne/ne/ne/ne/ne/0
4//Level 1-1/ne/ne/ne/ne/ne/ne/0
5//Level 1-1/ne/ne/ne/ne/ne/ne/0
6//Level 1-1/ne/ne/ne/ne/ne/ne/0
```

```
def spremi_igru():
    global igraci, koji_user,
ime,level_state,postignuce1,postignuce2,postignuce3,postignuce4,postignuce5,po
stignuce6, output, uništeniProtivnici
    igraci[int(koji_user)-1] =
[koji_user,ime,level_state,postignuce1,postignuce2,postignuce3,postignuce4,pos
tignuce5,postignuce6,str(uništeniProtivnici)]
    output = ""
    for i in range(len(igraci)):
        for l in range(len(igraci[i])):
            output += igraci[i][l]+"/"
        output = output[0:len(output)-1]
        output += "\n"
```

```
    with open("igraci.txt","w") as datoteka:
        datoteka.write(output)
```

Funkcija spremi_igru() služi za pohranjivanje podataka o igračima u txt file „igraci“. U kodu postoji lista nazvani igraci unutar koje se nalazi 6 lista za svakog igrača po jedna i svaka ta lista sadrži podatke o igraču ispisane iz txt filea odvojenih po „/“. Dakle na početku funkcije se u toj listi mijenja lista za određenog igrača. Nakon toga se kreira privremena varijabla u koju se kao string spremaju svi važni podaci o igraču, a to su redom redni broj igrača, ime, level na koji je igrač stigao, da ili ne ovisno o statusu dobivenog postignuća i broj uništenih protivnika. Svi se ti podaci zapisuju da izgledaju kao u već prikazanom txt fileu tj. da podaci o jednom igraču budu odvojeni s „/“, a podaci o svim igračima odvojeni s „\n“. I na samom kraju funckije se taj string ispisuje u txt.file „igraci“.

```
def user_birac():
...
      while run == True:

        USER1_BUTTON = Button(f"{igraci[0][1]}({igraci[0][2]})", 70, "Light
Grey", (screen.get_width()*0.4, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.3, screen.get_height()*0.25))
        USER1_DELETE = Button(f"x", 70, "Light Grey", (screen.get_width()*0.3,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.7,
screen.get_height()*0.25))
        USER2_BUTTON = Button(f"{igraci[1][1]}({igraci[1][2]})", 70, "Light
Grey", (screen.get_width()*0.4, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.3, screen.get_height()*0.37))
        USER2_DELETE = Button(f"x", 70, "Light Grey", (screen.get_width()*0.3,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.7,
screen.get_height()*0.37))
        USER3_BUTTON = Button(f"{igraci[2][1]}({igraci[2][2]})", 70, "Light
Grey", (screen.get_width()*0.4, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.3, screen.get_height()*0.49))
        USER3_DELETE = Button(f"x", 70, "Light Grey", (screen.get_width()*0.3,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.7,
screen.get_height()*0.49))
        USER4_BUTTON = Button(f"{igraci[3][1]}({igraci[3][2]})", 70, "Light
Grey", (screen.get_width()*0.4, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.3, screen.get_height()*0.61))
        USER4_DELETE = Button(f"x", 70, "Light Grey", (screen.get_width()*0.3,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.7,
screen.get_height()*0.61))
        USER5_BUTTON = Button(f"{igraci[4][1]}({igraci[4][2]})", 70, "Light
Grey", (screen.get_width()*0.4, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.3, screen.get_height()*0.73))
        USER5_DELETE = Button(f"x", 70, "Light Grey", (screen.get_width()*0.3,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.7,
screen.get_height()*0.73))
```

```
        USER6_BUTTON = Button(f"{igraci[5][1]}({igraci[5][2]})", 70, "Light
Grey", (screen.get_width()*0.4, screen.get_height()/9), "#AE7A60", "#715040",
(screen.get_width()*0.3, screen.get_height()*0.85))
        USER6_DELETE = Button(f"x", 70, "Light Grey", (screen.get_width()*0.3,
screen.get_height()/9), "#AE7A60", "#715040", (screen.get_width()*0.7,
screen.get_height()*0.85))

...
```

Funkcija user_birac() služi za proizvodnju user menua, unutar funkcije kreiraju se gumbi kojima se spremaju user slotovi i brišu ti useri slotovi, otvara se txt datoteka „igraci“ i kreira se lista nazvana igraci koja ima 6 članova, a svaki je član posebna lista čiji su članovi varijable koje su vezane uz igrača, a to su redni broj igrača, ime, level na koji je igrač stigao, da ili ne ovisno o statusu dobivenog postignuća i broj uništenih protivnika. Kada se pritisne određeni user slot poziva se funkcija load_igru()

```
def load_igru():
    global koji_user, ime,
level_state,postignuce1,postignuce2,postignuce3,postignuce4,postignuce5,postig
nuce6,igraci, uništeniProtivnici
    if igraci[int(koji_user)-1][1] == "":
        player_name()
    else:
        level_state = igraci[int(koji_user)-1][2]
        postignuce1 = igraci[int(koji_user)-1][3]
        postignuce2 = igraci[int(koji_user)-1][4]
        postignuce3 = igraci[int(koji_user)-1][5]
        postignuce4 = igraci[int(koji_user)-1][6]
        postignuce5 = igraci[int(koji_user)-1][7]
        postignuce6 = igraci[int(koji_user)-1][8]
        uništeniProtivnici = int(igraci[int(koji_user)-1][9])
        ime = igraci[int(koji_user)-1][1]
        main_menu()
```

Funkcija load_igru() poziva se kako bi se svakom igraču pripisale pripadne varijable iz txt filea u kojem su pohranjene. Ukoliko user na pritisnutom user slotu nema pohranjeno ime pokreće se funkcija player_name(). Ako je već pohranjeno ime na pritisnutom user slotu onda se iz txt filea pripisuje pohranjene varijable igraču tako što se izmjenjuju članovi njegove liste unutar liste igraci te se nakon toga pokreće glavni izbornik tj. main_menu().

```
def player_name():
...
 if event.type == pygame.MOUSEBUTTONDOWN:
                if POTVRDA_BUTTON.checkForCollision(MENU_MOUSE_POS) and
len(ime) > 1:
                    with open("igraci.txt", "r") as datoteka:
                        a = datoteka.read()
```

```
                    igraci = a.split("\n")
                    for i, clan in enumerate(igraci):
                        igraci[i] = clan.split("/")
                    igraci[int(koji_user)-1][1] = ime
                    level_state = igraci[int(koji_user)-1][2]
                    postignuce1 = igraci[int(koji_user)-1][3]
                    postignuce2 = igraci[int(koji_user)-1][4]
                    postignuce3 = igraci[int(koji_user)-1][5]
                    postignuce4 = igraci[int(koji_user)-1][6]
                    postignuce5 = igraci[int(koji_user)-1][7]
                    postignuce6 = igraci[int(koji_user)-1][8]
                    uništeniProtivnici = int(igraci[int(koji_user)-1][9])
                    run = False
                    main_menu()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_BACKSPACE:
                    ime = ime[:-1]
                else:
                    ime += event.unicode
...
```

Funkcija player_name() kreira poseban ekran na kojem se prikazuje tekst „Unesi ime“ ,gumb potvrde unosa imena i text box unutar kojeg se ime može unositi. Kada se klikne gumb „Potvrdi“ i unese ime dulje od jednog znaka, ažuriraju se informacije o igraču, a zatim se izlazi iz petlje i prelazi se na glavni izbornik main_menu(). Pirlikom ažuriranja informacija o igraču otvara se txt file „igraci“ i iz njega se iščitavaju varijable povezane za igrača na pritisnutom user slotu te se spremaju u varijable.

```
def postignuca():
...
if postignuce1 == "da":
        tekst1 = "Neprobojni Štit: Preživi neprekidni napad neprijatelja bez
gubitaka zdravlja."
    if postignuce2 == "da":
        tekst2 ="Branič Svemira: Uništi 100 protivničkih trupa."
    if postignuce3 == "da":
        tekst3="Gospodar Svjetova: Prođi sve svjetove i postani nepobjedivi
letač."
    if postignuce4 == "da":
        tekst4="HoHoHo: Prođi božićni level."
    if postignuce5 == "da":
        tekst5="Apsolutni Pobjednik: Prođi nemogući svijet."
    if postignuce6 == "da":
        tekst6="Zmajkova Legenda: Otključaj sva postignuća da postaneš legenda
poput Zmajka. "
...
```

Funkcija postignuca() kreira novi prozor unutar kojeg se prikazuju sva postignuća koja igrač ima otključana. Provjerava se je li vrijednost varijable uz povezano postignuće jednako „da“ ,ukoliko je na ekranu se pomoću funkcije draw_text() ispisuje tekst za određeno postignuće.

```
def tutorial():
...
tutorial_tekst = ["W - gore", "A - lijevo", "S - dolje", "D - desno", "SPACE - pucanje", "Lijevi klik/Desni klik - mijenjanje powerupova", "P/ESC - pauza"]
...
```

Funkcija tutorial() služi za kreiranje prozor unutar kojeg se prikazuje kratak tutorial za upravljanje igrom tj. prikazuju se određene komande kojima se upravlja unutar igre.

```
def pause_menu():
...
while paused:
        MENU_MOUSE_POS = pygame.mouse.get_pos()

        pauzirano_logo = pygame.transform.scale(pygame.image.load("pauzirano_logo.png"), (screen.get_width()/3, screen.get_height()/5))

        PLAY_BUTTON = Button("Nastavi", 70, "White", (220, 120), "Light Grey", "Green", (screen.get_width()/2, screen.get_height()/2))
        QUIT_BUTTON = Button("Izađi", 70, "White", (220, 120), "Light Grey", "Red", (screen.get_width()/2, screen.get_height()/1.5))

        screen.blit(pauzirano_logo, (screen.get_width()/3,screen.get_height()/10))
...
```

Funkcija pause_menu() služi za mogućnost pauziranja igre. Ukoliko se pritisne tipka „P“ ili „ESC“ igra će se pauzirati, prikazati će se tekst koji će igraču dati do znanja da je igra pauzirana i pojavit će se gumb za izlazak iz igre i za nastavak igre.

```
def game_over():
...
while run == True:
        MENU_MOUSE_POS = pygame.mouse.get_pos()
        PLAY_BUTTON = Button("Igraj ponovno", 70, "White", (360, 120), "Light Grey", "Green", (screen.get_width()/2, screen.get_height()/2))
        MAIN_BUTTON = Button("Glavni izbornik", 70, "White", (360, 120), "Light Grey", "dimgray", (screen.get_width()/2, screen.get_height()/1.5))
        QUIT_BUTTON = Button("Izađi", 70, "White", (360, 120), "Light Grey", "Red", (screen.get_width()/2, screen.get_height()/1.2))
```

```
        kraj_logo = pygame.transform.scale(pygame.image.load("kraj_logo.png"),
(screen.get_width()/3, screen.get_height()/5))
        screen.blit(kraj_logo, (screen.get_width()/3,screen.get_height()/10))
...
```

Funkcija game_over() služi prikaz opcija nakon izgubljene igre. Ponude se tri opcije kada igra završi: povratak na glavni izbornik, ponovno pokretanje levela i izlazaka iz igrice.

```
def promijeni_level():
...
if (level_state == "Level 1-1" and replay_state == False) or (odabrani_level
== "Level 1-1" and replay_state == True):
        pozadina = pygame.transform.scale(pygame.image.load("world1_bg.jpg"),
(screen.get_width(), screen.get_height()))
        brojPtica = 20
        brzinaStvaranja = 2.5
        final_vrijeme = 20
        avioni_state=False
        meteori_state = False
        vjetar_state = False
        vanzemaljac_state = False
...
```

Funkcija promijeni_level() služi za promjenu varijabla povezanih uz level. Ovisno o varijabli level_state koja govori na kojem je igrač levelu i replay_state koji govori je li igrač već prošao level, varijable povezane za level se postavljaju u skladu s tim levelom.

```
def impossibleLevel():
    global brojPtica, brzinaStvaranja, final_vrijeme, avioni_state,
meteori_state, vjetar_state, brojAviona, brojMeteora, brojVjetra,
vanzemaljac_state, brojVanzemaljca, pozadina, impossible_state
    pozadina = pygame.transform.scale(pygame.image.load("world5_bg.jpg"),
(screen.get_width(), screen.get_height()))
    brojPtica = 20
    brzinaStvaranja = 1.5
    final_vrijeme = 20
    avioni_state=True
    meteori_state = True
    vjetar_state = True
    vanzemaljac_state = True
    brojAviona = 5
    brojMeteora = 5
    brojVjetra = 1
    brojVanzemaljca = 5
    igra()
```

Funkcija impossibleLevel() služi za postavljanje varijabla koje su potrebne za nemoguć level, a na kraju se poziva sama funkcija igra() koja zapravo kreira level.

```
def bozicLevel():
    global brojPtica, brzinaStvaranja, final_vrijeme, avioni_state,
meteori_state, vjetar_state, brojAviona, brojMeteora, brojVjetra,
vanzemaljac_state, brojVanzemaljca, pozadina, impossible_state
    pozadina = pygame.transform.scale(pygame.image.load("world5_bg.jpg"),
(screen.get_width(), screen.get_height()))
    brojPtica = 15
    brzinaStvaranja = 1.5
    final_vrijeme = 20
    avioni_state=True
    meteori_state = True
    vjetar_state = True
    vanzemaljac_state = True
    brojAviona = 3
    brojMeteora = 3
    brojVjetra = 1
    brojVanzemaljca = 2
    igra()
```

Funkcija bozicLevel() služi za postavljanje varijabla koje su potrebne za božićni level, a na kraju se poziva sama funkcija igra() koja zapravo kreira level.

```
def level_menu():
...
 LEVEL1_GUMB = Button("1-1", 70, "White", (120, 120), "Light Grey", "Green",
(screen.get_width()*0.312, screen.get_height()*0.73))
...
if LEVEL1_GUMB.text_input[-1] <= level_state[-1]:
                            odabrani_level = "Level "+LEVEL1_GUMB.text_input
                            replay_state = True
                        else:
                            draw_text(f"Otključaj level
prije.",text_font,(0,0,0),screen.get_width()/3,screen.get_height()/3)
                            pygame.display.update()
                            time.sleep(2)
...
```

Funkcija level_menu() kreira izbornik za odabir levela. Ovisno o level_state varijabli dopušta da se level pokrene. Također, ukoliko se pritisne gumb za neki level do kojeg igrač još nije došao ispisat će mu se poruka da mora proći levele prije tog levela.

```
def igra():
...
```

Funkcija igra() jest najveća funkcija u kodu ove igrice. Ona zapravo kreira svaki level, sve protivnike unutar level, omogućuje svu fiziku u igrici kao što je pokretanje protivnika, igrača, uništavanje protivnika.

```
def stvoriProtivnike():
...
def stvoriAvione():
...
def stvoriMeteore():
...
def stvoriVjetar():
...
def stvoriVanzemaljca():
...
```

Funkcije stvoriProtivnike(), stvoriAvione(), stvoriMeteore(), stvoriVjetar() i stvoriVanzemaljca() služe za stvaranje protivnika koji su prisutni u igrici.

```
def vratiProtivnika(i):
...
def vratiAvion(i):
...
def vratiMeteor(i):
...
def vratiVjetar(i):
...
def vratiVanzemaljca(i):
...
```

Funkcije vratiProtivnika(i), vratiAvion(i), vratiMeteor(i), vratiVjetar(i) i vratiVanzemaljca(i) služe stvaranje iluzije da su protivnici uništeni. Dakle svaki level ima definiran broj protivnika npr. bit će 10 protivnika, 10 aviona, 15 meteora, 3 vjetra i 12 vanzemaljaca. Kod svake te funkcije postoji jedna for petlja koja ima raspon do broja određenih protivnika:

```
for i in range(brojAviona):
```

Svaki put kada se uništi neki od protivnika gornje funkcije će primati za argument broj od 0 do zadanog broja protivnika i pomaknuti će „uništenog protivnika“ na vrh ekrana i tako sve dok se for petlja ne završi.