

## Дино Џумп (Dino Jump)

**Дино Џумп (Dino Jump)** је игра у којој играч контролише диносауруса по имену Дино који мора да преживи разне препреке и да избегава своје летеће непријатеље. Дино не сме да се заустави и у асортиману својих помагала он може да скочи као и да се сагне како би избегао опасности. Игра постаје све напетија и бржа како време тече. Циљ игре је преживети што дуже и скупити што већи број поена.



### Код игрице

```
1  import pygame
2  import os
3  import random
4  import sys
5
6  pygame.init()
7  pygame.display.set_caption('DINO JUMP')
8  Icon = pygame.image.load('Assets/Dino/DinoStart.png')
9  pygame.display.set_icon(Icon)
10
11 # GLOBALS
12 SCREEN_HEIGHT = 600
13 SCREEN_WIDTH = 1100
14 SCREEN = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
15
16 RUNNING = [pygame.image.load(os.path.join("Assets/Dino", "DinoRun1.png")),
17             pygame.image.load(os.path.join("Assets/Dino", "DinoRun2.png"))]
18 JUMPING = [pygame.image.load(os.path.join("Assets/Dino", "DinoJump.png"))]
19 DUCKING = [pygame.image.load(os.path.join("Assets/Dino", "DinoDuck1.png")),
20            pygame.image.load(os.path.join("Assets/Dino", "DinoDuck2.png"))]
21
22 SMALL_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus1.png")),
```

```

23     pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus2.png")),
24     pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus3.png"))]
25
26 LARGE_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus1.png")),
27     pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus2.png")),
28     pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus3.png"))]
29
30 PTERODACTYL = [pygame.image.load(os.path.join("Assets/Bird", "Bird1.png")),
31     pygame.image.load(os.path.join("Assets/Bird", "Bird2.png"))]
32
33 CLOUD = pygame.image.load(os.path.join("Assets/Other", "Cloud.png"))
34
35 BACKGROUND = pygame.image.load(os.path.join("Assets/Other", "Track.png"))
36
37
38 # DINOSAUR CLASS
39 class Dinosaur:
40     X_POS = 80
41     Y_POS = 310
42     Y_POS_DUCK = 340
43     JUMP_VEL = 8.5
44
45     def __init__(self):
46         self.duck_img = DUCKING
47         self.run_img = RUNNING
48         self.jump_img = JUMPING
49
50         self.dino_duck = False
51         self.dino_run = True
52         self.dino_jump = False
53
54         self.step_index = 0
55         self.jump_vel = self.JUMP_VEL
56         self.image = self.run_img[0]
57         self.dino_rect = self.image.get_rect()
58         self.dino_rect.x = self.X_POS
59         self.dino_rect.y = self.Y_POS
60
61     # Gets user input
62     def update(self, userInput):
63         if self.dino_duck:
64             self.duck()
65         if self.dino_run:
66             self.run()
67         if self.dino_jump:
68             self.jump()
69
70         if self.step_index >= 10:
71             self.step_index = 0
72
73         if userInput[pygame.K_UP] and not self.dino_jump:
74             self.dino_duck = False
75             self.dino_run = False
76             self.dino_jump = True
77         elif userInput[pygame.K_DOWN] and not self.dino_jump:
78             self.dino_duck = True

```

```

79     self.dino_run = False
80     self.dino_jump = False
81     elif not (self.dino_jump or userInput[pygame.K_DOWN]):
82         self.dino_duck = False
83         self.dino_run = True
84         self.dino_jump = False
85
86     def duck(self):
87         self.image = self.duck_img[self.step_index // 5]
88         self.dino_rect = self.image.get_rect()
89         self.dino_rect.x = self.X_POS
90         self.dino_rect.y = self.Y_POS_DUCK
91         self.step_index += 1
92
93     def run(self):
94         self.image = self.run_img[self.step_index // 5]
95         self.dino_rect = self.image.get_rect()
96         self.dino_rect.x = self.X_POS
97         self.dino_rect.y = self.Y_POS
98         self.step_index += 1
99
100    def jump(self):
101        self.image = self.jump_img[0]
102        if self.dino_jump:
103            self.dino_rect.y -= self.jump_vel * 4
104            self.jump_vel -= 0.8
105        if self.jump_vel < - self.JUMP_VEL:
106            self.dino_jump = False
107            self.jump_vel = self.JUMP_VEL
108
109    def draw(self, SCREEN):
110        SCREEN.blit(self.image, (self.dino_rect.x, self.dino_rect.y))
111
112
113
114 # CLOUD CLASS
115 class Cloud:
116     def __init__(self):
117         self.x = SCREEN_WIDTH + random.randint(800, 1000)
118         self.y = random.randint(50, 100)
119         self.image = CLOUD
120         self.width = self.image.get_width()
121
122     def update(self):
123         self.x -= game_speed
124         if self.x < -self.width:
125             self.x = SCREEN_WIDTH + random.randint(2500, 3000)
126             self.y = random.randint(50, 100)
127
128     def draw(self, SCREEN):
129         SCREEN.blit(self.image, (self.x, self.y))
130
131
132 # OBSTACLE CLASS
133 class Obstacle:
134

```

```

135 def __init__(self, image, type):
136     self.image = image
137     self.type = type
138     self.rect = self.image[self.type].get_rect()
139     self.rect.x = SCREEN_WIDTH
140
141 def update(self):
142     self.rect.x -= game_speed
143     if self.rect.x < -self.rect.width:
144         obstacles.pop()
145
146 def draw(self, SCREEN):
147     SCREEN.blit(self.image[self.type], self.rect)
148
149 # SMALLCACTUS CLASS
150 class SmallCactus(Obstacle):
151     def __init__(self, image):
152         self.type = random.randint(0, 2)
153         super().__init__(image, self.type)
154         self.rect.y = 325
155
156 # LARGECACTUS CLASS
157 class LargeCactus(Obstacle):
158     def __init__(self, image):
159         self.type = random.randint(0, 2)
160         super().__init__(image, self.type)
161         self.rect.y = 300
162
163 # PTERODACTYL CLASS
164 class Pterodactyl(Obstacle):
165     def __init__(self, image):
166         self.type = 0
167         super().__init__(image, self.type)
168         self.rect.y = 260
169         self.index = 0
170
171 def draw(self, SCREEN):
172     if self.index >= 9:
173         self.index = 0
174     SCREEN.blit(self.image[self.index // 5], self.rect)
175     self.index += 1
176
177
178 # MAIN CLASS
179 def main():
180     global game_speed, x_pos_bg, y_pos_bg, points, obstacles
181     run = True
182     clock = pygame.time.Clock()
183     player = Dinosaur()
184     cloud = Cloud()
185     game_speed = 14
186     x_pos_bg = 0
187     y_pos_bg = 380
188     points = 0
189     font = pygame.font.Font('freesansbold.ttf', 20)
190     obstacles = []

```

```

191 death_count = 0
192
193 def commands():
194     text1 = font.render("Commands: Key UP for JUMP", True, (0, 0, 0))
195     textRect1 = text1.get_rect()
196     textRect1.center = (800, 520)
197     text2 = font.render("Key DOWN for DUCK", True, (0, 0, 0))
198     textRect2 = text2.get_rect()
199     textRect2.center = (882, 540)
200     SCREEN.blit(text1, textRect1)
201     SCREEN.blit(text2, textRect2)
202
203 def score():
204     global points, game_speed
205     points += 1
206     if points % 100 == 0:
207         game_speed += 1
208
209     text = font.render("Points: " + str(points), True, (0, 0, 0))
210     textRect = text.get_rect()
211     textRect.center = (1000, 40)
212     SCREEN.blit(text, textRect)
213
214 def background():
215     global x_pos_bg, y_pos_bg
216     image_width = BACKGROUND.get_width()
217     SCREEN.blit(BACKGROUND, (x_pos_bg, y_pos_bg))
218     SCREEN.blit(BACKGROUND, (image_width + x_pos_bg, y_pos_bg))
219     if x_pos_bg <= -image_width:
220         SCREEN.blit(BACKGROUND, (image_width + x_pos_bg, y_pos_bg))
221         x_pos_bg = 0
222         x_pos_bg -= game_speed
223
224 while run:
225     for event in pygame.event.get():
226         if event.type == pygame.QUIT:
227             run = False
228
229     SCREEN.fill((255, 255, 255))
230     userInput = pygame.key.get_pressed()
231
232     player.draw(SCREEN)
233     player.update(userInput)
234
235     if len(obstacles) == 0:
236         if random.randint(0, 2) == 0:
237             obstacles.append(SmallCactus(SMALL_CACTUS))
238         elif random.randint(0, 2) == 1:
239             obstacles.append(LargeCactus(LARGE_CACTUS))
240         elif random.randint(0, 2) == 2:
241             obstacles.append(Pterodactyl(PTERODACTYL))
242
243     for obstacle in obstacles:
244         obstacle.draw(SCREEN)
245         obstacle.update()
246         if player.dino_rect.colliderect(obstacle.rect):

```

```

247     #pygame.draw.rect(SCREEN, (255, 0, 0), player.dino_rect, 2)
248     pygame.time.delay(1000)
249     death_count += 1
250     menu(death_count)
251
252     background()
253
254     cloud.draw(SCREEN)
255     cloud.update()
256
257     score()
258     commands()
259
260     clock.tick(30)
261     pygame.display.update()
262
263
264 def menu(death_count):
265     global points
266     run = True
267     while run:
268         SCREEN.fill((255, 255, 255))
269         font = pygame.font.Font('freesansbold.ttf', 30)
270
271         if death_count == 0:
272             text = font.render("Press any Key to Start!", True, (0, 0, 0))
273         elif death_count > 0:
274             text = font.render("Game Over! Press Any key to Restart", True, (0, 0, 0))
275             score = font.render("Your score: " + str(points), True, (0, 0, 0))
276             scoreRect = score.get_rect()
277             scoreRect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2 + 50)
278             SCREEN.blit(score, scoreRect)
279             textRect = text.get_rect()
280             textRect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
281             SCREEN.blit(text, textRect)
282             SCREEN.blit(RUNNING[0], (SCREEN_WIDTH // 2 - 20, SCREEN_HEIGHT // 2 - 140))
283             pygame.display.update()
284         for event in pygame.event.get():
285             if event.type == pygame.QUIT:
286                 run = False
287                 pygame.quit()
288                 quit()
289                 sys.exit()
290             if event.type == pygame.KEYDOWN:
291                 main()
292
293
294 menu(death_count=0)

```

Почетни део кода и глобалне променљиве

```
1 import pygame
2 import os
3 import random
4 import sys
5
6 pygame.init()
7 pygame.display.set_caption('DINO JUMP')
8 Icon = pygame.image.load('Assets/Dino/DinoStart.png')
9 pygame.display.set_icon(Icon)
10
11 # GLOBALS
12 SCREEN_HEIGHT = 600
13 SCREEN_WIDTH = 1100
14 SCREEN = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
15
16 RUNNING = [pygame.image.load(os.path.join("Assets/Dino", "DinoRun1.png")),
17             pygame.image.load(os.path.join("Assets/Dino", "DinoRun2.png"))]
18 JUMPING = [pygame.image.load(os.path.join("Assets/Dino", "DinoJump.png"))]
19 DUCKING = [pygame.image.load(os.path.join("Assets/Dino", "DinoDuck1.png")),
20            pygame.image.load(os.path.join("Assets/Dino", "DinoDuck2.png"))]
21
22 SMALL_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus1.png")),
23                 pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus2.png")),
24                 pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus3.png"))]
25
26 LARGE_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus1.png")),
27                 pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus2.png")),
28                 pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus3.png"))]
29
30 PTERODACTYL = [pygame.image.load(os.path.join("Assets/Bird", "Bird1.png")),
31                 pygame.image.load(os.path.join("Assets/Bird", "Bird2.png"))]
32
33 CLOUD = pygame.image.load(os.path.join("Assets/Other", "Cloud.png"))
34
35 BACKGROUND = pygame.image.load(os.path.join("Assets/Other", "Track.png"))
```

У почетку потребно нам је да уведемо модуле `pygame` као основу за нашу игрицу, након тога уводимо модуле `os`, `random` и `sys` које ћемо касније користити у нашем коду. Наредбом `pygame.init()` вршимо укључивање рада библиотеке `pygame`, затим позивамо функцију `pygame.display.set_caption('DINO JUMP')` која подешава име нашег прозора на 'Dino Jump'. Након тога позивамо `pygame.display.set_icon(Icon)` која подешава ширину прозора игрице.

Како бисмо избегли понављање велике количине кода увели смо глобалне променљиве у којима смо сместили величину прозора игрице помоћу функције `pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))` као и глобалне променљиве за слике из фолдера **Assets** које користимо за графику наше игрице.

Класа `Dinosaur`

```
1 class Dinosaur:
2     X_POS = 80
3     Y_POS = 310
```

```

4  Y_POS_DUCK = 340
5  JUMP_VEL = 8.5
6
7  def __init__(self):
8      self.duck_img = DUCKING
9      self.run_img = RUNNING
10     self.jump_img = JUMPING
11
12     self.dino_duck = False
13     self.dino_run = True
14     self.dino_jump = False
15
16     self.step_index = 0
17     self.jump_vel = self.JUMP_VEL
18     self.image = self.run_img[0]
19     self.dino_rect = self.image.get_rect()
20     self.dino_rect.x = self.X_POS
21     self.dino_rect.y = self.Y_POS
22
23     # Gets user input
24     def update(self, userInput):
25         if self.dino_duck:
26             self.duck()
27         if self.dino_run:
28             self.run()
29         if self.dino_jump:
30             self.jump()
31
32         if self.step_index >= 10:
33             self.step_index = 0
34
35         if userInput[pygame.K_UP] and not self.dino_jump:
36             self.dino_duck = False
37             self.dino_run = False
38             self.dino_jump = True
39         elif userInput[pygame.K_DOWN] and not self.dino_jump:
40             self.dino_duck = True
41             self.dino_run = False
42             self.dino_jump = False
43         elif not (self.dino_jump or userInput[pygame.K_DOWN]):
44             self.dino_duck = False
45             self.dino_run = True
46             self.dino_jump = False
47
48     def duck(self):
49         self.image = self.duck_img[self.step_index // 5]
50         self.dino_rect = self.image.get_rect()
51         self.dino_rect.x = self.X_POS
52         self.dino_rect.y = self.Y_POS_DUCK
53         self.step_index += 1
54
55     def run(self):
56         self.image = self.run_img[self.step_index // 5]
57         self.dino_rect = self.image.get_rect()
58         self.dino_rect.x = self.X_POS
59         self.dino_rect.y = self.Y_POS

```



```

60     self.step_index += 1
61
62     def jump(self):
63         self.image = self.jump_img[0]
64         if self.dino_jump:
65             self.dino_rect.y -= self.jump_vel * 4
66             self.jump_vel -= 0.8
67         if self.jump_vel < - self.JUMP_VEL:
68             self.dino_jump = False
69             self.jump_vel = self.JUMP_VEL
70
71     def draw(self, SCREEN):
72         SCREEN.blit(self.image, (self.dino_rect.x, self.dino_rect.y))

```

**Dinosaur** класа представља играча односно диносауруса којег играч контролише. Дино може да има 3 стања DUCKING, RUNNING и JUMPING односно Дино може да се у једном тренутку сагне или да стално трчи или да скочи. На почетку ове класе постављамо почетну позицију диносауруса на прозору наше игрице, поцизију Диносауруса када се сагне као и брзину његовог скока. У функцији `init` додељујемо почетна стања, индекс који користимо да бројим кораке диносауруса, слику диносауруса и позицију на екрану.

Функција **update** купи улаз са тастатуре и позива функције за скок, сагињање или трчање. Подешава индекс корака на 0 након сваких 10 корака што нам помаже током анимирања корака. Затим испитујемо стања диносауруса тј. на основу улаза тастуре мењамо стања нашег диносауруса.

Функција **duck** нам помаже да анимирамо сагињање тако што ротира између различитих слика нашег диносауруса. Подешава `x` и `y` позицију диносауруса на екрану притом да је `y` позиција узета из променљиве `Y_POS_DUCK`. Такође повећавамо индекс корака за један.

Функција **run** нам помаже да анимирамо трчање тако што ротира између различитих слика нашег диносауруса. Подешава `x` и `y` позицију диносауруса на екрану притом да је `y` позиција узета из променљиве `Y_POS`. Такође повећавамо индекс корака за један.

Функција **jump** нам помаже да анимирамо скакање уколико је притиснут тастер `KEY_UP` на тастатури смањује `Y` позицију диносауруса чиме ће диносаурус отићи на горе на нашем екрану и док се то дешава такође се смањује и брзина скока нашег диносауруса како би се анимирала гравитација. Када се врати у почетну позицију стање скока се мења `false`.

Функција **draw** `koristi` функцију `blit` која црта диносауруса на екрану тако што узима слику и тренутну позицију.

## Класа Cloud

```

1 class Cloud:
2     def __init__(self):
3         self.x = SCREEN_WIDTH + random.randint(800, 1000)
4         self.y = random.randint(50, 100)
5         self.image = CLOUD
6         self.width = self.image.get_width()
7
8     def update(self):
9         self.x -= game_speed
10        if self.x < -self.width:

```

```

11     self.x = SCREEN_WIDTH + random.randint(2500, 3000)
12     self.y = random.randint(50, 100)
13
14     def draw(self, SCREEN):
15         SCREEN.blit(self.image, (self.x, self.y))

```

Класа **Cloud** нам помаже да анимирамо облак у нашој позадини екрана. Облак се креће са десна на лево и прати глобалну варијаблу брзине игрице како би се анимирао. Чим облак дође до краја екрана са леве стране после насумично одабраног времена може се опет појавити на екрану.

Класа **Obstacle** и њена “деца”

```

1  class Obstacle:
2
3      def __init__(self, image, type):
4          self.image = image
5          self.type = type
6          self.rect = self.image[self.type].get_rect()
7          self.rect.x = SCREEN_WIDTH
8
9      def update(self):
10         self.rect.x -= game_speed
11         if self.rect.x < -self.rect.width:
12             obstacles.pop()
13
14     def draw(self, SCREEN):
15         SCREEN.blit(self.image[self.type], self.rect)
16
17 # SMALLCACTUS CLASS
18 class SmallCactus(Obstacle):
19     def __init__(self, image):
20         self.type = random.randint(0, 2)
21         super().__init__(image, self.type)
22         self.rect.y = 325
23
24 # LARGECACTUS CLASS
25 class LargeCactus(Obstacle):
26     def __init__(self, image):
27         self.type = random.randint(0, 2)
28         super().__init__(image, self.type)
29         self.rect.y = 300
30
31 # PTERODACTYL CLASS
32 class Pterodactyl(Obstacle):
33     def __init__(self, image):
34         self.type = 0
35         super().__init__(image, self.type)
36         self.rect.y = 260
37         self.index = 0
38
39     def draw(self, SCREEN):
40         if self.index >= 9:

```

```
41     self.index = 0
42     SCREEN.blit(self.image[self.index // 5], self.rect)
43     self.index += 1
```

Класа **Obstacle** и њене екстензије нам помажу да анимирамо препреке у насумичним интервалима на нашем екрану у виду мало или великог кактуса и птице која лети ка нашем диносаурусу. Уколико препрека изађе ван граница нашег екрана она се избацује из глобалног низа препрека и ослобађа се место за следећу препреку. Екстензије класе **Obstacle** тј. **SmallCactus** и **LargeCactus** помажу нам да изаберемо између различитих слика кактуса и цртају се на “земљи” и разликују се само у њиховој Y позицији због разлике у величини препреке. Класа **Pterodactyl** црта летећег диносауруса који иде супротно од нашег и користи променљиву **index** како би анимирала лет **Pterodactyl**-а.

## Funkcija Main

```
1  def main():
2      global game_speed, x_pos_bg, y_pos_bg, points, obstacles
3      run = True
4      clock = pygame.time.Clock()
5      player = Dinosaur()
6      cloud = Cloud()
7      game_speed = 14
8      x_pos_bg = 0
9      y_pos_bg = 380
10     points = 0
11     font = pygame.font.Font('freesansbold.ttf', 20)
12     obstacles = []
13     death_count = 0
14
15     def commands():
16         text1 = font.render("Commands: Key UP for JUMP", True, (0, 0, 0))
17         textRect1 = text1.get_rect()
18         textRect1.center = (800, 520)
19         text2 = font.render("Key DOWN for DUCK", True, (0, 0, 0))
20         textRect2 = text2.get_rect()
21         textRect2.center = (882, 540)
22         SCREEN.blit(text1, textRect1)
23         SCREEN.blit(text2, textRect2)
24
25     def score():
26         global points, game_speed
27         points += 1
28         if points % 100 == 0:
29             game_speed += 1
30
31         text = font.render("Points: " + str(points), True, (0, 0, 0))
32         textRect = text.get_rect()
33         textRect.center = (1000, 40)
34         SCREEN.blit(text, textRect)
35
```

```

36 def background():
37     global x_pos_bg, y_pos_bg
38     image_width = BACKGROUND.get_width()
39     SCREEN.blit(BACKGROUND, (x_pos_bg, y_pos_bg))
40     SCREEN.blit(BACKGROUND, (image_width + x_pos_bg, y_pos_bg))
41     if x_pos_bg <= -image_width:
42         SCREEN.blit(BACKGROUND, (image_width + x_pos_bg, y_pos_bg))
43         x_pos_bg = 0
44     x_pos_bg -= game_speed
45
46 while run:
47     for event in pygame.event.get():
48         if event.type == pygame.QUIT:
49             run = False
50
51     SCREEN.fill((255, 255, 255))
52     userInput = pygame.key.get_pressed()
53
54     player.draw(SCREEN)
55     player.update(userInput)
56
57     if len(obstacles) == 0:
58         if random.randint(0, 2) == 0:
59             obstacles.append(SmallCactus(SMALL_CACTUS))
60         elif random.randint(0, 2) == 1:
61             obstacles.append(LargeCactus(LARGE_CACTUS))
62         elif random.randint(0, 2) == 2:
63             obstacles.append(Pterodactyl(PTERODACTYL))
64
65     for obstacle in obstacles:
66         obstacle.draw(SCREEN)
67         obstacle.update()
68         if player.dino_rect.colliderect(obstacle.rect):
69             #pygame.draw.rect(SCREEN, (255, 0, 0), player.dino_rect, 2)
70             pygame.time.delay(1000)
71             death_count += 1
72             menu(death_count)
73
74     background()
75
76     cloud.draw(SCREEN)
77     cloud.update()
78
79     score()
80     commands()
81
82     clock.tick(30)
83     pygame.display.update()

```

Функција **Main** садржи глобалне варијабле за брзину игрице,  $x$  и  $y$  координате позадине као и глобалну варијаблу за освојене поене. Стање трчања је на старту увек подешено на True. Инцијализује се наш диносаурус и класа облак и подешавају се остала потребна стања која се користе у функцијама које су дефинисане унутар класе main.

Функција **commands** исцртава помоћ у виду текста како би играчи знали које команде на тастатури да користе за контролу диносауруса.

Функција **score** користи глобалне променљиве за поене и брзину игрице тако што на сваких 100 освојених поена увећава брзину игрице за један степен. Такође ова функција прати резултат у реалном времену и у свакој итерацији петље наше игрице она приказује тренутни резултат у горњем десном углу екрана.

Функција **background** користи глобалне променљиве за *x* и *y* позицију позадине екрана и исцртава тло по којем се наш диносаурус креће које је уједно и тло где се наше препреке у виду кактуса појављују. Тло се исцртава тако што се прати брзина игре и уколико слика тла изађе са екрана са десне стране тло се опет анимира.

Затим имамо петљу у којој се наша игра извршава и која прати стање трчања нашег диносауруса. Уколико се притисне на дугме за излаз наша петља се прекида. Током сваке итерације наше петље попуњава се позадина екрана у белу боју, прати се улаз са тастатуре и исцртава се наш диносаурус у зависности да ли је притиснута жељена команда на тастатури. Такође позивају се функције које анимирају позадину,облак,освојене поене и исцртавају се помоћне команде на екрану.

Функција **clock.tick()** подешава време наше игрице док **pygame.display.update()** освежава екран наше игрице.

## Funkcija Menu

```
1 def menu(death_count):
2     global points
3     run = True
4     while run:
5         SCREEN.fill((255, 255, 255))
6         font = pygame.font.Font('freesansbold.ttf', 30)
7
8         if death_count == 0:
9             text = font.render("Press any Key to Start!", True, (0, 0, 0))
10        elif death_count > 0:
11            text = font.render("Game Over! Press Any key to Restart", True, (0, 0, 0))
12            score = font.render("Your score: " + str(points), True, (0, 0, 0))
13            scoreRect = score.get_rect()
14            scoreRect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2 + 50)
15            SCREEN.blit(score, scoreRect)
16            textRect = text.get_rect()
17            textRect.center = (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2)
18            SCREEN.blit(text, textRect)
19            SCREEN.blit(RUNNING[0], (SCREEN_WIDTH // 2 - 20, SCREEN_HEIGHT // 2 - 140))
20            pygame.display.update()
21        for event in pygame.event.get():
22            if event.type == pygame.QUIT:
23                run = False
24                pygame.quit()
25                quit()
26                sys.exit()
27            if event.type == pygame.KEYDOWN:
28                main()
29
30
31 menu(death_count=0)
```

Функција **Menu** прави мени наше игрице пре почетак и након завршетка извршења main функције. Такође она и позива main функцију нашег програма. Уколико је игрица први пут покренута исписаће “Press any Key to Start” тј. чекаће да се притисне било која тастатура како би се игра покренула. У супротном уколико се игра завршила исписаће освојене поене као и сличну поруку “Game Over! Press any Key to Restart” где се такође чека улаз са тастатуре како би се игра поново покренула. Да би знали да ли се игра опет покренула користимо варијаблу death\_count која је на почетку игрице подшена на 0 и након сваког завршетка се повећава за један. У сваком тренутку из игрице се може изаћи притиском миша на дугме за излаз на екрану.

На крају наша игрица се покреће позивом функције Menu са варијаблом death\_count која је подешена на 0.

### Слике менија игрице

