

Simulate

Како играти Simulate?

Simulate је копија игре Сајмон. Постоје четири обојена думета на екрану. Дугмад засветле неким одређеним насумичним обрасцем, и онда играч мора да понови овај образац притискајући дугмад у одговарајућем редоследу. Сваки пут када играч успешно симулира образац, образац постаје дужи. Играч покушава да поновља образац што је дуже могуће.

Изворни код за Simulate

Овај код можете преузети на следећој адреси: <http://invpy.com/simulate.py>.

Уколико постоји нека порука о грешци, погледајте која линија кода је поменута у тој поруци и у наведеној линији кода проверите да ли постоји нека грешка у куцању.

Такође, код можете да налепите и у „web“ форми на адреси <http://invpy.com/diff/simulate> и да видите разлику између вашег кода и кода из приручника.

Можете преузети четири звука која су коришћена у овом програму са следећих линкова:

- <http://invpy.com/beep1.ogg>
- <http://invpy.com/beep2.ogg>
- <http://invpy.com/beep3.ogg>
- <http://invpy.com/beep4.ogg>

```
1  # Simulate (a Simon clone)
2  # By Al Sweigart al@inventwithpython.com
3  # http://inventwithpython.com/pygame
4  # Released under a "Simplified BSD" license
5
6  import random, sys, time, pygame
7  from pygame.locals import *
8
9  FPS = 30
10 WINDOWWIDTH = 640
11 WINDOWHEIGHT = 480
12 FLASHSPEED = 500 # u milisekundama
13 FLASHDELAY = 200 # u milisekundama
14 BUTTONSIZE = 200
15 BUTTONGAPSIZE = 20
16 TIMEOUT = 4 # sekunde pre nego što se igra završi ukoliko nijedno dugme nije pritisnuto
17
18 #           R   G   B
19 WHITE      = (255, 255, 255)
20 BLACK      = (  0,   0,   0)
21 BRIGHTRED  = (255,   0,   0)
22 RED        = (155,   0,   0)
23 BRIGHTGREEN = (  0, 255,   0)
24 GREEN      = (  0, 155,   0)
25 BRIGHTBLUE = (  0,   0, 255)
26 BLUE       = (  0,   0, 155)
27 BRIGHTYELLOW = (255, 255,   0)
28 YELLOW     = (155, 155,   0)
29 DARKGRAY   = ( 40,  40,  40)
30 bgColor = BLACK
```

```

31 -
32 XMARGIN = int((WINDOWWIDTH - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
33 YMARGIN = int((WINDOWHEIGHT - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
34
35 # Pravougaonik za svaki od četiri dugmeta
36 YELLOWRECT = pygame.Rect(XMARGIN, YMARGIN, BUTTONSIZE, BUTTONSIZE)
37 BLUERECT = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN, BUTTONSIZE, BUTTONSIZE)
38 REDRECT = pygame.Rect(XMARGIN, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
39 GREENRECT = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE,
40 BUTTONSIZE)
41
42 def main():
43     global FPSLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3, BEEP4
44
45     pygame.init()
46     FPSLOCK = pygame.time.Clock()
47     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
48     pygame.display.set_caption('Simulate')
49
50     BASICFONT = pygame.font.Font('freesansbold.ttf', 16)
51     infoSurf = BASICFONT.render('Ponovi obrazac klikom na dugme ili koristi Q, W, A, S tastere.', 1, DARKGRAY)
52     infoRect = infoSurf.get_rect()
53     infoRect.topleft = (10, WINDOWHEIGHT - 25)
54
55     # učitavanje zvuka
56     BEEP1 = pygame.mixer.Sound('beep1.ogg')
57     BEEP2 = pygame.mixer.Sound('beep2.ogg')
58     BEEP3 = pygame.mixer.Sound('beep3.ogg')
59     BEEP4 = pygame.mixer.Sound('beep4.ogg')
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

90         elif event.key == K_a:
91             clickedButton = RED
92         elif event.key == K_s:
93             clickedButton = GREEN
94
95
96
97     if not waitingForInput:
98         # ponovi obrazac
99         pygame.display.update()
100         pygame.time.wait(1000)
101         pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
102         for button in pattern:
103             flashButtonAnimation(button)
104             pygame.time.wait(FLASHDELAY)
105         waitingForInput = True
106     else:
107         # sačekaj da igrač pritisne dugmad
108         if clickedButton and clickedButton == pattern[currentStep]:
109             # pushed the correct button
110             flashButtonAnimation(clickedButton)
111             currentStep += 1
112             lastClickTime = time.time()
113
114             if currentStep == len(pattern):
115                 # pritisnuto poslednje dugme iz obrazca
116                 changeBackgroundAnimation()
117                 score += 1
118                 waitingForInput = False
119                 currentStep = 0 # vrati se na prvi korak
120
121         elif (clickedButton and clickedButton != pattern[currentStep]) or (currentStep != 0 and time.time() -
122             TIMEOUT > lastClickTime):
123             # pritisnuto pogresno dugme, ili je vreme isteklo
124             gameOverAnimation()
125             # podesi promenljive za novu igru:
126             pattern = []
127             currentStep = 0
128             waitingForInput = False
129             score = 0
130             pygame.time.wait(1000)
131             changeBackgroundAnimation()
132
133     pygame.display.update()
134     FPSLOCK.tick(FPS)
135
136 def terminate():
137     pygame.quit()
138     sys.exit()
139
140
141 def checkForQuit():
142     for event in pygame.event.get(QUIT): # pokupi sve QUIT dogadjaje
143         terminate() # okoncaj sve QUIT dogadjaje koji su prisutni
144     for event in pygame.event.get(KEYUP): # pokupi KEYUP dogadjaje
145         if event.key == K_ESCAPE:
146             terminate() # okoncaj ako je za KEYUP dogodjaj pritisnuto Esc dugme
147     pygame.event.post(event) # vrati ostale KEYUP dogadjaje
148

```

```

149
150 def flashButtonAnimation(color, animationSpeed=50):
151     if color == YELLOW:
152         sound = BEEP1
153         flashColor = BRIGHTYELLOW
154         rectangle = YELLOWRECT
155     elif color == BLUE:
156         sound = BEEP2
157         flashColor = BRIGHTBLUE
158         rectangle = BLUERECT
159     elif color == RED:
160         sound = BEEP3
161         flashColor = BRIGHTRED
162         rectangle = REDRECT
163     elif color == GREEN:
164         sound = BEEP4
165         flashColor = BRIGHTGREEN
166         rectangle = GREENRECT
167
168     origSurf = DISPLAYSURF.copy()
169     flashSurf = pygame.Surface((BUTTONSIZE, BUTTONSIZE))
170     flashSurf = flashSurf.convert_alpha()
171     r, g, b = flashColor
172     sound.play()
173
174     for start, end, step in ((0, 255, 1), (255, 0, -1)): # petlja za animacije
175         for alpha in range(start, end, animationSpeed * step):
176             checkForQuit()
177             DISPLAYSURF.blit(origSurf, (0, 0))
178             flashSurf.fill((r, g, b, alpha))
179             DISPLAYSURF.blit(flashSurf, rectangle.topleft)
180             pygame.display.update()
181             FPSCLOCK.tick(FPS)
182     DISPLAYSURF.blit(origSurf, (0, 0))
183
184 def drawButtons():
185     pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
186     pygame.draw.rect(DISPLAYSURF, BLUE, BLUERECT)
187     pygame.draw.rect(DISPLAYSURF, RED, REDRECT)
188     pygame.draw.rect(DISPLAYSURF, GREEN, GREENRECT)
189
190
191 def changeBackgroundAnimation(animationSpeed=40):
192     global bgColor
193     newBgColor = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
194
195     newBgSurf = pygame.Surface((WINDOWWIDTH, WINDOWHEIGHT))
196     newBgSurf = newBgSurf.convert_alpha()
197     r, g, b = newBgColor
198     for alpha in range(0, 255, animationSpeed): # apetlja za animacije
199         checkForQuit()
200         DISPLAYSURF.fill(bgColor)

```

```

201
202     newBgSurf.fill((r, g, b, alpha))
203     DISPLAYSURF.blit(newBgSurf, (0, 0))
204
205     drawButtons() # ponovo nacrtaj dugmad
206
207     pygame.display.update()
208     FPSCLOCK.tick(FPS)
209     bgColor = newBgColor
210
211
212 def gameOverAnimation(color=WHITE, animationSpeed=50):
213     #пусти sve zvuke odjednom, onda nek pozadina treperi
214     origSurf = DISPLAYSURF.copy()
215     flashSurf = pygame.Surface(DISPLAYSURF.get_size())
216     flashSurf = flashSurf.convert_alpha()
217     BEEP1.play() #пусти sve zvuke odjednom
218     BEEP2.play()
219     BEEP3.play()
220     BEEP4.play()
221     r, g, b = color
222     for i in range(3): # neka blicne tri puta
223         for start, end, step in ((0, 255, 1), (255, 0, -1)):
224             # Prvo ponavljanje u ovoj petlji postavlja sledeću petlju da ide od 0 do 255
225             # a drugu od 255 do 0.
226             for alpha in range(start, end, animationSpeed * step): # petlja animacije
227                 # alpha oznacava providnost, 255 je neprozirno, 0 je prozirno
228
229
230         flashSurf.fill((r, g, b, alpha))
231         DISPLAYSURF.blit(origSurf, (0, 0))
232         DISPLAYSURF.blit(flashSurf, (0, 0))
233         drawButtons()
234         pygame.display.update()
235         FPSCLOCK.tick(FPS)
236
237
238 def getButtonClicked(x, y):
239     if YELLOWRECT.collidepoint( (x, y) ):
240         return YELLOW
241     elif BLUERECT.collidepoint( (x, y) ):
242         return BLUE
243     elif REDRECT.collidepoint( (x, y) ):
244         return RED
245     elif GREENRECT.collidepoint( (x, y) ):
246         return GREEN
247     return None
248
249
250 if __name__ == '__main__':
251     main()

```

Уобичајени почетак

```
1  # Simulate (a Simon clone)
2  # By Al Sweigart al@inventwithpython.com
3  # http://inventwithpython.com/pygame
4  # Released under a "Simplified BSD" license
5
6  import random, sys, time, pygame
7  from pygame.locals import *
8
9  FPS = 30
10 WINDOWWIDTH = 640
11 WINDOWHEIGHT = 480
12 FLASHSPEED = 500 # in milliseconds
13 FLASHDELAY = 200 # in milliseconds
14 BUTTONSIZE = 200
15 BUTTONGAPSIZE = 20
16 TIMEOUT = 4 # seconds before game over if no button is pushed.
17
18 #           R   G   B
19 WHITE      = (255, 255, 255)
20 BLACK      = ( 0,  0,  0)
21 BRIGHTRED  = (255,  0,  0)
22 RED        = (155,  0,  0)
23 BRIGHTGREEN = ( 0, 255,  0)
24 GREEN      = ( 0, 155,  0)
25 BRIGHTBLUE = ( 0,  0, 255)
26 BLUE       = ( 0,  0, 155)
27 BRIGHTYELLOW = (255, 255,  0)
28 YELLOW     = (155, 155,  0)
29 DARKGRAY   = ( 40,  40,  40)
30 bgColor = BLACK
31
32 XMARGIN = int((WINDOWWIDTH - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
33 YMARGIN = int((WINDOWHEIGHT - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
```

Овде подешавамо уобичајене константе за ствари које бисмо касније желели да изменимо као што су величина четири дугмета, нијансе боје које користимо (светлије боје се користе када су дугмад упаљена) и временски интервал који играч има да притисне следеће дугме у низу пре него се игра заврши.

Подешавање дугмади

```
35 # Pravougaonik za svaki od četiri dugmeta
36 YELLOWRECT = pygame.Rect(XMARGIN, YMARGIN, BUTTONSIZE, BUTTONSIZE)
37 BLUERECT    = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN, BUTTONSIZE, BUTTONSIZE)
38 REDRECT     = pygame.Rect(XMARGIN, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
39 GREENRECT   = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
```

Баш као и дугмад у „Sliding Puzzle“ игрици за „Reset“ , „Solve“ и „New Game“, „Simulate“ игрица има четири правоугаоне области и код који руководи када играч кликне унутар неке од ових области. Програму ће бити потребни „Rect“ објекти за четири дугмета па се могу позивати „collidepoint()“ методе за њих. Линије кода од 36 до 39 подешавају ове „Rect“ објекте са одговарајућим координатама и величинама.

The main () функција

```
41 def main():
42     global FPSLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3, BEEP4
43
44     pygame.init()
45     FPSLOCK = pygame.time.Clock()
46     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
47     pygame.display.set_caption('Simulate')
48
49     BASICFONT = pygame.font.Font('freesansbold.ttf', 16)
50     infoSurf = BASICFONT.render('Ponovi obrazac klikom na dugme ili koristi Q, W, A, S tastere.', 1, DARKGRAY)
51     infoRect = infoSurf.get_rect()
52     infoRect.topleft = (10, WINDOWHEIGHT - 25)
53
54     # učitavanje zvuka
55     BEEP1 = pygame.mixer.Sound('beep1.ogg')
56     BEEP2 = pygame.mixer.Sound('beep2.ogg')
57     BEEP3 = pygame.mixer.Sound('beep3.ogg')
58     BEEP4 = pygame.mixer.Sound('beep4.ogg')
```

The main() функција ће имплементирати главни део програма и позвати остале функције када су потребне. Уобичајне „Pygame“ уграђене функције су позване да иницијализују библиотеку, креирају „Clock“ објекат, креирају прозор, поставе наслов, и креирају „Font“ објекте који ће приказивати резултат и инструкције у прозору. Објекти креирани овим функцијама биће смештени у глобалне променљиве тако да могу бити коришћени и у другим функцијама. Али, у суштини они су константе пошто се њихова вредност никада не мења. Линије кода од 55 до 58 учитавају звук тако да „Simulate“ игрица може произвести звук сваки пут када играч кликне на неко дугме. „The pygame.mixer.Sound()“ функција враћа звучни објекат, који ми складиштимо у променљиве од „BEEP1“ до „BEEP4“ које су направљене као глобалне променљиве у 42. линији кода.

Неке локалне променљиве коришћене у овом програму

```
60     # Inicijalizacija promenljivih za igricu
61     pattern = [] # čuva obrasce boja
62     currentStep = 0 # sledeća boja koju igrač mora da pritisne
63     lastClickTime = 0 # vremenska oznaka poslednjeg pritiska dugmeta
64     score = 0
65     # kada je False, obrazac se ponavlja. kada je True, čeka se igrač da pritisne obojeno dugme
66     waitingForInput = False
```

„Pattern“ променљива ће бити листа вредности боја (било YELLOW, RED, BLUE, или GREEN) да се прати образац који би играч требало да запамти. На пример, ако је вредност „pattern“ променљиве [RED, RED, YELLOW, RED, BLUE, BLUE, RED, GREEN] онда играч прво мора да кликне црвено дугме два пута, онда жуто, онда црвено, и тако даље, све док не дође до последњег зеленог дугмета. Сваки пут кад играч заврши једну рунду, нова насумице одабрана боја биће додата на крај листе.

„currentStep“ променљива води рачуна коју следећу боју у обрасцу играч треба да кликне. Ако је „currentStep“ променљива 0 а „pattern“ је [GREEN, RED, RED, YELLOW], онда играч треба да кликне на зелено дугме. Ако кликне на било које друго дугме, доћи ће до завршетка игре.

Постоји „TIMEOUT“ константа која проузрокује да играч у року од неколико секунди треба да кликне на следећу боју, иначе ће доћи до завршетка игре.

Да би проверили да ли је прошло довољно времена од последњег клик, променљива „lastClickTime“ би требало да прати када је играч последњи пут кликнуо на дугме.

Променљива „score“ прати резултат.

Такође, постоје и два режима. Или програм показује образац који играч треба да понови (у том случају „waitingForInput“ је постављена на „False“), или програм чека да играч понови образац у одговарајућем редоследу (у том случају „waitingForInput“ је постављена на „True“).

Цртање табле и

```
68     while True: # glavna petlja
69         clickedButton = None # dugme koje je pritisnuto (YELLOW, RED, GREEN, or BLUE)
70         DISPLAYSURF.fill(bgColor)
71         drawButtons()
72
73         scoreSurf = BASICFONT.render('Score: ' + str(score), 1, WHITE)
74         scoreRect = scoreSurf.get_rect()
75         scoreRect.topleft = (WINDOWWIDTH - 100, 10)
76         DISPLAYSURF.blit(scoreSurf, scoreRect)
77
78         DISPLAYSURF.blit(infoSurf, infoRect)
```

Линија 68 је почетак главне петље. „clickedButton“ биће подешено на „None“ на почетку сваке итерације. Ако је дугме кликнуто у току итерације, онда ће „clickedButton“ бити подешено на једну од боја које одговарају том дугмету (YELLOW, RED, GREEN, или BLUE).

„fill()“ метода позвана у линији 70 префарбаће целу површину екрана тако да можемо да почнемо испочетка. Четири обојена дугмета су нацртана помоћу „drawButtons()“. Текст за резултат је креиран од 73. до 76. линије кода.

Такође, постоји текст који говори играчу који је његов тренутни резултат. За разлику од позива „render()“ методе у линији 51 за текст за инструкције, текст за резултат се мења. Креће од „Score: 0“, па постаје „Score: 1“, и тако даље. Због овог правимо нове објекте позивајући „render()“ методу у линији 73 унутар петље. Како се текст за инструкције („Попови образац кликом ...“) никад не мења, потребан нам је само један позив „render()“ функције изван главне петље у линији 50.

Провера клика мишем

```
80         checkForQuit()
81     for event in pygame.event.get(): # petlja dogadjaja
82         if event.type == MOUSEBUTTONDOWN:
83             mousex, mousey = event.pos
84             clickedButton = getButtonClicked(mousex, mousey)
```

Линија 80 проверава „QUIT“ догађаје, а линија 81 је почетак петље за обраду догађаја. „XY“ координате било ког клика мишем биће сачувана у променљивим „mousex“ и „mousey“. Ако је клик мишем био на неки од четири дугмета, онда „getButtonClicked()“ функција враћа објекат боје дугмета које је кликнуто (у сутротном враћа вредност „None“).

Провера притиска тастера на тастатури

```
85         elif event.type == KEYDOWN:
86             if event.key == K_q:
87                 clickedButton = YELLOW
88             elif event.key == K_w:
89                 clickedButton = BLUE
90             elif event.key == K_a:
91                 clickedButton = RED
92             elif event.key == K_s:
93                 clickedButton = GREEN
```

Линије од 85 до 93 проверавају „KEYDOWN“ догађаје (креиране кад акорисник притисне тастер на тастатури). „Q“, „W“, „A“ и „S“ тастери одговарају дугмадима јер су распоређени у квадратном облику на тастатури. „Q“ је горњи леви тастер од ова четири поменути, баш као што је жуто дугме горе лево на екрану, па ћемо направити да притисак тастера „Q“ буде исто као и клик на жуто дугме. Ово можемо урадити подешавајући да променљива „clickedButton“ има вредност константе „YELLOW“. Исто можемо урадити и за преостала три тастера. На овај начин, корисник може играти „Simulate“ и на тастатури и мишем.

Два режима у петљи игре

```
97         if not waitingForInput:
98             # ponovi obrazac
99             pygame.display.update()
100             pygame.time.wait(1000)
101             pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
102             for button in pattern:
103                 flashButtonAnimation(button)
104                 pygame.time.wait(FLASHDELAY)
105             waitingForInput = True
```

Постоје два различита режима у којима програм може да буде. Када „waitingForInput“ има вредност „False“, програм ће приказивати анимацију за образац. Када „waitingForInput“ има вредност „True“, програм ће чекати корисника да изабере дугме. Како је ово урађено на почетку игре или кад играч заврши образац, линија 101 ће додати произвољну боју у „pattern“ листу да образац продужи за један корак. Линије од 102 до 104 пролазе кроз сваку вредност у листи „pattern“ и позивају „flashButtonAnimation()“ функцију која омогућава да дугме засветли. Након што су сва дугмад засветлела у листи, програм ће променљивој „waitingForInput“ доделити вредност „True“.

Да ли је играч кликну на право дугме?

```
106 |         else:
107 |             # sačekaj da igrač pritisne dugmad
108 |             if clickedButton and clickedButton == pattern[currentStep]:
109 |                 # pushed the correct button
110 |                 flashButtonAnimation(clickedButton)
111 |                 currentStep += 1
112 |                 lastClickTime = time.time()
```

Ако „waitingForInput“ има вредност „True“ онда ће се код у 106. линји извршити. Линија 108 проверава да ли је играч кликнуо на дугме у току итерације и да ли је то право дугме. Променљива „currentStep“ чува податак о индексу дугмета у листи „pattern“ које играч треба следеће да притисне.

Ако је играч притиснуо одговарајуће дугме, желимо да оно засветли позивајући функцију „flashButtonAnimation()“, повећати „currentStep“ за следећи корак, и онда ажурирати променљиву „lastClickTime“ на тренутно време.

```
114 |         if currentStep == len(pattern):
115 |             # pritisnuto poslednje dugme iz obrazca
116 |             changeBackgroundAnimation()
117 |             score += 1
118 |             waitingForInput = False
119 |             currentStep = 0 # vrati se na prvi korak
```

Линије од 114 до 119 су унутар „else“ исказа који почиње у линији 106. Ако је извршење унутар тог „else“ исказа, знамо да је играч кликнуо на право дугме. Линија 114 проверава да ли је ово последње одговарајуће дугме у образцу проверавајући да ли је број сачуван у „currentStep“ једнак броју вредности у листи „pattern“. Ако је то „True“, онда желимо да променимо боју позадине позивајући функцију „changeBackgroundAnimation()“. Ово је једноставан начин да играч сазна да је поновио образац тачно. Резултат се повећава, „currentStep“ се враћа на 0, а променљива „waitingForInput“ сада има вредност „False“ да би у следећој итерацији петље код додао нову „Color“ вредност у листу „pattern“.

```
121 |         elif (clickedButton and clickedButton != pattern[currentStep]) or (currentStep != 0 and time.time() - lastClickTime >
```

Уколико играч није кликнуо на одговарајуће дугме, „elif“ исказ у линији 121 покрива случај где је играч кликнуо на погрешно дугме или је играч чекао превише дуго да кликне на дугме. Како год, треба да покажемо „game over“ анимацију и да започнемо нову игру. Овај „(clickedButton and clickedButton != pattern[currentStep])“ део „elif“ услова проверава да ли је дугме притиснуто и да ли је то било погрешно дугме. Можеш да упоредиш ово са 108. линијом „if“ услова „clickedButton and clickedButton == pattern[currentStep]“ која добија вредност „True“ ако је играч кликнуо дугме и ако је то било одговарајуће дугме.

Други део 121. линије кода „elif“ исказа је „(currentStep != 0 and time.time() - TIMEOUT > lastClickTime)“. Овим се проверава да играчу није истекло време за игру. Примети, да овај део услова има два услова повезана речју „and“. То значи да ова услова морају бити испуњена.

Цртање табле на екрану

```
132     pygame.display.update()
133     FPSLOCK.tick(FPS)
```

Баш као и у осталим програмима, последња ствар у петљи је цртање површине прозора на екрану позивајући „tick()“ метод.

terminate() функција

```
136     def terminate():
137         pygame.quit()
138         sys.exit()
139
140
141     def checkForQuit():
142         for event in pygame.event.get(QUIT): # pokupi sve QUIT dogadjaje
143             terminate() # okoncaj sve QUIT dogadjaje koji su prisutni
144         for event in pygame.event.get(KEYUP): # pokupi KEYUP dogadjaje
145             if event.key == K_ESCAPE:
146                 terminate() # okoncaj ako je za KEYUP dogodjaj pritisnuto Esc dugme
147             pygame.event.post(event) # vrati ostale KEYUP dogadjaje
```

„terminate()“ и „checkForQuit()“ функције су коришћене и објашњене у поглављу „Sliding Puzzle“, па их нећемо поново описивати.

Поново коришећење константних променљивих

```
--  
150 def flashButtonAnimation(color, animationSpeed=50):  
151     if color == YELLOW:  
152         sound = BEEP1  
153         flashColor = BRIGHTYELLOW  
154         rectangle = YELLOWRECT  
155     elif color == BLUE:  
156         sound = BEEP2  
157         flashColor = BRIGHTBLUE  
158         rectangle = BLUERECT  
159     elif color == RED:  
160         sound = BEEP3  
161         flashColor = BRIGHTRED  
162         rectangle = REDRECT  
163     elif color == GREEN:  
164         sound = BEEP4  
165         flashColor = BRIGHTGREEN  
166         rectangle = GREENRECT
```

Зависи која вредност променљиве „Color“ је прослеђена као аргумент за параметар „color“, звук, светлија боја блица, и правоугаона област блица биће различита. Линије од 151 до 166 подешавају три локалне променљиве другачије у зависности од вредности параметра „color“: „sound“, „flashColor“, и „rectangle“.

Анимирање „Flash“ тастера

```
168 origSurf = DISPLAYSURF.copy()  
169 flashSurf = pygame.Surface((BUTTONSIZE, BUTTONSIZE))  
170 flashSurf = flashSurf.convert_alpha()  
171 r, g, b = flashColor  
172 sound.play()
```

Овај процес анимације је једноставан: На сваки фрејм анимације, нацртана је нормална табла и изнад тога, светлија боја дугмета које трепери је нацртана изнад њега. Вредност „alpha“ светлије боје почиње од 0 за први фрејм анимације, али за сваки следећи након што се вредност „alpha“ полако повећава док не постане потпуно непрозирна и светлија боја потпуно обоји стандардну боју дугмета.

Осветљавање је први део анимације. Други део је да замагљивање дугмета. Ово је одрађено истим кодом, али сада уместо да се вредност „alpha“ повећава за сваки фрејм, она ће се смањивати. Како вредност „alpha“ постаје све мања и мања, светлија боја ће постајати све више прозирна, све док оригинална табла не постане видљива.

Да бисмо ово урадили у коду, линија 168 прави копију почетног екрана и чува је у „origSurf“. Линија 169 креира нови објекат величине једног дугмета и чува га у „flashSurf“. Метод „convert_alpha()“ је позван на „flashSurf“ тако да „Surface“ објекат може имати прозирне боје нацртане на њему (у супротном, вредност „alpha“ ће бити игнорисана и аутоматски постављена на 255) .

Линија 171 креира појединачне локалне променљиве назване „r“, „g“ и „b“ да сачува оригиналне RGB вредности. Пре него што почнемо са анимацијом блица за одрежемо дугме, линија 172 пушта звучни ефекат у то дугме.

```

173     for start, end, step in ((0, 255, 1), (255, 0, -1)): # petlja za animacije
174         for alpha in range(start, end, animationSpeed * step):
175             checkForQuit()
176             DISPLAYSURF.blit(origSurf, (0, 0))
177             flashSurf.fill((r, g, b, alpha))
178             DISPLAYSURF.blit(flashSurf, rectangle.topleft)
179             pygame.display.update()
180             FPSLOCK.tick(FPS)
181     DISPLAYSURF.blit(origSurf, (0, 0))

```

Да би смо одрадили анимацију, прво желимо да нацртамо „flashSurf“ бојом која има повећање вредности „alpha“ од 0 до 255 да бисмо одрадили светлији део анимације. Након тога желимо да замаглимо, желимо да вредност „alpha“ иде од 255 до 0. То можемо урадити следећим кодом:

```

for alpha in range(0, 255, animationSpeed): # brightening
    checkForQuit()
    DISPLAYSURF.blit(origSurf, (0, 0))
    flashSurf.fill((r, g, b, alpha))
    DISPLAYSURF.blit(flashSurf, rectangle.topleft)

```

```

pygame.display.update()
FPSLOCK.tick(FPS)
for alpha in range(255, 0, -animationSpeed): # dimming
    checkForQuit()
    DISPLAYSURF.blit(origSurf, (0, 0))
    flashSurf.fill((r, g, b, alpha))
    DISPLAYSURF.blit(flashSurf, rectangle.topleft)
    pygame.display.update()
    FPSLOCK.tick(FPS)

```

Приметимо да код унутар „for“ петљи служи за цртање оквира и сви су исти. Ако напишемо код као овај изнад, тада ће прва „for“ петља служити за посветљивање (вредност „alpha“ иде од 0 до 255), а друга „for“ петља ће служити за замагљивање (вредност „alpha“ иде од 255 до 0). Приметимо да је у другој „for“ петљи, трећи аргумент у позивању функције „range()“ негативан број. Кад год имамо идентичан код као сад, вероватно можемо да га скратимо па не морамо да га понављамо. То је оно што радимо у „for“ петљи у линији 173, која снабдева различите вредности за позив „range()“ функцији у линији 174:

```

173     for start, end, step in ((0, 255, 1), (255, 0, -1)): # petlja za animacije
174         for alpha in range(start, end, animationSpeed * step):

```

У првој итерацији у 173. линији кода „for“ петље, „start“ има вредност 0, „end“ има вредност 255, а „step“ вредност 1. Када се „for“ петља у линији 174 извршава, позива се „range(0, 255, animationSpeed“).

Након тога „for“ петља у 174. линији кода извршава анимацију за посветљивање.

У другој итерацији у 173. линији кода „for“ петља (уvek постоје две и само две итерације ове унутрашње „for“ петље), „start“ има вредност 255, „end“ има вредност 0, а „step“ има вредност -1. Када је „for“ петља у, 174. линији, извршена, позива „range(255, 0, -animationSpeed)“

Код који је унутар 174. линије кода:

```
175         checkForQuit()
176         DISPLAYSURF.blit(origSurf, (0, 0))
177         flashSurf.fill((r, g, b, alpha))
178         DISPLAYSURF.blit(flashSurf, rectangle.topleft)
179         pygame.display.update()
180         FPSCLOCK.tick(FPS)
181     DISPLAYSURF.blit(origSurf, (0, 0))
```

Када проверимо „QUIT“ догађаје (у случају да корисник покуша да затвори програм у току анимације), онда правимо „origSurf“ површину да прикажемо „Surface“. Онда бојимо „flashSurf Surface“ позивајући „fill()“. Тада је „flashSurf“ направљена да прикаже „Surface“. Онда, да направимо да се „Surface“ појављује на екрану, позивамо „pygame.display.update()“ у линији 179. Да бисмо били сигурни да се анимација не приказује тако брзо колико брзо компјутер може да је нацрта, додајемо кратку паузу позивајући метод „tick()“.

Цртање дугмади

```
184 def drawButtons():
185     pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
186     pygame.draw.rect(DISPLAYSURF, BLUE, BLUERECT)
187     pygame.draw.rect(DISPLAYSURF, RED, REDRECT)
188     pygame.draw.rect(DISPLAYSURF, GREEN, GREENRECT)
```

Како је свако дугме правоугаоник одређене боје на одређеном месту, само ћемо четири пута позвати „pygame.draw.rect()“ да нацрта дугмад на екрану. Објекти „Color“ и „Rect“ које користимо да би смо их позиционирали никад се не мењају, зато су представљени као константе, на пример „YELLOW“ и „YELLOWRECT“.

Анимација промене позадине

```
191 def changeBackgroundAnimation(animationSpeed=40):
192     global bgColor
193     newBgColor = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
194
195     newBgSurf = pygame.Surface((WINDOWWIDTH, WINDOWHEIGHT))
196     newBgSurf = newBgSurf.convert_alpha()
197     r, g, b = newBgColor
198     for alpha in range(0, 255, animationSpeed): # апетља за анимације
199         checkForQuit()
200         DISPLAYSURF.fill(bgColor)
201
202         newBgSurf.fill((r, g, b, alpha))
203         DISPLAYSURF.blit(newBgSurf, (0, 0))
204
205         drawButtons() # ponovo nacrtaj dugmad
206
207         pygame.display.update()
208         FPSCLOCK.tick(FPS)
209     bgColor = newBgColor
```

Анимација промене боје позадине дешава се сваки пут када играч заврши са читавим шаблоном коректно. У свакој итерацији кроз петљу која почиње на 198. линији, читава површина екрана мора бити поново исцртана. Кораци извршени у свакој итерацији петље су:

- Линија 200 попуњава целу површину екрана (сачувана у „DISPLAYSURF“) са старом бојом позадине (која је сачувана у „bgColor“)
- Линија 202 попуњава различите објекте на површини (сачувана у „newBgSurf“) са новом бојом позадине („alpha“ вредност се мења у свакој итерацији)
- Линија 203 исцртава „newBgSurf“ површину у „DISPLAYSURF“
- Сада када нам је позадина обојена на начин на који желимо, нацртаћемо дугмад преко тога позивајући „drawButtons()“ у линији 205
- Линије 207 и 208 исцртавају површину на екрану и додају паузе

Разлог зашто постоји глобални исказ на почетку „changeBackgroundAnimation()“ функције је зато што „bgColor“ променљива зато што ова функција модификује садржај променљиве у исказу у линији 209. Било која функција може прочитати вредност променљиве без специфичног глобалног исказа.

Ако ра функција додели вредност глобалној променљивој без глобалног исказа, онда Пајтон сматра да је та променљива локална променљива која има исти назив као и глобална променљива. „main()“ функција користи „bgColor“ променљиву али јој не треба глобални исказ за то зато што чита само садржаје променљиве „bgColor“ којој „main()“ функција никад није доделила нову вредност. Овај концепт је објашњен детаљније на следећем линку <http://invpy.com/global>.

„GAME OVER“анимација

```
212 def gameOverAnimation(color=WHITE, animationSpeed=50):
213     # pustiti sve zvuke odjednom, onda nek pozadina treperi
214     origSurf = DISPLAYSURF.copy()
215     flashSurf = pygame.Surface(DISPLAYSURF.get_size())
216     flashSurf = flashSurf.convert_alpha()
217     BEEP1.play() # pustiti sve zvuke odjednom
218     BEEP2.play()
219     BEEP3.play()
220     BEEP4.play()
221     r, g, b = color
222     for i in range(3): # neka blicne tri puta
```

Свака итерација „for“ петље у следећој линији (линија 223, испод) ће извршити блиц. Да бисмо имали три блица завршена, ставићемо цео тај код у „for“ петљу која има три итерације. Уколико желиш више или мање блицева, онда промени цео број који је прослеђен у „range()“ у линији 222.

```
223     for start, end, step in ((0, 255, 1), (255, 0, -1)):
```

„for“ петља у линији 223 је потпуно иста као она у линији 173. „start“, „end“ и „step“ променљиве ће бити коришћене у следећој петљи (линија 224) да би контролисале промену вредности „alpha“ променљиве.

```

224     # Prvo ponavljanje u ovoj petlji postavlja sledeću petlju da ide od 0 do 255
225     # a drugu od 255 do 0.
226     for alpha in range(start, end, animationsSpeed * step): # petlja animacije
227         # alpha oznacava providnost, 255 je neprozirno, 0 je prozirno
228         checkForQuit()
229         flashSurf.fill((r, g, b, alpha))
230         DISPLAYSURF.blit(origSurf, (0, 0))
231         DISPLAYSURF.blit(flashSurf, (0, 0))
232         drawButtons()
233         pygame.display.update()
234         FPSLOCK.tick(FPS)

```

Ова петља функционише исто као и претходна анимација у делу „[анимација промене позадине](#)“. Копија оригиналне позадине саувана у „origSurf“ је нацртана на екрану, онда је „flashSurf“ саграђена преко површине екрана. Након што је боја позадине намештена, дугмад су нацртана преко тога у линији 232. Коначно, површина је нацртана на екрану позивајући „pygame.display.update()“.

Претварање из координата пиксела у дугмад

```

238     def getButtonClicked(x, y):
239         if YELLOWRECT.collidepoint( (x, y) ):
240             return YELLOW
241         elif BLUERECT.collidepoint( (x, y) ):
242             return BLUE
243         elif REDRECT.collidepoint( (x, y) ):
244             return RED
245         elif GREENRECT.collidepoint( (x, y) ):
246             return GREEN
247         return None
248
249
250     if __name__ == '__main__':
251         main()

```

Функција „getButtonClicked()“ једноставно узима „XY“ координате пиксела и враћа једну од вредности YELLOW, BLUE, RED, или GREEN уколико је неко дугме кликнуто, или враћа „None“ уколико „XY“ координате пиксела нису преко неког од четири дугмета.