```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
1   #!/usr/bin/python
2   # Golf
3   # Code Angel
4
5
6   import sys
7   import os
8   import pygame
9   from pygame.locals import *
10  import random
11
12  # Define the colours
13  WHITE = (255, 255, 255)
14  GREY = (62, 87, 113)
15
16  # Define constants
17  SCREEN_WIDTH = 640
18  SCREEN_HEIGHT = 480
19
20  SCOREBOARD_MARGIN = 4
21  SCOREBOARD_HEIGHT = 48
22  SCOREBOARD_LINE = 20
23  SCOREBOARD_COLUMNS = 10
24
25  HOLE_MESSAGE_Y = 60
26
27  METER_X = 25
28  METER_Y = SCOREBOARD_HEIGHT + 20
29
30  SLIDER_BORDER = 5
31  SLIDER_X = 35
32  SLIDER_TOP_PADDDING = 8
33
34  SLIDER_SPEED = 5
35  SLOW_SLIDER_SPEED = 20
36  SLOW_PUTT_RANGE = 3
37
38  MAX_POWER = 30
39  MIN_POWER = 1
40
41  START_BALL_X = 20
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
BALL_Y = 436
BALL_STEP = 3
BALL_DESCENT = 5

FLAG_Y = 244
RANDOM_FLAG_MIN = 10
RANDOM_FLAG_MAX = 30
FLAG_STEP = 18
HOLE_CENTRE = 8

# Setup
os.environ['SDL_VIDEO_CENTERED'] = '1'
pygame.mixer.pre_init(44100, -16, 2, 512)
pygame.mixer.init()
pygame.init()
game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption('Golf')
clock = pygame.time.Clock()
font = pygame.font.SysFont('Helvetica', 16)

# Load images
background_image = pygame.image.load('golf_background.png').convert()
power_meter_image = pygame.image.load('power_meter.png').convert()
slider_image = pygame.image.load('slider.png').convert_alpha()
ball_image = pygame.image.load('ball.png').convert_alpha()

flag_1_image = pygame.image.load('flag_1.png').convert_alpha()
flag_2_image = pygame.image.load('flag_2.png').convert_alpha()
flag_3_image = pygame.image.load('flag_3.png').convert_alpha()

# Load sounds
putt_sound = pygame.mixer.Sound('putt.ogg')
clap_sound = pygame.mixer.Sound('clap.ogg')


def main():

    # Initialise variables
    slider_direction = 'up'
    slider_timer = SLOW_SLIDER_SPEED
    shot_power = 1
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
    meter_height = power_meter_image.get_rect().height - 2 * SLIDER_BORDER

    ball_x = START_BALL_X
    ball_distance = 0
    ball_direction = 'right'
    final_ball_location = 0
    moves_per_flag = FLAG_STEP / BALL_STEP

    flag_distance = random.randint(RANDOM_FLAG_MIN, RANDOM_FLAG_MAX)
    flag_x = flag_distance * FLAG_STEP + HOLE_CENTRE

    hole = 1

    hole_strokes = [0, 0, 0]
    round_strokes = 0
    best_round_strokes = 0

    in_the_hole = False

    # Main game loop
    while True:

        for event in pygame.event.get():
            key_pressed = pygame.key.get_pressed()

            # SPACE key pressed - hit shot
            if key_pressed[pygame.K_SPACE] and ball_distance == 0 and in_the_hole is False:
                slider_direction = 'none'
                ball_distance = shot_power * moves_per_flag

                hole_strokes[hole - 1] += 1

                if ball_direction == 'right':
                    final_ball_location += shot_power
                else:
                    final_ball_location -= shot_power

                putt_sound.play()

            # RETURN pressed when ball is in the hole - start new hole
            elif key_pressed[pygame.K_RETURN] and in_the_hole is True:
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
124
125                 if hole == 3:
126
127                     if round_strokes < best_round_strokes or best_round_strokes == 0:
128                         best_round_strokes = round_strokes
129
130                     hole = 1
131                     hole_strokes = [0, 0, 0]
132                     round_strokes = 0
133
134                 else:
135                     hole += 1
136
137                 in_the_hole = False
138                 shot_power = 1
139                 slider_direction = 'up'
140
141                 ball x = START BALL X
142                 ball_direction = 'right'
143                 final_ball_location = 0
144
145                 flag_distance = random.randint(RANDOM_FLAG_MIN, RANDOM_FLAG_MAX)
146                 flag_x = flag_distance * FLAG_STEP + HOLE_CENTRE
147
148                 slider_timer = SLOW_SLIDER_SPEED
149
150             if event.type == QUIT:
151                 pygame.quit()
152                 sys.exit()
153
154         # Update slider
155         slider_timer -= 1
156
157         if slider_timer == 0:
158
159             # Slider moving up, increase shot power
160             if slider_direction == 'up':
161                 shot_power += 1
162                 if shot_power == MAX_POWER:
163                     slider_direction = 'down'
164
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
165             # Slider moving down, decrease shot power
166             elif slider_direction == 'down':
167                 shot_power -= 1
168                 if shot_power == MIN_POWER:
169                     slider_direction = 'up'
170
171             # New timer pause
172             if shot_power <= SLOW_PUTT_RANGE:
173                 slider_timer = SLOW_SLIDER_SPEED
174             else:
175                 slider_timer = SLIDER_SPEED
176
177         # Update ball location
178         if ball_distance > 0:
179
180             if ball_direction == 'right':
181                 ball_x += BALL_STEP
182             else:
183                 ball_x -= BALL_STEP
184
185             # Ball gone off left or right hand edge of screen
186             if ball_x > SCREEN_WIDTH or ball_x < 0:
187
188                 # Reset ball location at left of screen
189                 ball_x = START_BALL_X
190                 ball_distance = 0
191                 ball_direction = 'right'
192                 shot_power = 1
193
194                 # Reset slider at bottom of meter
195                 slider_direction = 'up'
196                 final_ball_location = 0
197                 slider_timer = SLOW_SLIDER_SPEED
198
199             # Ball is still on screen so move ball closer to final_ball_position
200             else:
201                 ball_distance -= 1
202
203                 # Ball has stopped rolling
204                 if ball_distance == 0:
205
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
206                     # Ball in hole
207                     if final_ball_location == flag_distance:
208                         in_the_hole = True
209                         round_strokes += hole_strokes[hole - 1]
210
211                         clap_sound.play()
212
213                     # Ball missed hole
214                     else:
215                         if final_ball_location < flag_distance:
216                             ball_direction = 'right'
217                         else:
218                             ball_direction = 'left'
219
220                         # Reset slider at bottom of meter
221                         shot_power = 1
222                         slider_direction = 'up'
223                         slider timer = SLOW SLIDER SPEED
224
225         # Draw background
226         game_screen.blit(background_image, [0, 0])
227
228         # Draw meter and slider
229         if ball_distance == 0 and in_the_hole is False:
230             game_screen.blit(power_meter_image, [METER_X, METER_Y])
231
232             slider_step = (MAX_POWER - shot_power) * meter_height / MAX_POWER
233             slider_y = METER_Y + SLIDER_BORDER + slider_step - SLIDER_TOP_PADDDING
234             game_screen.blit(slider_image, [SLIDER_X, slider_y])
235
236         # Draw flag
237         if hole == 1:
238             game_screen.blit(flag_1_image, [flag_x, FLAG_Y])
239         elif hole == 2:
240             game_screen.blit(flag_2_image, [flag_x, FLAG_Y])
241         elif hole == 3:
242             game_screen.blit(flag_3_image, [flag_x, FLAG_Y])
243
244         # Draw ball
245         if in_the_hole is False:
246             game_screen.blit(ball_image, [ball_x, BALL_Y])
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
247         else:
248             game_screen.blit(ball_image, [ball_x, BALL_Y + BALL_DESCENT])
249
250         # Display scoreboard
251         display_scoreboard(hole_strokes, round_strokes, best_round_strokes)
252
253         # In the hole messages
254         if in_the_hole is True:
255             in_hole_message(hole, hole_strokes[hole - 1], round_strokes)
256
257         pygame.display.update()
258         clock.tick(30)
259
260
261 # Display the scoreboard
262 def display_scoreboard(hole_strokes, round_strokes, best):
263
264     scoreboard background rect = (0, 0, SCREEN WIDTH, SCOREBOARD HEIGHT)
265     pygame.draw.rect(game_screen, GREY, scoreboard_background_rect)
266
267     # Display holes 1-3
268     display_scoreboard_data('Hole:', 0, 0)
269
270     for hole_number in range(1, 4):
271         display_scoreboard_data(str(hole_number), hole_number, 0)
272
273     # Display strokes on each of the 3 holes
274     display_scoreboard_data('Strokes:', 0, 1)
275
276     for hole_number in range(0, 3):
277         if hole_strokes[hole_number] > 0:
278             display_scoreboard_data(str(hole_strokes[hole_number]), hole_number + 1, 1)
279         else:
280             display_scoreboard_data(str('-'), hole_number + 1, 1)
281
282     # Display total for round
283     display_scoreboard_data('Total', 6, 0)
284
285     if round_strokes > 0:
286         display_scoreboard_data(str(round_strokes), 6, 1)
287     else:
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
288         display_scoreboard_data(str('-'), 6, 1)
289
290     # Display best overall round
291     display_scoreboard_data('Best', 7, 0)
292
293     if best > 0:
294         display_scoreboard_data(str(best), 7, 1)
295     else:
296         display_scoreboard_data(str('-'), 7, 1)
297
298
299 # Display scoreboard text items
300 def display_scoreboard_data(scoreboard_text, column, line):
301
302     display_text = font.render(scoreboard_text, True, WHITE)
303
304     text_x = SCREEN_WIDTH / SCOREBOARD_COLUMNS * column + SCOREBOARD_MARGIN
305     text_y = SCOREBOARD_MARGIN + line * SCOREBOARD_LINE
306
307     game_screen.blit(display_text, [text_x, text_y])
308
309
310 # Display message at the end of each hole
311 def in_hole_message(hole_number, hole_strokes, round_strokes):
312
313     if hole_number == 3:
314         message = 'Round completed in ' + str(round_strokes) + '. Press RETURN to play another round.'
315         text = font.render(message, True, WHITE)
316     else:
317         message = 'In the hole in ' + str(hole_strokes) + '. Press RETURN to play next hole.'
318         text = font.render(message, True, WHITE)
319
320     background_x = SCOREBOARD_MARGIN * 4
321     background_width = SCREEN_WIDTH - SCOREBOARD_MARGIN * 8
322     background_height = 2 * SCOREBOARD_LINE
323     message_background_rect = (background_x, HOLE_MESSAGE_Y, background_width, background_height)
324     pygame.draw.rect(game_screen, GREY, message_background_rect)
325
326     text_rect = text.get_rect()
327     messsage_x = (SCREEN_WIDTH - text_rect.width) / 2
328     message_y = HOLE_MESSAGE_Y + SCOREBOARD_LINE / 2
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
329         game_screen.blit(text, [messsage_x, message_y])
330
331
332     if __name__ == '__main__':
333         main()
334
```

|0 |1 |2 |3 |4 |5 |6 |7 |8

