```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
1   #!/usr/bin/python
2   # Micro Racers
3   # Code Angel
4
5   import sys
6   import os
7   import pygame
8   from pygame.locals import *
9   import math
10
11  import track
12
13  # Define the colours
14  BLACK = (0, 0, 0)
15  WHITE = (255, 255, 255)
16  RED = (227, 0, 13)
17  BLUE = (8, 34, 255)
18
19  # Define constants
20  SCREEN_WIDTH = 640
21  SCREEN_HEIGHT = 480
22  SCOREBOARD_MARGIN = 8
23  SCOREBOARD_HEIGHT = 36
24  SCOREBOARD_CAR_WIDTH = 128
25  TEXT_LINE_HEIGHT = 18
26  RESULT_BLOCK_WIDTH = 320
27  RESULT_BLOCK_HEIGHT = 150
28
29  TRACK_BLOCK_SIZE = 80
30  TRACK_HEIGHT = 20
31
32  ACCELERATION = 0.2
33  OFF_TRACK_ACCELERATION = 0.05
34  MAX_SPEED = 3
35  DRAG = 0.93
36  ANGLE_DRAG = 0.92
37  TURN_SPEED = 0.5
38
39  COMPUTER_CAR_ACCELERATION = 0.15
40  COMPUTER_CAR_TURN_ANGLE = 5
41  COMPUTER_CAR_MAX_SPEED = 3
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
42
43  CAR_LENGTH = 16
44  CAR_WIDTH = 8
45  GRID_WIDTH = 10
46
47  RACE_LAPS = 3
48
49  # Setup
50  os.environ['SDL_VIDEO_CENTERED'] = '1'
51  pygame.init()
52  game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
53  pygame.display.set_caption('Micro Racers')
54  pygame.key.set_repeat(10, 20)
55  clock = pygame.time.Clock()
56  font = pygame.font.SysFont('Helvetica', 16)
57
58  background_image = track.load_image('general', 'background')
59  car_image = track.load_image('general', 'red_car')
60  computer_car_image = track.load_image('general', 'blue_car')
61
62  lights_off_image = track.load_image('general', 'lights_off')
63  lights_red_image = track.load_image('general', 'lights_red')
64  lights_amber_image = track.load_image('general', 'lights_amber')
65  lights_green_image = track.load_image('general', 'lights_green')
66
67
68  def main():
69
70      # Initialise variables
71      level = 1
72      race = 1
73
74      # Track details
75      track_pieces = track.load_track_pieces()
76
77      current_track = track.get_track(level, track_pieces)
78      start_grid = current_track[0]
79      start_grid_x = get_x_coord(start_grid.get('loc'))
80      start_grid_y = get_y_coord(start_grid.get('loc'))
81
82      lights_loc = track.get_lights_loc(level)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
83
84      # Player and computer car start positions
85      car_center_x = start_grid_x + TRACK_BLOCK_SIZE - CAR_LENGTH - GRID_WIDTH
86      computer_car_center_x = car_center_x
87
88      car_center_y = start_grid_y - CAR_WIDTH + TRACK_BLOCK_SIZE / 2
89      computer_car_center_y = start_grid_y + CAR_WIDTH * 2 + TRACK_BLOCK_SIZE / 2
90
91      # Player car starting data
92      speed_x = 0
93      speed_y = 0
94      angle = 90
95      angle_speed = 0
96      car_accelerator = False
97      car_on_track = True
98
99      laps = 0
100     lap_counted = True
101     check_point_passed = False
102
103     # Computer car starting data
104     computer_car_speed_x = 0
105     computer_car_speed_y = 0
106     computer_car_angle = 90
107     computer_car_new_angle = computer_car_angle
108     computer_laps = 0
109     computer_lap_counted = True
110     computer_car_acceleration = COMPUTER_CAR_ACCELERATION
111
112     race_start = True
113     race_over = False
114     race_winner = ''
115     green_light = False
116     start_light_time = pygame.time.get_ticks()
117
118     # Main game loop
119     while True:
120
121         for event in pygame.event.get():
122             key_pressed = pygame.key.get_pressed()
123
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
            if race_start is False and race_over is False:

                # SPACE key pressed - accelerate car
                if key_pressed[pygame.K_SPACE]:
                    if event.type == pygame.KEYDOWN:
                        car_accelerator = True

                # No SPACE pressed - no acceleration
                else:
                    car_accelerator = False

                # Right key pressed - adjust angle of car
                if key_pressed[pygame.K_RIGHT]:
                    angle_speed -= TURN_SPEED

                # Left key pressed - adjust angle of car
                elif key_pressed[pygame.K_LEFT]:
                    angle speed += TURN SPEED

            if key_pressed[pygame.K_RETURN] and race_over is True:

                if race_winner == 'player':
                    computer_car_acceleration += 0.01
                    race += 1
                    if level == 4:
                        level = 1
                else:
                    level = 1
                    race = 1
                    computer_car_acceleration = 0.15

                # New level / game - reset track and car data
                current_track = track.get_track(level, track_pieces)
                start_grid = current_track[0]
                start_grid_x = get_x_coord(start_grid.get('loc'))
                start_grid_y = get_y_coord(start_grid.get('loc'))

                lights_loc = track.get_lights_loc(level)

                # Player and computer car start positions
                car_center_x = start_grid_x + TRACK_BLOCK_SIZE - CAR_LENGTH - GRID_WIDTH
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
165                 computer_car_center_x = car_center_x
166
167                 car_center_y = start_grid_y - CAR_WIDTH + TRACK_BLOCK_SIZE / 2
168                 computer_car_center_y = start_grid_y + CAR_WIDTH * 2 + TRACK_BLOCK_SIZE / 2
169
170                 speed_x = 0
171                 speed_y = 0
172                 angle = 90
173                 angle_speed = 0
174                 car_accelerator = False
175                 car_on_track = True
176
177                 laps = 0
178                 lap_counted = True
179                 check_point_passed = False
180
181                 computer_car_speed_x = 0
182                 computer car speed y = 0
183                 computer_car_angle = 90
184                 computer_car_new_angle = computer_car_angle
185                 computer_laps = 0
186                 computer_lap_counted = True
187
188                 race_start = True
189                 race_over = False
190                 race_winner = ''
191                 green_light = False
192                 start_light_time = pygame.time.get_ticks()
193
194             if event.type == QUIT:
195                 pygame.quit()
196                 sys.exit()
197
198         # Accelerate player car increasing speed_x and speed_y (if space pressed)
199         if car_accelerator is True:
200
201             # If player car is on the track, accelerate by ACCELERATION
202             if car_on_track is True:
203                 speed_x += math.sin(math.radians(angle)) * ACCELERATION
204                 speed_y += math.cos(math.radians(angle)) * ACCELERATION
205
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
206                 # Don't let player car get above the maximum speed
207                 if speed_x > MAX_SPEED:
208                     speed_x = MAX_SPEED
209
210                 if speed_y > MAX_SPEED:
211                     speed_y = MAX_SPEED
212
213             # If player car is off the track, accelerate by OFF TRACK ACCELERATION
214             else:
215                 speed_x += math.sin(math.radians(angle)) * OFF_TRACK_ACCELERATION
216                 speed_y += math.cos(math.radians(angle)) * OFF_TRACK_ACCELERATION
217
218         # Factor in player car drag resistance
219         speed_x *= DRAG
220         speed_y *= DRAG
221
222         angle += angle_speed
223         angle speed *= ANGLE DRAG
224
225         # Accelerate computer car increasing car_speed_x and car_speed_y
226         computer_car_speed_x += math.sin(math.radians(computer_car_angle)) * computer_car_acceleration
227         computer_car_speed_y += math.cos(math.radians(computer_car_angle)) * computer_car_acceleration
228
229         # Don't let computer car get above the maximum speed
230         if computer_car_speed_x > COMPUTER_CAR_MAX_SPEED:
231             computer_car_speed_x = COMPUTER_CAR_MAX_SPEED
232
233         if computer_car_speed_y > COMPUTER_CAR_MAX_SPEED:
234             computer_car_speed_y = COMPUTER_CAR_MAX_SPEED
235
236         # Factor in computer car drag resistance
237         computer_car_speed_x *= DRAG
238         computer_car_speed_y *= DRAG
239
240         if computer_car_new_angle < computer_car_angle:
241             computer_car_angle -= COMPUTER_CAR_TURN_ANGLE
242         elif computer_car_new_angle > computer_car_angle:
243             computer_car_angle += COMPUTER_CAR_TURN_ANGLE
244
245         if computer_car_angle == -270:
246             computer_car_angle = 90
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
247             computer_car_new_angle = computer_car_angle
248
249         # Prevent the player car from leaving the screen
250         if car_center_x < 0:
251             car_center_x = 0
252
253         if car_center_x > SCREEN_WIDTH:
254             car_center_x = SCREEN_WIDTH
255
256         if car_center_y < SCOREBOARD_HEIGHT:
257             car_center_y = SCOREBOARD_HEIGHT
258
259         if car_center_y > SCREEN_HEIGHT:
260             car_center_y = SCREEN_HEIGHT
261
262         # Update the player car rectangle centre point by adding x and y speeds
263         car_rect = car_image.get_rect()
264         car center x = car center x + speed x
265         car_center_y = car_center_y + speed_y
266
267         car_rect.centerx = car_center_x
268         car_rect.centery = car_center_y
269
270         # Update the computer car rectangle centre point by adding x and y speeds
271         computer_car_rect = computer_car_image.get_rect()
272
273         if race_start is False and race_over is False:
274             computer_car_center_x = computer_car_center_x + computer_car_speed_x
275             computer_car_center_y = computer_car_center_y + computer_car_speed_y
276
277         computer_car_rect.centerx = computer_car_center_x
278         computer_car_rect.centery = computer_car_center_y
279
280         # Check if the player car is on the track, and if so, if it has crossed the start line
281         car_on_track = False
282         for track_piece in current_track:
283             track_piece_left = get_x_coord(track_piece.get('loc'))
284             track_piece_top = get_y_coord(track_piece.get('loc'))
285             track_piece_right = track_piece_left + TRACK_BLOCK_SIZE
286             track_piece_bottom = track_piece_top + TRACK_BLOCK_SIZE
287
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
            # check if the player car is on the track
            if (track_piece_left <= car_center_x <= track_piece_right and
                    track_piece_top <= car_center_y <= track_piece_bottom):

                # the player car is on the track
                car_on_track = True
                car_track = track_piece.get('image')

                # check if the player car is at the start line and has passed the check point
                if (car_track == track_pieces.get('start_grid_image') and
                        check_point_passed is True and
                        car_center_x > track_piece_right - CAR_LENGTH):

                    # player car has passed the start line
                    if lap_counted is False:
                        laps += 1
                        lap_counted = True
                        check_point_passed = False
                        if laps == RACE_LAPS:
                            race_over = True
                            race_winner = 'player'
                            level += 1
                            car_accelerator = False

                elif car_track != track_pieces.get('start_grid_image'):
                    lap_counted = False

                # The check point midway round track ensures player can't take a short cut
                if car_track == track_pieces.get('check_point_image'):
                    check_point_passed = True

            # check if the computer car is on the track
            if (track_piece_left <= computer_car_center_x <= track_piece_right and
                    track_piece_top <= computer_car_center_y <= track_piece_bottom):

                computer_car_track = track_piece.get('image')

                # check if the computer car needs to turn, and if so work out new angle
                if computer_car_track == track_pieces.get('right_down_image') and computer_car_angle == 90:
                    computer_car_new_angle = 0
                elif computer_car_track == track_pieces.get('down_right_image') and computer_car_angle == 0:
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
329                     computer_car_new_angle = 90
330                 elif computer_car_track == track_pieces.get('down_left_image') and computer_car_angle == 0:
331                     computer_car_new_angle = -90
332                 elif computer_car_track == track_pieces.get('down_right_image') and computer_car_angle == -90:
333                     computer_car_new_angle = -180
334                 elif computer_car_track == track_pieces.get('right_down_image') and computer_car_angle == -180:
335                     computer_car_new_angle = -90
336                 elif computer_car_track == track_pieces.get('up_right_image') and computer_car_angle == -180:
337                     computer_car_new_angle = -270
338                 elif computer_car_track == track_pieces.get('down_left_image') and computer_car_angle == 90:
339                     computer_car_new_angle = 180
340                 elif computer_car_track == track_pieces.get('up_right_image') and computer_car_angle == 180:
341                     computer_car_new_angle = 90
342                 elif computer_car_track == track_pieces.get('up_right_image') and computer_car_angle == -90:
343                     computer_car_new_angle = 0
344
345                 # check if the computer car is at the start finish line
346                 if (computer_car_track == track_pieces.get('start_grid_image') and
347                         computer_car_center_x > track_piece_right - CAR_LENGTH):
348                     if computer_lap_counted is False:
349                         computer_laps += 1
350                         computer_lap_counted = True
351                         if computer_laps == RACE_LAPS:
352                             race_over = True
353                             race_winner = 'computer'
354                             car_accelerator = False
355                 elif computer_car_track != track_pieces.get('start_grid_image'):
356                     computer_lap_counted = False
357
358         # Display background
359         game_screen.blit(background_image, [0, 0])
360
361         # Display track pieces
362         for track_piece in current_track:
363             track_x = get_x_coord(track_piece.get('loc'))
364             track_y = get_y_coord(track_piece.get('loc'))
365             track_image = track_piece.get('image')
366             game_screen.blit(track_image, [track_x, track_y])
367
368         # Display player car
369         display_car_image = pygame.transform.rotate(car_image, angle)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
370         game_screen.blit(display_car_image, car_rect)
371
372         # Display computer car
373         computer_display_car_image = pygame.transform.rotate(computer_car_image, computer_car_angle)
374         game_screen.blit(computer_display_car_image, computer_car_rect)
375
376         # Dsiplay scoreboard
377         display_scoreboard(laps, computer_laps, race)
378
379         # Display end of race messages
380         if race_over is True:
381             display_end_race_message(race_winner, level)
382
383         # Display start light sequence
384         if race_start is True:
385
386             # Start light sequence
387             new time = pygame.time.get ticks()
388             delta_time = new_time - start_light_time
389
390             if delta_time <= 1000:
391                 game_screen.blit(lights_off_image, lights_loc)
392
393             elif delta_time <= 2000:
394                 game_screen.blit(lights_red_image, lights_loc)
395
396             elif delta_time <= 3000:
397                 game_screen.blit(lights_amber_image, lights_loc)
398
399             elif delta_time <= 4000:
400                 game_screen.blit(lights_green_image, lights_loc)
401                 race_start = False
402                 green_light = True
403
404         # Pause on green light before hiding lights
405         if green_light is True:
406             new_time = pygame.time.get_ticks()
407             delta_time = new_time - start_light_time
408
409             if delta_time <= 5000:
410                 game_screen.blit(lights_green_image, lights_loc)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
411             else:
412                 green_light = False
413
414         pygame.display.update()
415         clock.tick(60)
416
417
418 # Get the actual x coordinate of the centre point of a track piece
419 def get_x_coord(coordinates):
420     return coordinates[0] * TRACK_BLOCK_SIZE - TRACK_BLOCK_SIZE / 2
421
422
423 # Get the actual y coordinate of the centre point of a track piece
424 def get_y_coord(coordinates):
425     return coordinates[1] * TRACK_BLOCK_SIZE + TRACK_HEIGHT - TRACK_BLOCK_SIZE / 2
426
427
428 # Display the scoreboard at the top of the screen
429 def display_scoreboard(laps, computer_laps, race_number):
430
431     # Draw the black, red and blue rectangles
432     scoreboard_back_rect = (0, 0, SCREEN_WIDTH, SCOREBOARD_HEIGHT)
433     pygame.draw.rect(game_screen, BLACK, scoreboard_back_rect)
434
435     scoreboard_red_car_rect = (0, 0, SCOREBOARD_CAR_WIDTH, SCOREBOARD_HEIGHT)
436     pygame.draw.rect(game_screen, RED, scoreboard_red_car_rect)
437
438     scoreboard_blue_car_rect = (SCREEN_WIDTH - SCOREBOARD_CAR_WIDTH, 0, SCOREBOARD_CAR_WIDTH, SCOREBOARD_HEIGHT)
439     pygame.draw.rect(game_screen, BLUE, scoreboard_blue_car_rect)
440
441     # Display player laps, computer laps and race number
442     laps_text = 'Player Laps: ' + str(laps)
443     text = font.render(laps_text, True, WHITE)
444     game_screen.blit(text, [SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])
445
446     computer_laps_text = 'Computer Laps: ' + str(computer_laps)
447     text = font.render(computer_laps_text, True, WHITE)
448     text_rect = text.get_rect()
449     game_screen.blit(text, [SCREEN_WIDTH - text_rect.width - SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])
450
451     race_text = 'RACE: ' + str(race_number)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
452     text = font.render(race_text, True, WHITE)
453     game_screen.blit(text, [centre_text_x(text), SCOREBOARD_MARGIN])
454
455
456 # Display box with end of race message
457 def display_end_race_message(winner, level):
458
459     # Display rectangle
460     message_block_x = (SCREEN_WIDTH - RESULT_BLOCK_WIDTH) / 2
461     message_block_y = (SCREEN_HEIGHT - RESULT_BLOCK_HEIGHT) / 2
462     race_over_background_rect = (message_block_x, message_block_y, RESULT_BLOCK_WIDTH, RESULT_BLOCK_HEIGHT)
463     pygame.draw.rect(game_screen, BLACK, race_over_background_rect)
464
465     # Display 'RESULT' centred
466     text = font.render('RESULT', True, WHITE)
467     game_screen.blit(text, [centre_text_x(text), message_box_line_y(2)])
468
469     # Display end of race message centred
470     if winner == 'player':
471         winner_text = 'Player Wins'
472         if level == 4:
473             return_text = 'Ultimate Champion - Hit RETURN for new game'
474         else:
475             return_text = 'Level Complete - Hit RETURN for new level'
476
477     else:
478         winner_text = 'Computer Wins'
479         return_text = 'Hit RETURN for new game'
480
481     text = font.render(winner_text, True, WHITE)
482     game_screen.blit(text, [centre_text_x(text), message_box_line_y(3)])
483
484     text = font.render(return_text, True, WHITE)
485     game_screen.blit(text, [centre_text_x(text), message_box_line_y(5)])
486
487
488 # Work out x coordinate to centre a block of text
489 def centre_text_x(text_image):
490     text_rect = text_image.get_rect()
491     return (SCREEN_WIDTH - text_rect.width) / 2
492
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
493
494 # Work out y coordinate to display a line of text in the end of game message box, given the line number
495 def message_box_line_y(line_number):
496     return (SCREEN_HEIGHT - RESULT_BLOCK_HEIGHT) / 2 + TEXT_LINE_HEIGHT * line_number
497
498
499 if __name__ == '__main__':
500     main()
```

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

