```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
1   #!/usr/bin/python
2   # Jupiter Landing
3   # Code Angel
4
5
6   import sys
7   import os
8   import pygame
9   from pygame.locals import *
10  import random
11  import math
12
13  # Define the colours
14  GREEN = (84, 216, 61)
15  AMBER = (255, 153, 0)
16
17  # Define constants
18  SCREEN_WIDTH = 640
19  SCREEN_HEIGHT = 480
20
21  DATA_DISPLAY_LEFT = 16
22  TEXT_LINE_SPACE = 24
23
24  FUEL_WARNING = 50
25  KGRAVITY = 0.01
26
27  MAX_TERRAIN_HEIGHT = 120
28  MIN_TERRAIN_HEIGHT = 90
29  LEFT_BUFFER_ZONE = 36
30  RIGHT_BUFFER_ZONE = 12
31
32  LANDER_START_Y = 40
33  ROTATE_ANGLE = 4
34
35  # Setup
36  os.environ['SDL_VIDEO_CENTERED'] = '1'
37  pygame.mixer.pre_init(44100, -16, 2, 512)
38  pygame.mixer.init()
39  pygame.init()
40  game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
41  pygame.display.set_caption('Jupiter Landing')
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
42  pygame.key.set_repeat(10, 20)
43  clock = pygame.time.Clock()
44  font = pygame.font.SysFont('Courier', 16)
45
46  # Load images
47  background_image = pygame.image.load('background.png').convert()
48
49  planet_high_low_image = pygame.image.load('planet_high_low.png').convert_alpha()
50  planet_high_mid_image = pygame.image.load('planet_high_mid.png').convert_alpha()
51  planet_low_high_image = pygame.image.load('planet_low_high.png').convert_alpha()
52  planet_low_mid_image = pygame.image.load('planet_low_mid.png').convert_alpha()
53  planet_mid_image = pygame.image.load('planet_mid.png').convert_alpha()
54  planet_mid_high_image = pygame.image.load('planet_mid_high.png').convert_alpha()
55  planet_mid_low_image = pygame.image.load('planet_mid_low.png').convert_alpha()
56  planet_blank_image = pygame.image.load('planet_blank.png').convert_alpha()
57  planet_landing_block_image = pygame.image.load('planet_landing_block.png').convert_alpha()
58
59  lander_image = pygame.image.load('lander.png').convert_alpha()
60  thrust_image = pygame.image.load('thrust_lander.png').convert_alpha()
61  exploded_lander_image = pygame.image.load('exploded_lander.png').convert_alpha()
62
63  # Load sounds
64  thrust_sound = pygame.mixer.Sound('thrust.ogg')
65  explosion_sound = pygame.mixer.Sound('explosion.ogg')
66  landed_sound = pygame.mixer.Sound('landed.ogg')
67
68
69  def main():
70
71      # Initialise variables
72      terrain_block_size = planet_blank_image.get_rect().width
73      terrain_blocks = int(SCREEN_WIDTH / terrain_block_size)
74
75      display_lander_image = lander_image
76      lander_width = lander_image.get_rect().width
77
78      fuel_left = True
79      velocity_ok = True
80      landing_location_ok = True
81      angle_ok = True
82      on_screen = True
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
    landed = False

    fuel = 500
    landing_blocks = 16
    landing_velocity = 1.0
    landing_angle = 24

    level = 1
    level_score = 0
    total_score = 0
    hi_score = 0

    planet = []

    lander_center_y = LANDER_START_Y
    angle = 0
    velocity_x = 0
    velocity y = 0
    gravity = KGRAVITY
    rocket_thrust = False
    exploded = False
    level_end_sound_played = False
    hit_terrain = False

    # Random planet, landing pad and start location
    landing_pad_start = random_landing_pad_start(landing_blocks, terrain_blocks)
    setup_planet(planet, landing_pad_start, landing_blocks, terrain_blocks)
    lander_column = random_lander_loc(landing_pad_start, landing_blocks, terrain_blocks)
    lander_center_x = lander_column * terrain_block_size

    # Main game loop
    while True:

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

            # SPACE key pressed - rocket thrust
            if key_pressed[pygame.K_SPACE]:
                rocket_thrust = True
            else:
                rocket_thrust = False
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
124
125             # Left or right key pressed - rotate ship
126             if key_pressed[pygame.K_RIGHT]:
127                 angle -= ROTATE_ANGLE
128                 if angle < -90:
129                     angle = -90
130             elif key_pressed[pygame.K_LEFT]:
131                 angle += ROTATE_ANGLE
132                 if angle > 90:
133                     angle = 90
134
135             # RETURN pressed and end of level/game
136             if key_pressed[pygame.K_RETURN] and (landed or hit_terrain is True):
137
138                 # Successful landing - level up
139                 if landed is True:
140                     level += 1
141                     total score += level score
142
143                     if level == 2:
144                         fuel = 400
145                         landing_blocks = 12
146                         landing_velocity = 1.0
147                         landing_angle = 24
148                     elif level == 3:
149                         fuel = 350
150                         landing_blocks = 12
151                         landing_velocity = 0.8
152                         landing_angle = 20
153                     elif level == 4:
154                         fuel = 300
155                         landing_blocks = 12
156                         landing_velocity = 0.8
157                         landing_angle = 16
158                     elif level == 5:
159                         fuel = 250
160                         landing_blocks = 8
161                         landing_velocity = 0.8
162                         landing_angle = 12
163                     elif level == 6:
164                         fuel = 200
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
                        landing_blocks = 8
                        landing_velocity = 0.6
                        landing_angle = 8
                    else:
                        fuel = 200 - (level * 10)
                        landing_blocks = 6
                        landing_velocity = 0.5
                        landing_angle = 0

                # Crashed - back to level 1
                elif hit_terrain is True:
                    level = 1
                    total_score = 0
                    level_score = 0

                    landing_blocks = 16
                    landing_velocity = 1.0
                    landing angle = 24
                    fuel = 500

                # Reset new game / new level variables
                lander_center_y = LANDER_START_Y
                angle = 0
                velocity_x = 0
                velocity_y = 0
                gravity = KGRAVITY
                rocket_thrust = False
                exploded = False
                level_end_sound_played = False
                hit_terrain = False
                landed = False

                fuel_left = True
                velocity_ok = True
                landing_location_ok = True
                angle_ok = True
                on_screen = True

                # Random planet, landing pad and start location
                landing_pad_start = random_landing_pad_start(landing_blocks, terrain_blocks)
                setup_planet(planet, landing_pad_start, landing_blocks, terrain_blocks)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
                lander_column = random_lander_loc(landing_pad_start, landing_blocks, terrain_blocks)
                lander_center_x = lander_column * terrain_block_size

                pygame.mixer.stop()

            if event.type == QUIT:
                pygame.quit()
                sys.exit()

        # Update lander location
        lander_rect = display_lander_image.get_rect()

        lander_center_x += velocity_x
        lander_left_x = lander_center_x - lander_width / 2
        lander_right_x = lander_center_x + lander_width / 2

        lander_center_y += velocity_y
        lander top y = lander center y - lander width / 2

        lander_rect.centerx = lander_center_x
        lander_rect.centery = lander_center_y

        # Check lander has not left screen left, right or top
        if lander_left_x < 0 or lander_right_x > SCREEN_WIDTH:
            velocity_x = 0
            exploded = True
            on_screen = False

        if lander_top_y < 0:
            velocity_x = 0
            velocity_y = 0
            exploded = True
            on_screen = False

        # Check for terrain collision by looping through all terrain blocks
        for block_number, block in enumerate(planet):

            terrain_y = block.get('terrain_y')
            block_x = block_number * terrain_block_size
            block_y = terrain_y * terrain_block_size

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

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
247             terrain_rect = pygame.Rect(
248                 block_x,
249                 block_y,
250                 terrain_block_size,
251                 terrain_block_size)
252 
253             if lander_rect.colliderect(terrain_rect):
254                 hit_terrain = True
255 
256                 # Check if landing velocity acceptable
257                 if abs(velocity_x) > landing_velocity or velocity_y > landing_velocity:
258                     exploded = True
259                     velocity_ok = False
260 
261                 # Check if landing angle acceptable
262                 if abs(angle) > landing_angle:
263                     exploded = True
264                     angle_ok = False
265 
266                 # Check if lander is within landing pad location
267                 landing_pad_left = (landing_pad_start - 1) * terrain_block_size
268                 landing_pad_right = (landing_pad_start + landing_blocks + 2) * terrain_block_size
269                 if lander_left_x >= landing_pad_left and lander_right_x <= landing_pad_right:
270                     if exploded is False:
271                         landed = True
272 
273                 else:
274                     exploded = True
275                     landing_location_ok = False
276 
277                 velocity_x = 0
278                 velocity_y = 0
279                 gravity = 0
280 
281         # Rocket thrust
282         if exploded is False:
283 
284             # If rocket thrust, calculate new velocity
285             if rocket_thrust is True:
286                 velocity_y = velocity_y - 2 * gravity - math.cos(math.radians(angle)) * 2 * gravity
287                 velocity_x += 2 * gravity * math.sin(math.radians(-angle))
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
288
289                 display_lander_image = pygame.transform.rotate(thrust_image, angle)
290
291                 # Reduce fuel
292                 fuel -= 1
293                 if fuel == 0:
294                     exploded = True
295                     fuel_left = False
296
297                 # Play thrust sound effect
298                 if pygame.mixer.get_busy() == 0:
299                     thrust_sound.play()
300
301             else:
302                 display_lander_image = pygame.transform.rotate(lander_image, angle)
303
304                 thrust_sound.stop()
305
306         else:
307             display_lander_image = pygame.transform.rotate(exploded_lander_image, angle)
308
309         # Gravity affects the y velocity
310         velocity_y += gravity
311
312         # Display background
313         game_screen.blit(background_image, [0, 0])
314
315         # Display planet
316         for block_number, block in enumerate(planet):
317             terrain_type = block.get('terrain_type')
318             terrain_y = block.get('terrain_y')
319
320             block_x = block_number * terrain_block_size
321             block_y = terrain_y * terrain_block_size
322
323             game_screen.blit(terrain_type, [block_x, block_y])
324
325             # Display blank blocks below terrain surface
326             for blank_block in range(terrain_y + 1, MAX_TERRAIN_HEIGHT):
327                 block_x = block_number * terrain_block_size
328                 block_y = blank_block * terrain_block_size
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
                game_screen.blit(planet_blank_image, [block_x, block_y])

        # Display lander
        game_screen.blit(display_lander_image, lander_rect)

        # Landed messages
        if hit_terrain is True:
            if landed is True:
                display_successful_landing()
                return_message_text = 'Hit return for next level.'
                level_score = fuel * level + (landing_angle - abs(angle)) * 10 * level

            else:
                display_failed_landing(on_screen, fuel_left, velocity_ok, landing_location_ok, angle_ok)
                level_score = 0
                return_message_text = 'Hit return for new game.'

            if total_score + level_score > hi_score:
                hi_score = total_score + level_score

            display_game_end(level_score, total_score + level_score, hi_score, return_message_text)

        else:
            display_status(fuel, velocity_x, landing_velocity, velocity_y, angle, landing_angle, level)

        if exploded is True:
            if level_end_sound_played is False:
                explosion_sound.play()
                level_end_sound_played = True

        if landed is True:
            if level_end_sound_played is False:
                landed_sound.play()
                level_end_sound_played = True

        pygame.display.update()
        clock.tick(60)


# Successful landing message
def display_successful_landing():
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
370
371     message_text = 'Congratulations Commander, a successful landing.'
372     text = font.render(message_text, True, GREEN)
373
374     text_rect = text.get_rect()
375     message_x = (SCREEN_WIDTH - text_rect.width) / 2
376     message_y = (SCREEN_HEIGHT - text_rect.height) / 2 - TEXT_LINE_SPACE * 6
377     game_screen.blit(text, [message_x, message_y])
378
379
380 # Failed landing message
381 def display_failed_landing(on_screen, fuel_left, velocity_ok, landing_location_ok, angle_ok):
382
383     message_text = 'Commander, you failed to land the spacecraft.'
384     text = font.render(message_text, True, GREEN)
385
386     text_rect = text.get_rect()
387     message_x = (SCREEN_WIDTH - text_rect.width) / 2
388     message_y = (SCREEN_HEIGHT - text_rect.height) / 2 - TEXT_LINE_SPACE * 7
389     game_screen.blit(text, [message_x, message_y])
390
391     fault_report_head = 'Fault Report'
392     text = font.render(fault_report_head, True, AMBER)
393
394     text_rect = text.get_rect()
395     message_x = (SCREEN_WIDTH - text_rect.width) / 2
396     message_y = (SCREEN_HEIGHT - text_rect.height) / 2 - TEXT_LINE_SPACE * 3
397     game_screen.blit(text, [message_x, message_y])
398
399     fault_text = ''
400     if on_screen is False:
401         fault_text += 'Left planet orbit'
402     else:
403         if fuel_left is False:
404             fault_text += 'No fuel. '
405         if velocity_ok is False:
406             fault_text += 'Approach velocity. '
407         if landing_location_ok is False:
408             fault_text += 'Missed landing pad. '
409         if angle_ok is False:
410             fault_text += 'Approach angle. '
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
411
412         text = font.render(fault_text, True, AMBER)
413         text_rect = text.get_rect()
414         message_x = (SCREEN_WIDTH - text_rect.width) / 2
415         message_y = (SCREEN_HEIGHT - text_rect.height) / 2 - TEXT_LINE_SPACE * 2
416         game_screen.blit(text, [message_x, message_y])
417
418
419     # End of game message
420     def display_game_end(level_score, game_score, hi_score, message):
421
422         score_text = 'Level ' + str(level_score) + '  -  Game ' + str(game_score) + '  -  Hi ' + str(hi_score)
423         text = font.render(score_text, True, GREEN)
424
425         text_rect = text.get_rect()
426         message_x = (SCREEN_WIDTH - text_rect.width) / 2
427         message_y = (SCREEN_HEIGHT - text_rect.height) / 2 - TEXT_LINE_SPACE * 5
428         game screen.blit(text, [message x, message y])
429
430         text = font.render(message, True, GREEN)
431         text_rect = text.get_rect()
432         message_x = (SCREEN_WIDTH - text_rect.width) / 2
433         message_y = (SCREEN_HEIGHT - text_rect.height) / 2 + TEXT_LINE_SPACE * 1
434         game_screen.blit(text, [message_x, message_y])
435
436
437     # On screen data display: Fuel, H Velocity, V Velocity, Angle and Level
438     def display_status(fuel, velocity_x, landing_velocity, velocity_y, angle, landing_angle, level):
439
440         fuel_text = 'Fuel:  ' + str(fuel)
441         if fuel < FUEL_WARNING:
442             text = font.render(fuel_text, True, AMBER)
443         else:
444             text = font.render(fuel_text, True, GREEN)
445         game_screen.blit(text, [DATA_DISPLAY_LEFT, TEXT_LINE_SPACE * 2])
446
447         rounded_vel_x = round(velocity_x, 2)
448         h_velocity_text = 'H Vel: ' + str(rounded_vel_x)
449         if abs(velocity_x) > landing_velocity:
450             text = font.render(h_velocity_text, True, AMBER)
451         else:
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
452         text = font.render(h_velocity_text, True, GREEN)
453     game_screen.blit(text, [DATA_DISPLAY_LEFT, TEXT_LINE_SPACE * 3])
454
455     rounded_vel_y = round(velocity_y, 2)
456     v_velocity_text = 'V Vel: ' + str(rounded_vel_y)
457     if velocity_y > landing_velocity:
458         text = font.render(v_velocity_text, True, AMBER)
459     else:
460         text = font.render(v_velocity_text, True, GREEN)
461
462     game_screen.blit(text, [DATA_DISPLAY_LEFT, TEXT_LINE_SPACE * 4])
463
464     angle_text = 'Angle: ' + str(-angle)
465     if abs(angle) > landing_angle:
466         text = font.render(angle_text, True, AMBER)
467     else:
468         text = font.render(angle_text, True, GREEN)
469     game screen.blit(text, [DATA DISPLAY LEFT, TEXT LINE SPACE * 5])
470
471     level_text = 'Level: ' + str(level)
472     text = font.render(level_text, True, GREEN)
473     game_screen.blit(text, [DATA_DISPLAY_LEFT, TEXT_LINE_SPACE * 6])
474
475
476 # Set up a planet terrain as list
477 def setup_planet(planet, landing_pad_start, landing_blocks, terrain_blocks):
478
479     del planet[:]
480
481     terrain_choice = ''
482
483     planet_images = {'hi_lo': planet_high_low_image,
484                      'hi_mid': planet_high_mid_image,
485                      'lo_hi': planet_low_high_image,
486                      'lo_mid': planet_low_mid_image,
487                      'mid': planet_mid_image,
488                      'mid_lo': planet_mid_low_image,
489                      'mid_hi': planet_mid_high_image,
490                      'blank': planet_blank_image,
491                      'landing': planet_landing_block_image}
492
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
493     new_block_last_mid = ['mid', 'higher', 'lower']
494
495     new_block_last_higher = ['higher', 'higher', 'higher', 'higher',
496                              'higher', 'higher', 'higher', 'higher',
497                              'higher', 'higher', 'mid', 'lower']
498
499     new_block_last_lower = ['lower', 'lower', 'lower', 'lower',
500                             'lower', 'lower', 'lower', 'lower',
501                             'lower', 'lower', 'mid', 'higher']
502
503     last_block = 'mid'
504
505     terrain_y_coord = int(MAX_TERRAIN_HEIGHT - ((MAX_TERRAIN_HEIGHT - MIN_TERRAIN_HEIGHT) / 2))
506
507     first_planet_block = {'terrain_type': planet_images['mid'], 'terrain_y': terrain_y_coord}
508     planet.append(first_planet_block)
509
510     for block in range(terrain_blocks):
511
512         if block == landing_pad_start - 1 or block == landing_pad_start + landing_blocks:
513             new_block = 'mid'
514         else:
515             if last_block == 'mid':
516                 new_block = random.choice(new_block_last_mid)
517             elif last_block == 'higher':
518                 new_block = random.choice(new_block_last_higher)
519             elif last_block == 'lower':
520                 new_block = random.choice(new_block_last_lower)
521
522         if landing_pad_start <= block < landing_pad_start + landing_blocks:
523             terrain_choice = planet_images['landing']
524         else:
525
526             if last_block == 'mid':
527
528                 if new_block == 'mid':
529                     terrain_choice = planet_images['mid']
530
531                 elif new_block == 'higher':
532                     terrain_choice = planet_images['mid_hi']
533
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
534                 elif new_block == 'lower':
535                     terrain_choice = planet_images['mid_lo']
536
537             elif last_block == 'higher':
538
539                 if new_block == 'mid':
540                     terrain_choice = planet_images['hi_mid']
541
542                 elif new_block == 'higher':
543                     terrain_choice = planet_images['lo_hi']
544                     terrain_y_coord -= 1
545
546                 elif new_block == 'lower':
547                     terrain_choice = planet_images['hi_lo']
548
549             elif last_block == 'lower':
550
551                 if new_block == 'mid':
552                     terrain_choice = planet_images['lo_mid']
553
554                 elif new_block == 'higher':
555                     terrain_choice = planet_images['lo_hi']
556
557                 elif new_block == 'lower':
558                     terrain_choice = planet_images['hi_lo']
559                     terrain_y_coord += 1
560
561         if terrain_y_coord < MIN_TERRAIN_HEIGHT:
562             terrain_y_coord = MIN_TERRAIN_HEIGHT
563             terrain_choice = planet_images['hi_mid']
564             new_block = random.choice(new_block_last_lower)
565
566         if terrain_y_coord > MAX_TERRAIN_HEIGHT - 1:
567             terrain_y_coord = MAX_TERRAIN_HEIGHT - 1
568             terrain_choice = planet_images['lo_mid']
569             new_block = random.choice(new_block_last_higher)
570
571         terrain_block = {'terrain_type': terrain_choice, 'terrain_y': terrain_y_coord}
572         planet.append(terrain_block)
573
574         last_block = new_block
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
575
576
577 # Generate a random x column location for the landing pad
578 def random_landing_pad_start(landing_blocks, terrain_blocks):
579     maximum_right_col = terrain_blocks - RIGHT_BUFFER_ZONE - landing_blocks
580     landing_pad_start = random.randint(LEFT_BUFFER_ZONE, maximum_right_col)
581
582     return landing_pad_start
583
584
585 # Generate a random starting x column for the lander
586 def random_lander_loc(landing_pad_start, landing_blocks, terrain_blocks):
587     maximum_right_col = terrain_blocks - RIGHT_BUFFER_ZONE - landing_blocks
588     lander_column = random.randint(LEFT_BUFFER_ZONE, maximum_right_col)
589
590     while landing_pad_start <= lander_column <= landing_pad_start + landing_blocks:
591         lander_column = random.randint(LEFT_BUFFER_ZONE, maximum_right_col)
592
593     return lander_column
594
595 if __name__ == '__main__':
596
597     main()
598
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

