SASTA OLIPHANT SCIENCE AWARDS

Prize Winner

Programming, Apps & Robotics

Year 7-8

Sandhu Sukhman Singh

Mount Carmel College

Atmospheric Adventure Report:

AIM, SCIENTIFIC PURPOSE AND POTENTIAL APPLICATIONS:

The Aim of this entry is to create a game using python to subtly educate students on various scientific topics using video games and programming, this could be used in schools to make learning extremely interactive and fun.

BIBLIOGRAPHY:

Sources that helped me learn to code in python:

Tech with tim (youtube channel)
Programming with Mosh (youtube channel)
Clear code (youtube channel)

Coding with russ (youtube channel)

Sources I accessed to find solutions to bugs:

Stack overflow (website)
Github (website)

IMPORTANT SECTIONS OF THE PROGRAM:

Sections:

1. Main loop (where all the functions are called and events are handled)

2. Movement (where movement is handled)

3. Levels class (where levels are handled)

The code is on the next page ->

1. The main loop:

```
# main loop of the game
def main():
  global pause_value
  global level_in_progress
  global task_in_progress
  global temp_asteroid_pos
  global mouse_pos
  global joysticks
  global gravity_button_pressed
  global variable_menu_active
  global controller_active
  run = True


  particle_event = pygame.USEREVENT + 1
  pygame.time.set_timer(particle_event, 100)

  while run:
    clock.tick(FPS)
    try:
      mouse_pos = pygame.mouse.get_pos()
      for event in pygame.event.get():
        if event.type == pygame.QUIT:
          run = False
          pygame.quit()
          sys.exit()


        # handling the lose and pause conditions
        if not lose:
          if event.type == pygame.KEYDOWN:
              if event.key == pygame.K_ESCAPE:

                if pause_value:
                  pause_value = False

                elif not pause_value:
                  pause_value = True

          if not pause_value:
            if event.type == pygame.JOYBUTTONDOWN:
              print(event)
```

```
            if level_in_progress == 2 and task_in_progress == 2:

                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_TAB and not controller_active:

                        if variable_menu_active:
                            variable_menu_active = False

                        elif not variable_menu_active:
                            variable_menu_active = True

                    if event.key == pygame.K_c and joysticks[0] != 0:

                        if controller_active:
                            controller_active = False

                        elif not controller_active:
                            controller_active = True

                        print(controller_active)

                elif event.type == pygame.JOYBUTTONDOWN and joysticks[0].get_button(6):

                    if variable_menu_active:
                        variable_menu_active = False

                    elif not variable_menu_active:
                        variable_menu_active = True

                if controller_active and not variable_menu_active and not
Next_level_Button.button_pressed():
                    if event.type == pygame.MOUSEBUTTONDOWN or event.type ==
pygame.JOYBUTTONDOWN and joysticks[0].get_button(7):
                        if joysticks[0].get_button(7):
                            if temp_asteroid_pos:
                                ast_to_sling = slingshot.create(temp_asteroid_pos, (end_of_line.centerx,
end_of_line.centery))
                                asteroids_to_slingshot.append(ast_to_sling)
                                temp_asteroid_pos = None
                            else:
                                temp_asteroid_pos = (launch_location.centerx, launch_location.centery)

                else:
                    if event.type == pygame.MOUSEBUTTONDOWN:
                        if not variable_menu_active and not Next_level_Button.button_pressed():
                            if temp_asteroid_pos:
                                ast_to_sling = slingshot.create(temp_asteroid_pos, mouse_pos)
                                asteroids_to_slingshot.append(ast_to_sling)
                                temp_asteroid_pos = None
                            else:
```

```
                        temp_asteroid_pos = mouse_pos

            if event.type == particle_event:
                if task_in_progress == 1:
                    particle1.add_cloud_particles()
                    particle1.add_fire_particles()
    except SystemError:
        print([x for x in pygame.event.get()])

    if not lose:
        if not pause_value:
            up_camera_offset()
            down_camera_offset()
            movement()
            altitude_meter_indicator()

            if satellite_rect.x > -325:
                satellite_rect.x -= 1
            else:
                satellite_rect.move_ip(WIDTH_screen + 325, 0)

            if satellite_rect2.x < WIDTH_screen:
                satellite_rect2.x += 1
            else:
                satellite_rect2.move_ip(-WIDTH_screen - 325, 0)
    else:
        pause_value = False
        if task_in_progress == 1:
            player_rect.x = 450 ; player_rect.y = 365

        elif level_in_progress == 1:
            player_rect.x = 500 ; player_rect.y = 200

    draw_screen()
    collision()
    pygame.display.update()


if __name__ == '__main__':
    main()
```

2. Movement function:

```
    # movement function
```

```
def movement():
  global PLAYER
  global lvl1_task2_ov_offset
  global player_rotation
  keys_pressed = pygame.key.get_pressed()

  if joysticks[0] != 0:

     if task_in_progress == 1:
        if keys_pressed[pygame.K_RIGHT] or keys_pressed[pygame.K_d] or
joysticks[0].get_axis(0) > .3:
           player_rotation = 270
           if player_rect.x < window_border.right_border():
              if joysticks[0].get_axis(0) > .3:
                 player_rect.x += joysticks[0].get_axis(0) * VELOCITY
              else:
                 player_rect.x += VELOCITY

        if keys_pressed[pygame.K_LEFT] or keys_pressed[pygame.K_a] or
joysticks[0].get_axis(0) < -.3:
           player_rotation = 90
           if player_rect.x > window_border.left_border():
              if joysticks[0].get_axis(0) < -.3:
                 player_rect.x += joysticks[0].get_axis(0) * VELOCITY
              else:
                 player_rect.x -= VELOCITY

        if keys_pressed[pygame.K_DOWN] or keys_pressed[pygame.K_s] or
joysticks[0].get_axis(1) > .3 and not keys_pressed[pygame.K_UP] and not
keys_pressed[pygame.K_w]:
           player_rotation = 180
           if not keys_pressed[pygame.K_w] or keys_pressed[pygame.K_UP]:
              if BACKGROUND_COLOUR[1] < 180:
                 BACKGROUND_COLOUR[1] += BACKGROUND_CHANGE_SPEED
              if BACKGROUND_COLOUR[2] < 255:
                 BACKGROUND_COLOUR[2] += BACKGROUND_CHANGE_SPEED
           if player_rect.y < window_border.bottom_border():
              player_rect.y += VELOCITY
```

```
        if keys_pressed[pygame.K_UP] or keys_pressed[pygame.K_w] or
joysticks[0].get_axis(1) < -.3 and not keys_pressed[pygame.K_DOWN] and not
keys_pressed[pygame.K_s]:
            player_rotation = 0
            if end_text_rect.y + ov_offset < 10:
                if not keys_pressed[pygame.K_s] or keys_pressed[pygame.K_DOWN]:
                    BACKGROUND_COLOUR[1] -= BACKGROUND_CHANGE_SPEED
                    BACKGROUND_COLOUR[2] -= BACKGROUND_CHANGE_SPEED
            if player_rect.y > window_border.top_border():
                player_rect.y -= VELOCITY

        if not keys_pressed[pygame.K_UP] and not keys_pressed[pygame.K_w] and not
keys_pressed[pygame.K_DOWN] and not keys_pressed[pygame.K_s] and not
keys_pressed[pygame.K_d] and not keys_pressed[pygame.K_RIGHT] and not
keys_pressed[pygame.K_a] and not keys_pressed[pygame.K_LEFT] and not
joysticks[0].get_axis(0) > .3 and not joysticks[0].get_axis(0) < -.3 and not
joysticks[0].get_axis(1) > .3 and not joysticks[0].get_axis(1) < -.3:
            player_rotation = 0
            PLAYER = pygame.transform.rotate(pygame.transform.scale(player_img,
(WIDTH_player, HEIGHT_player)), 0)
        else:
            PLAYER = pygame.transform.rotate(pygame.transform.scale(player_img,
(WIDTH_player, HEIGHT_player)), player_rotation)

    else:
        if level_in_progress == 1 and countdown // 100 < 0:
            if keys_pressed[pygame.K_RIGHT] or keys_pressed[pygame.K_d] or
joysticks[0].get_axis(0) > .3:
                if player_rect.x < window_border.right_border():
                    if joysticks[0].get_axis(0) > .3:
                        player_rect.x += joysticks[0].get_axis(0) * VELOCITY
                    else:
                        player_rect.x += VELOCITY


            if keys_pressed[pygame.K_LEFT] or keys_pressed[pygame.K_a] or
joysticks[0].get_axis(0) < -.3:
                if player_rect.x > window_border.left_border():
                    if joysticks[0].get_axis(0) < -.3:
                        player_rect.x += joysticks[0].get_axis(0) * VELOCITY
                    else:
                        player_rect.x -= VELOCITY
```

```
        if level_in_progress == 2 and controller_active:
            if temp_asteroid_pos:
                if joysticks[0].get_axis(0) > .3 and end_of_line.x < WIDTH_screen - 25:
                    end_of_line.x += joysticks[0].get_axis(0) * VELOCITY

                if joysticks[0].get_axis(0) < -.3 and end_of_line.x > 0:
                    end_of_line.x += joysticks[0].get_axis(0) * VELOCITY

                if joysticks[0].get_axis(1) > .3 and end_of_line.y < HEIGHT_screen - 25:
                    end_of_line.y += joysticks[0].get_axis(1) * VELOCITY

                if joysticks[0].get_axis(1) < -.3 and end_of_line.y > 0:
                    end_of_line.y += joysticks[0].get_axis(1) * VELOCITY
            else:
                if joysticks[0].get_axis(0) > .3 and launch_location.x < WIDTH_screen - 25:
                    launch_location.x += joysticks[0].get_axis(0) * VELOCITY

                if joysticks[0].get_axis(0) < -.3 and launch_location.x > 0:
                    launch_location.x += joysticks[0].get_axis(0) * VELOCITY

                if joysticks[0].get_axis(1) > .3 and launch_location.y < HEIGHT_screen - 25:
                    launch_location.y += joysticks[0].get_axis(1) * VELOCITY

                if joysticks[0].get_axis(1) < -.3 and launch_location.y > 0:
                    launch_location.y += joysticks[0].get_axis(1) * VELOCITY

        if level_in_progress == 3:
            pass

    print(joysticks[0].get_axis(0))

else:
    if task_in_progress == 1:
        if keys_pressed[pygame.K_RIGHT] or keys_pressed[pygame.K_d]:
            player_rotation = 270
            if player_rect.x < window_border.right_border():
                player_rect.x += VELOCITY

        if keys_pressed[pygame.K_LEFT] or keys_pressed[pygame.K_a]:
            player_rotation = 90
            if player_rect.x > window_border.left_border():
```

```
            player_rect.x -= VELOCITY

        if keys_pressed[pygame.K_DOWN] or keys_pressed[pygame.K_s] and not
keys_pressed[pygame.K_UP] and not keys_pressed[pygame.K_w]:
            player_rotation = 180
            if not keys_pressed[pygame.K_w] or keys_pressed[pygame.K_UP]:
                if BACKGROUND_COLOUR[1] < 180:
                    BACKGROUND_COLOUR[1] += BACKGROUND_CHANGE_SPEED
                if BACKGROUND_COLOUR[2] < 255:
                    BACKGROUND_COLOUR[2] += BACKGROUND_CHANGE_SPEED
            if player_rect.y < window_border.bottom_border():
                player_rect.y += VELOCITY


        if keys_pressed[pygame.K_UP] or keys_pressed[pygame.K_w] and not
keys_pressed[pygame.K_DOWN] and not keys_pressed[pygame.K_s]:
            player_rotation = 0
            if end_text_rect.y + ov_offset < 10:
                if not keys_pressed[pygame.K_s] or keys_pressed[pygame.K_DOWN]:
                    BACKGROUND_COLOUR[1] -= BACKGROUND_CHANGE_SPEED
                    BACKGROUND_COLOUR[2] -= BACKGROUND_CHANGE_SPEED
            if player_rect.y > window_border.top_border():
                player_rect.y -= VELOCITY


        if not keys_pressed[pygame.K_UP] and not keys_pressed[pygame.K_w] and not
keys_pressed[pygame.K_DOWN] and not keys_pressed[pygame.K_s] and not
keys_pressed[pygame.K_d] and not keys_pressed[pygame.K_RIGHT] and not
keys_pressed[pygame.K_a] and not keys_pressed[pygame.K_LEFT]:
            player_rotation = 0
            PLAYER = pygame.transform.rotate(pygame.transform.scale(player_img,
(WIDTH_player, HEIGHT_player)), 0)
        else:
            PLAYER = pygame.transform.rotate(pygame.transform.scale(player_img,
(WIDTH_player, HEIGHT_player)), player_rotation)

    else:
        if level_in_progress == 1 and countdown // 100 < 0:
            if keys_pressed[pygame.K_RIGHT] or keys_pressed[pygame.K_d]:
                if player_rect.x < window_border.right_border():
                    player_rect.x += VELOCITY
```

```
            if keys_pressed[pygame.K_LEFT] or keys_pressed[pygame.K_a]:
                if player_rect.x > window_border.left_border():
                    player_rect.x -= VELOCITY

        if level_in_progress == 2:
            pass

        if level_in_progress == 3:
            pass
```

3. The Levels class

```
class levels():

  def __init__(self, current_level, current_task) -> None:
      global player_rect
      self.current_level = current_level
      self.task = current_task
      self.level_text = word_font2.render(self.current_level + ' | ' + current_task + ': ' +
all_levels[self.current_level].get(current_task), True, 'white')
      if self.current_level == 'level 1' and self.task == 'task 2' and player_rect.x != -200 and
player_rect.y != 300:
          player_rect.x = 500 ; player_rect.y = 300


  def task_completetion(self):
      global Asteroids
      global lvl1_task2_ov_offset
      global countdown
      if self.task == 'task 1':

          if self.current_level == 'level 1':
              lvl1_task2_ov_offset = 0
              Asteroids = [[pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect1.x,
asteriod_rect1.y]],
                           [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect2.x,
asteriod_rect2.y]],
                           [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect3.x,
asteriod_rect3.y]],
```

```
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect4.x,
asteriod_rect4.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect5.x,
asteriod_rect5.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect6.x,
asteriod_rect6.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect7.x,
asteriod_rect7.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect8.x,
asteriod_rect8.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect9.x,
asteriod_rect9.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect10.x,
asteriod_rect10.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect11.x,
asteriod_rect11.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect12.x,
asteriod_rect12.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect13.x,
asteriod_rect13.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect14.x,
asteriod_rect14.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect15.x,
asteriod_rect15.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect16.x,
asteriod_rect16.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect17.x,
asteriod_rect17.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect18.x,
asteriod_rect18.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect19.x,
asteriod_rect19.y]]]

            return (satellite_rect.y + ov_offset) - player_rect.y in range(-300, 300) and
satellite_rect.x - player_rect.x in range(-300, 300)

        elif self.current_level == 'level 2':
            return (satellite_rect2.y + ov_offset) - player_rect.y in range(-300, 300) and
satellite_rect2.x - player_rect.x in range(-300, 300)

        elif self.current_level == 'level 3':
            pass
```

```
  def level_completetion(self):
    global lvl1_task2_ov_offset
    global Asteroids
    keys_pressed = pygame.key.get_pressed()

    if keys_pressed[pygame.K_1]:
      return [1,1]

    elif keys_pressed[pygame.K_2]:
      lvl1_task2_ov_offset = 0
      Asteroids = [[pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect1.x,
asteriod_rect1.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect2.x,
asteriod_rect2.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect3.x,
asteriod_rect3.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect4.x,
asteriod_rect4.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect5.x,
asteriod_rect5.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect6.x,
asteriod_rect6.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect7.x,
asteriod_rect7.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect8.x,
asteriod_rect8.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect9.x,
asteriod_rect9.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect10.x,
asteriod_rect10.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect11.x,
asteriod_rect11.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect12.x,
asteriod_rect12.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect13.x,
asteriod_rect13.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect14.x,
asteriod_rect14.y]],
                   [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect15.x,
asteriod_rect15.y]],
```

```
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect16.x,
asteriod_rect16.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect17.x,
asteriod_rect17.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect18.x,
asteriod_rect18.y]],
                [pygame.transform.rotate(asteriod1_img, 90), [asteriod_rect19.x,
asteriod_rect19.y]]]

        return [1,2]

    elif keys_pressed[pygame.K_3]:
        return [2,1]

    elif keys_pressed[pygame.K_4]:
        return [2,2]

    elif keys_pressed[pygame.K_5]:
        return [3,1]

    elif keys_pressed[pygame.K_6]:
        return [3,2]

    if self.task == 'task 2':
            if level_in_progress == 1:
                return round(lvl1_task2_ov_offset / 10000, 2) > 2.15

            if level_in_progress == 2:
                return Next_level_Button.button_pressed()

            if level_in_progress == 3:
                return 'win'


  def level_mainloop(self):
    global PLAYER
    global countdown
    global Asteroids
    global lose
    global temp_asteroid_pos
    global variable_menu_active
    global asteroid_mass_multiplier
```

```
global speed_multiplier
global Next_level_Button
keys_pressed = pygame.key.get_pressed()

if self.task == 'task 1':
    try:
        screen.fill(BACKGROUND_COLOUR)
    except Exception as e:
        screen.fill([0, 0, 75])
        print(e)

    particle1.control_fire_particles()
    screen.blit(PLAYER, (player_rect.x, player_rect.y))
    screen.blit(GROUND, (ground_rect.x, ground_rect.y + ov_offset))

    if not cloud_collision:
        screen.blit(CLOUD, (cloud_rect1.x, cloud_rect1.y + ov_offset))

    if not cloud_collision2:
        screen.blit(CLOUD2, (cloud_rect2.x, cloud_rect2.y + ov_offset))

    if not cloud_collision3:
        screen.blit(CLOUD3, (cloud_rect3.x, cloud_rect3.y + ov_offset))

    screen.blit(end_text, (end_text_rect.x, end_text_rect.y + ov_offset))
    screen.blit(SATELLITE, (satellite_rect.x ,satellite_rect.y + ov_offset))
    screen.blit(SATELLITE2, (satellite_rect2.x ,satellite_rect2.y + ov_offset))
    screen.blit(space_text, (space_text_rect.x, space_text_rect.y + ov_offset))
    screen.blit(ALTITUDE_INDICATOR, (altitude_pointer_rect.x, altitude_pointer_rect.y))
    screen.blit(ALTITUDE_METER, (0, 0))
    particle1.control_cloud_particles()

    if self.current_level == 'level 1':
        screen.blit(text_box_1, (255, 0))
        screen.blit(self.level_text, (260, 5))

    if self.current_level == 'level 2':
        screen.blit(text_box_2, (245, 0))
        screen.blit(self.level_text, (250, 5))

    if self.current_level == 'level 3':
        screen.blit(text_box_3, (170, 0))
```

```
            screen.blit(self.level_text, (175, 5))

        if pause_value:
            screen.blit(stop_screen_img, (50, 100))
            screen.blit(pause_screen_text, (300, 350))
            screen.blit(pause_screen_text2, (430, 630))

        if lose:
            screen.blit(stop_screen_img, (50, 100))
            screen.blit(lose_text, (375, 350))
            screen.blit(restart_text, (430, 630))

    else:
        if self.current_level == 'level 1':
            screen.fill([0, 0, 40])
            if not lose and not pause_value:
                if (countdown // 100) < 0:
                    distance_text = word_font2.render('Distance from Sun: ' +
str(round(lvl1_task2_ov_offset / 10000, 2)) + ' AU', True, 'white')
                    PLAYER = pygame.transform.rotate(pygame.image.load('oliphant
project/rocket.png'), 0)
                    screen.blit(PLAYER, (player_rect.x, player_rect.y))
                    screen.blit(sun_img, (sun_rect.x, sun_rect.y + lvl1_task2_ov_offset))
                    screen.blit(mercury_img, (mercury_rect.x, mercury_rect.y +
lvl1_task2_ov_offset))
                    screen.blit(mercury_text, (mercury_rect.x + 12, mercury_rect.y + 50 +
lvl1_task2_ov_offset))
                    screen.blit(mercury_text2, (mercury_rect.x + 30, mercury_rect.y + 95 +
lvl1_task2_ov_offset))
                    screen.blit(venus_img, (venus_rect.x, venus_rect.y + lvl1_task2_ov_offset))
                    screen.blit(venus_text, (venus_rect.x + 90, venus_rect.y + 125 +
lvl1_task2_ov_offset))
                    screen.blit(venus_text2, (venus_rect.x + 110, venus_rect.y + 175 +
lvl1_task2_ov_offset))
                    screen.blit(earth_img, (earth_rect.x, earth_rect.y + lvl1_task2_ov_offset))
                    screen.blit(earth_text, (earth_rect.x + 90, earth_rect.y + 125 +
lvl1_task2_ov_offset))
                    screen.blit(earth_text2, (earth_rect.x + 110, earth_rect.y + 175 +
lvl1_task2_ov_offset))
                    screen.blit(mars_img, (mars_rect.x, mars_rect.y + lvl1_task2_ov_offset))
                    screen.blit(mars_text, (mars_rect.x + 135, mars_rect.y + 145 +
lvl1_task2_ov_offset))
```

```
                screen.blit(mars_text2, (mars_rect.x + 122, mars_rect.y + 195 +
lvl1_task2_ov_offset))
                if len(Asteroids):
                    asteroid = Asteroids[0]
                    if asteroid[1][1] + lvl1_task2_ov_offset > HEIGHT_screen:
                        Asteroids.remove(asteroid)
                    else:
                        asteroid_mask = pygame.mask.from_surface(asteroid[0])
                        asteroid[1][1] += VELOCITY // 4
                        if asteroid_mask.overlap(player_mask, ((player_rect.x - asteroid[1][0]),
(player_rect.y) - (asteroid[1][1] + lvl1_task2_ov_offset))) and not
keys_pressed[pygame.K_i]:
                            lose = True
                        if mercury_mask.overlap(player_mask, ((player_rect.x - mercury_rect.x),
(player_rect.y) - (mercury_rect.y + lvl1_task2_ov_offset))) and not
keys_pressed[pygame.K_i]:
                            lose = True
                        if venus_mask.overlap(player_mask, ((player_rect.x - venus_rect.x),
(player_rect.y) - (venus_rect.y + lvl1_task2_ov_offset))) and not
keys_pressed[pygame.K_i]:
                            lose = True
                        if earth_mask.overlap(player_mask, ((player_rect.x - earth_rect.x),
(player_rect.y) - (earth_rect.y + lvl1_task2_ov_offset))) and not keys_pressed[pygame.K_i]:
                            lose = True
                        if mars_mask.overlap(player_mask, ((player_rect.x - mars_rect.x),
(player_rect.y) - (mars_rect.y + lvl1_task2_ov_offset))) and not keys_pressed[pygame.K_i]:
                            lose = True
                        screen.blit(asteroid[0], (asteroid[1][0], asteroid[1][1] +
lvl1_task2_ov_offset))

                screen.blit(self.level_text, (490, 5))
                screen.blit(distance_text, (0, 5))
                screen.blit(ratio_text, (0, 40))

            else:
                countdown -= 1
                if not countdown // 100 == 0:
                    if not countdown // 100 < 0:
                        screen.blit(word_font4.render(str(countdown // 100), True, 'white'),
(400, 200))
                else:
                    screen.blit(word_font4.render('GO', True, 'white'), (300, 200))
```

```
            if lose:
                screen.blit(stop_screen_img, (50, 100))
                screen.blit(lose_text, (375, 350))
                screen.blit(restart_text, (430, 630))

            if pause_value:
                screen.blit(stop_screen_img, (50, 100))
                screen.blit(pause_screen_text, (300, 350))
                screen.blit(pause_screen_text2, (430, 630))

        elif self.current_level == 'level 2':
            screen.fill([0, 0, 40])
            Planet.draw()
            screen.blit(self.level_text, (170, 5))
            screen.blit(word_font2.render('Speed Multiplier: ' + str((round(speed_multiplier,
1))) + 'x', True, 'white'), (700, 100))
            screen.blit(word_font2.render('Asteroid Mass: ' +
str((round(asteroid_mass_multiplier, 1))) + 'x', True, 'white'), (700, 150))
            screen.blit(word_font2.render('Planet Mass: ' + str((round(Planet.mass / 50, 1))) +
' Jupiter(s)', True, 'white'), (700, 200))
            screen.blit(word_font2.render('Gravitational Force: ' + str(round(force * 250)) + '
m/s²', True, 'white'), (700, 250))
            screen.blit(word_font2.render('Tip: click TAB to open menu and click on the top or
bottom to increase or decrease the variables', True, 'white'), (0, 700))
            if controller_active:
                screen.blit(word_font2.render('Controller active', True, 'white'), (700, 50))
            else:
                screen.blit(word_font2.render('Controller inactive', True, 'white'), (700, 50))

            if not controller_active:
                Asteroid_Mass_Button = Button(pygame.Rect(70, 310, 200, 100), mouse_pos,
'Object Mass', 10, 'dark gray')
                Speed_Button = Button(pygame.Rect(70, 110, 200, 100), mouse_pos, 'Speed',
10, 'dark gray')
                Planet_Mass_Button = Button(pygame.Rect(70, 510, 200, 100), mouse_pos,
'Planet Mass', 10, 'dark gray')
                Next_level_Button = Button(pygame.Rect(800, 650, 150, 75), mouse_pos, 'Next
>>', 10, 'dark gray')

                if Asteroid_Mass_Button.button_pressed() and variable_menu_active:
```

```
                if mouse_pos[1] < Asteroid_Mass_Button.rect.centery and
asteroid_mass_multiplier < 5:
                    asteroid_mass_multiplier += 0.01

                elif mouse_pos[1] > Asteroid_Mass_Button.rect.centery and
asteroid_mass_multiplier > 1:
                    asteroid_mass_multiplier -= 0.01

                elif mouse_pos[1] == Asteroid_Mass_Button.rect.centery:
                    asteroid_mass_multiplier += 0.01

            elif Speed_Button.button_pressed() and variable_menu_active:

                if mouse_pos[1] < Speed_Button.rect.centery:
                    speed_multiplier += 0.01

                elif mouse_pos[1] > Speed_Button.rect.centery and speed_multiplier > 1:
                    speed_multiplier -= 0.01

                elif mouse_pos[1] == Speed_Button.rect.centery:
                    speed_multiplier += 0.01

            elif Planet_Mass_Button.button_pressed() and variable_menu_active:

                if mouse_pos[1] < Planet_Mass_Button.rect.centery:
                    Planet.mass += 0.5

                elif mouse_pos[1] > Planet_Mass_Button.rect.centery and Planet.mass / 50 >
1:
                    Planet.mass -= 0.5

                elif mouse_pos[1] == Planet_Mass_Button.rect.centery:
                    Planet.mass += 0.5

        elif controller_active:
            Asteroid_Mass_Button = Button(pygame.Rect(70, 310, 200, 100),
(launch_location.x, launch_location.y), 'Object Mass', 10, 'dark gray')
            Speed_Button = Button(pygame.Rect(70, 110, 200, 100), (launch_location.x,
launch_location.y), 'Speed', 10, 'dark gray')
            Planet_Mass_Button = Button(pygame.Rect(70, 510, 200, 100),
(launch_location.x, launch_location.y), 'Planet Mass', 10, 'dark gray')
```

```
            Next_level_Button = Button(pygame.Rect(800, 650, 200, 100),
(launch_location.centerx, launch_location.centery), 'Next >>', 10, 'dark gray')

            if Asteroid_Mass_Button.button_pressed() and variable_menu_active:

                if launch_location.y < Asteroid_Mass_Button.rect.centery and
asteroid_mass_multiplier < 5:
                    asteroid_mass_multiplier += 0.01

                elif launch_location.y > Asteroid_Mass_Button.rect.centery and
asteroid_mass_multiplier > 1:
                    asteroid_mass_multiplier -= 0.01

                elif launch_location.y == Asteroid_Mass_Button.rect.centery:
                    asteroid_mass_multiplier += 0.01

            elif Speed_Button.button_pressed() and variable_menu_active:

                if launch_location.y < Speed_Button.rect.centery:
                    speed_multiplier += 0.01

                elif launch_location.y > Speed_Button.rect.centery and speed_multiplier > 1:
                    speed_multiplier -= 0.01

                elif launch_location.y == Speed_Button.rect.centery:
                    speed_multiplier += 0.01

            elif Planet_Mass_Button.button_pressed() and variable_menu_active:

                if launch_location.y < Planet_Mass_Button.rect.centery:
                    Planet.mass += 0.5

                elif launch_location.y > Planet_Mass_Button.rect.centery and Planet.mass /
50 > 1:
                    Planet.mass -= 0.5

                elif launch_location.y == Planet_Mass_Button.rect.centery:
                    Planet.mass += 0.5

        if not variable_menu_active:

            for asteroid in asteroids_to_slingshot:
```

```
                asteroid.draw()
                asteroid.movement(Planet)
                off_screen = asteroid.x < -10 or asteroid.x > WIDTH_screen + 10 or
asteroid.y < -10 or asteroid.y > HEIGHT_screen + 10
                collided = math.sqrt((asteroid.x - Planet.x) ** 2 + (asteroid.y - Planet.y) ** 2)
< 384 // 2 + 5
                if off_screen or collided:
                    asteroids_to_slingshot.remove(asteroid)

            if controller_active:
                if temp_asteroid_pos:
                    pygame.draw.line(screen, 'white', temp_asteroid_pos,
(end_of_line.centerx, end_of_line.centery), 4)
                    pygame.draw.circle(screen, 'dark gray', temp_asteroid_pos,
asteroid_to_slingshot_size)
            else:
                if temp_asteroid_pos:
                    pygame.draw.line(screen, 'white', temp_asteroid_pos, mouse_pos, 4)
                    pygame.draw.circle(screen, 'dark gray', temp_asteroid_pos,
asteroid_to_slingshot_size)

        else:
            screen.blit(variable_menu_img, (0, 10))
            Asteroid_Mass_Button.draw()
            Speed_Button.draw()
            Planet_Mass_Button.draw()
            Next_level_Button.draw()

        if controller_active:
            if not temp_asteroid_pos:
                pygame.draw.rect(screen, 'red', (launch_location.x, launch_location.y,
launch_location.width, launch_location.height))

        if lose:
            screen.blit(stop_screen_img, (50, 100))
            screen.blit(lose_text, (375, 350))
            screen.blit(restart_text, (430, 630))

        if pause_value:
            screen.blit(stop_screen_img, (50, 100))
            screen.blit(pause_screen_text, (300, 350))
            screen.blit(pause_screen_text2, (430, 630))
```

```
elif self.current_level == 'level 3':
    screen.fill([0, 0, 40])
    screen.blit(self.level_text, (350, 5))

    if lose:
        screen.blit(stop_screen_img, (50, 100))
        screen.blit(lose_text, (375, 350))
        screen.blit(restart_text, (430, 630))

    if pause_value:
        screen.blit(stop_screen_img, (50, 100))
        screen.blit(pause_screen_text, (300, 350))
        screen.blit(pause_screen_text2, (430, 630))
```