HeartBeatHorseRace Step-By-Step

main.py

1-77 Setting up the server and the global variables

86-202:

class MainScreen(self): creates the main screen and calibrates the horses to zero, no bluetooth function at all.

```python
class MainScreen(Screen):
    """
    Class to handle the main screen and its associated touch events
    """
    count = 0
    elapsed = ObjectProperty()

    # Automatically runs when MainScreen is created. Homes/Calibrates the oDrives if not already done, and sets up the
    # server. 'IF STATEMENT' WILL NOT FULLY RUN UNTIL A CONNECTION TO A CLIENT IS FOUND.
    def beginning_setup(self):
        global homed, serverCreated
        if homed is False:
            assert od_1.config.enable_brake_resistor is True, "Check for faulty brake resistor."
            assert od_2.config.enable_brake_resistor is True, "Check for faulty brake resistor."

            horse1.set_gains()
            horse2.set_gains()
            horse3.set_gains()
            horse4.set_gains()

            # Checks the ODrive Calibraton
            if not horse1.is_calibrated():
                print("calibrating horse1...")
                horse1.calibrate_with_current_lim(15)
            if not horse2.is_calibrated():
                print("calibrating horse2...")
                horse2.calibrate_with_current_lim(15)
            if not horse3.is_calibrated():
                print("calibrating horse3...")
                horse3.calibrate_with_current_lim(15)
            if not horse4.is_calibrated():
                print("calibrating horse4...")
                horse4.calibrate_with_current_lim(15)

            sleep(3)

            # Homes the Horses to Left Side
            horses = [horse1, horse2, horse3, horse4]
            for horse in horses:
                horse.set_ramped_vel(1, 1)
            sleep(1)
            for horse in horses:
                horse.wait_for_motor_to_stop()  # waiting until motor slowly hits wall
            for horse in horses:
                horse.set_pos_traj(horse.get_pos() - 0.5, 1, 2, 1)
            sleep(3)  # allows motor to start moving to offset position
            for horse in horses:
                horse.wait_for_motor_to_stop()
            for horse in horses:
                horse.set_home()

            horse1.set_vel(0)
            horse2.set_vel(0)
            horse3.set_vel(0)
            horse4.set_vel(0)

            od_1.clear_errors()
            od_2.clear_errors()

            if serverCreated is True:
                s.send_packet(PacketType.COMMAND0, b'game start')

            if not horse1.is_calibrated():
                print("calibrating horse1...")
                horse1.calibrate_with_current_lim(15)
            if not horse2.is_calibrated():
                print("calibrating horse2...")
                horse2.calibrate_with_current_lim(15)
            if not horse3.is_calibrated():
                print("calibrating horse3...")
                horse3.calibrate_with_current_lim(15)
            if not horse4.is_calibrated():
                print("calibrating horse4...")
                horse4.calibrate_with_current_lim(15)

            homed = True
            return homed

    def redraw(self, args):
        self.bg_rect.size = self.size
        self.bg_rect.pos = self.pos

    def switch_to_traj(self):
        # No button currently calls this function. Used for demonstrating control over the horses, but not necessary
        # for the game.
        SCREEN_MANAGER.transition.direction = "left"
        SCREEN_MANAGER.current = TRAJ_SCREEN_NAME

    def switch_to_beginning(self):
        global new_game
        new_game = False
        SCREEN_MANAGER.transition.direction = "down"
        SCREEN_MANAGER.current = BEGINNING_SCREEN_NAME
        print(str(new_game))
        return new_game

    def admin_action(self):
        """
        Hidden admin button touch event. Transitions to passCodeScreen.
        This method is called from pidev/kivy/PassCodeScreen.kv
        :return: None
        """
        SCREEN_MANAGER.current = 'passCode'

    print("screen 1 created")

    def quit(self):
        print("Exit")
        if serverCreated is True:
            s.send_packet(PacketType.COMMAND0, b'quitting')
            sleep(2)
            s.close_connection()
            print('connection closed')
            s.close_server()
            print('server closed')
            sleep(2)
        quit()
```

204-350:

class BeginningScreen(Screen): creates beginning screen and all the adapters needed (adapter is a pygatt.BGAPI object that might be like a server to connecting the horsebeat, might not be needed)

1. def stop_all_adapters(self): stops all the adapters
2. def quit(self): quits and closes connection to the server
3. then makes one, two, or three number of adapters start and one_player start or more

```python
class BeginningScreen(Screen):
    """
    Class to handle the beginning screen and its associated events for starting a race
    """

    def stop_all_adapters(self):
        try:
            adapter1.stop()
        except Exception as e:
            print(f'adapter 1 didnt stop: {e}')
        try:
            adapter2.stop()
        except Exception as e:
            print(f'adapter 1 didnt stop: {e}')
        try:
            adapter3.stop()
        except Exception as e:
            print(f'adapter 1 didnt stop: {e}')
        try:
            adapter4.stop()
        except Exception as e:
            print(f'adapter 1 didnt stop: {e}')
        finally:
            return

    def quit(self):
        print("Exit")
        s.close_connection()
        print('connection closed')
        s.close_server()
        print('server closed')
        quit()

    def switch_screen1(self):
        SCREEN_MANAGER.transition.direction = "down"
        SCREEN_MANAGER.current = MAIN_SCREEN_NAME

    def move_to_baseline_screen(self):
        SCREEN_MANAGER.transition.direction = "left"
        SCREEN_MANAGER.current = BASELINE_SCREEN_NAME

    # Starting the adapters does NOT require human interaction
    def one_adapater_start(self):
        try:
            adapter1.start()
        except:
            print("not working :(")
        finally:
            print('adapter1 started')

    def two_adapter_start(self):
        try:
            adapter1.start()
        except:
            print("not working p1:(")
        try:
            adapter2.start()
        except:
            print("not working p2:(")
        finally:
            print('adapter1 started')
            print('adapter2 started')

    def three_adapter_start(self):
        try:
            adapter1.start()
        except:
            print("not working p1:(")
        try:
            adapter2.start()
        except:
            print("not working p2:(")
        try:
            adapter3.start()
        except:
            print("not working p3:(")
        finally:
            print('adapter1 started')
            print('adapter2 started')
            print('adapter3 started')

    def four_adapter_start(self):
        try:
            adapter1.start()
        except:
            print("not working p1:(")
        try:
            adapter2.start()
        except:
            print("not working p2:(")
        try:
            adapter3.start()
        except:
            print("not working p3:(")
        try:
            adapter4.start()
        except:
            print("not working p4:(")

        finally:
            print('adapter1 started')
            print('adapter2 started')
            print('adapter3 started')
            print('adapter4 started')

    # Defines number of players for each game.
    def one_player(self):
        global numberOfPlayers

        print('a')
        self.one_adapater_start()
        print('b')
        self.move_to_baseline_screen()

        numberOfPlayers = 1
        return numberOfPlayers

    def two_players(self):
        global numberOfPlayers

        self.two_adapter_start()
        self.move_to_baseline_screen()

        numberOfPlayers = 2
        return numberOfPlayers

    def three_players(self):
        global numberOfPlayers

        self.three_adapter_start()
        self.move_to_baseline_screen()

        numberOfPlayers = 3
        return numberOfPlayers

    def four_players(self):
        global numberOfPlayers

        self.four_adapter_start()
        self.move_to_baseline_screen()

        numberOfPlayers = 4
        return numberOfPlayers

    print("Beginning Screen Created")

```

350-633:
class BaselineScreen(Screen): creates connection and finds average heartrate depending on number of players

1. quit FN
2. def find_baseline(self): Finds the baseline heartrate for specified number of players
    1. For every player, connects the adapter to the device id
    2. Subscribes to the UUID for heartrate
    3. Averages the heartrates
    4. Repeats for more players
    5. Returns baselines, the average heartrates, verniers - the adapters connected to HR sensors, i, and if it is homed
3. switch_screen(self): switches screen to BEGINNING_SCREEN_NAME variable

```python


        self.three_adapter_start()
        self.move_to_baseline_screen()

        numberOfPlayers = 3
        return numberOfPlayers

    def four_players(self):
        global numberOfPlayers

        self.four_adapter_start()
        self.move_to_baseline_screen()

        numberOfPlayers = 4
        return numberOfPlayers

    print("Beginning Screen Created")

class BaselineScreen(Screen):
    """
    Class to handle the baseline screen and finding each individual's heart rate
    """
    def quit(self):
        print("Exit")
        s.close_connection()
        print('connection closed')
        s.close_server()
        print('server closed')
        quit()

    # Finds the baseline heartrate for specified number of players after pressing the pixelated heart.
    def find_baseline(self):
        global baseline1, baseline2, baseline3, baseline4, vernier1, vernier2, vernier3, vernier4, i, homed, serverCreated
        baseline1List_real = True
        baseline2List_real = True
        baseline3List_real = True
        baseline4List_real = True

        if numberOfPlayers == 1:
            if serverCreated is True:
                s.send_packet(PacketType.COMMAND0, b'baseline')
            # Refer to dpea-p2p repo for more info on sending packets
            try:
                vernier1 = adapter1.connect(player1.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier2 = 0
                vernier3 = 0
                vernier4 = 0

            except:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return


            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(1))
                while i < 6:
                    sleep(1)
                    i += 1
                    # When loop breaks, the subscription breaks as well! Refer to HeartrateRate Example Repo

            except:
                print('weeeeee')
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return

            # Calls baseline to default at 60 if no data is pulled from the sensors.
            finally:
                if not baseline1List:
                    baseline1 = 60

                else:
                    baseline1 = round(average_heartrate(baseline1List))

                SCREEN_MANAGER.transition.direction = "right"
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'start')
                SCREEN_MANAGER.current = RUN_SCREEN_NAME

                print(homed)

        elif numberOfPlayers == 2:
            if serverCreated is True:
                s.send_packet(PacketType.COMMAND0, b'baseline')
            try:
                vernier1 = adapter1.connect(player1.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier2 = adapter2.connect(player2.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier3 = 0
                vernier4 = 0

            except:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return


            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(1))
                vernier2.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(2))
                while i < 6:
                    sleep(1)
                    i += 1

            except:
                print('L')
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return

            finally:
                if not baseline1List:
                    baseline1 = 60
                    if not baseline2List:
                        baseline2 = 60
                    else:
                        baseline2 = round(average_heartrate(baseline2List))

                elif not baseline2List:
                    baseline2 = 60
                    if not baseline1List:
                        baseline1 = 60
                    else:
                        baseline1 = round(average_heartrate(baseline1List))

                else:
                    baseline1 = round(average_heartrate(baseline1List))
                    baseline2 = round(average_heartrate(baseline2List))

                SCREEN_MANAGER.transition.direction = "right"
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'start')
                SCREEN_MANAGER.current = RUN_SCREEN_NAME

                print(homed)

        elif numberOfPlayers == 3:
            if serverCreated is True:
                s.send_packet(PacketType.COMMAND0, b'baseline')
            try:
                vernier1 = adapter1.connect(player1.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier2 = adapter2.connect(player2.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier3 = adapter3.connect(player3.deviceID)
                vernier4 = 0

            except:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return

            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(1))
                vernier2.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(2))
                vernier3.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(3))
                while i < 6:
                    sleep(1)
                    i += 1

            except:
                print('L')
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return

            finally:
                if not baseline1List:
                    baseline1List_real = False

                if not baseline2List:
                    baseline2List_real = False

                if not baseline3List:
                    baseline3List_real = False

                if baseline1List_real is False:
                    baseline1 = 60
                else:
                    baseline1 = round(average_heartrate(baseline1List))

                if baseline2List_real is False:
                    baseline2 = 60
                else:
                    baseline2 = round(average_heartrate(baseline2List))

                if baseline3List_real is False:
                    baseline3 = 60
                else:
                    baseline3 = round(average_heartrate(baseline3List))


                SCREEN_MANAGER.transition.direction = "right"
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'start')
                SCREEN_MANAGER.current = RUN_SCREEN_NAME

                print(homed)

        elif numberOfPlayers == 4:
            print('a')
            if serverCreated is True:
                s.send_packet(PacketType.COMMAND0, b'baseline')
            print('b')
            try:
                vernier1 = adapter1.connect(player1.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier2 = adapter2.connect(player2.deviceID, address_type=pygatt.BLEAddressType.random)
                vernier3 = adapter3.connect(player3.deviceID)
                vernier4 = adapter4.connect(player4.deviceID)
                print('c')
                vernier2 = adapter2.connect(player2.deviceID, address_type=pygatt.BLEAddressType.random)
                print('d')
                vernier3 = adapter3.connect(player3.deviceID)
                print('e')
                vernier4 = adapter4.connect(player4.deviceID)
                print('f')

            except:
                print('exception occured #1')
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return

            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(1))
                vernier2.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(2))
                vernier3.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(3))
                vernier4.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=heartrate_baseline(4))
                print('subscribing')
                while i < 6:
                    sleep(1)
                    i += 1

            except:
                print('L')
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'error')
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
                return


            finally:
                print('found finally')
                if not baseline1List:
                    baseline1List_real = False

                if not baseline2List:
                    baseline2List_real = False

                if not baseline3List:
                    baseline3List_real = False

                if not baseline4List:
                    baseline4List_real = False

                if baseline1List_real is False:
                    baseline1 = 60
                else:
                    baseline1 = round(average_heartrate(baseline1List))

                if baseline2List_real is False:
                    baseline2 = 60
                else:
                    baseline2 = round(average_heartrate(baseline2List))

                if baseline3List_real is False:
                    baseline3 = 60
                else:
                    baseline3 = round(average_heartrate(baseline3List))

                if baseline4List_real is False:
                    baseline4 = 60
                else:
                    baseline4 = round(average_heartrate(baseline4List))

                SCREEN_MANAGER.transition.direction = "right"
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'start')
                    sleep(1)
                SCREEN_MANAGER.current = RUN_SCREEN_NAME

                print(homed)

        else:
            print('not working L')
            return


        return baseline1, baseline2, baseline3, baseline4, vernier1, vernier2, vernier3, vernier4, i, homed

    def switch_screen(self):
        SCREEN_MANAGER.transition.direction = "right"
        SCREEN_MANAGER.current = BEGINNING_SCREEN_NAME

    print("Baseline Screen Created")
```

634-950:

class RunScreen(Screen):

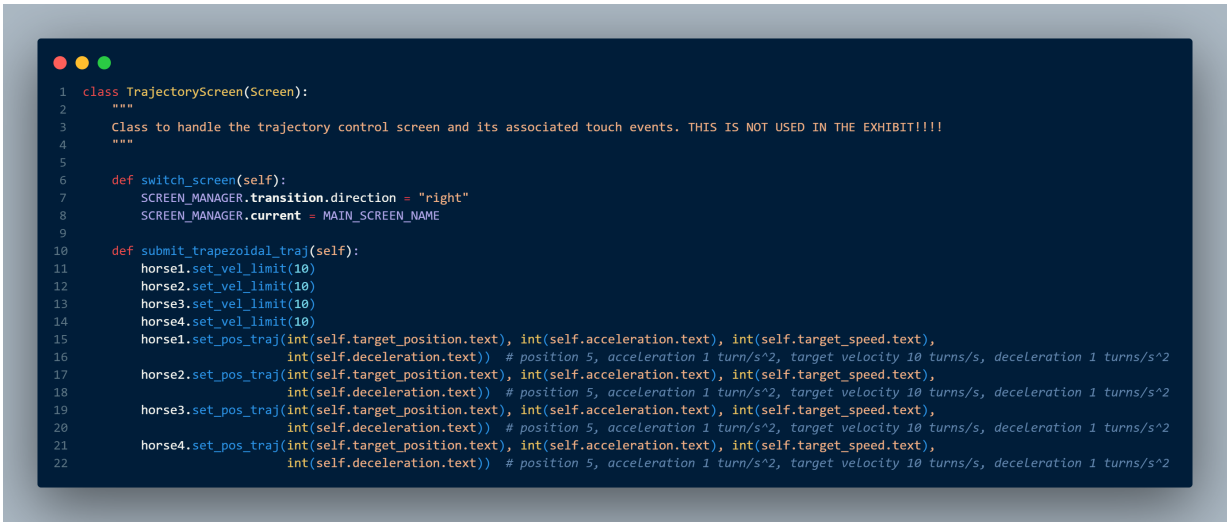
1. Quit FN as always
2. Updates baseline for all players
3. def start_game(self):
    1. Checks for the number of people
    2. subscribes to UUID through vernier
    3. Setting horse velocity to 0
    4. homes and changes screen to MAIN_SCREEN_NAME

```python
class RunScreen(Screen):
    """
    Class to handle the run screen and heart beat horse race
    """
    def quit(self):
        print("Exit")
        s.close_connection()
        print('connection closed')
        s.close_server()
        print('server closed')
        quit()

    def update_baseline(self):
        self.ids.player1Baseline.text = str(baseline1)
        self.ids.player2Baseline.text = str(baseline2)
        self.ids.player3Baseline.text = str(baseline3)
        self.ids.player4Baseline.text = str(baseline4)
        player1.baseline_rate = baseline1
        player2.baseline_rate = baseline2
        player3.baseline_rate = baseline3
        player4.baseline_rate = baseline4

    def start_game(self):
        global new_game, serverCreated
        t = "3"
        while int(t) > 0:
            t = str(int(t) - 1)
            sleep(1)
        self.ids.count.text = "GO!"
        sleep(.5)
        if numberOfPlayers == 1:
            try:
                print(1)
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(1))
                print(2)
                player1.start_game()
                print(3)

                while True:
                    print("in the true statement")
                    player_ticker(numberOfPlayers)
                    time.sleep(1)
                    print("while True is running")
                    if player1.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND0, b'WIN')
                        sleep(10)
                        break
                    # To end a subscription, you MUST break the while loop.

            except:
                print("unlucky race stopped")
                pass

            finally:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'done')
                horse1.set_vel(0)
                horse2.set_vel(0)
                horse3.set_vel(0)
                horse4.set_vel(0)

                player1.is_done = True
                player2.is_done = True
                player3.is_done = True
                player4.is_done = True
                player1.game_done()
                player2.game_done()
                player3.game_done()
                player4.game_done()
                player1.laps = 0
                sleep(3)
                print('new game')

                try:
                    adapter1.stop()
                except:
                    print('sucks')

                home_all_horses()

                try:
                    adapter1.stop()
                except:
                    print('sucks')

                SCREEN_MANAGER.transition.direction = "right"
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME

        elif numberOfPlayers == 2:
            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(1))
                vernier2.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(2))

                player1.start_game()
                player2.start_game()

                while True:
                    time.sleep(1)
                    player_ticker(numberOfPlayers)
                    print("while True is running")
                    if player1.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND0, b'WIN')
                        sleep(10)
                        break
                    elif player2.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND1, b'WIN')
                        sleep(10)
                        break

            except:
                print("unlucky race stopped")
                pass

            finally:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'done')
                horse1.set_vel(0)
                horse2.set_vel(0)
                horse3.set_vel(0)
                horse4.set_vel(0)

                player1.is_done = True
                player2.is_done = True
                player3.is_done = True
                player4.is_done = True
                player1.game_done()
                player2.game_done()
                player3.game_done()
                player4.game_done()
                player1.laps = 0
                player2.laps = 0
                sleep(3)
                print('new game')

                try:
                    adapter1.stop()
                    adapter2.stop()
                except:
                    print('sucks')

                home_all_horses()

                try:
                    adapter1.stop()
                    adapter2.stop()
                except:
                    print('sucks')


                SCREEN_MANAGER.transition.direction = "right"
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME


        elif numberOfPlayers == 3:
            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(1))
                vernier2.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(2))
                vernier3.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(3))

                player1.start_game()
                player2.start_game()
                player3.start_game()

                while True:
                    time.sleep(1)
                    player_ticker(numberOfPlayers)
                    print("while True is running")
                    if player1.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND0, b'WIN')
                        sleep(10)
                        break
                    elif player2.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND1, b'WIN')
                        sleep(10)
                        break
                    elif player3.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND2, b'WIN')
                        sleep(10)
                        break

            except:
                print("unlucky race stopped")
                pass

            finally:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'done')
                horse1.set_vel(0)
                horse2.set_vel(0)
                horse3.set_vel(0)
                horse4.set_vel(0)

                player1.is_done = True
                player2.is_done = True
                player3.is_done = True
                player4.is_done = True
                player1.game_done()
                player2.game_done()
                player3.game_done()
                player4.game_done()
                player1.laps = 0
                player2.laps = 0
                player3.laps = 0
                sleep(3)
                print('new game')

                try:
                    adapter1.stop()
                    adapter2.stop()
                    adapter3.stop()
                except:
                    print('sucks')


                home_all_horses()

                try:
                    adapter1.stop()
                    adapter2.stop()
                    adapter3.stop()
                except:
                    print('sucks')

                SCREEN_MANAGER.transition.direction = "right"
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME

        elif numberOfPlayers == 4:
            try:
                vernier1.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(1))
                vernier2.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(2))
                vernier3.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(3))
                vernier4.subscribe("00002a37-0000-1000-8000-00805f9b34fb", callback=setup(4))

                player1.start_game()
                player2.start_game()
                player3.start_game()
                player4.start_game()

                while True:
                    time.sleep(1)
                    player_ticker(numberOfPlayers)
                    player_ticker4()
                    print("while True is running")
                    if player1.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND0, b'WIN')
                        sleep(10)
                        break
                    elif player2.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND1, b'WIN')
                        sleep(10)
                        break
                    elif player3.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND2, b'WIN')
                        sleep(10)
                        break
                    elif player4.get_laps() >= total_laps:
                        if serverCreated is True:
                            s.send_packet(PacketType.COMMAND3, b'WIN')
                        sleep(10)
                        break

            except:
                print("unlucky race stopped")
                pass

            finally:
                if serverCreated is True:
                    s.send_packet(PacketType.COMMAND0, b'done')
                horse1.set_vel(0)
                horse2.set_vel(0)
                horse3.set_vel(0)
                horse4.set_vel(0)

                player1.is_done = True
                player2.is_done = True
                player3.is_done = True
                player4.is_done = True
                player1.game_done()
                player2.game_done()
                player3.game_done()
                player4.game_done()
                player1.laps = 0
                player2.laps = 0
                player3.laps = 0
                player4.laps = 0
                sleep(3)
                print('new game')

                try:
                    adapter1.stop()
                    adapter2.stop()
                    adapter3.stop()
                    adapter4.stop()
                except:
                    print('sucks')

                home_all_horses()

                try:
                    adapter1.stop()
                    adapter2.stop()
                    adapter3.stop()
                    adapter4.stop()
                except:
                    print('sucks')

                SCREEN_MANAGER.transition.direction = "right"
                SCREEN_MANAGER.current = MAIN_SCREEN_NAME
```

954-975:

class TrajectoryScreen(Screen):

1. switch_screen(self): switches screen to MAIN_SCREEN_NAME
2. submit_trapezoidal_traj(self): sets all horse velocity limits to 10 and then sets all horses' position trajectory to position 5, acceleration to 1 turn/sec^2, target velocity to 10 turns, deceleration to 1 turn/sec^2



```python
class TrajectoryScreen(Screen):
    """
    Class to handle the trajectory control screen and its associated touch events. THIS IS NOT USED IN THE EXHIBIT!!!!
    """

    def switch_screen(self):
        SCREEN_MANAGER.transition.direction = "right"
        SCREEN_MANAGER.current = MAIN_SCREEN_NAME

    def submit_trapezoidal_traj(self):
        horse1.set_vel_limit(10)
        horse2.set_vel_limit(10)
        horse3.set_vel_limit(10)
        horse4.set_vel_limit(10)
        horse1.set_pos_traj(int(self.target_position.text), int(self.acceleration.text), int(self.target_speed.text),
                            int(self.deceleration.text))  # position 5, acceleration 1 turn/s^2, target velocity 10 turns/s, deceleration 1 turns/s^2
        horse2.set_pos_traj(int(self.target_position.text), int(self.acceleration.text), int(self.target_speed.text),
                            int(self.deceleration.text))  # position 5, acceleration 1 turn/s^2, target velocity 10 turns/s, deceleration 1 turns/s^2
        horse3.set_pos_traj(int(self.target_position.text), int(self.acceleration.text), int(self.target_speed.text),
                            int(self.deceleration.text))  # position 5, acceleration 1 turn/s^2, target velocity 10 turns/s, deceleration 1 turns/s^2
        horse4.set_pos_traj(int(self.target_position.text), int(self.acceleration.text), int(self.target_speed.text),
                            int(self.deceleration.text))  # position 5, acceleration 1 turn/s^2, target velocity 10 turns/s, deceleration 1 turns/s^2
```

978-1024:

class AdminScreen(Screen):

1. __init__(self): builds the current screen and checks for a correct password to change to a screen
2. def transition_back(): transitions back to the MAIN SCREEN
3. def shutdown(): shuts down the system
4. def exit_program(): Quits the program

```python
class AdminScreen(Screen):
    """
    Class to handle the AdminScreen and its functionality.
    """

    def __init__(self, **kwargs):
        """
        Load the AdminScreen.kv file. Set the necessary names of the screens for the PassCodeScreen to transition to.
        Lastly super Screen's __init__
        :param kwargs: Normal kivy.uix.screenmanager.Screen attributes
        """
        Builder.load_file('AdminScreen.kv')

        PassCodeScreen.set_admin_events_screen(
            ADMIN_SCREEN_NAME)  # Specify screen name to transition to after correct password
        PassCodeScreen.set_transition_back_screen(
            MAIN_SCREEN_NAME)  # set screen name to transition to if "Back to Game is pressed"

        super(AdminScreen, self).__init__(**kwargs)

        print("admin screen created")

    @staticmethod
    def transition_back():
        """
        Transition back to the main screen
        :return:
        """
        SCREEN_MANAGER.current = MAIN_SCREEN_NAME

    @staticmethod
    def shutdown():
        """
        Shutdown the system. This should free all steppers and do any cleanup necessary
        :return: None
        """
        os.system("sudo shutdown now")

    @staticmethod
    def exit_program():
        """
        Quit the program. This should free all steppers and do any cleanup necessary
        :return: None
        """
        quit()

    print("static methods created")
```

1031-1043:

builds the files in the folder and runs the screens in order

```
Builder.load_file('main.kv')
Builder.load_file('BeginningScreen.kv')
Builder.load_file('BaselineScreen.kv')
Builder.load_file('RunScreen.kv')
SCREEN_MANAGER.add_widget(MainScreen(name=MAIN_SCREEN_NAME))
SCREEN_MANAGER.add_widget(TrajectoryScreen(name=TRAJ_SCREEN_NAME))
SCREEN_MANAGER.add_widget(BeginningScreen(name=BEGINNING_SCREEN_NAME))
SCREEN_MANAGER.add_widget(BaselineScreen(name=BASELINE_SCREEN_NAME))
SCREEN_MANAGER.add_widget(RunScreen(name=RUN_SCREEN_NAME))
SCREEN_MANAGER.add_widget(PassCodeScreen(name='passCode'))
SCREEN_MANAGER.add_widget(PauseScreen(name='pauseScene'))
SCREEN_MANAGER.add_widget(AdminScreen(name=ADMIN_SCREEN_NAME))
print("various screens created")
```