alien_invasion.py

```
import sys
from time import sleep

import pygame

from settings import Settings
from game_stats import GameStats
from scoreboard import Scoreboard
from button import Button
from ship import Ship
from bullet import Bullet
from alien import Alien
```

This is a Python script that imports various modules and classes for building a game using the Pygame library.

- The first line imports the built-in **sys** module, which provides access to some system-specific parameters and functions.
- The second line imports the **sleep** function from the **time** module, which is used to pause the program for a specified amount of time.
- The third line imports the **pygame** module, which is a set of Python modules designed for writing video games. This module provides access to various resources such as graphics, sound, and input devices.
- The next few lines import various classes from custom modules that are used to build the game. These modules include:
    - **Settings**: a class for storing game settings such as screen size and speed.
    - **GameStats**: a class for tracking game statistics such as score and level.
    - **Scoreboard**: a class for displaying the score and other relevant information.
    - **Button**: a class for creating buttons that can be clicked by the user.
    - **Ship**: a class for representing the player's spaceship.
    - **Bullet**: a class for representing bullets fired by the player's spaceship.
    - **Alien**: a class for representing the alien spaceships that the player must destroy.
- Overall, these import statements are used to ensure that all necessary modules and classes are available for building the game.

Class AlienInvasion

```
class AlienInvasion:
    """Overall class to manage game assets and behavior."""
```

```
    def __init__(self):
        """Initialize the game, and create game resources."""
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings()

        self.screen = pygame.display.set_mode(
            (self.settings.screen_width, self.settings.screen_height))
        pygame.display.set_caption("Alien Invasion")

        # Create an instance to store game statistics,
        #   and create a scoreboard.
        self.stats = GameStats(self)
        self.sb = Scoreboard(self)

        self.ship = Ship(self)
        self.bullets = pygame.sprite.Group()
        self.aliens = pygame.sprite.Group()

        self._create_fleet()

        # Start Alien Invasion in an inactive state.
        self.game_active = False

        # Make the Play button.
        self.play_button = Button(self, "Play")
```

This code block defines an **__init__** method for a class, which is called when a new instance of the class is created. This method initializes the game and creates various game resources.

- The first line is a docstring that briefly describes what the method does.
- **pygame.init()** initializes the Pygame library.
- **self.clock = pygame.time.Clock()** creates a Pygame clock object to help regulate the game's frame rate.
- **self.settings = Settings()** creates an instance of the **Settings** class, which stores game settings such as screen size and speed.
- **self.screen = pygame.display.set_mode((self.settings.screen_width, self.settings.screen_height))** creates a game window with the specified size using the **set_mode** method of the **display** module in Pygame.
- **pygame.display.set_caption("Alien Invasion")** sets the caption of the game window to "Alien Invasion".
- **self.stats = GameStats(self)** creates an instance of the **GameStats** class, which tracks game statistics such as score and level.

- **self.sb = Scoreboard(self)** creates an instance of the **Scoreboard** class, which displays the score and other relevant information.
- **self.ship = Ship(self)** creates an instance of the **Ship** class, which represents the player's spaceship.
- **self.bullets = pygame.sprite.Group()** creates an empty sprite group to hold the bullets fired by the player's spaceship.
- **self.aliens = pygame.sprite.Group()** creates an empty sprite group to hold the alien spaceships.
- **self._create_fleet()** calls a method to create the fleet of alien spaceships.
- **self.game_active = False** sets the game to be inactive when it starts.
- **self.play_button = Button(self, "Play")** creates a "Play" button that can be clicked to start the game. The button is an instance of the **Button** class.

```
    def run_game(self):
        """Start the main loop for the game."""
        while True:
            self._check_events()

            if self.game_active:
                self.ship.update()
                self._update_bullets()
                self._update_aliens()

            self._update_screen()
            self.clock.tick(60)
```

This code block defines a **run_game** method for a class, which starts the main loop for the game.

- The first line is a docstring that briefly describes what the method does.
- The **while** loop will run continuously until the program is terminated.
- **self._check_events()** is a method call that checks for user events such as key presses and mouse clicks.
- The **if** statement checks whether the game is currently active.
- If the game is active, **self.ship.update()** updates the position of the player's spaceship based on user input.
- **self._update_bullets()** updates the position of the bullets fired by the player's spaceship.
- **self._update_aliens()** updates the position of the alien spaceships and checks for collisions between the player's bullets and the aliens.
- **self._update_screen()** updates the screen with the latest game elements.

- **self.clock.tick(60)** limits the frame rate of the game to 60 frames per second by pausing the game for a short amount of time between each frame. This helps ensure that the game runs at a consistent speed across different computers.

```
    def _check_events(self):
        """Respond to keypresses and mouse events."""
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                self._check_keydown_events(event)
            elif event.type == pygame.KEYUP:
                self._check_keyup_events(event)
            elif event.type == pygame.MOUSEBUTTONDOWN:
                mouse_pos = pygame.mouse.get_pos()
                self._check_play_button(mouse_pos)
```

This code block defines a **_check_events** method for a class, which responds to key presses and mouse events.

- The first line is a docstring that briefly describes what the method does.
- The **for** loop iterates through a list of all the events that have occurred since the last time the loop was run.
- The **if** statement checks whether the **QUIT** event has occurred, which happens when the user clicks the close button on the game window or presses the "X" button on the window frame. If this event has occurred, the **sys.exit()** function is called to exit the game.
- The **elif** statement checks whether a **KEYDOWN** event has occurred, which happens when the user presses a key on the keyboard. If this event has occurred, the **_check_keydown_events** method is called with the **event** object as an argument.
- The **elif** statement checks whether a **KEYUP** event has occurred, which happens when the user releases a key on the keyboard. If this event has occurred, the **_check_keyup_events** method is called with the **event** object as an argument.
- The **elif** statement checks whether a **MOUSEBUTTONDOWN** event has occurred, which happens when the user clicks the mouse button. If this event has occurred, the position of the mouse cursor is obtained using the **pygame.mouse.get_pos()** function, and the **_check_play_button** method is called with the mouse position as an argument. This method checks whether the "Play" button has been clicked and starts the game if it has.

```
    def _check_play_button(self, mouse_pos):
        """Start a new game when the player clicks Play."""
        button_clicked = self.play_button.rect.collidepoint(mouse_pos)
        if button_clicked and not self.game_active:
```

```
            # Reset the game settings.
            self.settings.initialize_dynamic_settings()

            # Reset the game statistics.
            self.stats.reset_stats()
            self.sb.prep_score()
            self.sb.prep_level()
            self.sb.prep_ships()
            self.game_active = True

            # Get rid of any remaining bullets and aliens.
            self.bullets.empty()
            self.aliens.empty()

            # Create a new fleet and center the ship.
            self._create_fleet()
            self.ship.center_ship()

            # Hide the mouse cursor.
            pygame.mouse.set_visible(False)
```

This code block defines a **_check_play_button** method for a class, which starts a new game when the player clicks the "Play" button.

- The first line is a docstring that briefly describes what the method does.
- **mouse_pos** is a parameter that represents the current position of the mouse cursor.
- **self.play_button.rect.collidepoint(mouse_pos)** checks whether the mouse cursor is currently over the "Play" button.
- If the mouse cursor is over the "Play" button and the game is not currently active (**not self.game_active**), the following steps are performed to start a new game:
    - **self.settings.initialize_dynamic_settings()** resets the game settings to their default values.
    - **self.stats.reset_stats()** resets the game statistics to their default values.
    - **self.sb.prep_score()**, **self.sb.prep_level()**, and **self.sb.prep_ships()** update the scoreboard to display the new game statistics.
    - **self.game_active** is set to **True** to indicate that the game is now active.
    - **self.bullets.empty()** and **self.aliens.empty()** remove any remaining bullets and aliens from the previous game.
    - **self._create_fleet()** creates a new fleet of aliens.
    - **self.ship.center_ship()** centers the player's spaceship on the screen.
    - **pygame.mouse.set_visible(False)** hides the mouse cursor during gameplay.

```
    def _check_keydown_events(self, event):
        """Respond to keypresses."""
        if event.key == pygame.K_RIGHT:
            self.ship.moving_right = True
        elif event.key == pygame.K_LEFT:
            self.ship.moving_left = True
        elif event.key == pygame.K_q:
            sys.exit()
        elif event.key == pygame.K_SPACE:
            self._fire_bullet()
```

This code block defines a **_check_keydown_events** method for a class, which is called in response to keypress events.

- The first line is a docstring that briefly describes what the method does.
- **event** is a parameter that represents the keypress event that triggered the method.
- The method checks which key was pressed and performs a specific action for each key:
    - If the right arrow key (**pygame.K_RIGHT**) was pressed, the **moving_right** attribute of the player's spaceship is set to **True**.
    - If the left arrow key (**pygame.K_LEFT**) was pressed, the **moving_left** attribute of the player's spaceship is set to **True**.
    - If the "q" key (**pygame.K_q**) was pressed, the game is terminated using **sys.exit()**.
    - If the spacebar (**pygame.K_SPACE**) was pressed, the **_fire_bullet** method is called to fire a bullet from the player's spaceship.