

```
import sys

import pygame

from settings import Settings
from ship import Ship
from bullet import Bullet
from alien import Alien


class AlienInvasion:
    #Overall class to manage game assets and behavior

    def __init__(self):
        #Initialize the game, and create game resources

        pygame.init()
        self.settings = Settings()

        # Tell pygame to determine the size of the screen and set the screen
        width and height based on the players screen size
        self.screen = pygame.display.set_mode ((0,0), pygame.FULLSCREEN)
        self.settings.screen_width = self.screen.get_rect().width
        self.settings.screen_height = self.screen.get_rect().height

        pygame.display.set_caption ("Sharons Alien Invasion")

        # Set the background color - colors are RBG colors: amix of red, green,
        and blue. Each color range is 0 to 255
        self.bg_color = (200, 230, 230)

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

        # Add in the aliens
        self.aliens = pygame.sprite.Group()
        self._create_fleet()

    def run_game(self):
        #Start the main loop for the game

        while True:
            # call a method to check to see if any keyboard events have occurred
            self._check_events()
```

```

        self.ship.update()
        self._update_bullets()
        self._update.aliens()
        self._update_screen()

def _check_events(self):
    #Respond to keypresses and mouse events.
    # Did the player quit the game?
    for event in pygame.event.get():
        if event.type ==pygame.QUIT:
            sys.exit()
        # Did the player press the right or left arrow key?
        elif event.type == pygame.KEYDOWN:
            self._check_keydown_events(event)
        # Did the player stop holding down the arrow key?
        elif event.type == pygame.KEYUP:
            self._check_keyup_events(event)

def _check_keydown_events(self, event):
    # Is the key the right arrow or is it the left arrow
    if event.key == pygame.K_RIGHT:
        self.ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        self.ship.moving_left = True
    # Did the player hit the Q key to quite the game?
    elif event.key == pygame.K_q:
        sys.exit()
    # Did the player hit the space bar to shoot a bullet?
    elif event.key == pygame.K_SPACE:
        self._fire_bullet()

def _check_keyup_events(self, event):
    # Did the player stop holding down the arrow keys?
    if event.key == pygame.K_RIGHT:
        self.ship.moving_right = False
    elif event.key ==pygame.K_LEFT:
        self.ship.moving_left = False

def _fire_bullet(self):
    #Create a new bullet and add it to the bullets group
    #Limited the number of bullets a player can have at a time by adding a
constant to the settings file
    if len(self.bullets) < self.settings.bullets_allowed:
        new_bullet = Bullet(self)
        self.bullets.add(new_bullet)

```

```

def _update_bullets(self):
    #Update positions of the bullets and get rid of old bullets.
    self.bullets.update()

    # Get rid of bullets that have disappeared off the screen because they are
    # still there in the game and take up memory and execution time
    for bullet in self.bullets.copy():
        if bullet.rect.bottom <=0:
            self.bullets.remove(bullet)

    self._check_bullet_alien_collisions()

def _check_bullet_alien_collisions(self):
    # Respond to bullet-alien collisions
    # Check for any bullets that have hit aliens. If so, get rid of the
    # bullet and alien
    collisions = pygame.sprite.groupcollide(self.bullets, self.aliens, True,
    True)

    # Check to see if the aliens group is empty and if so, create a new
    # fleet
    if not self.aliens:
        # Destroy any existing bullets and create a new fleet
        self.bullets.empty()
        self._create_fleet()

def _update_aliens(self):
    # Update the position of all aliens in the fleet
    # Check if the fleet is at an edge then update the positions of all
    # aliens in the fleet
    self._check_fleet_edges()
    self.aliens.update()

    # Look for alien-ship collisions
    if pygame.sprite.spritecollideany(self.ship, self.aliens):
        print ("SHIP IT!!!")

# Add a method to create a fleet of Aliens
def _create_fleet(self):
    """Create the fleet of aliens"""
    # Make a single alien.
    aliens = Alien(self)
    alien_width, alien_height = aliens.rect.size
    # Determine how much space you have on the screen for aliens
    available_space_x = self.settings.screen_width - (2*alien_width)

```

```

number_aliens_x = available_space_x // (2 * alien_width)

#Determine the number of rows of aliens that fit on the screen
ship_height = self.ship.rect.height
available_space_y = (self.settings.screen_height -
                    (3 * alien_height) - ship_height)
number_rows = available_space_y // (2 * alien_height)

# Create the full fleet of aliens
for row_number in range (number_rows):
    for alien_number in range (number_aliens_x):
        self._create_alien(alien_number, row_number)

def _create_alien(self, alien_number, row_number):
    # Create an alien and place it in the row.
    aliens = Alien(self)
    alien_width, alien_height = aliens.rect.size
    alien_width = aliens.rect.width
    aliens.x = alien_width + 2 * alien_width * alien_number
    aliens.rect.x = aliens.x
    aliens.rect.y = alien_height + 2 * aliens.rect.height * row_number
    self.aliens.add(aliens)

def _check_fleet_edges(self):
    # Respond appropriately if any aliens have reached an edge
    for alien in self.aliens.sprites():
        if alien.check_edges():
            self._change_fleet_direction()
            break

def _change_fleet_direction(self):
    # Drop the entire fleet and change the fleet's direction
    for alien in self.aliens.sprites():
        alien.rect.y += self.settings.fleet_drop_speed
    self.settings.fleet_direction *= -1

def _update_screen(self):
    #Update images on the screen, and flip to the new screen.
    # Redraw the screen each pass through the loop
    self.screen.fill(self.settings.bg_color)
    self.ship.blitme()
    # Draw bullets on the screen
    for bullet in self.bullets.sprites():
        bullet.draw_bullet()

```

```
        #Draw the alien
        self.aliens.draw(self.screen)

        # Make the most recently drawn screen visible
        pygame.display.flip()

if __name__ == '__main__':
    # Make a game instance, and run the game
    ai = AlienInvasion()
    ai.run_game()

quit()
```