Python project from the book python crash course

ALIEN Invasion

```python
import pygame


def run_game():
    # Initialize pygame, settings, and screen object.

    pygame.init()
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")

    # Start the main loop for the game.
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

        pygame.display.flip()


run_game()
```

Step 1:

Creating a window:

Initially, we need to introduce pygame library and afterward import the library in alien_invasion.py. Presently begin creating a game. Make a capacity pygame.init. This strategy is utilized to make foundation settings. Also, we utilized pygame.display.set_mode to make the screen show. After that, we set the tallness and width of the screen to be shown. At that point, we utilized while circle and for circle. We made for the occasion which helps the player or client to a remarkable game.

```
import pygame
import sys


def run_game():
    # Initialize pygame, settings, and screen object.
    pygame.init()
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    # Set the background color.
    bg_color = (230, 230, 230)
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
        # Redraw the screen during each pass through the loop.
        screen.fill(bg_color)
        # Make the most recently drawn screen visible.
        pygame.display.flip()

run_game()
```

Setting the bg-color:

Pygame makes a dark screen naturally that is the reason we changed the tone by utilizing bg_color strategy. After that we use screen.fill (bg_color) to make the screen obvious.

OUTPUT:

```python
class Settings():
    """A class to store all settings for Alien Invasion."""

    def __init__(self):
        """Initialize the game's settings."""
        # Screen settings
        self.screen_width = 1200
        self.screen_height = 700
        self.bg_color = (230, 230, 230)
```

make another python record settings.py. Use class settings and added screen width, heigth and bg coloe tone.

```python
import sys
import pygame
from settings import Settings


def run_game():
    # Initialize pygame, settings, and screen object.
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode(
        (ai_settings.screen_width, ai_settings.screen_height))
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    bg_color = (230, 230, 230)
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

        screen.fill(ai_settings.bg_color)
        pygame.display.flip()


run_game()
```

Import settings in main program and store this in ai_settings. Now ai_settings.bg_color used to fill the background screen.

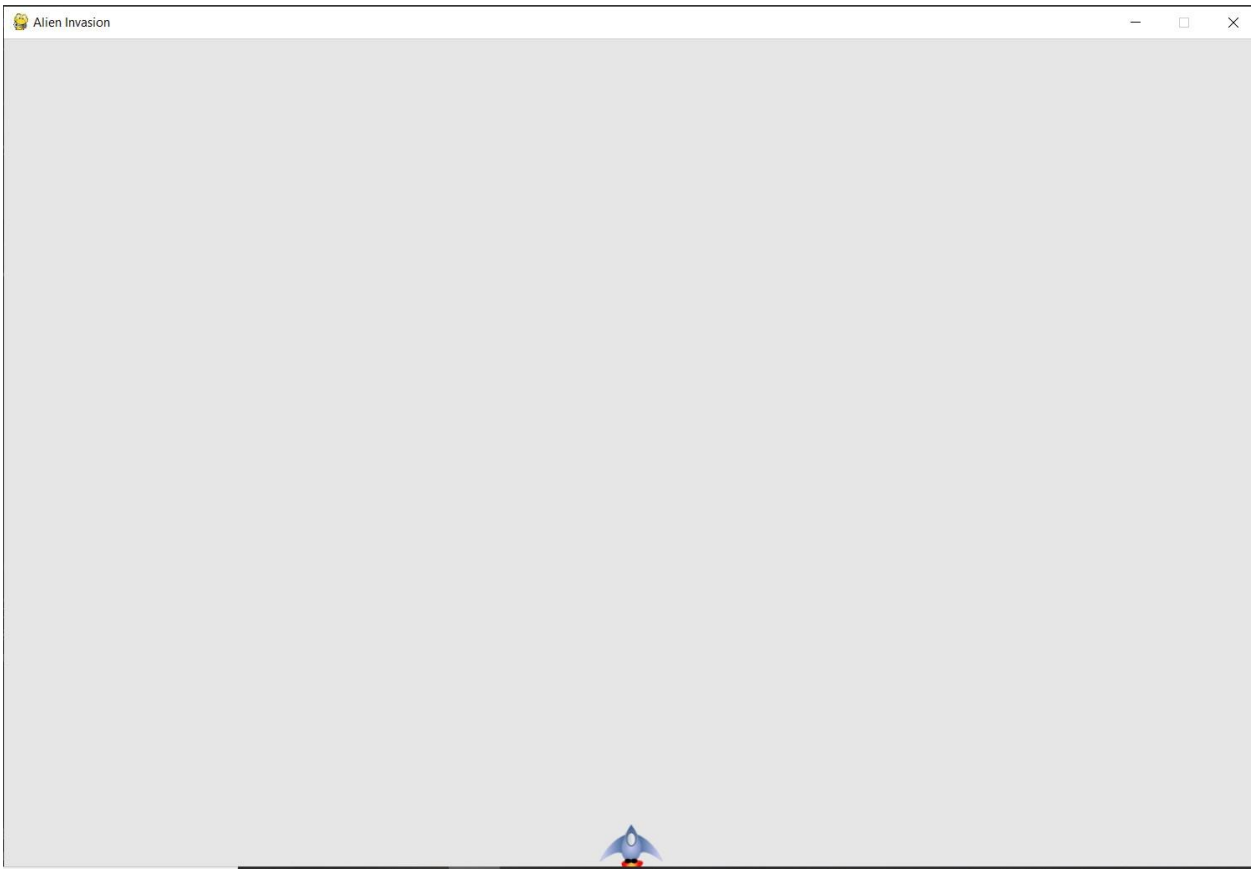
```python
import pygame


class Ship():

    def __init__(self, screen):
        """Initialize the ship and set its starting position."""
        self.screen = screen
        self.image = pygame.image.load('images/ship.bmp')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()
        self.rect.centerx = self.screen_rect.centerx
        self.rect.bottom = self.screen_rect.bottom

    def blitme(self):
        """Draw the ship at its current location."""
        self.screen.blit(self.image, self.rect)
```

Now, create a new python file name ship. In ship.py we used the class ship. In this class we created init function. After that we downloaded the ship image and used it. To extract the image from exact location we used image.rect method. Image appears at the bottom center of the screen. Secondly, we create a function blit. To update the screen.

```python
from ship import ship


def run_game():
    # Initialize pygame, settings, and screen object.
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode(
        (ai_settings.screen_width, ai_settings.screen_height))
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    bg_color = (230, 230, 230)
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

        screen.fill(ai_settings.bg_color)
        ship.blitme()
        pygame.display.flip()


run_game()
```

In main program we import a ship.py file and its method.

OUTPUT:




```python
import sys
import pygame


def check_events():
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

We create a new python file name game_function. in this file we created a new function called check events. In the function copy a for loop. Its already declare in a main program.

```python
from settings import Settings
from ship import Ship
import game_functions as gf


def run_game():
    # Initialize pygame, settings, and screen object.

    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode(
        (ai_settings.screen_width, ai_settings.screen_height))
    pygame.display.set_caption("Alien Invasion")
    ship = Ship(screen)
    bg_color = (230, 230, 230)

    while True:
        gf.check_events()
        screen.fill(ai_settings, bg_color)
        ship.bitme()
        pygame.display.flip()


run_game()
```

In a main program we import a game function file. and access its library by the help of gf method. In while loop we us gf.check events and remove a for loop path.

```python
import sys
import pygame


def check_events():
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()


def update_screen(ai_settings, screen, ship):
    screen.fill(ai_settings.bg_color)
    ship.blitme()
    pygame.display.flip()
```

In game function file we create a new function name update screen. And copy a method. Its already declare in main function.

```python
import sys

import pygame

from settings import Settings
from ship import Ship
import game_functions as gf


def run_game():
    # Initialize pygame, settings, and screen object.

    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode(
        (ai_settings.screen_width, ai_settings.screen_height))
    pygame.display.set_caption("Alien Invasion")
    ship = Ship(screen)
    bg_color = (230, 230, 230)

    while True:
        gf.check_events()
        gf.update_screen(ai_settings, screen, ship)


run_game()
```

Now, in main program we use update function and remove method which we declared in while loop because it is already declare in update function.

```python
import sys
import pygame


def check_events():
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                ship.rect.centerx += 1


def update_screen(ai_settings, screen, ship):
    screen.fill(ai_settings.bg_color)
    ship.bitme()
    pygame.display.flip()
```

We modify our program. In a game function file, we modify a check event function () and add ship in parameter. In check event function we add pygame.keydown. when user press right so ship move to right.

```python
import sys
import pygame
from settings import Settings
from ship import Ship
import game_functions as gf


def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode(
        (ai_settings.screen_width, ai_settings.screen_height))
    pygame.display.set_caption("Alien Invasion")
    ship = Ship(screen)
    bg_color = (230, 230, 230)

    while True:
        gf.check_events(ship)
        gf.update_screen(ai_settings, screen, ship)


run_game()
```

In this step we added ship check event parameter in a main program.

```python
    self.moving_right = False

    def update(self):
        if self.moving_right:
            self.rect.centerx += 1


def blitme(self):
    self.screen.blit(self.image, self.rect)
```

In a ship.py file we created update function () and we declare it when the user pressed right key the flag is true and the ship moves after that we added flag in a init function

```
elif event.type == pygame.KEYDOWN:
    if event.key == pygame.K_RIGHT:
        ship.rect.centerx += 1
        ship.moving_right = True
elif event.type == pygame.KEYUP:
    if event.key == pygame.K_RIGHT:
        ship.moving_right = False
```

In a game function, we modified check event function. In this function we create another elif block. In this block we use pygame.keyup so that when the user stop pressing a key the flag is flag is false.

```
while True:
    gf.check_events(ship)
    ship.update()
    gf.update_screen(ai_settings, screen, ship)
```

In a main program we create a ship update method.

```
    self.moving_right = False
    self.moving_left = False

def update(self):
        if self.moving_right:
            self.rect.centerx += 1
        if self.moving_left:
            self.rect.centerx -= 1
```

In ship.py file. We modify update function, it is same as we done above when the user press the left key the ship moves left.

```
    if event.key == pygame.K_RIGHT:
        ship.rect.centerx += 1
        ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        ship.moving_left = True
```

In a game function file we add pygame.leftkey in a check down block. Same process we did for right key.

```
class Settings():
    """A class to store all settings for Alien Invasion."""

    def __init__(self):
        """Initialize the game's settings."""
        # Screen settings
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230, 230, 230)
        self.ship_speed_factor = 1.5
```

In settings.py file we added ship speed.

```
import pygame


class Ship():

    def __init__(self, screen):

        self.screen = screen
        self.image = pygame.image.load('images/ship.bmp')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()
        self.rect.centerx = self.screen_rect.centerx
        self.rect.bottom = self.screen_rect.bottom
        self.center = float(self.rect.centerx)
        self.moving_right = False
        self.moving_left = False

    def update(self):
        if self.moving_right:
            self.rect.centerx += 1
        if self.moving_left:
            self.rect.centerx -= 1
```

Now, we added new things in ship file. Rect.centerx method is use to control the position of a ship. Self. Center adjust amount of speed.

```
def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_height))
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    ship = Ship(ai_settings, screen)

    bg_color = (230, 230, 230)

    while True:
```

In a main program we added ship to control speed.

```
def update(self):
    if self.moving_right and self.rect.right < self.screen_rect.right:
        self.center += self.ai_settings.ship_speed_factor
        if self.moving_right:
            self.rect.centerx += 1

    if self.moving_left and self.rect.left > 0:
        self.center -= self.ai_settings.ship_speed_factor

        if self.moving_left:
            self.rect.centerx -= 1
        self.rect.centerx = self.center
```

In ship.py file. We added some new command line in update function. These line use to stop the ship when it moves right and touches the corner. Same process we done for left key.

```
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
                check_keydown_events(event, ship)
        elif event.type == pygame.KEYUP:
                check_keyup_events(event, ship)
def check_keydown_events(event, ship):
    if event.key == pygame.K_RIGHT:
        ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        ship.moving_left = True

def check_keyup_events(event, ship):
    if event.key == pygame.K_RIGHT:
        ship.moving_right = False
    elif event.key == pygame.K_LEFT:
        ship.moving_left = False
```

In a game function file we created two functions key up and key down. And in key down we write the same lines which declare in elif key down block. Same process will be done for keyup.

```
class Settings():
    def __init__(self):
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230, 230, 230)
        self.ship_speed_factor = 1.5
        self.bullet_speed_factor = 1
        self.bullet_width = 3
        self.bullet_height = 15
        self.bullet_color = 60, 60, 60
```

Now we have to add some bullets and it speed that's why we write new lines in settings.py file. And set its width, height and color.

```
import pygame
from pygame.sprite import Sprite
class Bullet(Sprite):
    def __init__(self, ai_settings, screen, ship):
        super(Bullet, self).__init__()
        self.screen = screen
        self.rect = pygame.Rect(0, 0, ai_settings.bullet_width, ai_settings.bullet_height)
        self.rect.centerx = ship.rect.centerx
        self.rect.top = ship.rect.top

        self.y = float(self.rect.y)
        self.color = ai_settings.bullet_color
        self.speed_factor = ai_settings.bullet_speed_factor
```

Create a new file name bullet and in this file we import a sprite library to generate bullet. After that we created a class bullet. And in this class we added init function.in this function we added a bullet color, bullet position and height.

```python
def update(self):
    self.y -= self.speed_factor
    self.rect.y = self.y

def draw_bullet(self):
    pygame.draw.rect(self.screen, self.color, self.rect)
```

In the same file we created two function update and draw bullet. Update function mange a bullet speed. We set the bullet traveling direction as .y coordinate which means bullet will travel in vertical position.

In Second function we draw a bullet on the screen and set its color and position.

```python
import pygame
import sys
from pygame.sprite import Group
from settings import Settings
from ship import Ship
import game_functions as gf
def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width, ai_setting
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    ship = Ship(ai_settings, screen)
    bullets = Group()

    bg_color = (230, 230, 230)

    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        gf.update_screen(ai_settings, screen, ship)
```

In a main program, we imported spirit library and secondly we added a group of bullets.

```
    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        bullets.update()
        gf.update_screen(ai_settings, screen, ship, bullets)

run_game()
```

Also added the bullet on update screen.

```
from bullet import Bullet
def check_keydown_events(event,ai_settings,screen,ship,bullets):
    if event.key == pygame.K_RIGHT:
        ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        ship.moving_left = True
    elif event.key == pygame.K_SPACE:
        new_bullet = Bullet(ai_settings, screen, ship)
        bullets.add(new_bullet)

def check_keyup_events(event, ship):
    if event.key == pygame.K_RIGHT:
        ship.moving_right = False
    elif event.key == pygame.K_LEFT:
        ship.moving_left = False


def check_events(ai_settings, screen, ship, bullets):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
check_events() › for event in pygame.event.get()
```

In a game function.py file we created elif. And also we added a pygame. K space and added bullet, which means that when a user press enter the bullet release in ship.

```
    elif event.type == pygame.KEYDOWN:
        check_keydown_events(event,ai_settings,screen,ship,bullets)
    elif event.type == pygame.KEYUP:
            check_keyup_events(event, ship)
```

In this step, we added a bullet variable in a check keydown parameters. Also In check keyup.

```
    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        bullets.update()
        for bullet in bullets.copy():
            if bullet.rect.bottom <= 0:
                bullets.remove(bullet)
        print(len(bullets))
        gf.update_screen(ai_settings, screen, ship, bullets)

run_game()
```

In the main function. We enter a condition in while loop. This condition is for the bullet to disappear, when reaches the top of the screen.

```
        self.bullet_width = 3
        self.bullet_height = 15
        self.bullet_color = 60, 60, 60
        self.bullets_allowed = 3
```

In a setting function. We added bullets limit, which mean how many bullets will release at a time.

```
        ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        ship.moving_left = True
    elif event.key == pygame.K_SPACE:
        if len(bullets) < ai_settings.bullets_allowed:
            new_bullet = Bullet(ai_settings, screen, ship)
            bullets.add(new_bullet)
```

In a game function. We added a line in a K space. To fix a number of bullet release in a ship.

```
def update_bullets(bullets):
    bullets.update()
    for bullet in bullets.copy():
        if bullet.rect.bottom <= 0:
            bullets.remove(bullet)
```

In a game function we created a new function name update_bullets and from this function we copied for loop in a main program.

```
while True:
    gf.check_events(ai_settings, screen, ship, bullets)
    ship.update()
    gf.update_bullets(bullets)
    gf.update_screen(ai_settings, screen, ship, bullets)

n_game()
```

In main program we added update bullets function.

```
def check_keydown_events(event,ai_settings,screen,ship,bullets):
    if event.key == pygame.K_RIGHT:
        ship.moving_right = True
    elif event.key == pygame.K_LEFT:
        ship.moving_left = True
    elif event.key == pygame.K_SPACE:
        if len(bullets) < ai_settings.bullets_allowed:
            fire_bullet(ai_settings, screen, ship, bullets)
    elif event.key == pygame.K_q:
        sys.exit()
def fire_bullet(ai_settings, screen, ship, bullets):
    if len(bullets) < ai_settings.bullets_allowed:
        new_bullet = Bullet(ai_settings, screen, ship)
        bullets.add(new_bullet)
```

In a game function file. We created a fire bullet function. This function is used, when a user presses the space bar the bullets will appear on the screen. In check key down function () we call a function fire bullet.

```
            fire_bullet(ai_settings, screen, ship, bullet
    elif event.key == pygame.K_q:
        sys.exit()
def fire_bullet(ai_settings, screen, ship, bullets):
    if len(bullets) < ai_settings.bullets_allowed:
        new_bullet = Bullet(ai_settings, screen, ship)
        bullets.add(new_bullet)
```

In a game function file. We added a pygame.K_q method call in a pygame. Keydown.pygame.k_q, it used when the user press Q the game is exit.

```
from pygame.sprite import Sprite
class Alien(Sprite):
    def __init__(self, ai_settings, screen):
        super(Alien, self).__init__()
        self.screen = screen
        self.ai_settings = ai_settings
        self.image = pygame.image.load('alien.bmp')
        self.rect = self.image.get_rect()
        self.rect.x = self.rect.width
        self.rect.y = self.rect.height
        self.x = float(self.rect.x)

    def blitme(self):
        self.screen.blit(self.image, self.rect)

```

Now create alien file. In this file we added class alien and in this class we add function name init. And in this function we added lines. To add alien image same process as we did for ship image and set alien width, height. While Second function used for image and setting load.

alien_invasion.py bullet.py ship.py game_functions.py alien.py settings.py

```python
import pygame
import sys
from alien import Alien
from pygame.sprite import Group
from settings import Settings
from ship import Ship
import game_functions as gf
def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_he
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    ship = Ship(ai_settings, screen)
    bullets = Group()
    alien = Alien(ai_settings, screen)

    bg_color = (230, 230, 230)
```

In a main program we import alien file. And in the function we call alien setting.

```python
    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        gf.update_bullets(bullets)
        gf.update_screen(ai_settings, screen, ship, alien, bullets)
run_game()
```

After that we added alien in update screen.

```
def update_screen(ai_settings, screen, ship,alien,bullets):
    for bullet in bullets.sprites():
        bullet.draw_bullet()

    ship.blitme()
    alien.blitme()
    pygame.display.flip()
    screen.fill(ai_settings.bg_color)
```

In game function, we added alien.blitme function which is used for alien screen appearance.

```
    aliens = Group()
    gf.create_fleet(ai_settings, screen, aliens)

    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        gf.update_bullets(bullets)
        gf.update_screen(ai_settings, screen, ship, aliens, bullets)
run_game()
```

In a main program we added alien group. The program store a group of alien and call a create fleet function. After update a screen.

```python
import pygame
import sys
from bullet import Bullet
from alien import Alien
def create_fleet(ai_settings, screen, aliens):
    alien = Alien(ai_settings, screen)
    alien_width = alien.rect.width
    available_space_x = ai_settings.screen_width - 2 * alien_width
    number_aliens_x = int(available_space_x / (2 * alien_width))
    for alien_number in range(number_aliens_x):
        alien = Alien(ai_settings, screen)
        alien.x = alien_width + 2 * alien_width * alien_number
        alien.rect.x = alien.x
        aliens.add(alien)
```

In a game function file we created {create_fleet function}. In this function we added alien width, alien space alien place in horizontal place, and adding aliens in row.

```python
import pygame
import sys
from bullet import Bullet
from alien import Alien
def get_number_aliens_x(ai_settings, alien_width):
    available_space_x = ai_settings.screen_width - 2 * alien_width
    number_aliens_x = int(available_space_x / (2 * alien_width))
    return number_aliens_x
def create_alien(ai_settings, screen, aliens, alien_number):
    alien = Alien(ai_settings, screen)
    alien_width = alien.rect.width
    alien.x = alien_width + 2 * alien_width * alien_number
    alien.rect.x = alien.x
    aliens.add(alien)

def create_fleet(ai_settings, screen, aliens):
    alien = Alien(ai_settings, screen)
    number_aliens_x = get_number_aliens_x(ai_settings, alien.rect.width)
    for alien_number in range(number_aliens_x):
        create_alien(ai_settings, screen, aliens, alien_number)
```

In a game function. We added number of alien and to copy create_fleet line which we already declare in above step.

```python
    number_aliens_x = int(available_space_x / (2 * alien_width))
    return number_aliens_x
def create_alien(ai_settings, screen, aliens, alien_number,row_number):
    alien = Alien(ai_settings, screen)
    alien_width = alien.rect.width
    alien.x = alien_width + 2 * alien_width * alien_number
    alien.rect.x = alien.x
    alien.rect.y = alien.rect.height + 2 * alien.rect.height * row_number
    aliens.add(alien)
def get_number_rows(ai_settings, ship_height, alien_height):
    available_space_y = (ai_settings.screen_height-(3 * alien_height) - ship_height)
    number_rows = int(available_space_y / (2 * alien_height))
    return number_rows

def create_fleet(ai_settings,ship,screen, aliens):
    alien = Alien(ai_settings, screen)
    number_aliens_x = get_number_aliens_x(ai_settings, alien.rect.width)
    number_rows = get_number_rows(ai_settings,ship.rect.height,alien.rect.height)
    for row_number in range(number_rows):
        for alien_number in range(number_aliens_x):
            create_alien(ai_settings, screen, aliens, alien_number)
```

In create fleet function to add get number rows and create alien function. In a setting file we add alien speed in init function.

```
def update(self):
    self.x += self.ai_settings.alien_speed_factor
    self.rect.x = self.x

def blitme(self):
    self.screen.blit(self.image, self.rect)
```

In alien.py file, we added a speed factor and alien position in update function.

```
while True:
    gf.check_events(ai_settings, screen, ship, bullets)
    ship.update()
    gf.update_bullets(bullets)
    gf.update_aliens(aliens)
    gf.update_screen(ai_settings, screen, ship, aliens, bullets)
un_game()
```

Now, in main program, we added update aliens.

```
self.alien_speed_factor = 1
self.fleet_drop_speed = 10
self.fleet_direction = 1
```

In a setting.py file we add alien drop speed and its direction. How the alien moves and what its speed.

```
        self.rect.x = self.x

    def check_edges(self):
        screen_rect = self.screen.get_rect()
        if self.rect.right >= screen_rect.right:
            return True
        elif self.rect.left <= 0:
            return True

    def blitme(self):
        self.screen.blit(self.image, self.rect)
```

In alien file. Create a new function name check edges. In this function, we added a condition that when alien value is greater than right attribute alien moves left. Same process as for left.

```
    def update(self):
        self.x += (self.ai_settings.alien_speed_factor * self.ai_settings.fleet_direction)
        self.rect.x = self.x

    def blitme(self):
        self.screen.blit(self.image, self.rect)
```

In the same file, add alien speed and right and left.

```python
def change_fleet_direction(ai_settings, aliens):
    for alien in aliens.sprites():
        alien.rect.y += ai_settings.fleet_drop_speed
    ai_settings.fleet_direction *= -1

def check_fleet_edges(ai_settings, aliens):
    for alien in aliens.sprites():
        if alien.check_edges():
            change_fleet_direction(ai_settings, aliens)
            break
def update_aliens(ai_settings,aliens):
    check_fleet_edges(ai_settings, aliens)
    aliens.update()
```

In a game function file, add two function name fleet direction and fleet edges. Flat direction function use to control alien direction and fleet edges function used to check fleet direction value when a function is return true the alien change its direction and to break a loop.

After add check fleet edges function in update function.

```python
    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        gf.update_bullets(bullets)
        gf.update_aliens(ai_settings, aliens)
        gf.update_screen(ai_settings, screen, ship, aliens, bullets)
run_game()
```

After that we add update alien in a main program.

```python
def check_bullet_alien_collisions(ai_settings, screen, ship, aliens, bullets):
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)
    if len(aliens) == 0:
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
```

In a game function.py file. Create a new function name check bullet alien collision. In this function add pygame.spirit.groupcollide method. Means when a bullet hits alien. It will disappear from the screen, after ship shots all alien it will appear again on the screen.

```python
def update_bullets(ai_settings, screen, stats, sb, ship, aliens, bullets):
    bullets.update()
    for bullet in bullets.copy():
        if bullet.rect.bottom <= 0:
            bullets.remove(bullet)
    check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship,aliens, bullets)
```

Secondly, in update bullet function call a check bullet alien collision function.

```python
while True:
    gf.check_events(ai_settings, screen, stats, sb, play_button, ship,aliens, bullets)
    if stats.game_active:
        ship.update()
        gf.update_bullets(ai_settings, screen, ship, aliens, bullets)
        gf.update_aliens(ai_settings, screen, stats, sb, ship, aliens,bullets)
    gf.update_screen(ai_settings, screen, stats, sb, ship, aliens,bullets, play_button)
```

_game() › while True › if stats.game_active

In main program update bullet.

```
        self.bullet_speed_factor = 1
        self.bullet_width = 3
        self.bullet_speed_factor = 3
        self.bullet_height = 15
        self.bullet_color = 60, 60, 60
        self.bullets_allowed = 3
        self.alien_speed_factor = 1
        self.fleet_drop_speed = 10
        self.fleet_direction = 1
```

In a setting file add bullet speed.

```
def update_aliens(ai_settings, ship, aliens):
    check_fleet_edges(ai_settings, aliens)
    aliens.update()
    if pygame.sprite.spritecollideany(ship, aliens):
        print("Ship hit!!!")
```

To update alien function. In this function we used pygame.spirit.spiritecollideany method. In this method, any alien hits a ship. Ship will be disappeared from the screen and print ship hit. After again game start.

```
    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        ship.update()
        gf.update_bullets(aliens, bullets)
        gf.update_aliens(ai_settings, ship, aliens)
        gf.update_screen(ai_settings, screen, ship, aliens, bullets)
run_game()
```

In a main program we added update alien function.

```
alien_invasion.py  game_stats.py  bullet.py  ship.py  game_functions.py  alien.py  settings.py
class GameStats():
    def __init__(self, ai_settings):
        self.ai_settings = ai_settings
        self.reset_stats()


    def reset_stats(self):
        self.ships_left = self.ai_settings.ship_limit
```

Create a new file name game stats. In this file, add game stats class and add function name init. In this function. We used reset stats. When alien collide to ship. Ship disappear from the screen and again it appears on the screen.

alien_invasion.py game_stats.py bullet.py ship.py game_functions.py alien.py settings.py

```python
import pygame
import sys
from alien import Alien
from game_stats import GameStats
from pygame.sprite import Group
from settings import Settings
from ship import Ship
import game_functions as gf
def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_height))
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    stats = GameStats(ai_settings)
    ship = Ship(ai_settings, screen)
    bullets = Group()
    alien = Alien(ai_settings, screen)
    bg_color = (230, 230, 230)
```

In main program, we imported game stats file.

```python
    def __init__(self):
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230, 230, 230)
        self.ship_speed_factor = 1.5
        self.ship_limit = 3
        self.bullet_speed_factor = 1
        self.bullet_width = 3
        self.bullet_speed_factor = 3
        self.bullet_height = 15
        self.bullet_color = 60, 60, 60
        self.bullets_allowed = 3
        self.alien_speed_factor = 1
        self.fleet_drop_speed = 10
        self.fleet_direction = 1
```

In setting.py file add a ship limit.

```python
from bullet import Bullet
from alien import Alien
from time import sleep
def ship_hit(ai_settings, stats, screen, ship, aliens, bullets):
    if stats.ships_left > 0:
        stats.ships_left -= 1
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
        sleep(0.5)
    else:
        stats.game_active = False
```

In game function file. Create a new function name ship hit. In this function, we use sleep method which means when alien hits a ship. The game will pause. After that ship will appear again in a center and store bullet.

```python
def center_ship(self):
        self.center = self.screen_rect.centerx
```

In a ship.py file create a new function name center ship, it is because when ship disappear from the screen after few seconds it will again appear on a center of the screen.

```python
def check_aliens_bottom(ai_settings, screen, stats, sb, ship, aliens,bullets):
    screen_rect = screen.get_rect()
    for alien in aliens.sprites():
        if alien.rect.bottom >= screen_rect.bottom:
            ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets)
            break


def update_aliens(ai_settings, stats, screen, ship, aliens, bullets):
    check_fleet_edges(ai_settings, aliens)
    aliens.update()
    if pygame.sprite.spritecollideany(ship, aliens):
        ship_hit(ai_settings, stats, screen, ship, aliens, bullets)
```

In a game function file create a new function name alien bottom. In this function add alien bottom value. When alien bottom value is greater than the declared value. It will break a loop. And after few second. Update all setting.

```python
class GameStats():
    def __init__(self, ai_settings):
        self.ai_settings = ai_settings
        self.reset_stats()
        self.game_active = True


    def reset_stats(self):
        self.ships_left = self.ai_settings.ship_limit
```

In a game stats.py file, we add a condition that if there any ship if left than continue a game.

```python
from bullet import Bullet
from alien import Alien
from time import sleep
def ship_hit(ai_settings, stats, screen, ship, aliens, bullets):
    if stats.ships_left > 0:
        stats.ships_left -= 1
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
        sleep(0.5)
    else:
        stats.game_active = False
```

In the game function file, we added a condition in a ship hit function. Whereas, else is a number of ship that did not left which means to stop the game.

```python
    while True:
        gf.check_events(ai_settings, screen, ship, bullets)
        if stats.game_active:
            ship.update()
            gf.update_bullets(ai_settings, screen, ship, aliens, bullets)
            gf.update_aliens(ai_settings, stats, screen, ship, aliens, bullets)
        gf.update_screen(ai_settings, screen, ship, aliens, bullets)
run_game()
```

In the main program, we added the condition in a while loop. If the user press q the all program stop at the time and the game window disappear.

```python
class GameStats():
    def __init__(self, ai_settings):
        self.ai_settings = ai_settings
        self.reset_stats()
        self.game_active = False



    def reset_stats(self):
        self.ships_left = self.ai_settings.ship_limit
```

In a game stats file, we add a line game active which is used to stop the game when user does not want to play the game. The game does not start.

```python
    def __init__(self, ai_settings, screen, msg):
        self.screen = screen
        self.screen_rect = screen.get_rect()
        self.width, self.height = 200, 50
        self.button_color = (0, 255, 0)
        self.text_color = (255, 255, 255)
        self.font = pygame.font.SysFont(None, 48)
        self.rect = pygame.Rect(0, 0, self.width, self.height)
        self.rect.center = self.screen_rect.center
        self.prep_msg(msg)

    def prep_msg(self, msg):
        self.msg_image = self.font.render(msg, True, self.text_color, self.button_color)
        self.msg_image_rect = self.msg_image.get_rect()
        self.msg_image_rect.center = self.rect.center

    def draw_button(self):
        self.screen.fill(self.button_color, self.rect)
        self.screen.blit(self.msg_image, self.msg_image_rect)
```

Create a button file, in this file we added call few method. Like font, color. In this file we imported font and create a class and in this class create a function. In this function call button font color and txt color and its width, which position it appears.

Second function is prep_msg in this function we used Boolean argument. Like off and on. When a player click a button the game is start. Third function is draw button. This function is used to draw a button on a screen.

```python
import pygame
import sys
from alien import Alien
from game_stats import GameStats
from button import Button
from pygame.sprite import Group
from settings import Settings
from ship import Ship
import game_functions as gf
def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_height))
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    play_button = Button(ai_settings, screen, "Play")
    stats = GameStats(ai_settings)
    ship = Ship(ai_settings, screen)
    bullets = Group()
    alien = Alien(ai_settings, screen)
    bg_color = (230, 230, 230)
run_game()
```

In the main program we imported a button file to add play button

In a game function file, we added play button. It function as If the user not click a button the game does not start.

```python
while True:
    gf.check_events(ai_settings, screen, ship, bullets)
    if stats.game_active:
        ship.update()
        gf.update_bullets(ai_settings, screen, ship, aliens, bullets)
        gf.update_aliens(ai_settings, stats, screen, ship, aliens, bullets)
    gf.update_screen(ai_settings, screen, stats, ship, aliens, bullets,play_button)

run_game()
```

After that edit update screen to add play button.

```python
def check_play_button(ai_settings, screen, stats, play_button, ship, aliens,bullets, mouse_x, mouse_y):
    if play_button.rect.collidepoint(mouse_x, mouse_y):
        stats.reset_stats()
        stats.game_active = True
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
```

In a game function, update a check play button to act when the user click a play button or when the user refresh the game. It reset the setting.

```python
def check_events(ai_settings, screen, stats, play_button, ship, aliens,bullets):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            check_keydown_events(event,ai_settings,screen,ship,bullets)
        elif event.type == pygame.KEYUP:
                check_keyup_events(event, ship)
        elif event.type == pygame.MOUSEBUTTONDOWN:
            mouse_x,mouse_y = pygame.mouse.get_pos()
            check_play_button(ai_settings, screen, stats, play_button, ship, aliens, bullets, mouse_x, mouse_y)
```

To modify a check event add pygame mouse button down which means to restrict a function for a user. If user click anywhere on a screen. The game does not start.

```python
def check_play_button(ai_settings, screen, stats, play_button, ship, aliens,bullets, mouse_x, mouse_y):
    button_clicked = play_button.rect.collidepoint(mouse_x, mouse_y)
    if button_clicked and not stats.game_active:
        pygame.mouse.set_visible(False)
    if play_button.rect.collidepoint(mouse_x, mouse_y):
        stats.reset_stats()
        stats.game_active = True
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
```

In a game function. Update check play button. To add condition. If player clicked a play button mouse cursor is not visible on pygame screen. This condition is pygame.mouse.set visible is false and the mouse cursor is hidden.

```
from time import sleep
def ship_hit(ai_settings, stats, screen, ship, aliens, bullets):
    if stats.ships_left > 0:
        stats.ships_left -= 1
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
        sleep(0.5)
    else:
        stats.game_active = False
        pygame.mouse.set_visible(True)
```

To update ship hit, when a game is inactive mouse cursor is visible on a screen. This condition is pygame.mouse.set_visible. True. When a game restart.

```
while True:
    gf.check_events(ai_settings, screen, stats, play_button, ship,aliens, bullets)
    if stats.game_active:
        ship.update()
        gf.update_bullets(ai_settings, screen, ship, aliens, bullets)
        gf.update_aliens(ai_settings, stats, screen, ship, aliens, bullets)
    gf.update_screen(ai_settings, screen, stats, ship,aliens, bullets,play_button)
```

In a main program to update a check event function.

```
        self.alien_speed_factor = 1
        self.fleet_drop_speed = 10
        self.fleet_direction = 1
        self.speedup_scale = 1.1
        self.initialize_dynamic_settings()
    def initialize_dynamic_settings(self):
        self.ship_speed_factor = 1.5
        self.bullet_speed_factor = 3
        self.alien_speed_factor = 1
        self.fleet_direction = 1
    def increase_speed(self):
        self.ship_speed_factor *= self.speedup_scale
        self.bullet_speed_factor *= self.speedup_scale
        self.alien_speed_factor *= self.speedup_scale
```

In a setting file, we added game speed. When player reach a new level. Game speed is increases. By the help speedup scale. Second a new function create. In this function. To set initial value of a ship, alien. And a bullet. It increase when a player reach a new level. When restart a game it reset again. Last function is where a ship kill last alien. It increase speed. To all object.

```
def check_bullet_alien_collisions(ai_settings, screen, ship, aliens, bullets):
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)
    if len(aliens) == 0:
        bullets.empty()
        ai_settings.increase_speed()
        create_fleet(ai_settings, screen, ship, aliens)
```

In a check bullet alien collision. Call increase speed function.

```python
def check_play_button(ai_settings, screen, stats, play_button, ship, aliens,bullets, mouse_x, mouse_y):
    button_clicked = play_button.rect.collidepoint(mouse_x, mouse_y)
    if button_clicked and not stats.game_active:
        ai_settings.initialize_dynamic_settings()
        pygame.mouse.set_visible(False)
    if play_button.rect.collidepoint(mouse_x, mouse_y):
        stats.reset_stats()
        stats.game_active = True
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
```

In a game function. Add initialize dynamic setting. To reach a new level. It clear a screen and again appear alien. This time its speed increase and also a game speed. It more difficult.

```python
def reset_stats(self):
    self.ships_left = self.ai_settings.ship_limit
    self.score = 0
```

In a setting file. Add initial score is 0. To increase a score, player have kill alien. To restart a new game it will be again 0.

```python
import pygame.font
class Scoreboard():
        def __init__(self, ai_settings, screen, stats):
            self.screen = screen
            self.screen_rect = screen.get_rect()
            self.ai_settings = ai_settings
            self.stats = stats
            self.text_color = (30, 30, 30)
            self.font = pygame.font.SysFont(None, 48)
            self.prep_score()

        def prep_score(self):
                score_str = str(self.stats.score)
                self.score_image = self.font.render(score_str, True, self.text_color,self.ai_settings.bg_color)
                self.score_rect = self.score_image.get_rect()
                self.score_rect.right = self.screen_rect.right - 20
                self.score_rect.top = 20

        def show_score(self):
            self.screen.blit(self.score_image, self.score_rect)
```

Add another file named scoreboard. In this document to import textual style and to add class name scoreboard. In this class case another capacity name init. To seem a score on a screen. Also, its tone. Second make another capacity name prep score. In this capacity to seem a score on screen as a picture and score esteem proclaim in setting. This picture show a privilege and to set a width and tallness to set this utilization score.rect.right and score.rect.top. To set a worth. And furthermore pronounce its tone. To make Third capacity is show score.in this capacity to utilize bilit technique. By the assistance of this technique. To show up picture on screen

```python
def update_screen(ai_settings, screen, stats, sb, ship, aliens, bullets,play_button):
    for bullet in bullets.sprites():
        bullet.draw_bullet()
    ship.blitme()
    aliens.draw(screen)
    sb.show_score()
    if not stats.game_active:
        play_button.draw_button()
    pygame.display.flip()
    screen.fill(ai_settings.bg_color)
```

In a game function. Add show score function in update screen function.

```
alien_invasion.py  button.py  game_stats.py  scoreboard.py  bullet.py  ship.py  game_functions.py  alien.py  settings.py
import pygame
import sys
from scoreboard import Scoreboard
from alien import Alien
from game_stats import GameStats
from button import Button
from pygame.sprite import Group
from settings import Settings
from ship import Ship
import game_functions as gf
def run_game():
    pygame.init()
    ai_settings = Settings()
    screen = pygame.display.set_mode((ai_settings.screen_width, ai_settings.screen_height))
    screen = pygame.display.set_mode((1200, 800))
    pygame.display.set_caption("Alien Invasion")
    play_button = Button(ai_settings, screen, "Play")
    stats = GameStats(ai_settings)
    sb = Scoreboard(ai_settings, screen, stats)
    ship = Ship(ai_settings, screen)
    bullets = Group()
```

In a main program, import a scoreboard file. And save its value in the sb variable.

```
while True:
    gf.check_events(ai_settings, screen, stats, play_button, ship,aliens, bullets)
    if stats.game_active:
        ship.update()
        gf.update_bullets(ai_settings, screen, ship, aliens, bullets)
        gf.update_aliens(ai_settings, stats, screen, ship, aliens, bullets)
    gf.update_screen(ai_settings, screen, stats, sb, ship, aliens,bullets, play_button)


run_game()
```

After add screen.

```
def initialize_dynamic_settings(self):
    self.ship_speed_factor = 1.5
    self.bullet_speed_factor = 3
    self.alien_speed_factor = 1
    self.fleet_direction = 1
    self.alien_points = 50
```

In a setting.py file, add alien points. On kill adds 50 points.

```
def check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship,aliens, bullets):
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)
    if collisions:
        for aliens in collisions.values():
            stats.score += ai_settings.alien_points * len(aliens)
            sb.prep_score()

    if len(aliens) == 0:
        bullets.empty()
```

In function file. add alien collision line. When a bullets hits alien the score board changes.

```
def update_bullets(ai_settings, screen, stats, sb, ship, aliens, bullets):
    bullets.update()
    for bullet in bullets.copy():
        if bullet.rect.bottom <= 0:
            bullets.remove(bullet)

    check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship, aliens, bullets)
```

To add bullets function call check bullet alien collision function.

```
    while True:
        gf.check_events(ai_settings, screen, stats, play_button, ship,aliens, bullets)
        if stats.game_active:
            ship.update()
            gf.update_bullets(ai_settings, screen, stats, sb, ship, aliens, bullets)
            gf.update_aliens(ai_settings, stats, screen, ship, aliens, bullets)
        gf.update_screen(ai_settings, screen, stats, sb, ship, aliens,bullets, play_button)


run_game()
```

In the main program, add parameters in bullets function.

```
        self.bullet_color = 60, 60, 60
        self.bullets_allowed = 3
        self.alien_speed_factor = 1
        self.fleet_drop_speed = 10
        self.fleet_direction = 1
        self.speedup_scale = 1.1
        self.score_scale = 1.5
        self.initialize_dynamic_settings()
    def initialize_dynamic_settings(self):
        self.ship_speed_factor = 1.5
        self.bullet_speed_factor = 3
        self.alien_speed_factor = 1
        self.fleet_direction = 1
        self.alien_points = 50
    def increase_speed(self):
        self.ship_speed_factor *= self.speedup_scale
        self.bullet_speed_factor *= self.speedup_scale
        self.alien_speed_factor *= self.speedup_scale
        self.alien_points = int(self.alien_points * self.score_scale)

Settings > increase_speed()
```

In the setting file, add alien points. And when hits every alien increase speed of game.

```python
def increase_speed(self):
    self.ship_speed_factor *= self.speedup_scale
    self.bullet_speed_factor *= self.speedup_scale
    self.alien_speed_factor *= self.speedup_scale
    self.alien_points = int(self.alien_points * self.score_scale)
    print(self.alien_points)
```

In increase speed function add alien points. Each alien hits. Its print a points onthe program.

```python
class GameStats():
    def __init__(self, ai_settings):
        self.ai_settings = ai_settings
        self.reset_stats()
        self.game_active = False
        self.high_score = 0

    def reset_stats(self):
        self.ships_left = self.ai_settings.ship_limit
        self.score = 0
```

In stats.py file, add high score. And the initial value is 0.

```
def __init__(self, ai_settings, screen, stats):
    self.screen = screen
    self.screen_rect = screen.get_rect()
    self.ai_settings = ai_settings
    self.stats = stats
    self.text_color = (30, 30, 30)
    self.font = pygame.font.SysFont(None, 48)
    self.prep_score()
    self.prep_high_score()
```

In scoreboard file, add prep high score function.

```
def prep_high_score(self):
    high_score = int(round(self.stats.high_score, -1))
    high_score_str = "{:,}".format(high_score)
    self.high_score_image = self.font.render(high_score_str, True, self.text_color, self.ai_settings.bg_color)
    self.high_score_rect = self.high_score_image.get_rect()
    self.high_score_rect.centerx = self.screen_rect.centerx
    self.high_score_rect.top = self.score_rect.top

def show_score(self):
    self.screen.blit(self.score_image, self.score_rect)
    self.screen.blit(self.high_score_image, self.high_score_rect)
```

In score board file, create a function prep high score. It is used to declare its width, height and position and its image, and add high score image and its position.

```
def check_high_score(stats, sb):
    if stats.score > stats.high_score:
        stats.high_score = stats.score
        sb.prep_high_score()
```

In function create a new function check high score in this function there are two parameters. First to check a current score and high score and sb to modify high score. so If current score is greater than high score so updates high score.

```python
def check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship,aliens, bullets):
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)
    if collisions:
        for aliens in collisions.values():
            stats.score += ai_settings.alien_points * len(aliens)
            sb.prep_score()
        check_high_score(stats, sb)
```

In a check bullet alien collision function call check high score function.

```python
class GameStats():
    def __init__(self, ai_settings):
        self.ai_settings = ai_settings
        self.reset_stats()
        self.game_active = False
        self.high_score = 0


    def reset_stats(self):
        self.ships_left = self.ai_settings.ship_limit
        self.score = 0
        self.level = 1
```

In the stats.py file, add game level in a reset stats having initial value 1.

```python
import pygame.font
class Scoreboard():
    def __init__(self, ai_settings, screen, stats):
        self.screen = screen
        self.screen_rect = screen.get_rect()
        self.ai_settings = ai_settings
        self.stats = stats
        self.text_color = (30, 30, 30)
        self.font = pygame.font.SysFont(None, 48)
        self.prep_score()
        self.prep_high_score()
        self.prep_level()
```

In score board call prep level function.

```
def prep_level(self):
    self.level_image = self.font.render(str(self.stats.level), True, self.text_color, self.ai_settings.bg_color)
    self.level_rect = self.level_image.get_rect()
    self.level_rect.right = self.score_rect.right
    self.level_rect.top = self.score_rect.bottom + 10


def prep_score(self):
```

In score board create a new function named prep level. In this function add image height, position to declare image position value is 10.

```
def show_score(self):
    self.screen.blit(self.score_image, self.score_rect)
    self.screen.blit(self.high_score_image, self.high_score_rect)
    self.screen.blit(self.level_image, self.level_rect)
```

After adding show score, add level image, and position.

```
def check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship,aliens, bullets):
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)
    if collisions:
        for aliens in collisions.values():
            stats.score += ai_settings.alien_points * len(aliens)
            sb.prep_score()
        check_high_score(stats, sb)

    if len(aliens) == 0:
        bullets.empty()
        ai_settings.increase_speed()
        stats.level += 1
        sb.prep_level()
        create_fleet(ai_settings, screen, ship, aliens)
```

In the function.py file, add check bullet alien collision function. In this function add stats level line and prep level function, which means to hits all alien level image is update. And increment 1.

```
def check_play_button(ai_settings, screen, stats, sb, play_button, ship,aliens, bullets, mouse_x, mouse_y):
    button_clicked = play_button.rect.collidepoint(mouse_x, mouse_y)
    if button_clicked and not stats.game_active:
        ai_settings.initialize_dynamic_settings()
        pygame.mouse.set_visible(False)
    if play_button.rect.collidepoint(mouse_x, mouse_y):
        stats.reset_stats()
        stats.game_active = True
        sb.prep_score()
        sb.prep_high_score()
        sb.prep_level()
        aliens.empty()
        bullets.empty()
```

Also add a check play button function. The score board is reset and it opens after update.

```
def check_events(ai_settings, screen, stats, sb, play_button, ship, aliens,bullets):

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            check_keydown_events(event,ai_settings,screen,ship,bullets)
        elif event.type == pygame.KEYUP:
                check_keyup_events(event, ship)
        elif event.type == pygame.MOUSEBUTTONDOWN:
            mouse_x,mouse_y = pygame.mouse.get_pos()
            check_play_button(ai_settings, screen, stats, sb, play_button,ship, aliens, bullets, mouse_x, mouse_y)
```

From check event, add sb and stats parameter in play button.

```
    while True:
        gf.check_events(ai_settings, screen, stats, sb, play_button, ship, aliens,bullets)
        if stats.game_active:
            ship.update()
            gf.update_bullets(ai_settings, screen, stats, sb, ship, aliens, bullets)
            gf.update_aliens(ai_settings, stats, screen, ship, aliens, bullets)
        gf.update_screen(ai_settings, screen, stats, sb, ship, aliens,bullets, play_button)


run_game()
```

Add a check event function. In a main program

```python
import pygame
from pygame.sprite import Sprite
class Ship(Sprite):
 def __init__(self, ai_settings, screen):
        super(Ship, self).__init__()
        self.screen = screen
        self.ai_settings = ai_settings
        self.image = pygame.image.load('ship.bmp')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()
        self.rect.centerx = self.screen_rect.centerx
        self.rect.bottom = self.screen_rect.bottom
        self.center = float(self.rect.centerx)
        self.moving_right = False
        self.moving_left = False
```

In a ship, import spirit. Spirit library use in graphic in class ship to add spirit parameters..

```python
import pygame.font
from pygame.sprite import Group
from ship import Ship
class Scoreboard():
        def __init__(self, ai_settings, screen, stats):
                self.screen = screen
                self.screen_rect = screen.get_rect()
                self.ai_settings = ai_settings
                self.stats = stats
                self.text_color = (30, 30, 30)
                self.font = pygame.font.SysFont(None, 48)
                self.prep_score()
                self.prep_high_score()
                self.prep_level()
                self.prep_ships()
```

In the score board file, add the prep ship function.

```
def prep_ships(self):
    self.ships = Group()
    for ship_number in range(self.stats.ships_left):
        ship = Ship(self.ai_settings, self.screen)
        ship.rect.x = 10 + ship_number * ship.rect.width
        ship.rect.y = 10
        self.ships.add(ship)
```

In a score board record, make new capacity name prep transport. This capacity used to hold gathering of boat. Besides, apply condition by the assistance of for circle. To run a circle. Will situate a boat and each boat annihilated. To include a boat screen. Also, to less an all out number of boat. This picture shows on upper screen. So we have as of now proclaim picture position.

```
def show_score(self):
    self.screen.blit(self.score_image, self.score_rect)
    self.screen.blit(self.high_score_image, self.high_score_rect)
    self.screen.blit(self.level_image, self.level_rect)
    self.ships.draw(self.screen)
```

So to create a show score function add ship draw method. To show a total number of ship.

```python
def check_play_button(ai_settings, screen, stats, sb, play_button, ship,aliens, bullets, mouse_x, mouse_y):
    button_clicked = play_button.rect.collidepoint(mouse_x, mouse_y)
    if button_clicked and not stats.game_active:
        ai_settings.initialize_dynamic_settings()
        pygame.mouse.set_visible(False)
    if play_button.rect.collidepoint(mouse_x, mouse_y):
        stats.reset_stats()
        stats.game_active = True
        sb.prep_score()
        sb.prep_high_score()
        sb.prep_level()
        sb.prep_ships()
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
```

In function.py file, update check play button function to call prep ship function as this function used to display a number of ships left.

```python
def update_aliens(ai_settings, screen, stats, sb, ship, aliens, bullets):
    check_fleet_edges(ai_settings, aliens)
    aliens.update()
    if pygame.sprite.spritecollideany(ship, aliens):
        ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets)
    check_aliens_bottom(ai_settings, screen, stats, sb, ship, aliens, bullets)
```

In thhe game, modify update alien function in this function add sb parameters in the ship hit function.

```python
def ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets):
    if stats.ships_left > 0:
        stats.ships_left -= 1
        sb.prep_ships()
        aliens.empty()
        bullets.empty()
        create_fleet(ai_settings, screen, ship, aliens)
        ship.center_ship()
        sleep(0.5)
    else:
        stats.game_active = False
        pygame.mouse.set_visible(True)
```

After in boat hit capacity to call prep transport work by the assistance of sb boundaries add sb boundary in boat hit work. This capacity assists with showing the right number of boats left.

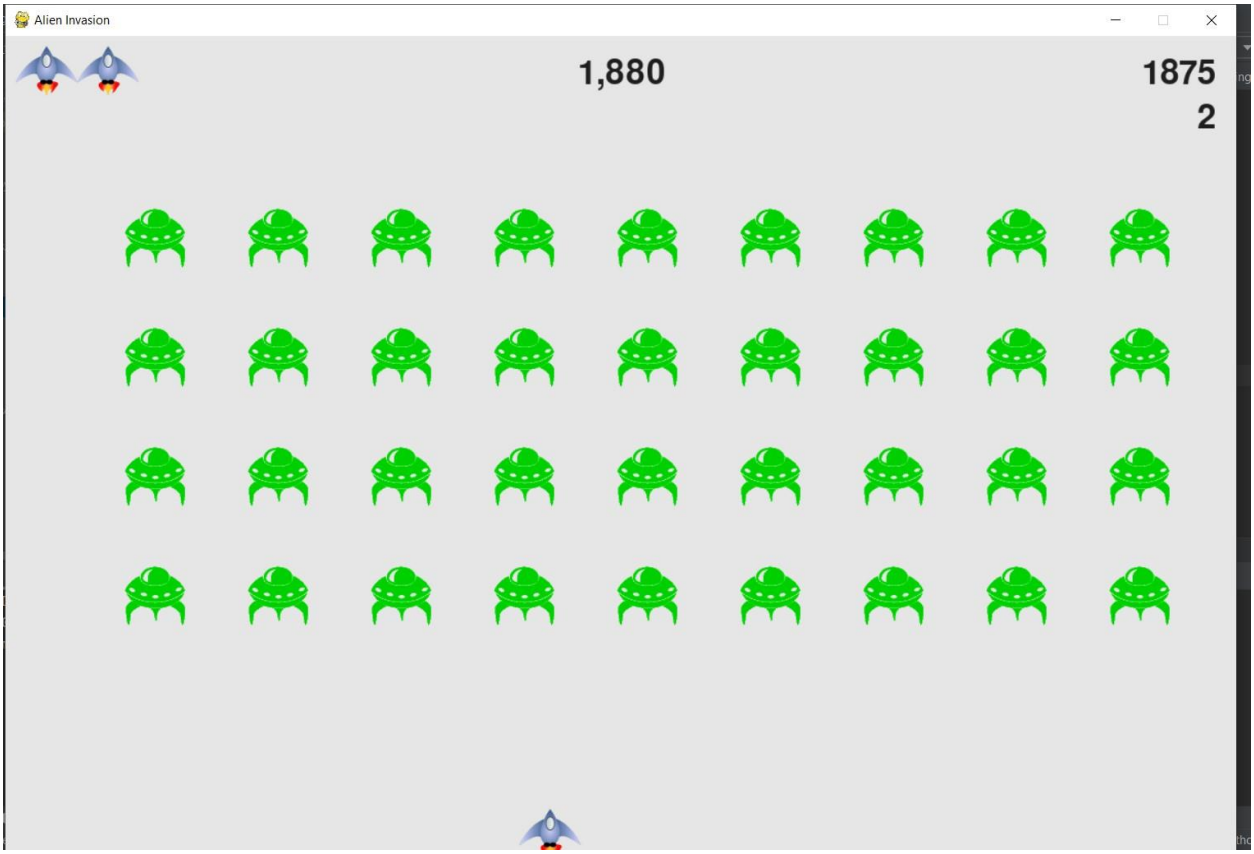
```python
def check_aliens_bottom(ai_settings, screen, stats, sb, ship, aliens,bullets):

    screen_rect = screen.get_rect()
    for alien in aliens.sprites():
        if alien.rect.bottom >= screen_rect.bottom:
            ship_hit(ai_settings, screen, stats, sb, ship, aliens, bullets)
            break
```

After updating ship hit parameters which gives check alien bottom.

```python
while True:
    gf.check_events(ai_settings, screen, stats, sb, play_button, ship, aliens,bullets)
    if stats.game_active:
        ship.update()
        gf.update_bullets(ai_settings, screen, stats, sb, ship, aliens, bullets)
        gf.update_aliens(ai_settings, screen, stats, sb, ship, aliens,bullets)
    gf.update_screen(ai_settings, screen, stats, sb, ship, aliens,bullets, play_button)
```

Sb pass in update aliens, In the main program.

OUTPUT:



To see one possibility is misfortune a player. After number of ships left its show up on the left of the top.