Docstring

1. Tạo file config để tiện sử dụng lại bằng cách import:

config.py > ...

```python
import pygame
pygame.display.set_mode((750, 700))
pygame.font.init()
pygame.display.init()
screen_width = 750
screen_height = 700
offset = 50
blue = (0,206,209)
speed = 6
laser_delay = 300
```

2. Tạo class SpaceShip:
   - Khởi tạo con tàu vũ trụ:

spaceship.py > Spaceship

```python
import pygame
from laser import Laser
from config import screen_width, screen_height, offset, speed, laser_delay

class Spaceship(pygame.sprite.Sprite):
    def __init__(self, screen_width, screen_height, offset):
        """
        Khởi tạo tàu vũ trụ.

        Tham số
        ----------
        screen_width : int
            Chiều rộng của màn hình trò chơi.
        screen_height : int
            Chiều cao của màn hình trò chơi.
        offset : int
            Khoảng cách lề bên trái của màn hình.
        """
        super().__init__()
        self.image = pygame.transform.scale(pygame.image.load("Graphics/spaceship.png").convert_alpha(), (45, 45))
        self.rect = self.image.get_rect(midbottom=((screen_width + offset) / 2, screen_height))
        self.lasers_group = pygame.sprite.Group()
        self.laser_ready = True
        self.laser_time = 0
        self.laser_sound = pygame.mixer.Sound("Sounds/laser.ogg")
```

- Xử lý đầu vào từ người chơi: di chuyển lên, xuống, trái, phải

```
26
27    def get_user_input(self):
28        """Xử lý các sự kiện đầu vào từ người chơi."""
29        keys = pygame.key.get_pressed()
30
31        if keys[pygame.K_RIGHT]:
32            self.rect.x += speed
33
34        if keys[pygame.K_LEFT]:
35            self.rect.x -= speed
36
37        if keys[pygame.K_UP]:
38            self.rect.y -= speed
39
40        if keys[pygame.K_DOWN]:
41            self.rect.y += speed
42
43        if keys[pygame.K_SPACE] and self.laser_ready:
44            self.laser_ready = False
45            laser = Laser(self.rect.center, 5, screen_height)
46            self.lasers_group.add(laser)
47            self.laser_time = pygame.time.get_ticks()
48            self.laser_sound.play()
49
```

- Cập nhật trạng thái con tàu:

```
50    def update(self):
51        """Cập nhật trạng thái của tàu vũ trụ."""
52        self.get_user_input()
53        self.limit_movement()
54        self.lasers_group.update()
55        self.recharge_laser()
```

- Giới hạn phạm vi di chuyển con tàu để không vượt ra khỏi màn hình, nếu vượt khỏi thì nó vẫn đúng tại góc đó

```
57    def limit_movement(self):
58        """Giới hạn phạm vi di chuyển của tàu vũ trụ để không vượt ra khỏi màn hình."""
59        if self.rect.right > screen_width:
60            self.rect.right = screen_width
61        if self.rect.left < offset:
62            self.rect.left = offset
63        if self.rect.bottom > screen_height:
64            self.rect.bottom = screen_height
65        if self.rect.top < 0:
66            self.rect.top = 0
```

- Nạp lại laser khi đã bắn xong:

```
102    def recharge_laser(self):
103        """Thực hiện việc nạp lại tia laser sau mỗi lần bắn."""
104        if not self.laser_ready:
105            current_time = pygame.time.get_ticks()
106            if current_time - self.laser_time >= laser_delay:
107                self.laser_ready = True
108
```

3. Tạo class Block: đây là tạo vật cản

```python
class Block(pygame.sprite.Sprite):
    """
    Đối tượng đại diện cho một khối chướng ngại vật trong trò chơi.
    """
    def __init__(self, x, y):
        """
        Khởi tạo một Block mới.

        Tham số:
            x (int): Tọa độ x của khối.
            y (int): Tọa độ y của khối.
        """
        super().__init__()
        self.image = pygame.Surface((3,3))
        self.image.fill((100,149,237))
        self.rect = self.image.get_rect(topleft = (x,y))

grid = [
[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0],
[0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0],
[0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0],
[0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
[1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1],
[1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1],
[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1]]
```

4. Tạo class Obstacles để quản lý tất cả các chướng ngại vật trong Game:

```python
class Obstacle:
    """
    Lớp đại diện cho chướng ngại vật trong trò chơi.
    """
    def __init__(self, x, y):
        """
        Khởi tạo một chướng ngại vật mới.

        Tham số:
            x (int): Tọa độ x của chướng ngại vật.
            y (int): Tọa độ y của chướng ngại vật.
        """
        self.blocks_group = pygame.sprite.Group()
        for row in range(len(grid)):
            for column in range(len(grid[0])):
                if grid[row][column] == 1:
                    pos_x = x + column * 3
                    pos_y = y + row * 3
                    block = Block(pos_x, pos_y)
                    self.blocks_group.add(block)

```

5. Tạo class Aliens:
   - Khởi tạo thuộc tính cần thiết:

```python
class Alien(pygame.sprite.Sprite):
    def __init__(self, type, x, y):
        """
        Khởi tạo tất cả các thuộc tính cần thiết cho đối tượng người ngoài hành tinh.

        Tham số:
        type : str
            Loại người ngoài hành tinh.
        x : int
            Tọa độ x ban đầu của người ngoài hành tinh.
        y : int
            Tọa độ y ban đầu của người ngoài hành tinh.
        """
        super().__init__()
        self.type = type
        path = f"Graphics/alien_{type}.png"
        self.image = pygame.image.load(path)
        self.rect = self.image.get_rect(topleft=(x, y))

```

   - Cập nhật vị trí di chuyển:

```python
    def update(self, direction):
        """
        Di chuyển người ngoài hành tinh theo hướng đã cho.

        Tham số:
        direction : int
            Hướng di chuyển của người ngoài hành tinh (giá trị âm để di chuyển sang trái và giá trị dương để di chuyển sang phải).
        """
        self.rect.x += direction

```

6. Tạo class con tàu bí ẩn:
   - Khởi tạo thuộc tính cần thiết

```python

class MysteryShip(pygame.sprite.Sprite):

    def __init__(self, screen_width, offset):
        """
        Khởi tạo tất cả các thuộc tính cần thiết cho đối tượng tàu bí ẩn.

        Tham số:
        screen_width : int
            Chiều rộng của màn hình.
        offset : int
            Giá trị bù trừ để xác định vị trí bắt đầu của tàu bí ẩn.
        """
        super().__init__()
        self.image = pygame.image.load("Graphics/mystery.png")

        x = random.choice([offset/2, screen_width + offset - self.image.get_width()])
        if x == offset/2:
            self.speed = 3
        else:
            self.speed = -3

        self.rect = self.image.get_rect(topleft=(x, 90))

```

- Cập nhật vị trí con tàu bí ẩn:

```python

    def update(self):
        """
        Cập nhật vị trí của tàu bí ẩn. Nếu tàu bí ẩn đi ra khỏi màn hình, nó sẽ bị xóa.
        """
        self.rect.x += self.speed
        if self.rect.right > screen_width + offset/2:
            self.kill()
        elif self.rect.left < offset/2:
            self.kill()

```

7. Tạo class Laser:
   - Khởi tạo tia Laser:

laser.py > Laser

```python
import pygame
from config import screen_height

class Laser(pygame.sprite.Sprite):
    def __init__(self, position, speed, screen_height):
        """
        Khởi tạo một tia laser.

        Tham số
        ----------
        position : tuple
            Tọa độ (x, y) ban đầu của tia laser.
        speed : int
            Tốc độ di chuyển của tia laser.
        screen_height : int
            Chiều cao của màn hình trò chơi.
        """
        super().__init__()
        self.image = pygame.Surface((4, 15))
        self.image.fill((0, 206, 209))
        self.rect = self.image.get_rect(center=position)
        self.speed = speed

```

- Cập nhật vị trí Laser, nếu Laser ra khỏi màn hình thì sẽ xoá nó ra khỏi nhóm:

```python
    def update(self):
        """
        Cập nhật vị trí của tia laser.

        Nếu tia laser vượt ra khỏi màn hình, nó sẽ bị xóa khỏi nhóm sprite.
        """
        self.rect.y -= self.speed
        if self.rect.y > screen_height + 15 or self.rect.y < 0:
            self.kill()
```

8. Tạo class Game để quản lý các thành phần trong trò chơi:
   - Khởi tạo trò chơi:

```python

class Game:
    def __init__(self, screen_width, screen_height, offset):
        """
        Khởi tạo trò chơi với kích thước màn hình và độ lệch đã cho.

        Tham số
        ----------
        screen_width : int
            Chiều rộng của màn hình trò chơi.
        screen_height : int
            Chiều cao của màn hình trò chơi.
        offset : int
            Giá trị độ lệch để định vị các phần tử trên màn hình.
        """
        self.spaceship_group = pygame.sprite.GroupSingle()
        self.spaceship_group.add(Spaceship(screen_width, screen_height, offset))
        self.obstacles = self.create_obstacles()
        self.aliens_group = pygame.sprite.Group()
        self.create_aliens()
        self.aliens_direction = 1
        self.alien_lasers_group = pygame.sprite.Group()
        self.mystery_ship_group = pygame.sprite.GroupSingle()
        self.lives = 3
        self.mystery_ship_hit_count = 0
        self.mystery_ship_destroyed = False
        self.run = True
        self.score = 0
        self.highscore = 0
        self.explosion_sound = pygame.mixer.Sound("Sounds/explosion.ogg")
        self.check_and_load_highscore()
        pygame.mixer.music.load("Sounds/music.ogg")
        pygame.mixer.music.play(-1)
```

- Tạo các chướng ngại vật:

```python
    def create_obstacles(self):
        """
        Tạo và trả về danh sách chướng ngại vật.

        Trả về
        -------
        list
            Danh sách các đối tượng Obstacle.
        """
        obstacle_width = len(grid[0]) * 3
        gap = (screen_width + offset - (3 * obstacle_width)) / 4
        obstacles = []
        for i in range(3):
            offset_x = (i + 1) * gap + i * obstacle_width
            obstacle = Obstacle(offset_x, screen_height - 100)
            obstacles.append(obstacle)
        return obstacles

```

- Tạo các Aliens và thêm vào aliens_group:

```python
    def create_aliens(self):
        """
        Tạo các đối tượng alien và thêm vào nhóm aliens_group.
        """
        for row in range(5):
            for column in range(12):
                x = 75 + column * 55
                y = 110 + row * 55

                if row == 0:
                    alien_type = 3
                elif row in (1, 2):
                    alien_type = 2
                else:
                    alien_type = 1

                alien = Alien(alien_type, x + offset / 2, y)
                self.aliens_group.add(alien)
```

- Di chuyển của Alien: thay đổi hướng và và dịch xuống nếu chạm cạnh của màn hình:

```python
    def move_aliens(self):
        """
        Di chuyển các alien và thay đổi hướng nếu chạm vào cạnh màn hình.
        """
        self.aliens_group.update(self.aliens_direction)
        alien_sprites = self.aliens_group.sprites()
        for alien in alien_sprites:
            if alien.rect.right >= screen_width + offset / 2:
                self.aliens_direction = -1
                self.alien_move_down(1)
            elif alien.rect.left <= offset / 2:
                self.aliens_direction = 1
                self.alien_move_down(1)

    def alien_move_down(self, distance):
        """
        Di chuyển tất cả các alien xuống một khoảng cách nhất định.

        Tham số
        ----------
        distance : int
            Khoảng cách để di chuyển các alien xuống.
        """
        if self.aliens_group:
            for alien in self.aliens_group.sprites():
                alien.rect.y += distance
```

- Sử dụng random để chọn 1 alien bắn laser:

```
106
107     def alien_shoot_laser(self):
108         """
109         Cho một alien ngẫu nhiên bắn tia laser.
110         """
111         if self.aliens_group.sprites():
112             random_alien = random.choice(self.aliens_group.sprites())
113             laser_sprite = Laser(random_alien.rect.center, -6, screen_height)
114             self.alien_lasers_group.add(laser_sprite)
115
```

- Tạo tàu bí ẩn:

```
16     def create_mystery_ship(self):
17         """
18         Tạo và thêm tàu bí ẩn vào trò chơi.
19         """
20         self.mystery_ship_group.add(MysteryShip(screen_width, offset))
21
```

- Tạo laser cho mysteryship bắn: bắn 1 lượt ra 20 laser và vị trí bắn random từ (-50, 50) so với tâm

```
123
124     def mystery_ship_shoot_laser(self):
125         """
126         Cho mysteryship bắn 20 tia laser.
127         """
128         if self.mystery_ship_laser_timer >= 40:  # số frame giữa mỗi lần bắn laser của MysteryShip
129             if self.mystery_ship_group.sprites():
130                 mystery_ship = self.mystery_ship_group.sprites()[0]
131
132                 for _ in range(20):
133                     x = mystery_ship.rect.centerx + random.randint(-50, 50)
134                     y = mystery_ship.rect.bottom
135                     laser_sprite = Laser((x, y), random.choice([-7, -6, -5, -4]), screen_height)
136                     self.mystery_ship_lasers_group.add(laser_sprite)
137
138             self.mystery_ship_laser_timer = 0
139         else:
140             self.mystery_ship_laser_timer += 1
141
```

- Màn hình kết thúc trò chơi:

```
143    def game_over(self):
144        """
145        Kết thúc trò chơi và hiển thị màn hình kết thúc.
146        """
147        pygame.init()
148        screen = pygame.display.set_mode((750, 700))
149        pygame.display.set_caption("Game Over")
150        background = pygame.image.load("Graphics/game_over.jpg")
151        background = pygame.transform.scale(background, (750, 700))
152        font = pygame.font.Font(None, 40)
153        text = font.render(str(self.score), True, (255, 255, 255))
154        text_rect = text.get_rect()
155        text_rect.center = (215, 325)
156
157        highscore_font = pygame.font.Font(None, 40)
158        highscore_text = highscore_font.render(str(self.highscore), True, (255, 255, 255))
159        highscore_rect = highscore_text.get_rect()
160        highscore_rect.center = (310, 373)
161
162        running = True
163        pygame.mixer.music.stop()
164        game_over_sound = pygame.mixer.Sound("Sounds/game_over.mp3")
165        game_over_sound.play()
166
167        while running:
168            for event in pygame.event.get():
169                if event.type == pygame.QUIT:
170                    running = False
171            screen.blit(background, (0, 0))
172            screen.blit(text, text_rect)
173            screen.blit(highscore_text, highscore_rect)
174            pygame.display.flip()
175
176        pygame.quit()
177        exit()
```

Kiểm tra và cập nhật điểm số: kiểm tra và đọc điểm từ tệp highscore.txt. Nếu không có sẽ bằng 0. Nếu cao hơn highscore trong file thì sẽ cập nhật lại.

- Màn hình thắng trò chơi:

```
179    def game_win(self):
180        """
181        Kết thúc trò chơi và hiển thị màn hình chiến thắng.
182        """
183        pygame.init()
184        screen = pygame.display.set_mode((750, 700))
185        pygame.display.set_caption("Game Win")
186        background = pygame.image.load("Graphics/game_win.jpg")
187        background = pygame.transform.scale(background, (750, 700))
188        pygame.mixer.music.stop()
189        game_win_sound = pygame.mixer.Sound("Sounds/game_win.mp3")
190        game_win_sound.play()
191
192        running = True
193        while running:
194            for event in pygame.event.get():
195                if event.type == pygame.QUIT:
196                    running = False
197                    break
198            screen.blit(background, (0, 0))
199            pygame.display.flip()
200
201        pygame.quit()
202        exit()
```

- Xử lý va chạm:

```
197
198     def check_for_collisions(self):
199         """
200         Kiểm tra và xử lý va chạm giữa các thành phần trong trò chơi.
201         """
202         if self.spaceship_group.sprite.lasers_group:
203             aliens_hit = []
204             for laser_sprite in self.spaceship_group.sprite.lasers_group:
205                 aliens_hit += pygame.sprite.spritecollide(laser_sprite, self.aliens_group, True)
206                 if pygame.sprite.spritecollide(laser_sprite, self.mystery_ship_group, False):
207                     self.explosion_sound.play()
208                     laser_sprite.kill()
209                     self.mystery_ship_hit_count += 1
210                     if self.mystery_ship_hit_count == 3:
211                         self.score += 1000
212                         self.check_and_load_highscore()
213                         self.mystery_ship_group.empty()
214                         self.game_win()
215                 for obstacle in self.obstacles:
216                     if pygame.sprite.spritecollide(laser_sprite, obstacle.blocks_group, True):
217                         laser_sprite.kill()
218             if aliens_hit:
219                 self.explosion_sound.play()
220                 for alien in aliens_hit:
221                     self.score += 100
222                     self.check_and_load_highscore()
223         if self.alien_lasers_group:
224             for laser_sprite in self.alien_lasers_group:
225                 if pygame.sprite.spritecollide(laser_sprite, self.spaceship_group, False):
226                     laser_sprite.kill()
227                     self.lives -= 1
228                     if self.lives == 0:
229                         self.game_over()
230                 for obstacle in self.obstacles:
231                     if pygame.sprite.spritecollide(laser_sprite, obstacle.blocks_group, True):
232                         laser_sprite.kill()
233         if self.mystery_ship_lasers_group:
234             for laser_sprite in self.mystery_ship_lasers_group:
235                 if pygame.sprite.spritecollide(laser_sprite, self.spaceship_group, False):
236                     laser_sprite.kill()
237                     self.lives -= 1
238                     if self.lives == 0:
239                         self.game_over()
240                 for obstacle in self.obstacles:
241                     if pygame.sprite.spritecollide(laser_sprite, obstacle.blocks_group, True):
242                         laser_sprite.kill()
243         if self.aliens_group:
244             for alien in self.aliens_group:
245                 for obstacle in self.obstacles:
246                     pygame.sprite.spritecollide(alien, obstacle.blocks_group, True)
247                 if pygame.sprite.spritecollide(alien, self.spaceship_group, False):
248                     self.game_over()
249
```

Hàm check_for_collisions kiểm tra và xử lý va chạm giữa các thành phần trong trò chơi. Dưới đây là giải thích chi tiết từng phần của hàm:

1. **Va chạm giữa laser của spaceship và aliens:**
   - Đầu tiên, kiểm tra xem nhóm lasers của spaceship có không. Nếu có, tạo một danh sách trống để lưu các alien bị tác động.
   - Tiếp theo, lặp qua từng laser trong nhóm lasers của spaceship.

- Sử dụng pygame.sprite.spritecollide để kiểm tra va chạm giữa laser hiện tại và nhóm aliens. Nếu có va chạm, thêm tất cả các alien bị tác động vào danh sách aliens_hit.
- Nếu có va chạm giữa laser và nhóm tàu bí ẩn, kiểm tra xem tàu bí ẩn đã bị bắn 3 lần chưa. Nếu đã bị bắn 3 lần, tăng điểm và gọi hàm game_win() để kết thúc trò chơi.
- Kiểm tra va chạm giữa laser và các viên gạch của các chướng ngại vật. Nếu có va chạm, loại bỏ laser.
- Nếu aliens_hit không rỗng, phát âm thanh nổ và tăng điểm cho mỗi alien bị tác động.

2. **Va chạm giữa laser của aliens và spaceship:**
   - Kiểm tra va chạm giữa laser của aliens và spaceship. Nếu có va chạm, loại bỏ laser và giảm số lượt chơi (lives) của người chơi.
   - Kiểm tra va chạm giữa laser và các viên gạch của các chướng ngại vật. Nếu có va chạm, loại bỏ laser.
3. **Va chạm giữa laser của tàu bí ẩn và spaceship:**
   - Tương tự như va chạm giữa laser của aliens và spaceship, kiểm tra va chạm giữa laser của tàu bí ẩn và spaceship.
   - Nếu có va chạm, loại bỏ laser và giảm số lượt chơi (lives) của người chơi.
   - Kiểm tra va chạm giữa laser và các viên gạch của các chướng ngại vật. Nếu có va chạm, loại bỏ laser.
4. **Va chạm giữa aliens và spaceship:**
   - Kiểm tra va chạm giữa mỗi alien và spaceship. Nếu có va chạm, gọi hàm game_over() để kết thúc trò chơi.

- Kiểm tra và cập nhật highscore:

```
206     def check_and_load_highscore(self):
207         """
208         Kiểm tra và cập nhật điểm cao nhất nếu điểm hiện tại cao hơn.
209         """
210         try:
211             with open("highscore.txt", "r") as file:
212                 self.highscore = int(file.read())
213         except FileNotFoundError:
214             self.highscore = 0
215 
216         if self.score > self.highscore:
217             self.highscore = self.score
218             with open("highscore.txt", "w") as file:
219                 file.write(str(self.highscore))
```

9. Tạo class Main để chạy trò chơi:

- Khởi tạo thành phần chính và đtawj các sự kiện diễn ra

```python
class Main:
    def __init__(self):
        """
        Khởi tạo các thành phần chính của trò chơi và đặt sự kiện bắn laser và tạo tàu bí ẩn.
        """
        pygame.init()
        self.play = pygame.transform.scale(pygame.image.load("Graphics/play.jpg"), (2000, 2000))
        self.font = pygame.font.Font("Graphics/monogram.ttf", 40)
        self.score_text_surface = self.font.render("SCORE", False, blue)
        self.highscore_text_surface = self.font.render("HIGH-SCORE", False, blue)
        self.screen = pygame.display.set_mode((screen_width + offset, screen_height + 2 * offset))
        pygame.display.set_caption("Python Space Invaders")
        self.clock = pygame.time.Clock()
        self.game = Game(screen_width, screen_height, offset)

        self.SHOOT_LASER = pygame.USEREVENT
        pygame.time.set_timer(self.SHOOT_LASER, 300)

        self.MYSTERYSHIP = pygame.USEREVENT + 1
        pygame.time.set_timer(self.MYSTERYSHIP, random.randint(4000, 8000))
```

- Cập nhật trạng thái, sự kiện và vẽ cái phần tử lên màn hình

```python
    def run(self):
        """
        Bắt sự kiện, cập nhật trạng thái và vẽ các phần tử trên màn hình.
        """
        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
                if event.type == self.SHOOT_LASER and self.game.run:
                    self.game.alien_shoot_laser()
                if event.type == self.MYSTERYSHIP and self.game.run:
                    self.game.create_mystery_ship()
                    pygame.time.set_timer(self.MYSTERYSHIP, random.randint(4000, 8000))
            if self.game.run:
                self.game.spaceship_group.update()
                self.game.move_aliens()
                self.game.alien_lasers_group.update()
                self.game.mystery_ship_group.update()
                self.game.check_for_collisions()
            self.screen.blit(self.play, (-100, -100))
            pygame.draw.rect(self.screen, blue, (10, 10, 780, 780), 2)
            pygame.draw.line(self.screen, blue, (25, 700), (775, 700), 3)
            x = 50
            for life in range(self.game.lives):
                self.screen.blit(self.game.spaceship_group.sprite.image, (x, 720))
                x += 50
            self.screen.blit(self.score_text_surface, (30, 15, 50, 50))
            formatted_score = str(self.game.score).zfill(5)
            score_surface = self.font.render(formatted_score, False, blue)
            self.screen.blit(score_surface, (30, 40, 50, 50))
            self.screen.blit(self.highscore_text_surface, (610, 710, 50, 50))
            formatted_highscore = str(self.game.highscore).zfill(5)
            highscore_surface = self.font.render(formatted_highscore, False, blue)
            self.screen.blit(highscore_surface, (680, 740, 50, 50))
            self.game.spaceship_group.draw(self.screen)
            self.game.spaceship_group.sprite.lasers_group.draw(self.screen)
            for obstacle in self.game.obstacles:
                obstacle.blocks_group.draw(self.screen)
            self.game.aliens_group.draw(self.screen)
            self.game.alien_lasers_group.draw(self.screen)
            self.game.mystery_ship_group.draw(self.screen)
            pygame.display.update()
            self.clock.tick(60)
```

10. Tạo class Button để làm Menu:
    - Khởi tạo thuộc tính:

button.py > Button

```python
class Button():
    def __init__(self, image, pos, text_input, font, base_color, hovering_color):
        """
        Khởi tạo tất cả các thuộc tính cần thiết cho đối tượng nút.

        Tham số:
        image : pygame.Surface
            Hình ảnh của nút.
        pos : tuple
            Tọa độ (x, y) của nút.
        text_input : str
            Văn bản hiển thị trên nút.
        font : pygame.font.Font
            Phông chữ được sử dụng để hiển thị văn bản trên nút.
        base_color : tuple
            Màu sắc cơ bản của văn bản khi nút không được hover.
        hovering_color : tuple
            Màu sắc của văn bản khi nút được hover.
        """
        self.image = image
        self.x_pos = pos[0]
        self.y_pos = pos[1]
        self.font = font
        self.base_color, self.hovering_color = base_color, hovering_color
        self.text_input = text_input
        self.text = self.font.render(self.text_input, True, self.base_color)
        if self.image is None:
            self.image = self.text
        self.rect = self.image.get_rect(center=(self.x_pos, self.y_pos))
        self.text_rect = self.text.get_rect(center=(self.x_pos, self.y_pos))

```

    - Cập nhật vị trí và vẽ nút:

```python
    def update(self, screen):
        """
        Cập nhật và vẽ nút lên màn hình.

        Tham số:
        screen : pygame.Surface
            Màn hình trò chơi để vẽ nút.
        """
        if self.image is not None:
            screen.blit(self.image, self.rect)
        screen.blit(self.text, self.text_rect)

```

- Kiểm tra yêu cầu người chơi:

```
    def checkForInput(self, position):
        """
        Kiểm tra xem vị trí con trỏ chuột có nằm trong vùng của nút hay không.

        Tham số:
        position : tuple
            Vị trí (x, y) của con trỏ chuột.

        Trả về:
        bool
            True nếu con trỏ chuột nằm trong vùng của nút, ngược lại False.
        """                                                                    (function) top: Any
        if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top, self.rect.bottom):
            return True
        return False

```

11. Tạo class MainGame để quản lý menu chính của trò chơi
    - Khởi tạo lớp và lấy font để làm menu

```
class MainGame:
    def __init__(self):
        """Khởi tạo lớp MainGame."""
        pygame.init()
        self.SCREEN = pygame.display.set_mode((750, 700))
        pygame.display.set_caption("Menu")
        self.BG_Menu = pygame.image.load("Graphics/Background.png").convert()

    def get_font(self, size):
        """Lấy một đối tượng font Pygame với kích thước đã cho.

        Args:
            size (int): Kích thước của font.

        Returns:
            pygame.font.Font: Đối tượng font Pygame.
        """
        return pygame.font.Font("Graphics/font.ttf", size // 4 * 3)
```

- Chạy chương trình

```
    def run(self):
        """Chạy vòng lặp menu chính."""
        running = True
        while running:
            self.SCREEN.blit(self.BG_Menu, (0, 0))
            MENU_MOUSE_POS = pygame.mouse.get_pos()
            MENU_TEXT = self.get_font(100).render("MAIN MENU", True, (255, 255, 204))
            MENU_RECT = MENU_TEXT.get_rect(center=(375, 150))
            PLAY_BUTTON = Button(image=pygame.image.load("Graphics/Play Rect.jpg"), pos=(375, 400),
                                 text_input="PLAY", font=self.get_font(80), base_color="#333333", hovering_color="#333333")
            self.SCREEN.blit(MENU_TEXT, MENU_RECT)

            for button in [PLAY_BUTTON]:
                button.update(self.SCREEN)

            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False
                if event.type == pygame.MOUSEBUTTONDOWN:
                    if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                        game_app = Main()
                        game_app.run()
                        running = False

            pygame.display.update()

        pygame.quit()
        sys.exit()

main = MainGame()
main.run()
```