Difficulty: 🌶🌶🌶🌶🌶	Language: Python	Requires: Laptop with Python, coderdojo [games]

Waves of aliens are attacking the Earth! We need to defend our planet by shooting them down with our laser cannon. In this you will learn how to use the pygame library to make a game, along with classes and lists.

This sheet recommends using Thonny in Python 3 mode.

Assets

Get the assets with `kenney-assets install space-shooter-redux`

Starting up pygame

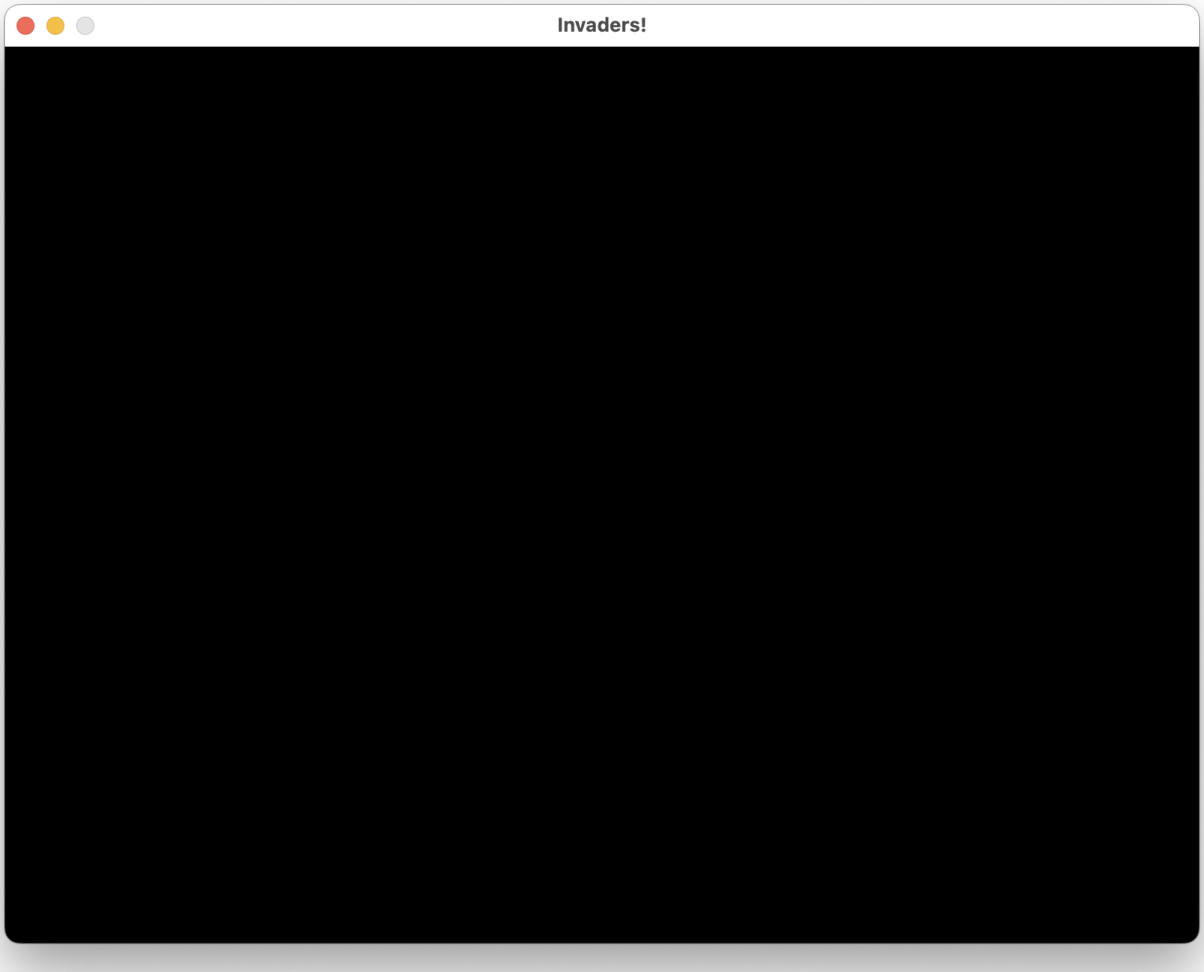
Pygame needs a couple of things to get started:

```
import pygame

WIDTH=800
HEIGHT=600
FPS=30

pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Invaders!")
clock = pygame.time.Clock()

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    clock.tick(FPS)
```

We've set a display size, you can adjust to something bigger if you like. There's a caption, which is the text that appears in the window title bar.

We also have a clock so the game can run at a consistent speed.

The `running` variable is used to control the main loop, and the `for` loop is used to process events. Events are things like mouse clicks, key presses, and window close events. At this point, this only handles closing the window.


Invaders!

Drawing a player

We need a player to shoot the aliens, so let's draw one.

We'll show added code with a + sign, with lines around it to show where. Do not type the + sign.

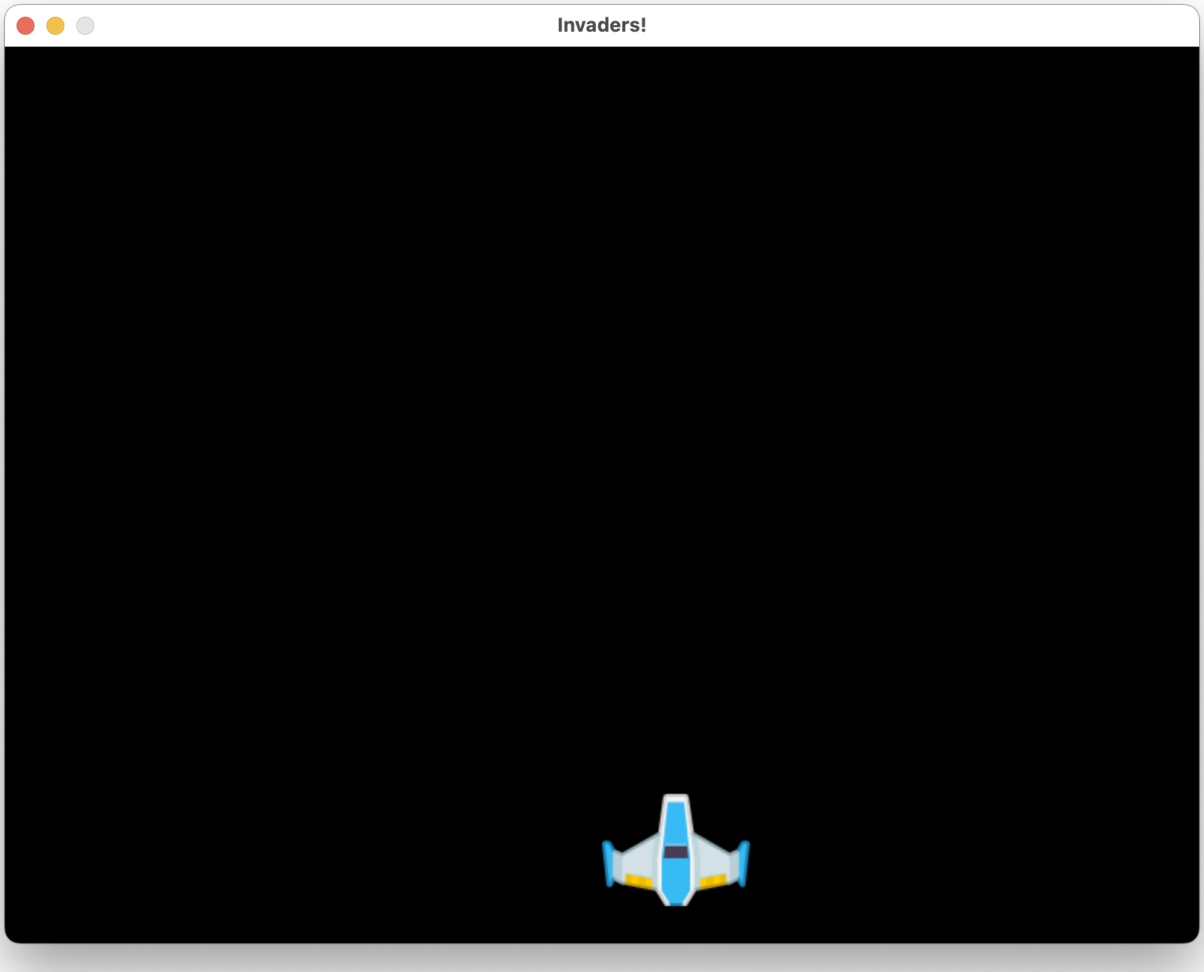
```
clock = pygame.time.Clock()
+ asset_path = "assets/space-shooter-redux/PNG/"
+ player_img = pygame.image.load(asset_path + "playerShip1_blue.png")
```

We are loading the image here, and storing it in the `player_img` variable. The variable `asset_path` will help us load assets.

Then in the main loop we need to draw it:

```
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
+   screen.blit(player_img, (WIDTH/2, HEIGHT-100))
+   pygame.display.update()
    clock.tick(FPS)
```

You should now see a player ship near the bottom of the screen.


Invaders!

Centering the player

You might have noticed that the player ship is not centred on the screen.

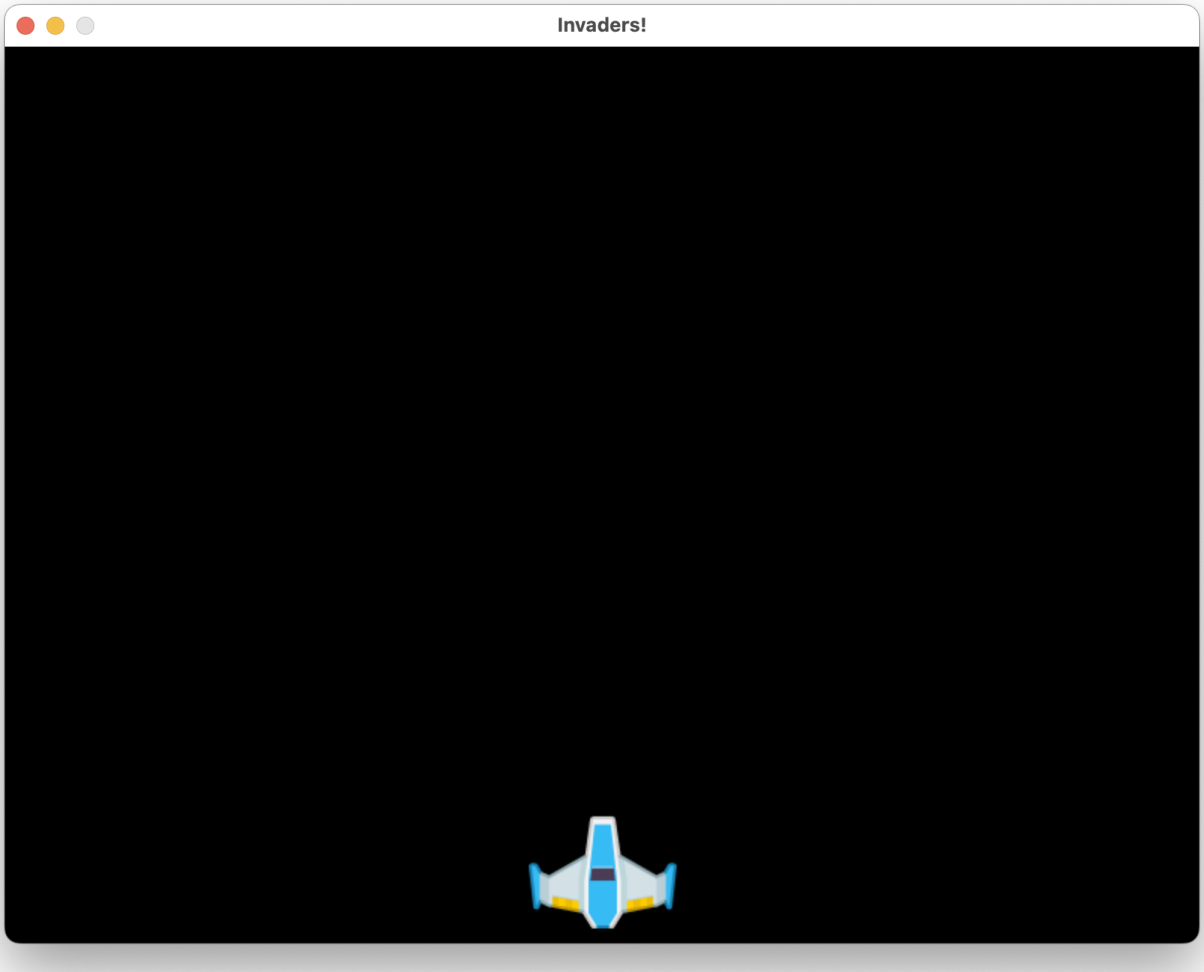
```
  player_img = pygame.image.load(asset_path + "playerShip1_blue.png")
+ player_rect = player_img.get_rect()
+ player_rect.centerx = WIDTH/2
+ player_rect.bottom = HEIGHT-10
```

The – sign in the code means that we've removed a line and swapped it for something else. Do not type the – sign. Make this change in the main loop:

```
-   screen.blit(player_img, (WIDTH/2, HEIGHT-100))
+   screen.blit(player_img, player_rect))
    pygame.display.update()
```

We've fixed this by using the `get_rect()` method to get the size of the image, and then centring it. We can also set the bottom of the image to be 10 pixels above the bottom of the screen.

We've also changed the `blit` call to use the `player_rect` variable instead of the coordinates.

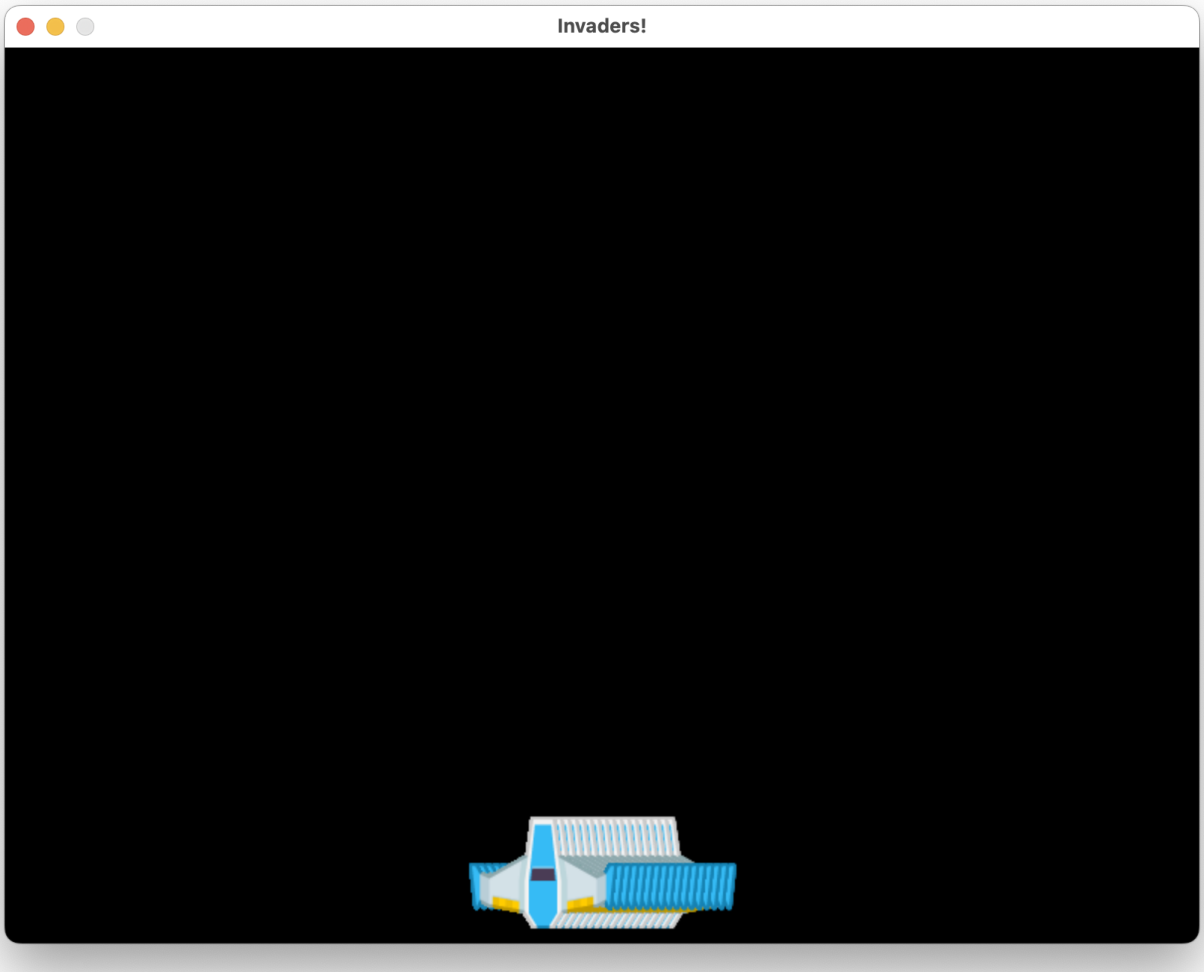

Invaders!

Moving the player

This isn't much fun yet. We want things to move. We'll use the left and right arrow keys to move the player.

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
+   keys = pygame.key.get_pressed()
+   if keys[pygame.K_LEFT]:
+       player_rect.centerx -= 5
+   elif keys[pygame.K_RIGHT]:
+       player_rect.centerx += 5
    screen.blit(player_img, player_rect)
    pygame.display.update()
    clock.tick(FPS)
```

This code checks for key presses, and moves the player left or right by 5 pixels. Let's run this!



Hmm - this doesn't quite look right! It's gliding around leaving trails.

We need to clear the screen before we draw the player. We can do this by filling the screen with a colour:

```
+   screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
    pygame.display.update()
```

Checkpoint

Your code at this stage should look like:

```
import pygame

WIDTH=800
HEIGHT=600
FPS=30

pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Invaders!")
clock = pygame.time.Clock()
asset_path = "assets/space-shooter-redux/PNG/"
player_img = pygame.image.load(asset_path + "playerShip1_blue.png")
player_rect = player_img.get_rect()
player_rect.centerx = WIDTH/2
player_rect.bottom = HEIGHT-10


running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        player_rect.centerx -= 5
    elif keys[pygame.K_RIGHT]:
        player_rect.centerx += 5
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
    pygame.display.update()
    clock.tick(FPS)
```

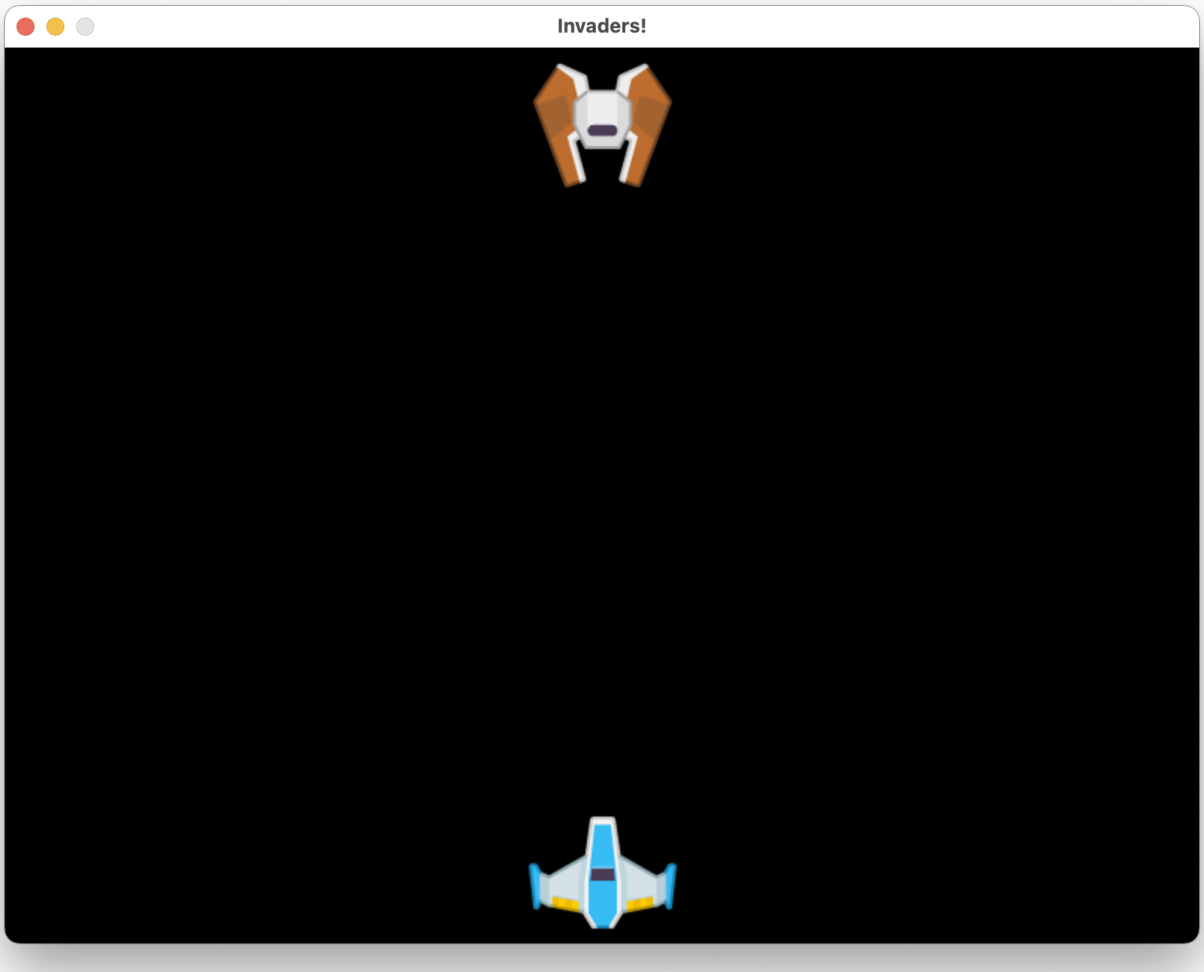
Adding an alien

We need some aliens to shoot at. Let's add one.

```
+ alien_img = pygame.image.load(asset_path + "Enemies/enemyRed1.png")
+ alien_rect = alien_img.get_rect()
+ alien_rect.centerx = WIDTH/2
+ alien_rect.top = 10
```

We can then draw it in the main loop:

```
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
+   screen.blit(alien_img, alien_rect)
    pygame.display.update()
```



This is ok, but it's not very exciting. We are going to make an alien sprite class, and use this to move the alien.

```
- alien_img = pygame.image.load(asset_path + "Enemies/enemyRed1.png")
- alien_rect = alien_img.get_rect()
- alien_rect.centerx = WIDTH/2
- alien_rect.top = 10
+ class Alien(pygame.sprite.Sprite):
+     def __init__(self):
+         pygame.sprite.Sprite.__init__(self)
+         self.image = pygame.image.load(asset_path +
"Enemies/enemyRed1.png")
+         self.rect = self.image.get_rect()
+         self.rect.centerx = WIDTH/2
+         self.rect.top = 10
+         self.speedx = 5
+
+   def update(self):
+       self.rect.centerx += self.speedx
+       if self.rect.right > WIDTH:
+         self.speedx = -5
+       if self.rect.left < 0:
+         self.speedx = 5
+
+  def draw(self):
+       screen.blit(self.image, self.rect)
+ alien = Alien()
```

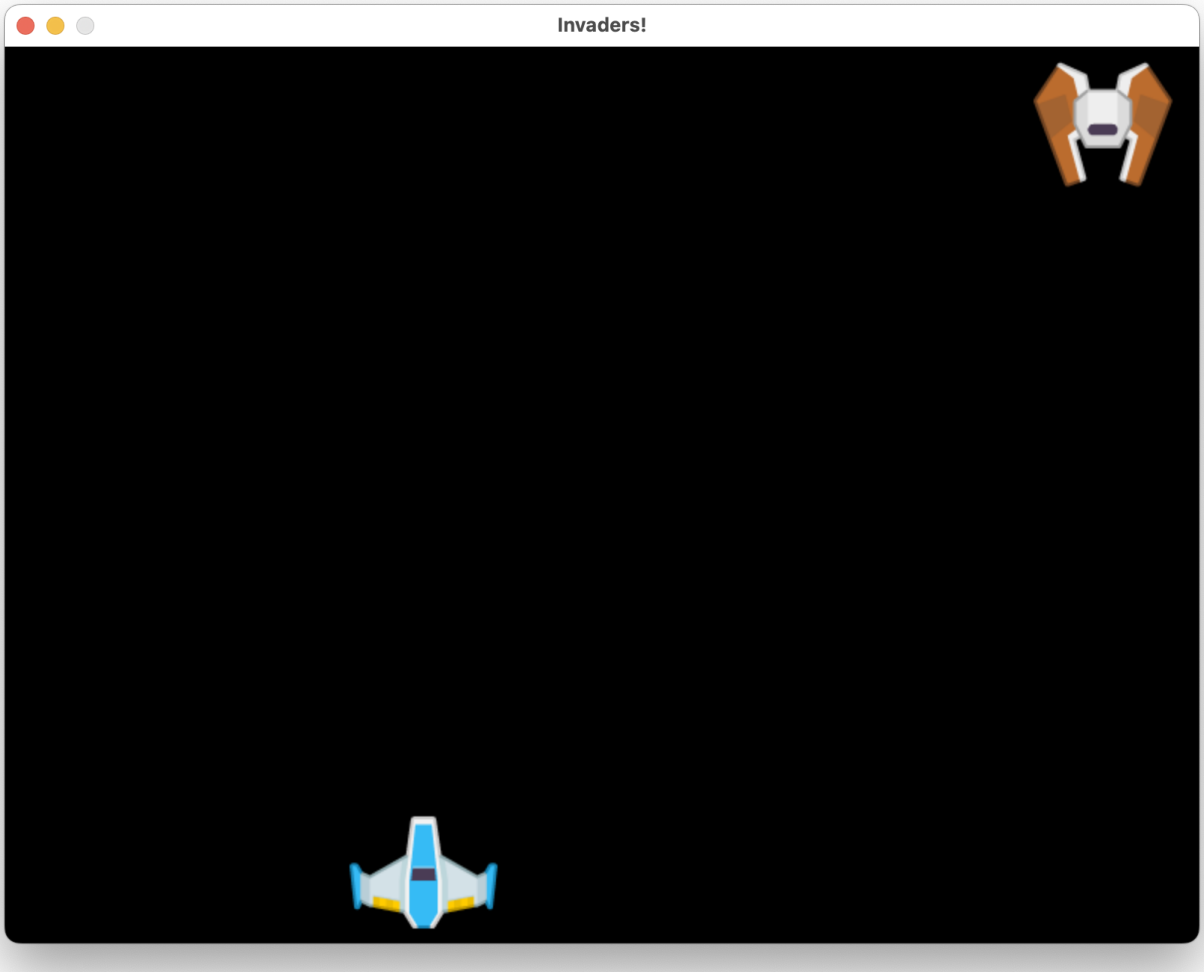
We've moved the code for creating the alien into this class. The class lets us group together the code for the alien, and makes it easier to add more aliens later. It's built from a Pygame sprite, so it can borrow code from the Pygame sprite class.

A `method` is like a function, but can use variables from the class.

The class has an `__init__` method, which is a recipe to make an alien. The `Alien` also has an `update` method to move it. This uses a `speedx` (as in speed in the x direction) variable to move, and when the alien reaches the edge of the screen, it reverses direction. We've also added a `draw` method to draw the alien.

Let's use this in the main loop:

```
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
-   screen.blit(alien_img, alien_rect)
+   alien.update()
+   alien.draw()
    pygame.display.update()
```


Invaders!

Checkpoint

Your code at this stage should look like:

```
import pygame

WIDTH=800
HEIGHT=600
FPS=30

pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Invaders!")
clock = pygame.time.Clock()
asset_path = "assets/space-shooter-redux/PNG/"
player_img = pygame.image.load(asset_path + "playerShip1_blue.png")
player_rect = player_img.get_rect()
player_rect.centerx = WIDTH/2
player_rect.bottom = HEIGHT-10

alien_img = pygame.image.load(asset_path + "Enemies/enemyRed1.png")
alien_rect = alien_img.get_rect()
alien_rect.centerx = WIDTH/2
alien_rect.top = 10

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        player_rect.centerx -= 5
    elif keys[pygame.K_RIGHT]:
        player_rect.centerx += 5
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
    screen.blit(alien_img, alien_rect)
    pygame.display.update()
    clock.tick(FPS)
```

Adding a bullet

The player should shoot lasers at the alien. Let's add a bullet class for these.

```
+ class Bullet(pygame.sprite.Sprite):
+     def __init__(self):
+         pygame.sprite.Sprite.__init__(self)
+         self.image = pygame.image.load(asset_path +
"Lasers/laserBlue01.png")
+         self.rect = self.image.get_rect()
+         self.rect.centerx = player_rect.centerx
+         self.rect.bottom = player_rect.top
+         self.speedy = -10
+
+     def update(self):
+         self.rect.y += self.speedy
+         if self.rect.bottom < 0:
+             self.kill()
+
+     def draw(self):
+         screen.blit(self.image, self.rect)

alien = Alien()
```

This has a `speedy` variable to move the bullet up the screen. It also has a `kill` method to remove the bullet from the game.

Pygame can manage a group of sprites for us, so we can add a group for the bullets:

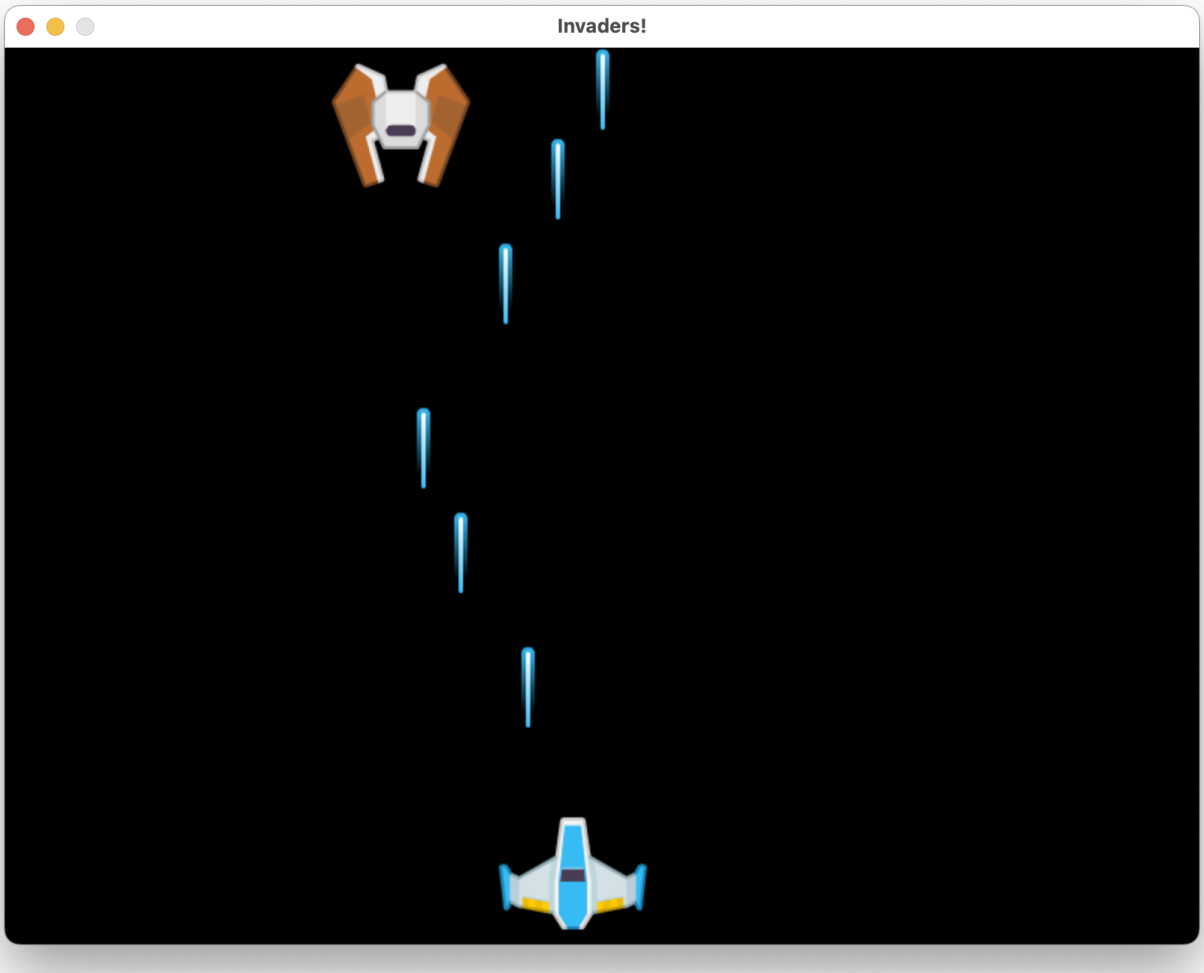
```


alien = Alien()

+ all_sprites = pygame.sprite.Group()
+ bullets = pygame.sprite.Group()
+ all_sprites.add(alien)
```

We can then add a bullet to the game when the player presses the space bar:

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
+       elif event.type == pygame.KEYDOWN:
+           if event.key == pygame.K_SPACE:
+               bullet = Bullet()
+               all_sprites.add(bullet)
+               bullets.add(bullet)
    if keys[pygame.K_LEFT]:
        player_rect.centerx -= 5
    elif keys[pygame.K_RIGHT]:
        player_rect.centerx += 5
```

We can then update and draw the alien and bullet sprites in the main loop:

```
+   all_sprites.update()
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
-    alien.update()
-    alien.draw()
+   all_sprites.draw(screen)
    pygame.display.update()
```



These bullets are nice, but they are sailing right through the enemy. We need to add some collision detection.

Checkpoint

```
import pygame

WIDTH=800
HEIGHT=600
FPS=30

pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Invaders!")
clock = pygame.time.Clock()
asset_path = "assets/space-shooter-redux/PNG/"
player_img = pygame.image.load(asset_path + "playerShip1_blue.png")
player_rect = player_img.get_rect()
player_rect.centerx = WIDTH/2
player_rect.bottom = HEIGHT-10

class Alien(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(asset_path +
"Enemies/enemyRed1.png")
        self.rect = self.image.get_rect()
        self.rect.centerx = WIDTH/2
        self.rect.top = 10
        self.speedx = 5

    def update(self):
        self.rect.centerx += self.speedx
        if self.rect.right > WIDTH:
          self.speedx = -5
        if self.rect.left < 0:
          self.speedx = 5

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

class Bullet(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(asset_path +
"Lasers/laserBlue01.png")
        self.rect = self.image.get_rect()
        self.rect.centerx = player_rect.centerx
        self.rect.bottom = player_rect.top
        self.speedy = -10

    def update(self):
        self.rect.y += self.speedy
        if self.rect.bottom < 0:
            self.kill()

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

```
alien = Alien()
all_sprites = pygame.sprite.Group()
bullets = pygame.sprite.Group()
all_sprites.add(alien)

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                bullet = Bullet()
                all_sprites.add(bullet)
                bullets.add(bullet)
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        player_rect.centerx -= 5
    elif keys[pygame.K_RIGHT]:
        player_rect.centerx += 5
    all_sprites.update()
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
    all_sprites.draw(screen)
    pygame.display.update()
    clock.tick(FPS)
```

Collisions!

We want to stop those aliens somehow. Let's help the bullets make contact.

```
    all_sprites.update()
+   hits = pygame.sprite.spritecollide(alien, bullets, True)
+   if hits:
+       alien.kill()
    screen.fill((0,0,0))
    screen.blit(player_img, player_rect)
    all_sprites.draw(screen)
    pygame.display.update()
    clock.tick(FPS)
```

The good news is that the sprite groups, along with the rectangles make this easy! We can use the `spritecollide` method to check if any of the bullets have collided with the alien. If they have, we can remove the alien and the bullet.

We can run this, and you should be able to fire and shoot down the alien. However, there's not much fanfare, the alien simply disappears. Let's add some explosions.

Explosions

We can add an explosion class to the game. This will be a sprite that will be added to the game when the alien is killed.

```

```