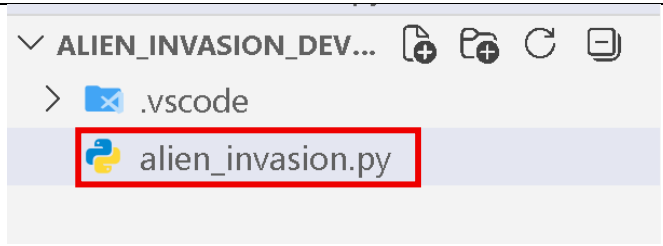
Práctica 1:	**Planificación del Proyecto**
Cuando creamos un proyecto grande, es importante trazar un plan antes de empezar a escribir código. ¿Este plan? Le ayudará a mantenerse centrado y aumenta las posibilidades de completar el proyecto. Vamos a escribir una descripción de la mecánica del juego. Aunque la siguiente descripción no cubre todos los detalles de alien invasión, da una idea clara de cómo empezar a montar El juego.	
En _____, el jugador controla una nave que aparece en el centro de la pantalla, en la parte inferior. El _____ puede mover la nave hacia la _____ y hacia la _____ con las _____ y disparar balas con la _____. Cuando comienza el juego, una flota de extraterrestres llena el cielo y se mueve hacia abajo por la pantalla. El _____ dispara a los aliens y los destruye. Cuando el _____ **consiga acabar con todos los** _____, aparece una **nueva flota que se** _____ **más rápido** que el anterior. Si un alien _____ la nave del jugador o llega al _____ de la pantalla, el jugador _____. El juego **termina** cuando el jugador pierde _____.	
Para la **primera fase de desarrollo**, haremos una **nave** que se pueda desplazar hacia la **derecha** y hacia la **izquierda** cuando el jugador pulse las **flechas de dirección del teclado** y que **dispare** cuando el jugador **presione la barra espaciadora**. Tras configurar este comportamiento, podemos crear los aliens y refinar la mecánica del juego.	
Sello de realizado en clase	

Práctica 2:	**Instalar Pygame**
1. **Crear una nueva carpeta: alien_invasion** 2. **Abrir carpeta desde VSCode** 3. **En consola escribir:**	
	
Sello de culminado en clase	

Práctica 3: Crear ventana y responder a entrada de usuario

alien_invasion.py > ...

```python
# CECyTEM Plantel Villa del Carbón
# Codifica Software de Sistemas Informáticos
# Mi primer juego con Pygame: Alien Invasion
# Programadora: LizDev

import sys # Para salir del juego cuando el jugador lo desee.

import pygame # Funciones para videojuegos con gráficos y sonidos.

class AlienInvasion: # Clase para gestionar recursos y comportamiento del juego.

    def __init__(self): # Inicializa el juego y crea los recursos del juego.
        pygame.init() #Inicializa configuración de fondo, para funcionar correctamente.

        self.screen = pygame.display.set_mode((1200, 800))# Ventana de 1200x800 para el juego.
        pygame.display.set_caption("Alien Invasion") # Título de la ventana.

    def run_game(self): # Controla el juego.

        while True: # Continuamente escucha eventos de teclado y de ratón y se actualiza la pantalla.
            for event in pygame.event.get():# Responde a eventos de teclado y de ratón.
                if event.type == pygame.QUIT: # Si da clic en el Botón cerrar.
                    sys.exit() # Sale del juego.

        pygame.display.flip() # Hace visible la últma pantalla dibujada.

if __name__ == '__main__':    # Crea una instancia del juego y lo ejecuta.
    ai = AlienInvasion()
    ai.run_game()
```

Clase:

Métodos:

¿Qué es un evento?

Sello de realizado en clase

Práctica 4. Reloj pygame

```python
# CECyTEM Plantel Villa del Carbón
# Codifica Software de Sistemas Informáticos
# Mi primer juego con Pygame: Alien Invasion
# Programadora: LizDev

import sys

import pygame

class AlienInvasion:

    def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock() # Controla la velocidad del juego.

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

    def run_game(self):

        while True:
            pygame.display.flip() # Actualiza la pantalla en cada iteración para mostrar los cambios.
            self.clock.tick(60) # Limita la velocidad del juego a 60 FPS.
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    sys.exit()


if __name__ == '__main__':
    ai = AlienInvasion()
    ai.run_game()
```

Lo ideal es que los juegos se ejecuten a la misma velocidad o con la misma tasa de *frames* o cuadros por segundo *FPS* en cualquier sistema. Pygame ofrece, un reloj, es decir, calculará la cantidad de tiempo correcto para hacer una pausa. Con el objetivo de que el juego se ejecute en una taza consistente.

Explica el **método tick**: _____

El blucle cuantas veces se ejecuta exactamente **por segundo**:

_____

Sello de culminado en clase

Práctica 5: Configurar el fondo de color

Paint crea una pantalla negra por defecto, pero es aburrido. Vamos a configurar un color de fondo distinto. Lo haremos al final del método __init__( ).

```python
import pygame

class AlienInvasion:

    def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock()

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

        #Configurar el fondo del juego
        self.bg_color = (230, 230, 230) # Color de fondo gris claro.

    def run_game(self):

        while True:
            self.screen.fill(self.bg_color) # Rellena la pantalla con el color de fondo.
            pygame.display.flip()
            self.clock.tick(60)

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

Pygame y el Sistema **RGB**: mezcla de **rojo**, **verde** y **azul**.

Cada color tiene un valor que puede ir de cero a _____.

El valor de color **ROJO** (**255**,0,0).

El valor de color **VERDE** (0, **255**,0).

El valor del color **AZUL** (, ,).

Podemos mezclar distintos valores RGB para crear hasta 16 millones de colores. **Por ejemplo:** **Red**, **Green**, **Blue**

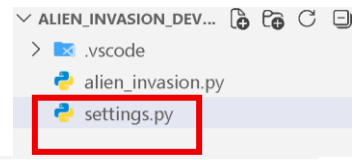
El color _____ es con los valores (230, 230, 230).

El método fill() permite: _____

Sello de realizado en clase

Práctica 6. Clase Settings

Cada vez que introducimos una funcionalidad en el juego, crearemos una nueva configuración.



settings.py > ...

```python
# Clase Settings para almacenar la configuración de la aplicación
class Settings:
    """Una clase para almacenar todas las configuraciones de Alien Invasion."""

    def __init__(self):
        """Inicializa las configuraciones estáticas del juego."""
        # Configuraciones de la pantalla
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230, 230, 230)  # Color de fondo gris claro.
```

El archivo _____ contiene la clase Settings. Esta clase sólo tiene un método __init__( ), que inicializa atributos que controlan el aspecto del juego y la velocidad de la nave.

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

class AlienInvasion:

    def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings() # intanciamos Settings para acceder a sus atributos

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


    def run_game(self):

        while True:
            self.screen.fill(self.settings.bg_color) # Rellena la pantalla con el color de fondo.
            pygame.display.flip()
            self.clock.tick(60)

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

Sello de culminado en clase

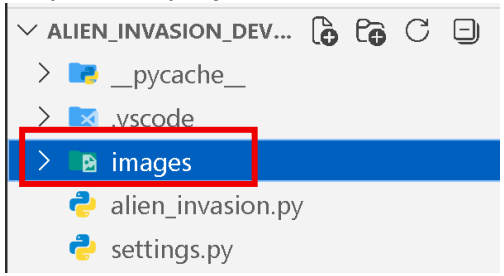
Práctica 7: Añadir la imagen de la nave

Para el manejo de imágenes se recomienda utilizar **mapa de bits (.bmp)** porque Pygame carga estos archivos por defecto, aunque se puede configurar para que utilice otro tipo de archivos. Como **.JPG** o **.PNG**. Pero hay que convertirlas en mapa de bits con herramientas como photoshop o paint.

Existen sitios como: https://opengameart.org/
https://icon-icons.com/

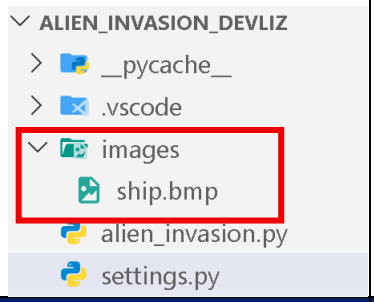
ship.bmp

1. Crea una carpeta llamada imágenes dentro de la carpeta del proyecto:

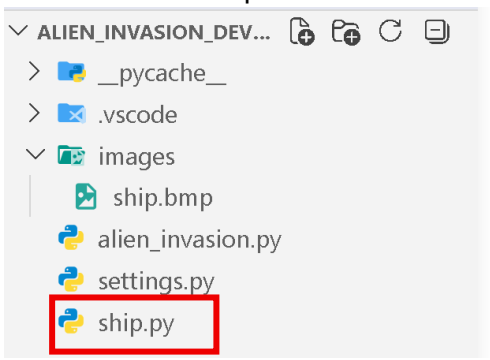


2. Guada el archivo ship.bmp dentro de la carpeta:

3. Arrastrar y soltar:



Crear la clase Ship



ship.py. = Gestiona la _____ de la nave.

Blitme= _____ la nave en la pantalla.

ship.bmp= _____

_____

ship.py

ship.py > Ship > blitme

```python
import pygame
class Ship:
    """Una clase para administrar la nave espacial."""

    def __init__(self, ai_game):
        """Inicializa la nave espacial y establece su posición inicial."""
        self.screen = ai_game.screen
        self.screen_rect = ai_game.screen.get_rect()

        # Carga la imagen de la nave y obtiene su rectángulo.
        self.image = pygame.image.load('images/ship.bmp')
        self.rect = self.image.get_rect()

        # Inicia cada nueva nave en la parte inferior central de la pantalla.
        self.rect.midbottom = self.screen_rect.midbottom

    def blitme(self):
        """Dibuja la nave en su ubicación actual."""
        self.screen.blit(self.image, self.rect)
```

Sello de realizado en clase

Práctica 8: **Dibuja la nave en la pantalla**

alien_invasion.py > AlienInvasion > __init__

```python
import sys
import pygame
from settings import Settings
from ship import Ship #importamos Ship

class AlienInvasion:

    def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings()

        self.screen = pygame.display.set_mode(
            (self.settings.screen_width, self.settings.screen_height))
        pygame.display.set_caption("Alien Invasion")
        self.ship = Ship(self) # intanciamos Ship para acceder a sus atributos


    def run_game(self):

        while True:
            self.screen.fill(self.settings.bg_color)
            self.ship.blitme() # Dibuja la nave en su ubicación actual.
            pygame.display.flip()
            self.clock.tick(60)

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

Dibuja que observas en pantalla

Refactorización de métodos

La refactorización simplifica la estructura del código que tenemos escrito, de modo que sea más fácil. Es decir. Lo que hacemos es dividir el código largo, en este caso lo vamos a aplicar con el método run_ game.

alien_invasion.py

alien_invasion.py > AlienInvasion > __init__

```python
# Programadora: LizDev

import sys
import pygame
from settings import Settings
from ship import Ship #importamos Ship

class AlienInvasion:

    def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings()

        self.screen = pygame.display.set_mode(
            (self.settings.screen_width, self.settings.screen_height))
        pygame.display.set_caption("Alien Invasion")
        self.ship = Ship(self) # intanciamos Ship para acceder a sus atributos


    def run_game(self): # Inicia el bucle principal del juego.

        while True:
            self._check_events()
            self._update_screen()
            self.clock.tick(60)

    def _check_events(self):# Responde a los eventos de teclado y ratón.
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

    def _update_screen(self):# Actualiza las imágenes en la pantalla y cambia a la pantalla nueva.
        self.screen.fill(self.settings.bg_color)
        self.ship.blitme()
        pygame.display.flip()

if __name__ == '__main__':
    ai = AlienInvasion()
    ai.run_game()
```

Sello de realizado en clase	Sello de culminado en clase

Práctica 9: Pilotear la nave

alien_invasion.py

```python
# CECyTEM Plantel Villa del Carbón
# Codifica Software de Sistemas Informáticos
# Mi primer juego con Pygame: Alien Invasion
# Programadora: LizDev

import sys
import pygame
from settings import Settings
from ship import Ship

class AlienInvasion:

    def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings()

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

    def run_game(self):
        while True:
            self._check_events()
            self.ship.update()
            self._update_screen()
            self.clock.tick(60)

    def _check_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RIGHT:
                    self.ship.moving_right = True
                elif event.key == pygame.K_LEFT:
                    self.ship.moving_left = True
                elif event.key == pygame.K_q:
                    sys.exit()
            elif event.type == pygame.KEYUP:
                if event.key == pygame.K_RIGHT:
                    self.ship.moving_right = False
                elif event.key == pygame.K_LEFT:
                    self.ship.moving_left = False

    def _update_screen(self):
        self.screen.fill(self.settings.bg_color)
        self.ship.blitme()
        pygame.display.flip()

if __name__ == '__main__':
    ai = AlienInvasion()
    ai.run_game()
```

ship.py

```python
import pygame

class Ship: # Clase para administrar la nave espacial.

    def __init__(self, ai_game):# Inicializa la nave espacial y establece su posición inicial.
        self.screen = ai_game.screen
        self.screen_rect = ai_game.screen.get_rect()

        # Carga la imagen de la nave y obtiene su rectángulo.
        self.image = pygame.image.load('images/ship.bmp')
        self.rect = self.image.get_rect()

        # Inicia cada nueva nave en la parte inferior central de la pantalla.
        self.rect.midbottom = self.screen_rect.midbottom

        # Bandera de movimiento; empezamos sin movernos.
        self.moving_right = False
        self.moving_left = False

    def update(self): # Actualiza la posición de la nave según las teclas presionadas.
        if self.moving_right and self.rect.right < self.screen_rect.right:
            self.rect.x += 1
        if self.moving_left and self.rect.left > 0:
            self.rect.x -= 1

    def blitme(self): # Dibuja la nave en su ubicación actual.
        self.screen.blit(self.image, self.rect)
```

Ajustra velocidad de la nave:

settings.py

```python
# Clase Settings para almacenar la configuración de la aplicación
class Settings:
    """Una clase para almacenar todas las configuraciones de Alien Invasion."""

    def __init__(self):
        """Inicializa las configuraciones estáticas del juego."""
        # Configuraciones de la pantalla
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230, 230, 230)  # Color de fondo gris claro.

        # Configuraciones de la nave
        self.ship_speed = 1.5
```

Sello de realizado en clase

Sello de culminado en clase