Name: Michael Clair
Class: Python Programming
CSC 121-OA1
Date: April 20, 2025

Table of Contents

---

1. Project Overview

This project is a 1980s style shooting game created using Python and Pygame, following the *Python Crash Course* textbook chapters 12–14. I chose this project out of all the options because I played many games similar to this as a youth.

The game is a classic game with a similar setup to Galaga or Space Invaders. The player controls a ship that moves left and right at the bottom of the screen, firing bullets to destroy fleets of aliens moving across and downward in unison.

2. Installation Instructions

Requirements:

- Python 3.x installed
- Pygame installed (`pip install pygame`)
- Ship and Alien (or image of your choosing) saved to “images” files inside an `images/` folder

How to Run:

1. Open a Terminal or Command Prompt window.
2. Navigate to the folder where you saved the project files (for my computer this is

   `Documents)`.

3. Run the following command: python3 alien_invasion.py

```
alien_invasion_project — -bash — 80×24
Last login: Sat Apr 19 19:01:01 on ttys000
Michaels-MacBook-Pro:~ michaelclair$ cd Documents
Michaels-MacBook-Pro:Documents michaelclair$ cd alien_invasion_project
Michaels-MacBook-Pro:alien_invasion_project michaelclair$ python3 alien_invasion
.py
pygame 2.6.1 (SDL 2.28.4, Python 3.13.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
Michaels-MacBook-Pro:alien_invasion_project michaelclair$ python3 alien_invasion
.py
```

The game window should open and you can begin playing! If not, rely on your ol pal ChatGPT to help steer you on course.

---

3. How to Play

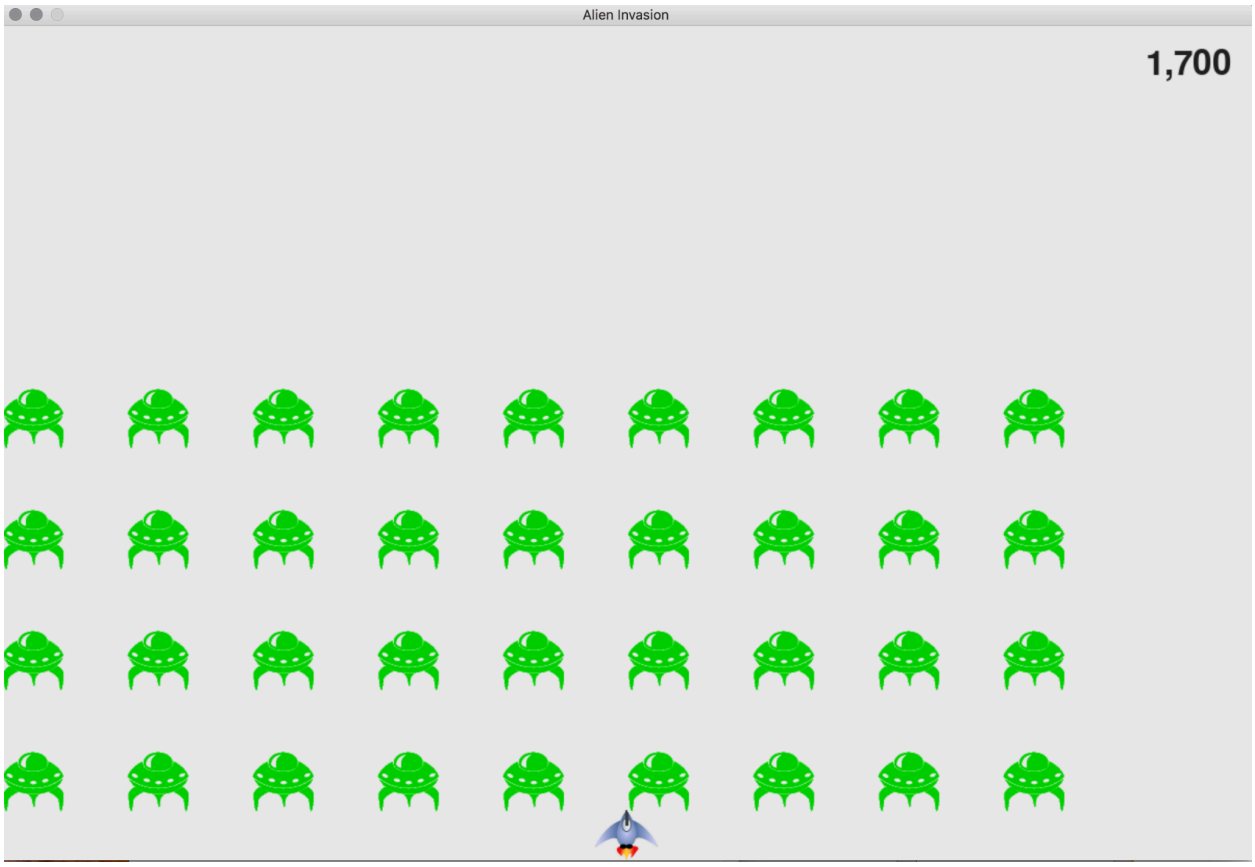
- **Arrow Right (→):** Move ship right
- **Arrow Left (←):** Move ship left
- **Space Bar:** Fire bullets
- **Q Key:** Quit the game

Destroy all aliens to complete the level and advance!
Each destroyed alien adds points to your score. I strongly recommend shooting the ones closest to you, as those will be the ones to kill you first!

You have **3 lives** (ships) before the game ends.

---

4. Screenshot of Actual Alien Invasion Game



---

5. Resources Used

- *Python Crash Course, 3rd Edition* by Eric Matthes
- Pygame Official Documentation: https://www.pygame.org/docs/

- Additional formatting and structure by ChatGPT

---

6. Full Code Listing

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

```
ALIEN INVASION
import sys
import pygame
from settings import Settings
from ship import Ship
from bullet import Bullet
from alien import Alien
from game_stats import GameStats
from scoreboard import Scoreboard

class AlienInvasion:
    """Overall class to manage game assets and behavior."""

    def __init__(self):
        """Initialize the game and create game resources."""
        pygame.init()
        self.settings = Settings()

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

        self.stats = GameStats(self)
        self.sb = Scoreboard(self)

        self.ship = Ship(self)
```

```
        self.bullets = pygame.sprite.Group()
        self.aliens = pygame.sprite.Group()

        self._create_fleet()

    def run_game(self):
        """Start the main loop for the game."""
        while True:
            self._check_events()

            if self.stats.game_active:
                self.ship.update()
                self._update_bullets()
                self._update_aliens()

            self._update_screen()

    def _check_events(self):
        """Respond to keypresses and mouse events."""
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                self._check_keydown_events(event)
            elif event.type == pygame.KEYUP:
                self._check_keyup_events(event)

    def _check_keydown_events(self, event):
        """Respond to keypresses."""
        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_SPACE:
            self._fire_bullet()
        elif event.key == pygame.K_q:
            sys.exit()

    def _check_keyup_events(self, event):
        """Respond to key releases."""
        if event.key == pygame.K_RIGHT:
            self.ship.moving_right = False
        elif event.key == pygame.K_LEFT:
            self.ship.moving_left = False
```

```
def _fire_bullet(self):
    """Create a new bullet and add it to the bullets group."""
    if len(self.bullets) < self.settings.bullets_allowed:
        new_bullet = Bullet(self)
        self.bullets.add(new_bullet)

def _update_bullets(self):
    """Update bullet positions and delete old bullets."""
    self.bullets.update()

    for bullet in self.bullets.copy():
        if bullet.rect.bottom <= 0:
            self.bullets.remove(bullet)

    self._check_bullet_alien_collisions()

def _check_bullet_alien_collisions(self):
    """Respond to bullet-alien collisions."""
    collisions = pygame.sprite.groupcollide(
        self.bullets, self.aliens, True, True)

    if collisions:
        for aliens in collisions.values():
            self.stats.score += self.settings.alien_points * len(aliens)
        self.sb.prep_score()

    if not self.aliens:
        self.bullets.empty()
        self._create_fleet()

def _update_aliens(self):
    """Update aliens' positions."""
    self._check_fleet_edges()
    self.aliens.update()

    if pygame.sprite.spritecollideany(self.ship, self.aliens):
        self._ship_hit()

    self._check_aliens_bottom()

def _ship_hit(self):
    """Respond to the ship being hit by an alien."""
    if self.stats.ships_left > 0:
```

```
        self.stats.ships_left -= 1
        self.sb.prep_ships()

        self.aliens.empty()
        self.bullets.empty()

        self._create_fleet()
        self.ship.center_ship()

        pygame.time.delay(500)
    else:
        self.stats.game_active = False

def _check_aliens_bottom(self):
    """Check if any aliens have reached the bottom."""
    screen_rect = self.screen.get_rect()
    for alien in self.aliens.sprites():
        if alien.rect.bottom >= screen_rect.bottom:
            self._ship_hit()
            break

def _create_fleet(self):
    """Create a fleet of aliens."""
    alien = Alien(self)
    alien_width, alien_height = alien.rect.size
    available_space_x = self.settings.screen_width - (2 * alien_width)
    number_aliens_x = available_space_x // (2 * alien_width)

    ship_height = self.ship.rect.height
    available_space_y = (self.settings.screen_height -
                         (3 * alien_height) - ship_height)
    number_rows = available_space_y // (2 * alien_height)

    for row_number in range(number_rows):
        for alien_number in range(number_aliens_x):
            self._create_alien(alien_number, row_number)

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

```
        self.aliens.add(alien)

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

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

    def _update_screen(self):
        """Update images on the screen and flip to the new screen."""
        self.screen.fill(self.settings.bg_color)
        self.ship.blitme()
        for bullet in self.bullets.sprites():
            bullet.draw_bullet()
        self.aliens.draw(self.screen)
        self.sb.show_score()

        pygame.display.flip()

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

ALIEN

```
import pygame
from pygame.sprite import Sprite

class Alien(Sprite):
    """A class to represent a single alien in the fleet."""

    def __init__(self, ai_game):
        """Initialize the alien and set its starting position."""
```

```
        super().__init__()
        self.screen = ai_game.screen
        self.settings = ai_game.settings

        # Load the alien image and set its rect attribute
        self.image = pygame.image.load('images/alien.bmp')
        self.rect = self.image.get_rect()

        # Start each new alien near the top left of the screen
        self.rect.x = self.rect.width
        self.rect.y = self.rect.height

        # Store the alien's exact horizontal position
        self.x = float(self.rect.x)

    def check_edges(self):
        """Return True if alien is at edge of screen."""
        screen_rect = self.screen.get_rect()
        if self.rect.right >= screen_rect.right or self.rect.left <= 0:
            return True

    def update(self):
        """Move the alien right or left."""
        self.x += (self.settings.alien_speed * self.settings.fleet_direction)
        self.rect.x = self.x
```

BULLET

```
import pygame
from pygame.sprite import Sprite

class Bullet(Sprite):
    """A class to manage bullets fired from the ship."""

    def __init__(self, ai_game):
        """Create a bullet object at the ship's current position."""
        super().__init__()
        self.screen = ai_game.screen
        self.settings = ai_game.settings
        self.color = self.settings.bullet_color
```

```
        # Create a bullet rect at (0, 0) and then set correct position
        self.rect = pygame.Rect(0, 0, self.settings.bullet_width,
                                self.settings.bullet_height)
        self.rect.midtop = ai_game.ship.rect.midtop

        # Store the bullet's position as a decimal value
        self.y = float(self.rect.y)

    def update(self):
        """Move the bullet up the screen."""
        # Update the decimal position of the bullet
        self.y -= self.settings.bullet_speed
        self.rect.y = self.y

    def draw_bullet(self):
        """Draw the bullet to the screen."""
        pygame.draw.rect(self.screen, self.color, self.rect)
```

GAME STATS

```
# Game Stats Tracker

class GameStats:
    """Track statistics for Alien Invasion."""

    def __init__(self, ai_game):
        """Initialize statistics."""
        self.settings = ai_game.settings
        self.reset_stats()

        # Start Alien Invasion in an active state
        self.game_active = True

        # High score should never be reset
        self.high_score = 0

    def reset_stats(self):
        """Initialize statistics that can change during the game."""
        self.ships_left = 3
```

```
        self.score = 0
```

SCOREBOARD

```
# Scoreboard to show scores on screen
import pygame.font

class Scoreboard:
    """A class to report scoring information."""

    def __init__(self, ai_game):
        """Initialize scorekeeping attributes."""
        self.ai_game = ai_game
        self.screen = ai_game.screen
        self.screen_rect = self.screen.get_rect()
        self.settings = ai_game.settings
        self.stats = ai_game.stats

        # Font settings for scoring
        self.text_color = (30, 30, 30)
        self.font = pygame.font.SysFont(None, 48)

        # Prepare the initial score image
        self.prep_score()

    def prep_score(self):
        """Turn the score into a rendered image."""
        rounded_score = round(self.stats.score, -1)
        score_str = f"{rounded_score:,}"
        self.score_image = self.font.render(score_str, True,
                        self.text_color, self.settings.bg_color)

        # Display the score at the top right of the screen
        self.score_rect = self.score_image.get_rect()
        self.score_rect.right = self.screen_rect.right - 20
        self.score_rect.top = 20

    def prep_ships(self):
        """Placeholder if you want to show ships left later."""
        pass
```

```
    def show_score(self):
        """Draw the score to the screen."""
        self.screen.blit(self.score_image, self.score_rect)
```

SETTINGS

```
# Settings for Alien Invasion Game

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)

        # Ship settings
        self.ship_speed = 1.5

        # Bullet settings
        self.bullet_speed = 1.0
        self.bullet_width = 3
        self.bullet_height = 15
        self.bullet_color = (60, 60, 60)
        self.bullets_allowed = 5

        # Alien settings
        self.alien_speed = 1.0
        self.fleet_drop_speed = 10
        self.fleet_direction = 1  # 1 = right, -1 = left

        # Scoring
        self.alien_points = 50
```

SHIP

```python
import pygame

class Ship:
    """A class to manage the ship."""

    def __init__(self, ai_game):
        """Initialize the ship and set its starting position."""
        self.screen = ai_game.screen
        self.settings = ai_game.settings
        self.screen_rect = self.screen.get_rect()

        # Load the ship image
        self.image = pygame.image.load('images/ship.bmp')
        self.rect = self.image.get_rect()

        # Start each new ship at the bottom center of the screen
        self.rect.midbottom = self.screen_rect.midbottom

        # Store a decimal value for the ship's horizontal position
        self.x = float(self.rect.x)

        # Movement flags
        self.moving_right = False
        self.moving_left = False

    def update(self):
        """Update the ship's position based on movement flags."""
        if self.moving_right and self.rect.right < self.screen_rect.right:
            self.x += self.settings.ship_speed
        if self.moving_left and self.rect.left > 0:
            self.x -= self.settings.ship_speed

        # Update rect object from self.x
        self.rect.x = self.x

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

    def center_ship(self):
        """Center the ship on the screen."""
        self.rect.midbottom = self.screen_rect.midbottom
        self.x = float(self.rect.x)
```