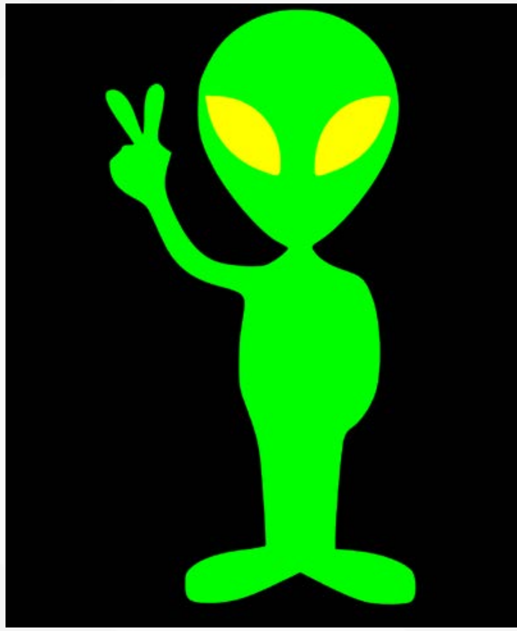
Here come the aliens!

Team members:
朱芙蓉、刘颖、
周韩韵、Wendy

2018



目 录

CONTENT

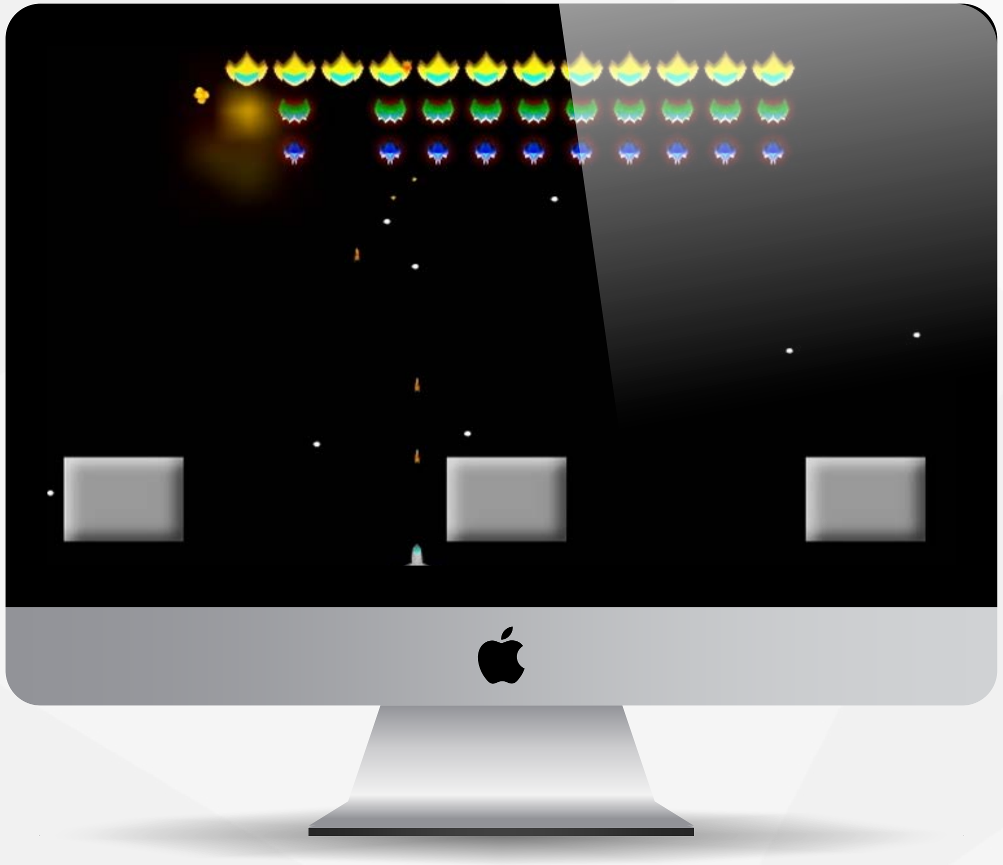
PART 1

INTRODUCTION

THE OPPORTUNITY

ONLINE VIDEO GAME INDUSTRY

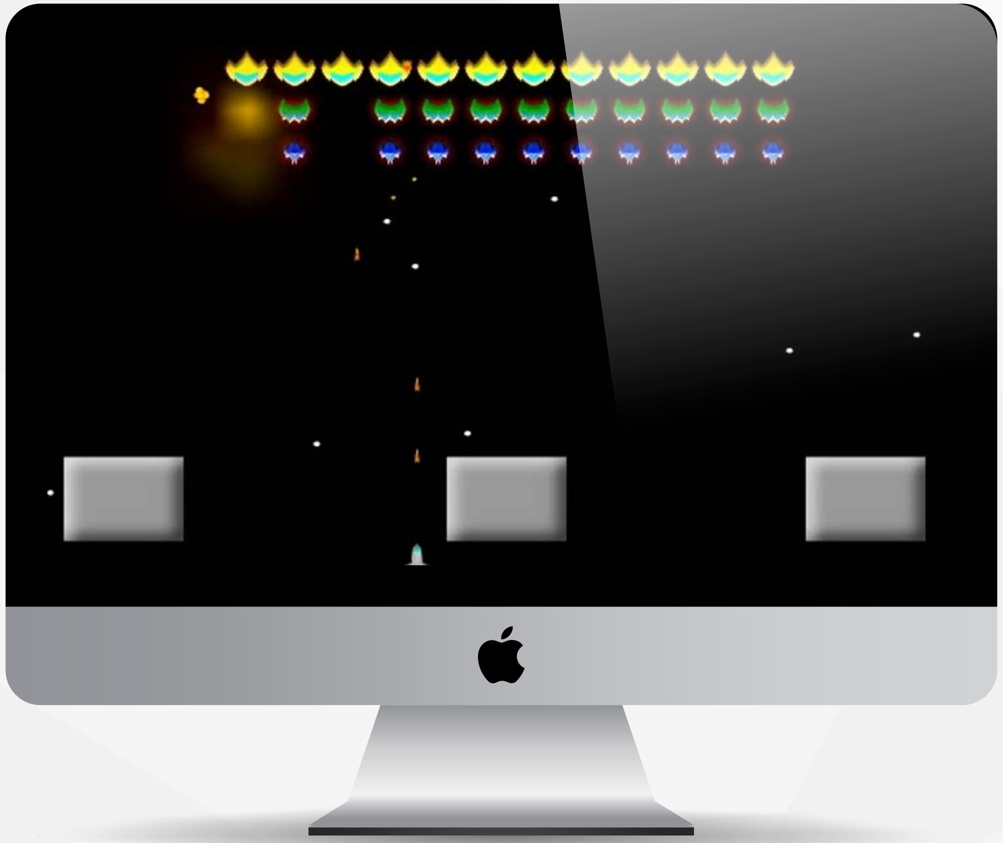
- Increasing amount of online gamers
- Data volume of global online gaming expected to grow to 568PB in 2020
- Worldwide market projected to reach 2.2T USD by 2021



OUR IDEA

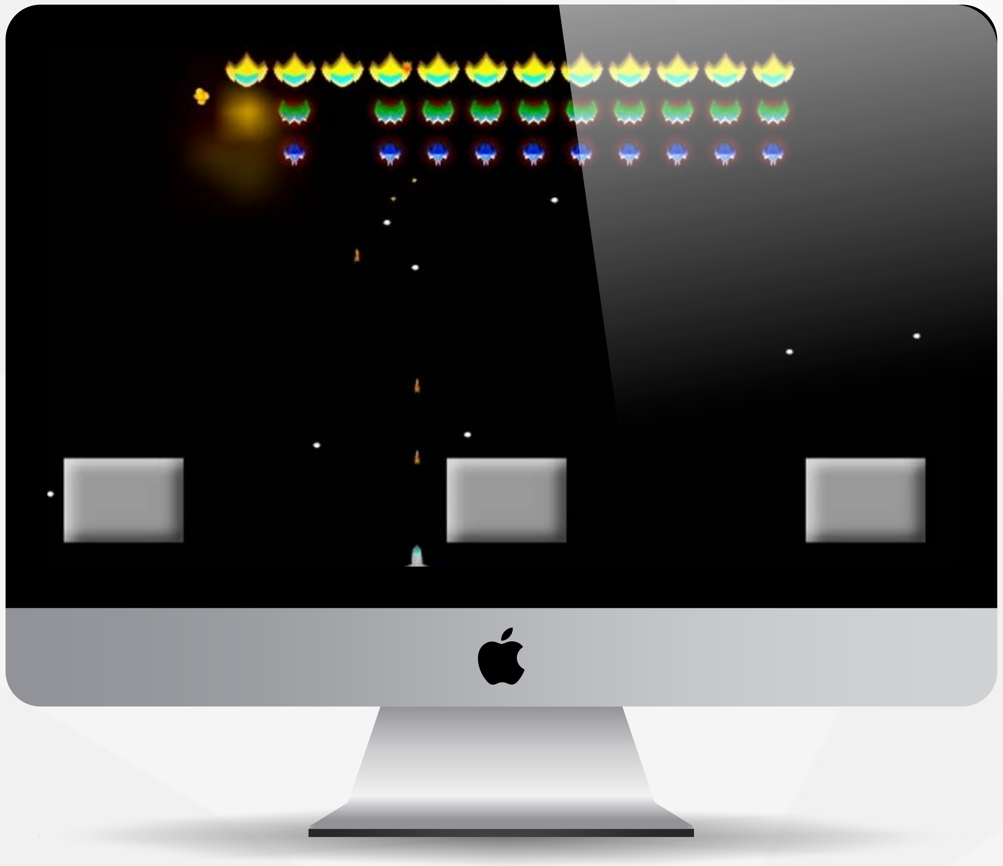
“ALIEN INVASION” GAME

- Simple and easy-to-learn 1 player game
- Fun and engaging for players
- Good for children and adults of all ages
- Use Pygame module for media and graphics



HOW TO PLAY

- Player controls a spacecraft and shoots Aliens
- Aliens slowly move down the screen during the game
- Once all Aliens are shot, they reappear on the screen and move down at a faster rate
- Player loses a life when:
    - Alien touches Player
    - Alien touches bottom of the screen
- Game is over when:
    - Player loses 3 lives



HOW TO PLAY

IMPORTANT KEYS

- ARROW KEYS: Move spacecraft LEFT or RIGHT
- SPACEBAR: Allows Player to shoot Aliens

01 INTRODUCTION

1. RESEARCH

Research Python code, modules, and interfaces to use

2. PREPARATION

Download PyGame

Install updated Python version and pip

3. DRAW SHAPES & OBJECTS

Design game elements such as spacecraft and aliens

Set background color and other images

Manage other visual elements of the game interface

4. GAME FUNCTIONS

Add spacecraft, aliens to game interface

Have game respond to user input, controls, and events

Establish rules of the game within the code

Enable increased difficulty between levels

5. FINISHING UP

Add Play button and Scoring capabilities

Fine-tune visuals

Test and play!

PART 02 Explanation of the code

Install Pygame

(open anaconda prompt)
code:pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pygame

02 Explanation of the code

A brief introduction of Pygame

pygame.cdrom	访问光驱
pygame.cursors	加载光标
pygame.display	访问显示设备
pygame.draw	绘制形状、线和点
pygame.event	管理事件
pygame.font	使用字体
pygame.image	加载和存储图片
pygame.joystick	使用游戏手柄或者 类似的东西
pygame.key	读取键盘按键
pygame.mixer	声音
pygame.mouse	鼠标
pygame.movie	播放视频
pygame.music	播放音频
pygame.overlay	访问高级视频叠加
pygame	it is what we are doing
pygame.rect	管理矩形区域
pygame.sndarray	操作声音数据
pygame.sprite	操作移动图像
pygame.surface	管理图像和屏幕
pygame.surfarray	管理点阵图像数据
pygame.time	管理时间和帧信息
pygame.transform	缩放和移动图像

02 Explanation of the code

Create the Pygame window and respond to user's input

```
import sys
import pygame
def run_game():
    #initialize game and create a dispaly object
    pygame.init()
    screen = pygame.display.set_mode((1200,800))    ←  the size of the window
    pygame.display.set_caption("Alien Invasion")
    # set backgroud color
    bg_color = (230,230,230)    ←  red, green and blue(the maximum of each color is 255)

    # game loop
    while True:
        # supervise keyboard and mouse item
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()    ←  to exit the game
        # fill color
        screen.fill(bg_color)
        # visualiaze the window
        pygame.display.flip()    ←
run_game()
```

every time it executes a while loop, it will move away the old window and create a new window.It means when we move the object, it will continually show the new location of the object and hide the old one.

Create settings (to store all settings in one place for future modification)

```
class Settings(object):
    def __init__(self):
        # initialize setting of game

        # screen setting
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230,230,230)
```

They are easy to understand!

Then let's import the settings to alien_invasion.py

To be brief, we add a line of code 'from settings import Settings' and replace the exact number with the descriptions in settings.py .

(e.g. ai_settings = Settings()

```
        screen = pygame.display.set_mode((ai_settings.screen_width,ai_settings.screen_height))
```

02 Explanation of the code

Create ship.py

```
import pygame
class Ship():

    def __init__(self,screen):
        #initialize spaceship and its location
        self.screen = screen

        # load bmp image and get rectangle
        self.image = pygame.image.load('image/ship.bmp')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()

        #put spaceship on the bottom of window
        self.rect.centerx = self.screen_rect.centerx
        self.rect.bottom = self.screen_rect.bottom

    def blitme(self):
        #build the spaceship at the specific location
        self.screen.blit(self.image,self.rect)
```

'rect' makes us able to treat the screen just like a rectangle. But we have to know that the original point is in the top left corner. Whether we move to the right or the under, the number both becomes larger.

Refactoring: module game_functions

To simplify!

Move the code to a function. Like this:

```
import sys
import pygame

def check_events():
    #respond to  keyboard and
mouse item
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

In this module, Sys and pyGame are used to import the event check loop.

Drive the ship.

① Press the right (left) button to control the ship moving to the right (left)

```
def check_events(ship):
    #respond to  keyboard and mouse item
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                #move right
                ship.rect.centerx +=1
```

Drive the ship.

② Adjust the speed of the ship.

```
class Settings(object):
    """docstring for Settings"""
    def __init__(self):
        # initialize setting of game

        # screen setting
        self.screen_width = 1200
        self.screen_height = 800
        self.bg_color = (230,230,230)
        self.ship_speed_factor = 1.5
```

Reconstruction

Remember the check_events() function?

Here we'll focus on refactoring the check_events() function, breaking some of the code into two parts, one dealing with KEYDOWN events and one dealing with KEYUP events.

02 Explanation of the code

Create bullets

```
# Create bullet rect at (0, 0), then set correct position.
        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
```

Here, we first set the bullet at the (0,0). The we move it to the position of the ship.

```
        self.y = float(self.rect.y)
```

We make the y decimals so that we can adjust the speed of the ship precisely.

02 Explanation of the code

Fire! Fire! Fire!

We modify the check_keydown_events() function to listen for events when a player presses a space key.We also modify the update_screen() function to ensure that each bullet is redrawn every time the screen is updated.

02 Explanation of the code

Delete the missing bullet

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

Create the first alien

Here’s the same way as creating a ship.

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

    def __init__(self, ai_settings, screen):
        """Initialize the alien, and set its starting position."""
        super().__init__()
        self.screen = screen
        self.ai_settings = ai_settings

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

Then we can create a group of aliens.

Limit the number of bullets

We have a rule that only three bullets can exist on the screen at the same time. We only need to check whether the number of bullets that remain on the screen is less than three before each bullet is created.

```
def fire_bullet(ai_settings, screen, ship, bullets):
    """Fire a bullet, if limit not reached yet."""
    # Create a new bullet, add to bullets group.
    if len(bullets) < ai_settings.bullets_allowed:
        new_bullet = Bullet(ai_settings, screen, ship)
        bullets.add(new_bullet)
```

Move and shoot the aliens !

We set the alien's speed in the Settings class and then use the update method in the Alien class to implement the move.

➡ **Move**

 Detect the collision immediately after we have updated the position of the bullet.

⬅ **Shoot**

When to end the game

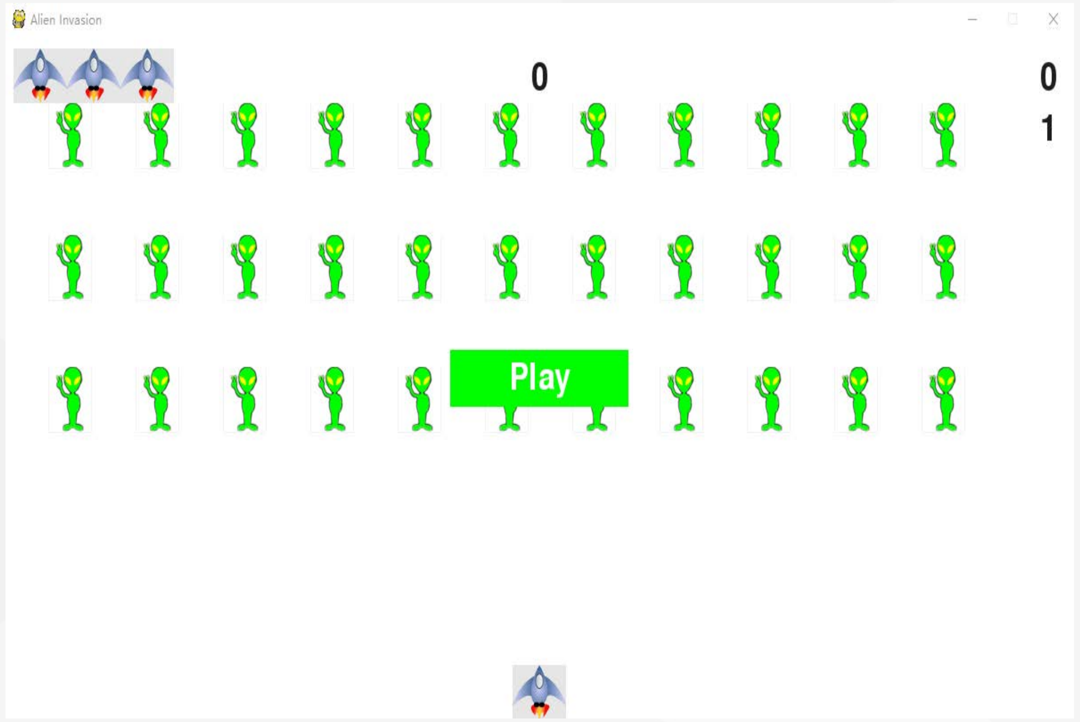
IF The ship was completely destroyed

OR Aliens reach the bottom of the screen

Last step !

Add a Play button to the game to start the game as needed and restart the game after the game is over.

Implement a scoring system that will speed up the tempo as the player level increases.



Let's run the game!

THANKS FOR WATCHING!