```
|0   |1   |2   |3   |4   |5   |6   |7   |8
#!/usr/bin/python
# Alien Invasion
# Code Angel

import sys
import os
import pygame
from pygame.locals import *
import random

# Define the colours
LIGHT_YELLOW = (255, 255, 204)
WHITE = (255, 255, 255)

# Define constants
SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480
SCOREBOARD_MARGIN = 4

MISSILE_PLATFORM = 31
MISSILE_SPEED = 10
GAME_MISSILES = 20

UFO_UPPER_Y = 20
UFO_LOWER_Y = 240
UFO_HIT_TIME = 20
UFO_OFF_TIME = 60
UFO_SCORE = 50

RANDOM_VERTICAL_CHANGE = 20
RANDOM_HORIZONTAL_CHANGE = 100
UFO_DIRECTIONS = ['left', 'right', 'up', 'down']

RANDOM_RAY = 200
RANDOM_RAY_TIME_MAX = 120
RANDOM_RAY_TIME_MIN = 30

BASE_SPEED = 6

# Setup
os.environ['SDL_VIDEO_CENTERED'] = '1'
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
pygame.mixer.pre_init(44100, -16, 2, 512)
pygame.mixer.init()
pygame.init()
game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption('Alien Invasion')
pygame.key.set_repeat(10, 20)
clock = pygame.time.Clock()
font = pygame.font.SysFont('Helvetica', 16)

# Load images
background_image = pygame.image.load('background.png').convert()
base_image = pygame.image.load('base.png').convert_alpha()
missile_image = pygame.image.load('missile.png').convert_alpha()
missile_fired_image = pygame.image.load('missile_fired.png').convert_alpha()

ufo_1_image = pygame.image.load('ufo 1.png').convert_alpha()
ufo_2_image = pygame.image.load('ufo 2.png').convert_alpha()
ufo_1_exploded_image = pygame.image.load('ufo 1 exploded.png').convert_alpha()
ufo_2_exploded_image = pygame.image.load('ufo 2 exploded.png').convert_alpha()
ufo_ray_image_1 = pygame.image.load('ufo ray 1.png').convert_alpha()
ufo_ray_image_2 = pygame.image.load('ufo ray 2.png').convert_alpha()

# Load sounds
spaceship_hit_sound = pygame.mixer.Sound('spaceship_hit.ogg')
launch_sound = pygame.mixer.Sound('launch.ogg')


def main():

    # Initialise variables
    base_x = 300
    base_y = 430
    base_width = base_image.get_rect().width

    ufo_width = ufo_1_image.get_rect().width
    ufo_height = ufo_1_image.get_rect().height

    ray_width = ufo_ray_image_1.get_rect().width

    ufo_1_x = SCREEN_WIDTH - ufo_width
    ufo_1_y = random.randint(UFO_UPPER_Y, UFO_LOWER_Y)
    |0   |1   |2   |3   |4   |5   |6   |7   |8

    # UFO 1 dicitionary
    ufo_1 = {'x_loc': ufo_1_x, 'y_loc': ufo_1_y, 'direction': 'left', 'hit': False, 'hit_time': 0, 'off_time': 0,
             'ray_time': 0, 'speed': 5}

    ufo_2_y = random.randint(UFO_UPPER_Y, UFO_LOWER_Y)

    # UFO 2 dictionary
    ufo_2 = {'x_loc': 0, 'y_loc': ufo_2_y, 'direction': 'right', 'hit': False, 'hit_time': 0, 'off_time': 0,
             'ray_time': 0, 'speed': 3}

    missile_x = 0
    missile_y = 0
    missile_firing = False

    missile_width = missile_image.get_rect().width
    missile_height = missile_image.get_rect().height

    score = 0
    hi_score = 0
    missiles = GAME_MISSILES
    game_over = False

    # Main game loop
    while True:

        for event in pygame.event.get():
            key_pressed = pygame.key.get_pressed()

            # Left key pressed, move base left
            if key_pressed[pygame.K_LEFT]:
                base_x -= BASE_SPEED
                if base_x < 0:
                    base_x = 0

            # Right key pressed, move base right
            elif key_pressed[pygame.K_RIGHT]:
                base_x += BASE_SPEED
                if base_x > SCREEN_WIDTH - base_width:
                    base_x = SCREEN_WIDTH - base_width

    |0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
            # Space pressed, fire missile
            elif key_pressed[pygame.K_SPACE] and missile_firing is False and game_over is False:
                missile_firing = True
                missile_x = base_x + MISSILE_PLATFORM
                missile_y = base_y - missile_height
                missiles -= 1
                launch_sound.play()
                if missiles == 0:
                    game_over = True

            # Return pressed at end of game, start new game
            elif key_pressed[pygame.K_RETURN] and game_over is True:
                game_over = False
                score = 0
                missiles = GAME_MISSILES

            # User quits
            if event.type == QUIT:
                pygame.quit()
                sys.exit()

        # Update missile location
        if missile_firing is True:
            missile_y -= MISSILE_SPEED
            if missile_y < 0:
                missile_firing = False

        # Update UFO locations
        move_ufo(ufo_1, ufo_width)
        move_ufo(ufo_2, ufo_width)

        # Update UFO rays
        update_ray(ufo_1)
        update_ray(ufo_2)

        # Check if missile hits a UFO
        missile_rect = pygame.Rect(missile_x, missile_y, missile_width, missile_height)

        if ufo_1.get('hit') is False and missile_firing is True:
            ufo_hit = check_ufo_hit(ufo_1, missile_rect, ufo_width, ufo_height)
            if ufo_hit == 'missile destroyed':
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
                missile_firing = False
                pygame.mixer.stop()

            elif ufo_hit == 'direct hit':
                missile_firing = False
                score += UFO_SCORE * 2
                ufo_1['hit_time'] = UFO_HIT_TIME
                ufo_1['hit'] = True

                pygame.mixer.stop()
                spaceship_hit_sound.play()

        if ufo_2.get('hit') is False and missile_firing is True:
            ufo_hit = check_ufo_hit(ufo_2, missile_rect, ufo_width, ufo_height)
            if ufo_hit == 'missile destroyed':
                missile_firing = False
                pygame.mixer.stop()

            elif ufo_hit == 'direct hit':
                missile_firing = False
                score += UFO_SCORE
                ufo_2['hit_time'] = UFO_HIT_TIME
                ufo_2['hit'] = True

                pygame.mixer.stop()
                spaceship_hit_sound.play()

        # Update hit UFOs
        update_hit_ufo(ufo_1, SCREEN_WIDTH - ufo_width, 'left')
        update_hit_ufo(ufo_2, 0, 'right')

        # Draw background
        game_screen.blit(background_image, [0, 0])

        # Draw base
        game_screen.blit(base_image, [base_x, base_y])

        # Draw missile
        if missile_firing is True:
            game_screen.blit(missile_fired_image, [missile_x, missile_y])
        else:
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
            game_screen.blit(missile_image, [base_x + MISSILE_PLATFORM, base_y - missile_height])

        # Draw UFOs
        if ufo_1.get('hit_time') > 0:
            game_screen.blit(ufo_1_exploded_image, [ufo_1.get('x_loc'), ufo_1.get('y_loc')])
        elif ufo_1.get('hit') is False:
            game_screen.blit(ufo_1_image, [ufo_1.get('x_loc'), ufo_1.get('y_loc')])

        if ufo_2.get('hit_time') > 0:
            game_screen.blit(ufo_2_exploded_image, [ufo_2.get('x_loc'), ufo_2.get('y_loc')])
        elif ufo_2.get('hit') is False:
            game_screen.blit(ufo_2_image, [ufo_2.get('x_loc'), ufo_2.get('y_loc')])

        # Draw UFO defence rays
        if ufo_1.get('ray_time') > 0:
            ray_x = ufo_1.get('x_loc') + (ufo_width - ray_width) / 2
            ray_y = ufo_1.get('y_loc') + ufo_height
            if ufo_1.get('ray_time') % 4 == 0 or ufo_1.get('ray_time') % 5 == 0:
                game_screen.blit(ufo_ray_image_2, [ray_x, ray_y])
            else:
                game_screen.blit(ufo_ray_image_1, [ray_x, ray_y])

        if ufo_2.get('ray_time') > 0:
            ray_x = ufo_2.get('x_loc') + (ufo_width - ray_width) / 2
            ray_y = ufo_2.get('y_loc') + ufo_height
            if ufo_2.get('ray_time') % 4 == 0 or ufo_2.get('ray_time') % 5 == 0:
                game_screen.blit(ufo_ray_image_2, [ray_x, ray_y])
            else:
                game_screen.blit(ufo_ray_image_1, [ray_x, ray_y])

        # Game over
        if game_over is True and missile_firing is False:
            if score > hi_score:
                hi_score = score

            display_game_over()

        # Display score board
        score_text = 'Score: ' + str(score)
        display_scoreboard_data(score_text, 'left')

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```python
        missile_text = 'Missiles: ' + str(missiles)
        display_scoreboard_data(missile_text, 'centre')

        hi_score_text = 'Hi: ' + str(hi_score)
        display_scoreboard_data(hi_score_text, 'right')

        pygame.display.update()
        clock.tick(30)


# Move the UFO
def move_ufo(ufo, ufo_width):
    if ufo.get('hit') is False:
        if ufo.get('direction') == 'left':
            ufo['x_loc'] -= ufo.get('speed')
        elif ufo.get('direction') == 'right':
            ufo['x_loc'] += ufo.get('speed')
        elif ufo.get('direction') == 'up':
            ufo['y_loc'] -= ufo.get('speed')
        elif ufo.get('direction') == 'down':
            ufo['y_loc'] += ufo.get('speed')

        # If the UFO goes off the screen left, reset x coordinate and change direction
        if ufo.get('x_loc') < 0:
            ufo['x_loc'] = 0
            ufo['direction'] = 'right'

        # If the UFO goes off the screen right, reset x coordinate and change direction
        elif ufo.get('x_loc') > SCREEN_WIDTH - ufo_width:
            ufo['x_loc'] = SCREEN_WIDTH - ufo_width
            ufo['direction'] = 'left'

        # If the UFO goes too high, reset y coordinate and change direction
        elif ufo.get('y_loc') < UFO_UPPER_Y:
            ufo['y_loc'] = UFO_UPPER_Y
            ufo['direction'] = 'down'

        # If the UFO goes too low, reset y coordinate and change direction
        elif ufo.get('y_loc') > UFO_LOWER_Y:
            ufo['y_loc'] = UFO_LOWER_Y
            ufo['direction'] = 'up'
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

        # If none of the above, then random chance of changing direction
        else:
            if ufo.get('direction') == 'up' or ufo.get('direction') == 'down':
                ufo_direction_chance = random.randint(0, RANDOM_VERTICAL_CHANGE)
            else:
                ufo_direction_chance = random.randint(0, RANDOM_HORIZONTAL_CHANGE)

            if ufo_direction_chance == 1:
                ufo['direction'] = random.choice(UFO_DIRECTIONS)


# Update the status of the UFO ray
def update_ray(ufo):

    # If there is not already a ray, then random chance of there being a ray
    if ufo.get('ray_time') == 0 and ufo.get('hit') is False:
        random_ray = random.randint(0, RANDOM_RAY)
        if random_ray == 1:
            ufo['ray_time'] = random.randint(RANDOM_RAY_TIME_MIN, RANDOM_RAY_TIME_MAX)

    # If there is a ray, decrease its time
    elif ufo.get('ray_time') > 0:
        ufo['ray_time'] -= 1


# Has the UFO been hit my the missile
def check_ufo_hit(ufo, missile_rect, ufo_width, ufo_height):

    ufo_rect = pygame.Rect(ufo.get('x_loc'), ufo.get('y_loc'), ufo_width, ufo_height)

    if missile_rect.colliderect(ufo_rect):

        # If the missile collides with the UFO and there is no defence ray, direct hit
        if ufo.get('ray_time') == 0:
            ufo_hit = 'direct hit'

        # If the missile collides with the UFO and there is a defence ray, missile is destroyed
        else:
            ufo_hit = 'missile destroyed'

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0   |1   |2   |3   |4   |5   |6   |7   |8
    # If the missile has not collided with the UFO, no hit
    else:
        ufo_hit = 'no hit'

    return ufo_hit


# Update status of UFO if it has been hit
def update_hit_ufo(ufo, new_x_loc, new_direction):

    # UFO has been hit, redice the hit time
    if ufo.get('hit_time') > 0:
        ufo['hit_time'] -= 1

        # When hit time reaches zero, UFO should go off screen
        if ufo.get('hit_time') == 0:
            ufo['off_time'] = UFO_OFF_TIME

    # UFO is off screen, reduce the off screen time
    elif ufo.get('off_time') > 0:
        ufo['off_time'] -= 1

        # When off screen time reaches 0, set new UFO location and direction
        if ufo.get('off_time') == 0:
            ufo['y_loc'] = random.randint(UFO_UPPER_Y, UFO_LOWER_Y)
            ufo['x_loc'] = new_x_loc
            ufo['direction'] = new_direction
            ufo['hit'] = False


# Display the scoreboard data
def display_scoreboard_data(scoreboard_text, alignment):
    display_text = font.render(scoreboard_text, True, LIGHT_YELLOW)
    text_rect = display_text.get_rect()

    text_loc = [0, 0]

    if alignment == 'left':
        text_loc = [SCOREBOARD_MARGIN, SCOREBOARD_MARGIN]

    elif alignment == 'right':
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        text_loc = [SCREEN_WIDTH - text_rect.width - SCOREBOARD_MARGIN, SCOREBOARD_MARGIN]

    elif alignment == 'centre':
        text_loc = [(SCREEN_WIDTH - text_rect.width) / 2, SCOREBOARD_MARGIN]

    game_screen.blit(display_text, text_loc)


# Display the game over message
def display_game_over():
    text_line_1 = font.render('GAME OVER', True, WHITE)
    text_rect_1 = text_line_1.get_rect()
    text_line_1_loc = [(SCREEN_WIDTH - text_rect_1.width) / 2, (SCREEN_HEIGHT / 2) - 16]

    text_line_2 = font.render('Hit RETURN for new game', True, WHITE)
    text_rect_2 = text_line_2.get_rect()
    text_line_2_loc = [(SCREEN_WIDTH - text_rect_2.width) / 2, (SCREEN_HEIGHT / 2) + 16]

    game_screen.blit(text_line_1, text_line_1_loc)
    game_screen.blit(text_line_2, text_line_2_loc)


if __name__ == '__main__':
    main()

|0  |1  |2  |3  |4  |5  |6  |7  |8
```