```python
#!/usr/bin/env python
# Space Invaders Created by Lee Robinson
# Modified for learning purposes by Daniel Sedó

from pygame import *
import sys
from os.path import abspath, dirname
from random import choice

BASE_PATH = abspath(dirname(__file__))
FONT_PATH = BASE_PATH + '/dat/'   #modificada ruta
IMAGE_PATH = BASE_PATH + '/img/'   #modificada ruta
SOUND_PATH = BASE_PATH + '/dat/'   #modificada ruta

WHITE = (255, 255, 255)
GREEN = (78, 255, 87)
YELLOW = (241, 255, 0)
BLUE = (80, 255, 239)
PURPLE = (203, 0, 255)
RED = (237, 28, 36)
BLACK = (0, 0, 0)

SCREEN = display.set_mode((800, 600))
FONT = FONT_PATH + 'space_invaders.ttf'
IMG_NAMES = ['ship', 'mystery', 'enemy1_1', 'enemy1_2', 'enemy2_1', 'enemy2_2', 'enemy3_1',
'enemy3_2', 'explosionblue', 'explosiongreen', 'explosionpurple', 'laser', 'enemylaser']
IMAGES = {name: image.load(IMAGE_PATH + '{}.png'.format(name)).convert_alpha() for name in
IMG_NAMES}

ENEMY_DEFAULT_POSITION = 65  # Initial value for a new game
ENEMY_MOVE_DOWN = 35

class Ship(sprite.Sprite):
    def __init__(self):
        sprite.Sprite.__init__(self)
        self.image = IMAGES['ship']
        self.rect = self.image.get_rect(topleft=(375, 540))   #modificada ubicación ppal nave
        self.speed = 4   #modificada velocidad ppal nave
    def update(self, keys, *args):
        if keys[K_LEFT] and self.rect.x > 10:    #limites pantalla izq
            self.rect.x -= self.speed
        if keys[K_RIGHT] and self.rect.x < 750:   #limites pantalla der
            self.rect.x += self.speed
        game.screen.blit(self.image, self.rect)

class Bullet(sprite.Sprite):
    def __init__(self, xpos, ypos, direction, speed, filename, side):
        sprite.Sprite.__init__(self)
        self.image = IMAGES[filename]
        self.rect = self.image.get_rect(topleft=(xpos, ypos))
        self.speed = speed
        self.direction = direction
        self.side = side
        self.filename = filename
    def update(self, keys, *args):
        game.screen.blit(self.image, self.rect)
        self.rect.y += self.speed * self.direction
        if self.rect.y < 15 or self.rect.y > 600:
            self.kill()

class Enemy(sprite.Sprite):
    def __init__(self, row, column):
        sprite.Sprite.__init__(self)
        self.row = row
        self.column = column
        self.images = []
        self.load_images()
        self.index = 0
        self.image = self.images[self.index]
        self.rect = self.image.get_rect()
    def toggle_image(self):
        self.index += 1
        if self.index >= len(self.images):
            self.index = 0
        self.image = self.images[self.index]
    def update(self, *args):
        game.screen.blit(self.image, self.rect)
    def load_images(self):
        images = {0: ['1_2', '1_1'], 1: ['2_2', '2_1'], 2: ['2_2', '2_1'], 3: ['3_1', '3_2'], 4: ['3_1', '3_2'], 5: ['3_1', '3_2'],6: ['2_2', '2_1']}
        img1, img2 = (IMAGES['enemy{}'.format(img_num)] for img_num in images[self.row])
        self.images.append(transform.scale(img1, (40, 35)))
        self.images.append(transform.scale(img2, (40, 35)))

class EnemiesGroup(sprite.Group):
    def __init__(self, columns, rows):
        sprite.Group.__init__(self)
        self.enemies = [[None] * columns for _ in range(rows)]
        self.columns = columns
        self.rows = rows
        self.leftAddMove = 0
        self.rightAddMove = 0
        self.moveTime = 600
        self.direction = 1
        self.rightMoves = 30
        self.leftMoves = 30
        self.moveNumber = 15
        self.timer = time.get_ticks()
        self.bottom = game.enemyPosition + ((rows - 1) * 45) + 35
        self._aliveColumns = list(range(columns))
        self._leftAliveColumn = 0
        self._rightAliveColumn = columns - 1
    def update(self, current_time):
        if current_time - self.timer > self.moveTime:
            if self.direction == 1:
                max_move = self.rightMoves + self.rightAddMove
            else:
                max_move = self.leftMoves + self.leftAddMove
            if self.moveNumber >= max_move:
                self.leftMoves = 30 + self.rightAddMove
                self.rightMoves = 30 + self.leftAddMove
                self.direction *= -1
                self.moveNumber = 0
                self.bottom = 0
                for enemy in self:
                    enemy.rect.y += ENEMY_MOVE_DOWN
                    enemy.toggle_image()
                    if self.bottom < enemy.rect.y + 35:
                        self.bottom = enemy.rect.y + 35
            else:
                velocity = 10 if self.direction == 1 else -10
                for enemy in self:
                    enemy.rect.x += velocity
                    enemy.toggle_image()
                self.moveNumber += 1
            self.timer += self.moveTime
```

```python
    def add_internal(self, *sprites):
        super(EnemiesGroup, self).add_internal(*sprites)
        for s in sprites:
            self.enemies[s.row][s.column] = s
    def remove_internal(self, *sprites):
        super(EnemiesGroup, self).remove_internal(*sprites)
        for s in sprites:
            self.kill(s)
        self.update_speed()
    def is_column_dead(self, column):
        return not any(self.enemies[row][column] for row in range(self.rows))
    def random_bottom(self):
        col = choice(self._aliveColumns)
        col_enemies = (self.enemies[row - 1][col]
                       for row in range(self.rows, 0, -1))
        return next((en for en in col_enemies if en is not None), None)
    def update_speed(self):
        if len(self) == 1:
            self.moveTime = 200
        elif len(self) <= 10:
            self.moveTime = 400
    def kill(self, enemy):
        self.enemies[enemy.row][enemy.column] = None
        is_column_dead = self.is_column_dead(enemy.column)
        if is_column_dead:
            self._aliveColumns.remove(enemy.column)
        if enemy.column == self._rightAliveColumn:
            while self._rightAliveColumn > 0 and is_column_dead:
                self._rightAliveColumn -= 1
                self.rightAddMove += 5
                is_column_dead = self.is_column_dead(self._rightAliveColumn)
        elif enemy.column == self._leftAliveColumn:
            while self._leftAliveColumn < self.columns and is_column_dead:
                self._leftAliveColumn += 1
                self.leftAddMove += 5
                is_column_dead = self.is_column_dead(self._leftAliveColumn)

class Mystery(sprite.Sprite):
    def __init__(self):
        sprite.Sprite.__init__(self)
        self.image = IMAGES['mystery']
        self.image = transform.scale(self.image, (75, 35))
        self.rect = self.image.get_rect(topleft=(-80, 45))
        self.row = 5
        self.moveTime = 25000
        self.direction = 1
        self.timer = time.get_ticks()
        self.mysteryEntered = mixer.Sound(SOUND_PATH + 'mysteryentered.wav')
        self.mysteryEntered.set_volume(0.3)
        self.playSound = True
    def update(self, keys, currentTime, *args):
        resetTimer = False
        passed = currentTime - self.timer
        if passed > self.moveTime:
            if (self.rect.x < 0 or self.rect.x > 800) and self.playSound:
                self.mysteryEntered.play()
                self.playSound = False
            if self.rect.x < 840 and self.direction == 1:
                self.mysteryEntered.fadeout(4000)
                self.rect.x += 2
                game.screen.blit(self.image, self.rect)
            if self.rect.x > -100 and self.direction == -1:
                self.mysteryEntered.fadeout(4000)
                self.rect.x -= 2
                game.screen.blit(self.image, self.rect)
        if self.rect.x > 830:
            self.playSound = True
            self.direction = -1
            resetTimer = True
        if self.rect.x < -90:
            self.playSound = True
            self.direction = 1
            resetTimer = True
        if passed > self.moveTime and resetTimer:
            self.timer = currentTime

class EnemyExplosion(sprite.Sprite):
    def __init__(self, enemy, *groups):
        super(EnemyExplosion, self).__init__(*groups)
        self.image = transform.scale(self.get_image(enemy.row), (40, 35))
        self.image2 = transform.scale(self.get_image(enemy.row), (50, 45))
        self.rect = self.image.get_rect(topleft=(enemy.rect.x, enemy.rect.y))
        self.timer = time.get_ticks()
    @staticmethod
    def get_image(row):
        img_colors = ['purple', 'blue', 'blue', 'green', 'green', 'green', 'blue'] #modificada ampliación enemigos
        return IMAGES['explosion{}'.format(img_colors[row])]
    def update(self, current_time, *args):
        passed = current_time - self.timer
        if passed <= 100:
            game.screen.blit(self.image, self.rect)
        elif passed <= 200:
            game.screen.blit(self.image2, (self.rect.x - 6, self.rect.y - 6))
        elif 400 < passed:
            self.kill()

class MysteryExplosion(sprite.Sprite):
    def __init__(self, mystery, score, *groups):
        super(MysteryExplosion, self).__init__(*groups)
        self.text = Text(FONT, 20, str(score), WHITE,
                         mystery.rect.x + 20, mystery.rect.y + 6)
        self.timer = time.get_ticks()
    def update(self, current_time, *args):
        passed = current_time - self.timer
        if passed <= 200 or 400 < passed <= 600:
            self.text.draw(game.screen)
        elif 600 < passed:
            self.kill()

class ShipExplosion(sprite.Sprite):
    def __init__(self, ship, *groups):
        super(ShipExplosion, self).__init__(*groups)
        self.image = IMAGES['ship']
        self.rect = self.image.get_rect(topleft=(ship.rect.x, ship.rect.y))
        self.timer = time.get_ticks()
    def update(self, current_time, *args):
        passed = current_time - self.timer
        if 300 < passed <= 600:
            game.screen.blit(self.image, self.rect)
        elif 900 < passed:
            self.kill()

class Life(sprite.Sprite):
    def __init__(self, xpos, ypos):
        sprite.Sprite.__init__(self)
        self.image = IMAGES['ship']
        self.image = transform.scale(self.image, (23, 23))
        self.rect = self.image.get_rect(topleft=(xpos, ypos))
    def update(self, *args):
        game.screen.blit(self.image, self.rect)

class Text(object):
    def __init__(self, textFont, size, message, color, xpos, ypos):
        self.font = font.Font(textFont, size)
        self.surface = self.font.render(message, True, color)
        self.rect = self.surface.get_rect(topleft=(xpos, ypos))
    def draw(self, surface):
        surface.blit(self.surface, self.rect)

class SpaceInvaders(object):
    def __init__(self):
        mixer.pre_init(44100, -16, 1, 4096)   # It seems, in Linux buffersize=512 is not enough,
use 4096 to prevent:  ALSA lib pcm.c:7963:(snd_pcm_recover) underrun occurred
        init()
        self.clock = time.Clock()
        self.caption = display.set_caption('Space Invaders Simpson Cutre!!')   #Aquí comienza el
destrozo
        self.screen = SCREEN
        self.background = image.load(IMAGE_PATH + 'background.jpg').convert()  #Bg de Simpson
incluido
        self.startGame = False
        self.mainScreen = True
        self.gameOver = False
        # Counter for enemy starting position (increased each new round)
        self.enemyPosition = ENEMY_DEFAULT_POSITION
        self.titleText = Text(FONT, 70, 'Space Invaders', RED, 64, 50)    # modificada título,
posición y tamaño
        self.titleText2 = Text(FONT, 25, 'Press any key to continue', RED, 201, 225)   # modificada
colores y posiciones
        self.titleText3 = Text(FONT, 20, 'Destrozado por D.Sedo', YELLOW, 250, 130)   # modificada
colores y posiciones
        self.gameOverText = Text(FONT, 50, 'You are dead!!!', PURPLE, 250, 250)
        self.nextRoundText = Text(FONT, 50, 'Next Round', PURPLE, 240, 270)
        self.enemy1Text = Text(FONT, 25, '   =  10 pts', GREEN, 368, 270)
        self.enemy2Text = Text(FONT, 25, '   =  20 pts', BLUE, 368, 320)
        self.enemy3Text = Text(FONT, 25, '   =  30 pts', PURPLE, 368, 370)
        self.enemy4Text = Text(FONT, 25, '   =  ?? pts', RED, 368, 420)
        self.scoreText = Text(FONT, 20, 'Score', WHITE, 5, 5)
        self.livesText = Text(FONT, 20, 'Lives ', WHITE, 600, 5)
        self.life1 = Life(688, 3)                                           #modificadas vidas,
añadida 1 vida
        self.life2 = Life(715, 3)
        self.life3 = Life(742, 3)
        self.life4 = Life(769, 3)
        self.livesGroup = sprite.Group(self.life1, self.life2, self.life3, self.life4)
    def reset(self, score):
        self.player = Ship()                #creado jugador con clase ship
        self.playerGroup = sprite.Group(self.player)
        self.explosionsGroup = sprite.Group()
        self.bullets = sprite.Group()
        self.mysteryShip = Mystery()            #creado mystery con clase Mystery
        self.mysteryGroup = sprite.Group(self.mysteryShip)
        self.enemyBullets = sprite.Group()
        self.make_enemies()
        self.allSprites = sprite.Group(self.player, self.enemies, self.livesGroup,
self.mysteryShip)
        self.keys = key.get_pressed()
        self.timer = time.get_ticks()
```

```python
        self.noteTimer = time.get_ticks()
        self.shipTimer = time.get_ticks()
        self.score = score
        self.create_audio()
        self.makeNewShip = False
        self.shipAlive = True
    def create_audio(self):
        self.sounds = {}
        for sound_name in ['shoot', 'shoot2', 'invaderkilled', 'mysterykilled', 'shipexplosion']:
            self.sounds[sound_name] = mixer.Sound(SOUND_PATH + '{}.wav'.format(sound_name))
            self.sounds[sound_name].set_volume(0.2)
        self.musicNotes = [mixer.Sound(SOUND_PATH + '{}.wav'.format(i)) for i in range(4)]
        for sound in self.musicNotes:
            sound.set_volume(0.5)
        self.noteIndex = 0
    def play_main_music(self, currentTime):
        if currentTime - self.noteTimer > self.enemies.moveTime:
            self.note = self.musicNotes[self.noteIndex]
            if self.noteIndex < 3:
                self.noteIndex += 1
            else:
                self.noteIndex = 0
            self.note.play()
            self.noteTimer += self.enemies.moveTime
    @staticmethod
    def should_exit(evt):
        # type: (pygame.event.EventType) -> bool
        return evt.type == QUIT or (evt.type == KEYUP and evt.key == K_ESCAPE)
    def check_input(self):
        self.keys = key.get_pressed()
        for e in event.get():
            if self.should_exit(e):
                sys.exit()
            if e.type == KEYDOWN:
                if e.key == K_SPACE:
                    if len(self.bullets) == 0 and self.shipAlive:
                        if self.score < 100:                                  #modificada disparos a partir de 100pts
                            bullet = Bullet(self.player.rect.x + 23, self.player.rect.y + 5, -1, 15, 'laser', 'center')
                            self.bullets.add(bullet)
                            self.allSprites.add(self.bullets)
                            self.sounds['shoot'].play()
                        elif 100 <= self.score < 200:
                            leftbullet = Bullet(self.player.rect.x + 8, self.player.rect.y + 5, -1, 15, 'laser', 'left')
                            rightbullet = Bullet(self.player.rect.x + 38, self.player.rect.y + 5, -1, 15, 'laser', 'right')
                            self.bullets.add(leftbullet)
                            self.bullets.add(rightbullet)
                            self.allSprites.add(self.bullets)
                            self.sounds['shoot2'].play()
                        elif self.score >= 200:            #triple disparo mítico añadido, indestructible!!
                            bullet = Bullet(self.player.rect.x + 23, self.player.rect.y + 5, -1, 15, 'laser', 'center')
                            leftbullet = Bullet(self.player.rect.x + 8, self.player.rect.y + 5, -1, 15, 'laser', 'left')
                            self.bullets.add(bullet)
                            rightbullet = Bullet(self.player.rect.x + 38, self.player.rect.y + 5, -1, 15, 'laser', 'right')
                            self.bullets.add(leftbullet)
                            self.bullets.add(rightbullet)
                            self.allSprites.add(self.bullets)
                            # self.sounds['shoot'].play()
                            self.sounds['shoot2'].play()
    def make_enemies(self):
        enemies = EnemiesGroup(10, 7)                                               #modificada grupo
enemigos a 7
        for row in range(7):
            for column in range(10):
                enemy = Enemy(row, column)
                enemy.rect.x = 157 + (column * 50)
                enemy.rect.y = self.enemyPosition + (row * 45)
                enemies.add(enemy)
        self.enemies = enemies
    def make_enemies_shoot(self):
        if (time.get_ticks() - self.timer) > 700 and self.enemies:
            enemy = self.enemies.random_bottom()
            self.enemyBullets.add(Bullet(enemy.rect.x + 14, enemy.rect.y + 20, 1, 5, 'enemylaser',
'center'))
            self.allSprites.add(self.enemyBullets)
            self.timer = time.get_ticks()
    def calculate_score(self, row):
        scores = {0: 30, 1: 20, 2: 20, 3: 10, 4: 10, 5: 10, 6:10, 7 : choice([50, 100, 150, 300])}
#modificada puntuación
        score = scores[row]
        self.score += score
        return score
    def create_main_menu(self):
        self.enemy1 = IMAGES['enemy3_1']
        self.enemy1 = transform.scale(self.enemy1, (40, 40))                      #escala en menu
principal
        self.enemy2 = IMAGES['enemy2_2']
        self.enemy2 = transform.scale(self.enemy2, (40, 40))
        self.enemy3 = IMAGES['enemy1_2']
        self.enemy3 = transform.scale(self.enemy3, (40, 40))
        self.enemy4 = IMAGES['mystery']
        self.enemy4 = transform.scale(self.enemy4, (80, 40))
        self.screen.blit(self.enemy1, (318, 270))                                 #posición en menu
principal
        self.screen.blit(self.enemy2, (318, 320))
        self.screen.blit(self.enemy3, (318, 370))
        self.screen.blit(self.enemy4, (299, 420))
    def check_collisions(self):
        sprite.groupcollide(self.bullets, self.enemyBullets, True, True)
        for enemy in sprite.groupcollide(self.enemies, self.bullets, True, True).keys():
            self.sounds['invaderkilled'].play()
            self.calculate_score(enemy.row)
            EnemyExplosion(enemy, self.explosionsGroup)
            self.gameTimer = time.get_ticks()
        for mystery in sprite.groupcollide(self.mysteryGroup, self.bullets, True, True).keys():
            mystery.mysteryEntered.stop()
            self.sounds['mysterykilled'].play()
            score = self.calculate_score(mystery.row)
            MysteryExplosion(mystery, score, self.explosionsGroup)
            newShip = Mystery()
            self.allSprites.add(newShip)
            self.mysteryGroup.add(newShip)
        for player in sprite.groupcollide(self.playerGroup, self.enemyBullets, True, True).keys():
            if self.life4.alive():
                self.life4.kill()
            elif self.life3.alive():
                self.life3.kill()
            elif self.life2.alive():
                self.life2.kill()
            elif self.life1.alive():
                self.life1.kill()
            else:
                self.gameOver = True
                self.startGame = False
            self.sounds['shipexplosion'].play()
            ShipExplosion(player, self.explosionsGroup)
            self.makeNewShip = True
            self.shipTimer = time.get_ticks()
            self.shipAlive = False
        if self.enemies.bottom >= 540:
            sprite.groupcollide(self.enemies, self.playerGroup, True, True)
            if not self.player.alive() or self.enemies.bottom >= 600:
                self.gameOver = True
                self.startGame = False
    def create_new_ship(self, createShip, currentTime):
        if createShip and (currentTime - self.shipTimer > 900):
            self.player = Ship()
            self.allSprites.add(self.player)
            self.playerGroup.add(self.player)
            self.makeNewShip = False
            self.shipAlive = True
    def create_game_over(self, currentTime):
        self.screen.blit(self.background, (0, 0))
        passed = currentTime - self.timer
        if passed < 750:
            self.gameOverText.draw(self.screen)
        elif 750 < passed < 1500:
            self.screen.blit(self.background, (0, 0))
        elif 1500 < passed < 2250:
            self.gameOverText.draw(self.screen)
        elif 2250 < passed < 2750:
            self.screen.blit(self.background, (0, 0))
        elif passed > 3000:
            self.mainScreen = True
        for e in event.get():
            if self.should_exit(e):
                sys.exit()
    def main(self):
        while True:
            if self.mainScreen:
                # self.screen.blit(self.background, (0, 0))
                self.titleText.draw(self.screen)
                self.titleText2.draw(self.screen)
                self.titleText3.draw(self.screen)
                self.enemy1Text.draw(self.screen)
                self.enemy2Text.draw(self.screen)
                self.enemy3Text.draw(self.screen)
                self.enemy4Text.draw(self.screen)
                self.create_main_menu()
                for e in event.get():
                    if self.should_exit(e):
                        sys.exit()
                    if e.type == KEYUP:
                        self.livesGroup.add(self.life1, self.life2, self.life3)
                        self.reset(0)
                        self.startGame = True
                        self.mainScreen = False
            elif self.startGame:
                if not self.enemies and not self.explosionsGroup:
                    currentTime = time.get_ticks()
                    if currentTime - self.gameTimer < 3000:
                        self.screen.blit(self.background, (0, 0))
                        self.scoreText2 = Text(FONT, 20, str(self.score), GREEN, 85, 5)
                        self.scoreText.draw(self.screen)
                        self.scoreText2.draw(self.screen)
                        self.nextRoundText.draw(self.screen)
                        self.livesText.draw(self.screen)
                        self.livesGroup.update()
                        self.check_input()
                    if currentTime - self.gameTimer > 3000:
                        # Move enemies closer to bottom
                        self.enemyPosition += ENEMY_MOVE_DOWN
                        self.reset(self.score)
                        self.gameTimer += 3000
                else:
                    currentTime = time.get_ticks()
                    self.play_main_music(currentTime)
                    self.screen.blit(self.background, (0, 0))
                    # self.allBlockers.update(self.screen)                 eliminados Blockers
                    self.scoreText2 = Text(FONT, 20, str(self.score), GREEN,85, 5)
                    self.scoreText.draw(self.screen)
                    self.scoreText2.draw(self.screen)
                    self.livesText.draw(self.screen)
                    self.check_input()
                    self.enemies.update(currentTime)
                    self.allSprites.update(self.keys, currentTime)
                    self.explosionsGroup.update(currentTime)
                    self.check_collisions()
                    self.create_new_ship(self.makeNewShip, currentTime)
                    self.make_enemies_shoot()
            elif self.gameOver:
                currentTime = time.get_ticks()
                # Reset enemy starting position
                self.enemyPosition = ENEMY_DEFAULT_POSITION
                self.create_game_over(currentTime)
            display.update()
            self.clock.tick(60)

if __name__ == '__main__':
    game = SpaceInvaders()
    game.main()
```