Khi chạy chương trình, ta thấy thiên thạch đã rơi liên tục tại cùng một vị trí. Chúng ta sẽ bổ sung thư viện **random** bằng lệnh **import random**, sau đó thiết lập tọa độ x nhận giá trị ngẫu nhiên từ 0 đến 480 bằng lệnh **random.randint(0, 480).** Lúc này thiên thạch đã rơi được ở những vị trí ngẫu nhiên.

```
3   import random
…   ...

16  while True:
…       ...
29      if meteor_y > 360:
30          meteor_y = 0
# đặt meteor_x giá trị ngẫu nhiên từ 0 đến 480
31          meteor_x = random.randint(0, 480)
```

3.2. Lập trình chức năng tính điểm

Điểm là một đoạn văn bản được vẽ trên màn hình. Chúng ta cần tạo hai biến, một biến lưu giá trị điểm (ví dụ: **score,** dòng 16) và một biến có kiểu văn bản (ví dụ: **score_text,** dòng 17) có thể chuyển đổi thành ảnh và vẽ được trên màn hình. Biến có kiểu văn bản được khởi tạo bằng lệnh **pygame.font.SysFont("<tên font chữ>", <cỡ chữ>).**

```
    # tạo biến score để lưu điểm
16  score = 0
    # tạo biến score_text là đối tượng văn bản có thể chuyển
    # đổi thành ảnh và vẽ lên màn hình
17  score_text = pygame.font.SysFont("Arial", 24)
18  while True:
```

Trong trò chơi, điểm sẽ được tăng thêm 1 khi tàu vũ trụ tránh được 1 thiên thạch. Vậy chúng ta lập trình tăng điểm thêm 1 sau khi thiên thạch đã rơi hết màn hình và xuất hiện lại bên trên.

```
31      if meteor_y > 360:
31          meteor_y = 0
33          meteor_x = random.randint(0, 480)
            # tăng điểm thêm 1
34          score += 1
```

Ta sử dụng lệnh **score_text.render(<đoạn văn bản cần vẽ>, True, <mã màu RGB>)** sử dụng để chuyển đổi biến có kiểu văn bản bên trên (biến **score_text**) trở thành thành ảnh bằng, qua đó ta có thể hiển thị lên màn hình. Trong đó:

- Tham số đầu tiên là đoạn văn bản cần chuyển, gồm chữ "Score: " ghép với biến **score** (cần chuyển **score** thành kiểu dữ liệu string) thành biểu thức **"Score: " + str(score).**
- Tham số thứ 2 quy định có hiệu ứng khử răng cưa (antialias) hay không, giá trị **True** tương ứng với có hiệu ứng, **False** là ngược lại.

- Tham số thứ 3 là ba giá trị (R, G, B) tương ứng với định dạng màu chữ RGB. Ảnh chuyển đổi được lưu vào một biến khác (ví dụ: **score_render**), sau đó chúng ta lập trình vẽ điểm lên màn hình bằng lệnh **screen.blit()** như các ảnh tàu và thiên thạch. Các bạn lưu ý đặt lệnh vẽ score bên dưới lệnh vẽ hình nền và thiên thạch để không bị hai hình ảnh này đè lên.

```
38    screen.blit(meteor_img, (meteor_x, meteor_y))
      # chuyển đổi văn bản thành ảnh có thể vẽ và lưu vào biến
      # score_render
39    score_render = score_text.render("Score: " + str(score),
                                       True, (255, 255, 255))
      # vẽ ảnh lên màn hình
40    screen.blit(score_render, (0, 0))
41    pygame.display.update()
```

Lưu ý: Ta cần chuyển score thành kiểu dữ liệu string vì trong python hai kiểu dữ liệu khác nhau sẽ không tương tác (cộng, trừ, nối,...) được với nhau. Chính vì vậy, ta cần chuyển đổi kiểu dữ liệu số thành kiểu dữ liệu chữ, sau đó nối chúng lại với nhau.

Tóm tắt lý thuyết và bài tập thực hành

Bài tập 1. Lập trình ngôi sao xuất hiện và rơi xuống với tốc độ nhanh gấp 2 lần tốc độ thiên thạch.

Bài tập 2. Lập trình thiết lập các giá trị màu sau cho giá trị điểm và nêu màu thiết lập được là màu gì:

+ 0, 255, 255: .....

+ 255, 0, 255: .....

+ 255, 255, 0: .....

BÀI 4 XỬ LÝ VA CHẠM VÀ XUẤT CHƯƠNG TRÌNH THÀNH TỆP TIN THỰC THI

Trong bài học này, chúng ta sẽ lập trình xử lý va chạm giữa thiên thạch và tàu vũ trụ, sau đó tìm hiểu cách xuất chương trình thành tệp tin thực thi (.exe).

4.1. Xử lý va chạm khi thiên thạch chạm vào tàu

Khi thiên thạch chạm vào tàu, thiên thạch sẽ dừng lại, tàu biến mất và âm thanh nổ được phát ra. Các bạn có thể thấy chương trình gồm 2 trạng thái: Kết thúc và chưa kết thúc. Chúng ta sẽ lập trình để khi ở trạng thái kết thúc, lệnh di chuyển của thiên thạch và lệnh vẽ tàu sẽ không được thực hiện nữa.

Các bạn tạo một biến để lưu trạng thái chương trình đã kết thúc hay chưa (ví dụ: **game_over**), chúng ta quy ước:

- Nếu **game_over** có giá trị **False** tức là chương trình chưa kết thúc,
- Nếu **game_over** có giá trị **True** tức là chương trình đã kết thúc.

Ban đầu, biến cần được đặt giá trị **False** bằng lệnh **game_over = False** đặt trước lệnh lặp **while**.

Để biết được khi nào thiên thạch chạm vào tàu, chúng ta sẽ liên tục tính khoảng cách giữa thiên thạch và tàu thông qua tọa độ, nếu khoảng cách này nhỏ hơn một giá trị nào đó mà chúng ta quy định thì thiên thạch và tàu được coi là va chạm với nhau.

Trong mặt phẳng tọa độ, khoảng cách giữa 2 điểm A (x_A, y_A) và B (x_B, y_B) được tính theo công thức:

$$distance = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Chúng ta sẽ tạo một biến để lưu khoảng cách (ví dụ: **distance**). Biến **distance** chỉ sử dụng trong lệnh lặp chính, vì vậy chúng ta không cần khai báo trước từ bên ngoài. Chúng ta bổ sung thư viện **math** bằng lệnh **import math**, sau đó dùng lệnh **math.sqrt(<số cần khai căn>)** để thực hiện phép khai căn. Giá trị **distance** được tính theo công thức như sau:

```
distance = math.sqrt((ship_x - meteor_x)**2
                     + (ship_y - meteor_y)**2)
```

Lưu ý: Trong python, dấu "**" để tính số mũ, ví dụ: x**2 = x^2; x**3 = x^3; x**4 = x^4...

Chúng ta đang sử dụng tọa độ của đối tượng là vị trí góc trên bên trái của hình ảnh, việc sử dụng điểm chính giữa của hình ảnh không mang đến sự khác biệt đáng kể. Chúng ta sẽ ước lượng khoảng cách này, sau đó thực hiện "thử và sửa". "Thử và sửa" là quá trình chúng ta chạy chương trình với một giá trị thử nghiệm nhất định, tùy thuộc vào kết quả hoạt động mà chúng ta điều chỉnh giá trị sao cho phù hợp, sau đó lại lặp lại tới khi đạt được kết quả mong muốn. Với kích thước của hình ảnh có sẵn trong tài liệu, giá trị khoảng cách phù hợp là khoảng 60. Các bạn có thể tự mình thay đổi giá trị này theo ý muốn của mình.

Các bạn lập trình nếu khoảng cách nhỏ hơn 60 thì giá trị biến **game_over** sẽ được đặt thành **True**.

```
distance = math.sqrt((ship_x - meteor_x)**2
                                    + (ship_y - meteor_y)**2)
if distance < 60:
    game_over = True
```

Khi đã thiết lập được giá trị cho biến **game_over**, chúng ta lập trình việc cập nhật vị trí tàu, đá, vẽ tàu, vẽ đá chỉ khi khi biến **game_over** mang giá trị **False**. Điều này tương đương với nếu **game_over** bằng **True** thì những công việc này sẽ không được thực hiện. Việc kiểm tra khoảng cách cũng ghép vào bên trong lệnh kiểm tra **game_over** để tránh bị thực hiện lặp lại khi dừng chương trình.

```
...     ...
32      if not game_over:
33          ship_x += x_change
34          meteor_y += 2
35          distance = math.sqrt((ship_x - meteor_x)**2
                                    + (ship_y - meteor_y)**2)
36          if distance < 60:
37              game_over = True
...     ...
43      if not game_over:
44          screen.blit(ship_img, (ship_x, ship_y))
```

Chúng ta có thể gộp chung lệnh vẽ tàu và cập nhật trạng thái di chuyển vào chung, nhưng các bạn cần lưu ý chuyển vẽ hình nền lên trước.

Cuối cùng, chúng ta bổ sung các lệnh để có thể phát âm thanh nổ **explosion.wav** trong tài liệu đã tải về. Các bạn tải âm thanh vào chương trình bằng lệnh **pygame.mixer.Sound(<đường dẫn đến tệp tin âm thanh>)** và gán vào một biến mới (ví dụ: **ship_sound**).

```
ship_sound = pygame.mixer.Sound("resources/explosion.wav")
```

Để phát được âm thanh, chúng ta sử dụng lệnh **ship_sound.play()** và đặt ở vị trí sau khi kiểm tra va chạm xảy ra. Dưới đây là toàn bộ mã nguồn chương trình.

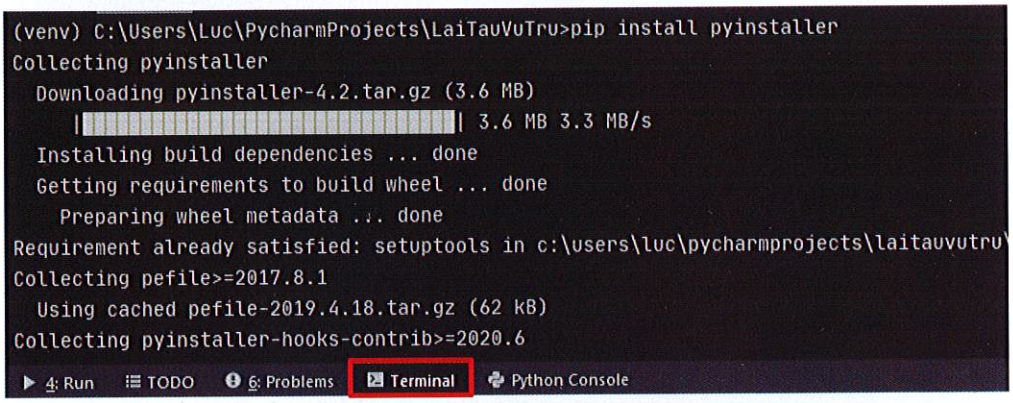
```
import pygame
import sys
import random
import math

pygame.init()
screen = pygame.display.set_mode((480, 360))
pygame.display.set_caption("Lái tàu vũ trụ")
ship_img = pygame.image.load("resources/rocketship.png")
pygame.display.set_icon(ship_img)
bg_image = pygame.image.load("resources/bg_galaxy.png")
ship_x = 220
ship_y = 280
ship_sound = pygame.mixer.Sound("resources/explosion.wav")

meteor_img = pygame.image.load("resources/meteoroid.png")
meteor_x = 220
meteor_y = 0
x_change = 0
score = 0
score_text = pygame.font.SysFont("Arial", 24)
game_over = False

while True:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      pygame.quit()
      sys.exit()
    if event.type == pygame.KEYDOWN:
      if event.key == pygame.K_LEFT:
        x_change = -5
      elif event.key == pygame.K_RIGHT:
        x_change = 5
    if event.type == pygame.KEYUP:
      x_change = 0
  if not game_over:
    ship_x += x_change
    meteor_y += 2
    distance = math.sqrt((ship_x - meteor_x)**2 +
                         (ship_y - meteor_y)**2)
    if distance < 60:
      game_over = True
      ship_sound.play()
      if meteor_y > 360:
        meteor_y = 0
        meteor_x = random.randint(0, 480)
        score += 1
```

```
44    screen.blit(bg_image, (0, 0))
45    if not game_over:
46      screen.blit(ship_img, (ship_x, ship_y))
47    screen.blit(meteor_img, (meteor_x, meteor_y))
48    score_render = score_text.render("Score: " + str(score),
                                       True, (255, 255, 255))
49    screen.blit(score_render, (0, 0))
50    pygame.display.update()
```

Các bạn lưu ý, hình trên là mã nguồn phần cơ bản, các bạn đọc và làm bài tập trong các bài học để tự bổ sung thêm những chức năng khác cho chương trình.

4.2. Xuất chương trình thành tệp tin thực thi

Hiện tại, chúng ta chỉ có thể chạy chương trình khi sử dụng một trình dịch Python. Để có thể chuyển thành tệp tin thực thi (tệp tin có phần mở rộng **.exe**), các bạn sẽ sử dụng thư viện **pyinstaller**. Các bạn cài đặt thư viện này bằng cách mở thẻ **Terminal** và gõ lệnh `pip install pyinstaller`, sau đó nhấn **Enter**.



Thẻ Terminal mặc định đã truy cập vào thư mục hiện tại của dự án, các bạn tiếp tục gõ lệnh `pyinstaller [tùy chọn] <tên tệp tin>` để xuất dự án thành tệp tin thực thi (ví dụ: `pyinstaller main.py`). Mặc định, việc xuất dự án như vậy sẽ tạo ra rất nhiều tệp tin phụ thuộc đi kèm với tệp tin thực thi và khi chạy sẽ tạo thêm một cửa sổ console. Chúng ta nên sử dụng thêm một số tùy chọn để giảm bớt những chức năng bất tiện như vậy:

- Tùy chọn **--onefile** giúp gói gọn các tệp tin phụ thuộc và tệp tin thực thi thành một tệp tin thực thi duy nhất
- Tùy chọn **-w** giúp loại bỏ cửa sổ console

Các bạn chạy lệnh `pyinstaller –onefile –w main.py`, một vài thư mục và tệp tin sẽ được tạo ra trong thư mục dự án.