



BÀI 2

LẬP TRÌNH TÀU VŨ TRỤ DI CHUYỂN BẰNG CÁC PHÍM MŨI TÊN

Trong bài học này, chúng ta sẽ tìm hiểu về cách bắt sự kiện trong pygame, sau đó lập trình khi nhấn nút  (góc phải phía trên cửa sổ chương trình) sẽ thoát trò chơi và tàu sẽ di chuyển khi các phím mũi tên trên bàn phím được nhấn.

2.1. Bắt sự kiện thoát chương trình

“Bắt sự kiện” là việc lập trình kiểm tra xem một sự kiện nào đó có xảy ra hay không, từ đó thực hiện các công việc tương ứng, trong trường hợp này là kiểm tra xem người dùng có nhấn nút  hay không để dừng chương trình.

`pygame.event.get()` là danh sách chứa các sự kiện trong **pygame** (được gọi là danh sách sự kiện), mỗi phần tử trong danh sách là một sự kiện, nếu muốn bắt (giám sát và biết) một sự kiện xảy ra (ví dụ nhấn nút trên bàn phím, nhấn chuột, đóng cửa sổ...) thì sự kiện đó sẽ được thêm vào danh sách này chứ không có sẵn trong danh sách.

Ví dụ:

Danh sách sự kiện

phần tử 0	Nhấn phím sang trái
phần tử 1	Nhấn phím sang phải
phần tử 2	Nhấn phím a
...	...

Để bắt một sự kiện, chúng ta cần duyệt toàn bộ danh sách và kiểm tra từng phần tử xem sự kiện chúng ta cần tìm có xuất hiện trong danh sách hay không, sau đó thực hiện các công việc cần thiết khi gặp đúng sự kiện giám sát. Chúng ta sẽ sử dụng vòng lặp **for** để duyệt toàn bộ phần tử trong danh sách theo cú pháp dưới đây:

```
for <biến lặp> in <danh sách>:
    # Các lệnh nằm bên trong vòng lặp for được lặp đi lặp
    # lại, chú ý thụt lề một cấp so với lệnh for
    <các lệnh cần thực hiện lặp>

# Vị trí này là các lệnh bên ngoài vòng lặp for và có cùng cấp
# (cùng mức thụt lề) với vòng lặp for, sẽ được thực hiện sau
# khi vòng lặp kết thúc
```

Trong đó, biến lặp là biến được chúng ta đặt tên tùy ý và biến đó ta chỉ sử dụng được bên trong vòng lặp. Với mỗi lần lặp, biến này nhận giá trị là một phần tử trong danh sách (lần lượt từ phần tử đầu tiên đến phần tử cuối cùng) và đoạn lệnh bên trong vòng lặp sẽ được thực hiện. Ví dụ lệnh dưới đây sẽ sử dụng biến lặp **so** (số) thực hiện in ra các phần tử trong danh sách [1, 2, 3]. Chúng ta sẽ tìm hiểu về danh sách kỹ hơn trong các bài học sau. Hình dưới đây minh họa cách sử dụng vòng lặp **for** để duyệt các phần tử trong danh sách.




```
>>> for so in [1, 2, 3]:
...     print(so)
...
1
2
3
```

Vòng lặp for

Trong Python, vòng lặp **for** luôn được sử dụng để duyệt danh sách. Nếu muốn sử dụng để lặp lại công việc với số lần nhất định, chúng ta có thể sử dụng lệnh **range(n)** với **n** là số lần lặp. Về bản chất, lệnh **range(n)** sẽ tạo một danh sách có **n** phần tử từ 0 đến **n-1**, việc duyệt qua **n** phần tử này tương đương với việc thực hiện lặp lại **n** lần và chúng ta không cần quan tâm đến giá trị biến lặp. Hình dưới đây mô tả cách sử dụng vòng lặp **for** thực hiện việc tăng biến **i** thêm 2 tổng cộng 5 lần.

```
>>> i = 0
>>> for so in range(5):
...     i = i + 2
...
>>> print(i)
10
```

Trong hình dưới đây chúng ta bổ sung vòng lặp **for** bên trong lệnh lặp **while** để duyệt từng phần tử trong danh sách sự kiện, biến lặp có thể đặt tên là **event**. Vòng lặp **while**, **for** và một số cấu trúc khác bắt buộc không được để trống bên trong, nếu chúng ta chưa lập trình đến chức năng bên trong vòng lặp và muốn tạm thời để trống vị trí này, chúng ta có thể sử dụng lệnh **pass**. Lệnh này không thực hiện bất cứ công việc nào mà chỉ có chức năng “giữ chỗ” cho các lệnh khác.

```
8 while True:
9     for event in pygame.event.get():
10         # thực hiện việc kiểm tra sự kiện ở đây
11         pass
12         # hai lệnh dưới đây cùng cấp với vòng lặp for, được thực
13         # hiện sau khi vòng lặp for kết thúc
14     screen.blit(bg_img, (0, 0))
15     pygame.display.update()
```



Các phần tử trong danh sách sự kiện **pygame.event.get()** có thuộc tính **type** quy định đó là kiểu sự kiện nào, chúng ta cần kiểm tra giá trị **event.type** có bằng với các giá trị của kiểu sự kiện được quy ước trong pygame không, ví dụ kiểu sự kiện nhấn nút thoát được quy ước có giá trị là **pygame.QUIT**. Để thực hiện kiểm tra, chúng ta sử dụng lệnh rẽ nhánh **if** với cú pháp:

```
if <điều kiện>:
    # Các lệnh nằm bên trong lệnh if được thực hiện khi <điều
    # kiện> có giá trị đúng, chú ý thụt lề một cấp
    <các lệnh cần thực hiện khi điều kiện đúng>
# Vị trí này là các lệnh bên ngoài lệnh if và có cùng cấp
# (cùng mức thụt lề) với lệnh if
```

Chúng ta so sánh **event.type** có giá trị bằng **pygame.QUIT** hay không bằng biểu thức **event.type == pygame.QUIT**.

```
8 while True:
9     for event in pygame.event.get():
10        if event.type == pygame.QUIT:
11            # thực hiện việc thoát chương trình
12            pass
13
14 screen.blit(bg_img, (0, 0))
15 pygame.display.update()
```

Cấu trúc đầy đủ của lệnh rẽ nhánh **if**

Với cấu trúc **if** vừa sử dụng, chúng ta không quan tâm đến trường hợp điều kiện sai, nếu điều kiện sai thì chương trình sẽ bỏ qua lệnh bên trong mà tiếp tục thực hiện các lệnh tiếp theo cùng cấp với lệnh **if**. Các bạn có thể bổ sung từ khóa **else** để chương trình thực hiện các công việc khi điều kiện sai:

```
if <điều kiện>:
    <các lệnh thực hiện khi điều kiện đúng>
else:
    <các lệnh thực hiện khi điều kiện sai>
# Vị trí này là các lệnh bên ngoài lệnh if và có cùng cấp
# (cùng mức thụt lề) với lệnh if
```

Với cấu trúc có **else** được gọi là cấu trúc **if-else**, chương trình sẽ thực hiện một trong hai đoạn lệnh <thực hiện khi điều kiện đúng> hoặc <thực hiện khi điều kiện sai> tùy thuộc vào kết quả kiểm tra điều kiện, đoạn lệnh còn lại sẽ bị bỏ qua.



Để thoát chương trình, chúng ta cần import thêm thư viện **sys**, sử dụng 2 lệnh **pygame.quit()** và **sys.exit()**. Hình dưới đây mô tả toàn bộ chương trình đến thời điểm hiện tại, lúc này các bạn có thể nhấn nút  để thoát chương trình.

```
1 import pygame
2 import sys      # bổ sung thư viện sys

3 pygame.init()
4 screen = pygame.display.set_mode((480, 360))
5 pygame.display.set_caption("Lái tàu vũ trụ")
6 ship_img = pygame.image.load("resources/rocketship.png")
7 pygame.display.set_icon(ship_img)
8 bg_img = pygame.image.load("resources/bg_galaxy.png")

9 while True:
10     for event in pygame.event.get():
11         if event.type == pygame.QUIT:
12             # bổ sung 2 lệnh thoát chương trình
13             pygame.quit()
14             sys.exit()

15     screen.blit(bg_img, (0, 0))
16     pygame.display.update()
```

2.2. Thêm tàu vũ trụ và bắt sự kiện nhấn phím trên bàn phím

Để thêm con tàu, chúng ta cũng cần thực hiện các bước tải ảnh vào chương trình và vẽ ra màn hình. Chúng ta đã có biến **ship_img** đã chứa dữ liệu ảnh con tàu từ trước, chúng ta chỉ cần bổ sung việc vẽ ảnh con tàu bằng lệnh **screen.blit()** phía dưới lệnh vẽ hình nền. Các bạn bổ sung thêm lệnh vẽ con tàu ở tọa độ (100, 100) và chạy chương trình.

Lưu ý: Hình ảnh các đối tượng được vẽ trên màn hình được sắp xếp đứng trước hay đứng sau theo thứ tự lập trình, các hình ảnh vẽ sau sẽ đè lên trên các hình ảnh được vẽ trước đó, vì vậy lệnh vẽ con tàu cần đặt phía dưới lệnh vẽ hình nền.

```
14 screen.blit(bg_image, (0, 0))
15     # bổ sung lệnh vẽ tàu ở vị trí sau lệnh vẽ nền
16     screen.blit(ship_img, (100, 100))
17     pygame.display.update()
```