บทที่ 17
การปฏิสัมพันธ์ด้วย PyGame

บทนำ

การเขียนโปรแกรมคอมพิวเตอร์กราฟิกส์จัดเป็นการเขียนโปรแกรมเพื่อสร้างส่วนเชื่อมประสานกับผู้ใช้แบบกราฟิกส์ (GUI : Graphics User Interface) ประเภทหนึ่ง โดยผู้ใช้มองผลลัพธ์ทางจอแสดงผล และนำเข้าข้อมูลเพื่อปฏิสัมพันธ์กับโปรแกรมผ่านทางอุปกรณ์นำเข้า ได้แก่ เมาส์ แป้นพิมพ์ และเกมคอนโทรลเลอร์ หรือจอสัมผัส

ดังนั้น ในบทนี้เป็นเนื้อหาเกี่ยวกับการเขียนโปรแกรมเพื่อปฏิสัมพันธ์กับผู้ใช้ ผ่านทางไลบรารี PyGame ซึ่งเป็นไลบรารีที่ครอบการทำงานของไลบรารี SDL2 ทำให้รองรับการทำงานทั้งระบบปฏิบัติการวินโดวส์ แม็คโอเอส และลินุกซ์ พร้อมทั้งเป็นพื้นฐานสำหรับการเชื่อมประสารกับไลบรารี OpenGL ทำให้สามารถโหลดไฟล์ภาพแบบ JPG, TIFF และ PNG เพื่อใช้เป็นลายผิว (Texture) และรองรับการเรนเดอร์ข้อความจากตัวอักษรไฟล์

ความหมายและความสำคัญของการปฏิสัมพันธ์

การปฏิสัมพันธ์ (interactive) คือ การกระทำหรือการประกอบกิจกรรมระหว่างสิ่งสองสิ่งหรือหลายสิ่งเพื่อให้ได้ผลลัพธ์ต่อสิ่งที่กำลังตอบโต้หรือกระทำอยู่ ในทางคอมพิวเตอร์เรามีการศึกษาถึงการปฏิสัมพันธ์ในประเด็นเกี่ยวกับการปฏิสัมพันธ์กับมนุษย์ (HCI: Human Computer Interaction), การเชื่อมประสานกับผู้ใช้ (UI: User Interface), การเชื่อมประสานกับผู้ใช้แบบกราฟิกส์หรือจียูไอ เป็นต้น

หน้าที่ของ PyGame

PyGame เป็นไลบรารีแบบเปิดเผยโค้ดโปรแกรมที่ออกแบบเพื่อใช้กับภาษาไพธอนเพื่อสร้างโปรแกรมประยุกต์ด้านสื่อผสมประเภทเกม โดยทำงานอยู่เหนือไลบรารี SDL2 และสามารถทำงานร่วมกับ OpenGL โดยไม่ขึ้นกับระบบปฏิบัติการ มีความสามารถรองรับการทำงานกับหน่วยประมวลผลแบบหลายแกน (Multi core CPUs) และมีการเพิ่มประสิทธิภาพการทำงานด้วยโค้ดภาษาซีและแอสเซมบลีทำให้มีความเร็วสูงกว่าการเขียนด้วยโค้ดภาษาไพธอนในแบบปกติ TTF ซึ่งคำสั่งสำหรับติดตั้งไลบรารี PyGame เป็นดังนี้

```
pip3  install pysdl2  pygame  pyopengl
```

โค้ดหลักของการเรียกใช้ PyGame เป็นดังโปรแกรมที่ 17.1 ซึ่งการใช้งาน PyGame มีขั้นตอนเบื้องต้นอยู่ 3 ส่วน คือ

1. เรียกใช้ pygame
2. เรียกให้ไลบนรารีเริ่มต้นทำงาน

```
pygame.init()
```

3. เมื่อสิ้นสุดโปรแกรมจะต้องสั่งปิดการทำงานของ PyGame ด้วยคำสั่งดังนี้

```
pygame.quit()
```

โปรแกรม 17.1 โปรแกรมตัวอย่างการใช้ PyGame

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`import sys`
4	`pygame.init()`
5	`screen = pygame.display.set_mode((800,600))`
6	`pygame.display.set_caption("Hello PyGame")`
7	`background = pygame.Color(100, 149, 237)`
8	`while True:`
9	`    screen.fill(background)`
10	`    for event in pygame.event.get():`
11	`        if event.type == pygame.QUIT:`
12	`            pygame.quit()`
13	`            sys.exit()`
14	`    pygame.display.flip()`

หน้าต่างและการแสดงผล

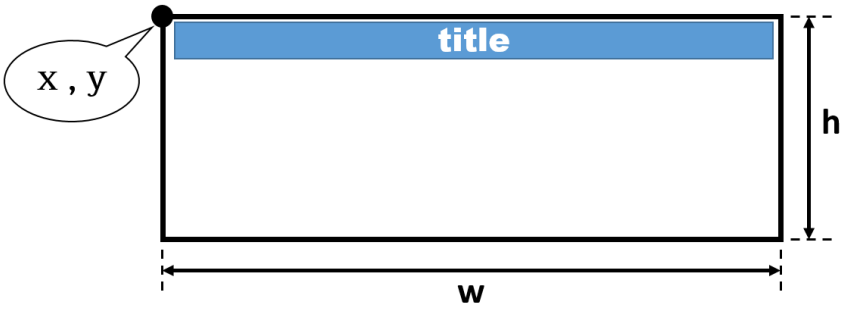
หน้าต่างเป็นส่วนแสดงผลการทำงานเพื่อให้ผู้ใช้ทราบถึงผลการกระทำที่ส่งผ่านทางแป้นพิมพ์หรือเมาส์ ในการเตรียมหน้าต่างเพื่อแสดงผลของโปรแกรมมีรูปแบบของคำสั่งการเปิดหน้าต่างเป็นดังนี้

```
ตัวแปรหน้าต่าง = pygame.display.set_mode( resolution=(w, h), flags=0)
```

โดยที่ w คือ ความกว้างของหน้าต่าง

h คือ ความสูงของหน้าต่าง

flags คือ การตั้งค่าคุณสมบัติของหน้าต่างซึ่งมีค่าตามตารางที่ 17.1



ภาพที่ 17.1 รูปแบบของหน้าต่าง

ตารางที่ 17.1 ค่า flags ของ display.set_mode

flags	ความหมาย
pygame.FULLSCREEN	เปิดหน้าต่างแบบเต็มจอ
pygame.DOUBLEBUF	ใช้เทคนิคบัฟเฟอร์ 2 ชุด
pygame.HWSURFACE	เข้าถึงหน้าจอโดยตรงกับฮาร์ดแวร์
pygame.OPENGL	ใช้ไลบรารี OpenGL ในการทำงาน
pygame.RESIZABLE	ยอมให้หน้าต่างถูกปรับขนาดได้
pygame.NOFRAME	กำหนดให้หน้าต่างไม่มีเส้นขอบ
pygame.OPENGLBLIT	ใช้เทคการบลิต (Blit หรือ การคัดลอกหน่วยความจำส่งไปที่หน่วยความจำแสดงผลของการ์ดแสดงผล) ผ่านทางไลบรารี OpenGL

คำสั่งเปลี่ยนชื่อหน้าต่างที่แสดงผลบนไตเติลบาร์ (title bar) หรือแคปชัน (Caption) ดังภาพที่ 17.1 มีรูปแบบดังนี้

```
pygame.display.set_caption("ข้อความ")
```

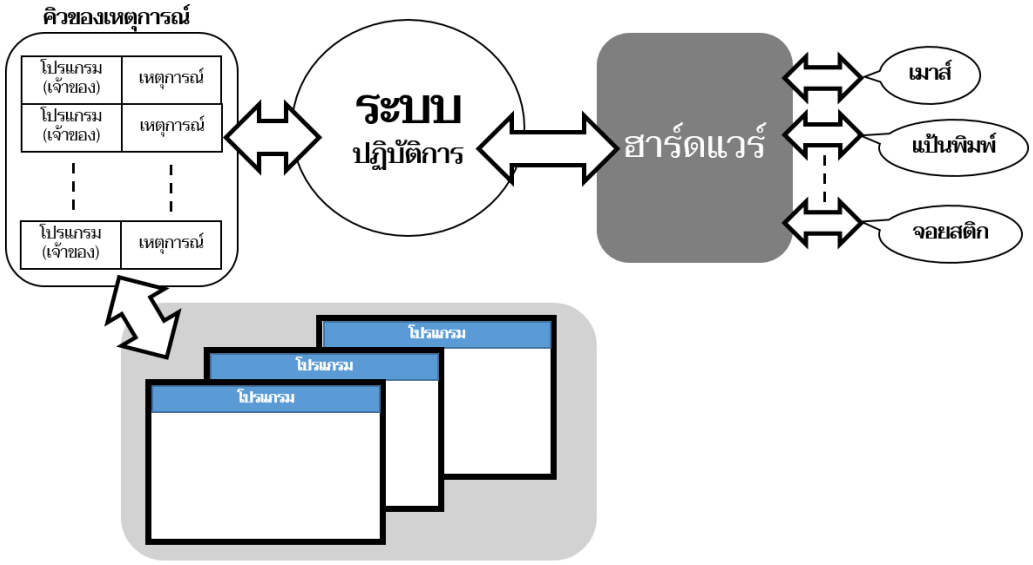
คำสั่งสำหรับให้ PyGame โอนข้อมูลจากบัฟเฟอร์ของการ์ดแสดงผลไปแสดงบนหน้าต่าง มีรูปแบบดังนี้

```
pygame.display.flip()
```

เหตุการณ์ (event) คือ สิ่งที่เกิดขึ้นในระบบปฏิบัติการ โดยผู้เขียนโปรแกรมมีหน้าที่คอยตรวจสอบเหตุการณ์ที่เกิดขึ้นในระบบว่ามีเหตุการณ์ใดเป็นของโปรแกรมที่เราเขียน และเหตุการณ์ที่เกิดขึ้นนั้นเราต้องการตอบสนองกับไปหรือไม่ ดังภาพที่ 17.2 การดักเหตุการณ์ที่เกิดขึ้นกับโปรแกรมมีรูปแบบการเขียนโค้ดดังนี้

```
for ตัวแปรเหตุการณ์ in pygame.event.get():
    if ตัวแปรเหตุการณ์.type == pygame.QUIT:
        pygame.quit()
        sys.exit()
```

จากโค้ดข้างต้นจะพบว่าคำสั่ง pygame.event.get() มีหน้าที่ร้องขอตรวจสอบเหตุการณ์จากระบบปฏิบัติการ ถ้าพบว่ามีเหตุการณ์ที่เกี่ยวกับโปรแกรมเกิดขึ้น จะนำข้อมูลเหตุการณ์เก็บในตัวแปรที่กำหนด หลังจากนั้นสามารถตรวจสอบประเภทเหตุการณ์ได้จากคุณสมบัติ type และถ้ามีค่าเป็น pygame.QUIT หมายความว่า ผู้ใช้คลิกปิดหน้าต่าง



ภาพที่ 17.2 การทำงานของระบบปฏิบัติการแบบขับเคลื่อนด้วยเหตุการณ์ (event driven)

พื้นที่แสดงผลของจอภาพมีชื่อเรียกว่า พื้นผิว (Surface) ซึ่งเป็นหน่วยความจำสำหรับเก็บข้อมูลสีของบริเวณแสดงผล โดยแต่ละจุดแสดงผลมีโครงสร้างตามการทำงานของการ์ดแสดงผล ซึ่งปัจจุบันมักทำงานในโหมด RGB หรือ RGBA ซึ่งหมายความว่า แต่ละจุดสีที่แสดงประกอบไปด้วยข้อมูลระดับความเข้ม (มีค่าอยู่ในช่วง 0 ถึง 255) ของแม่สีแดง (R: Red) เขียว (G: Green) น้ำเงิน (B:Blue) หรือการโปร่งแสง (A: Alpha)

วิธีการเปลี่ยนสีพื้นหลังของหน้าต่างทำได้โดยการสร้างตัวแปรเก็บค่าสี หลังจากนั้นสั่ง fill() สีลงหน้าจอ ซึ่งรูปแบบของของสั่งทั้ง 2 เป็นดังนี้

```
ตัวแปรเก็บสี = pygame.Color(แดง,เขียว,น้ำเงิน)
ตัวแปรหน้าต่าง.fill( ตัวแปรเก็บสี )
```

การนำเข้าข้อมูล

การนำเข้าข้อมูลในบทนี้กล่าวถึงการนำเข้าจากแป้นพิมพ์ เมาส์และเกมคอนโทรลเลอร์ ซึ่งคุณลักษณ์ของข้อมูลและวิธีการใช้งานมีความแตกต่างกัน

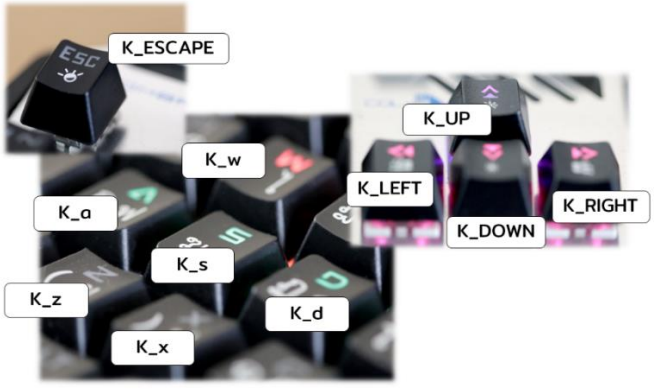
การเชื่อมประสานกับแป้นพิมพ์

แป้นพิมพ์เป็นอุปกรณ์นำเข้าข้อมูลที่ถูกนำมาใช้อย่างยาวนาน โดยสิ่งที่ผู้เขียนโปรแกรมได้รับจากอุปกรณ์ประเภทนี้ คือ ตัวอักขระที่ผู้ใช้กด และสถานะของแป้นพิมพ์ ซึ่งประเภทของเหตุการณ์หรือ type ที่ได้รับจาก PyGame มี 2 ประเภท คือ pygame.KEYDOWN และ pygame.KEYUP เพื่อแจ้งให้ทราบว่ามีการกดแป้นพิมพ์และปล่อยแป้มพิมพ์ ส่วนค่าของแป้นพิมพ์ที่ถูกกดจะถูกเก็บในคุณสมบัติ key ของเหตุการณ์ที่เกิดขึ้น โดยชื่อแป้นที่ถูกใช้ใน PyGame จะขึ้นต้นด้วย K_ (ภาพที่ 17.3) จะมีรูปแบบของชื่อเรียกเป็นดังนี้

```
pygame.ชื่อแป้นที่กด
```

ซึ่งรายชื่อแป้นที่กดมีดังนี้

K_BACKSPACE	K_TAB	K_CLEAR	K_RETURN	K_PAUSE
K_ESCAPE	K_SPACE	K_EXCLAIM	K_QUOTEDBL	K_HASH
K_DOLLAR	K_AMPERSAND	K_QUOTE	K_LEFTPAREN	K_RIGHTPAREN
K_ASTERISK	K_PLUS	K_COMMA	K_MINUS	K_PERIOD
K_SLASH	K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7	K_8
K_9	K_NUMLOCK	K_CAPSLOCK	K_SCROLLOCK	K_RSHIFT
K_LSHIFT	K_RCTRL	K_LCTRL	K_RALT	K_LALT
K_RMETA	K_LMETA	K_LSUPER	K_RSUPER	K_MODE
K_HELP	K_PRINT	K_SYSREQ	K_BREAK	K_MENU
K_POWER	K_EURO	K_F1	K_F2	K_F3
K_F4	K_F5	K_F6	K_F7	K_F8
K_F9	K_F10	K_F11	K_F12	K_F13
K_F14	K_F15	K_DELETE	K_KP0	K_KP1
K_KP2	K_KP3	K_KP4	K_KP5	K_KP6
K_KP7	K_KP8	K_KP9	K_KP_PERIOD	K_KP_DIVIDE
K_KP_MULTIPLY	K_KP_MINUS	K_KP_PLUS	K_KP_ENTER	K_KP_EQUALS
K_UP	K_DOWN	K_RIGHT	K_LEFT	K_INSERT
K_HOME	K_END	K_PAGEUP	K_PAGEDOWN	K_COLON
K_SEMICOLON	K_LESS	K_EQUALS	K_GREATER	K_QUESTION
K_AT	K_LEFTBRACKET	K_BACKSLASH	K_RIGHTBRACKET	
K_CARET	K_UNDERSCORE	K_BACKQUOTE	K_a	K_b
K_c	K_d	K_e	K_f	K_g
K_h	K_i	K_j	K_k	K_l
K_m	K_n	K_o	K_p	K_q
K_r	K_s	K_t	K_u	K_v
K_w	K_x	K_y	K_z	



ภาพที่ 17.3 ตัวอย่างชื่อเรียกแป้นพิมพ์

ตัวอย่างโปรแกรมที่ 17.2 เป็นโปรแกรมเปลี่ยนสีพื้นตามแป้นที่กด คือ

กด r แสดงสีแดง

กด g แสดงสีเขียว

กด b แสดงสีน้ำเงิน

และกด ESC สำหรับออกจากโปรแกรม

โปรแกรม 17.2 โปรแกรมเปลี่ยนสีพื้นหน้าต่างตามแป้นที่กด

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`pygame.init()`
4	`screen = pygame.display.set_mode((800,600))`
5	`pygame.display.set_caption("Ex 17.2")`
6	`black = pygame.Color(0,0,0)`
7	`red = pygame.Color(255,0,0)`
8	`green = pygame.Color(0,255,0)`
9	`blue = pygame.Color(0,0,255)`
10	`bgcolor = black`
11	`running = True`
12	`while running:`
13	`    screen.fill(bgcolor)`
14	`    for event in pygame.event.get():`
15	`        if event.type == pygame.QUIT:`
16	`            running = False`
17	`        if event.type == pygame.KEYDOWN:`
18	`            if event.key == pygame.K_r:`
19	`                bgcolor = red`
20	`            elif event.key == pygame.K_g:`
21	`                bgcolor = green`
22	`            elif event.key == pygame.K_b:`
23	`                bgcolor = blue`
24	`            elif event.key == pygame.K_ESCAPE:`
25	`                running = False`
26	`    pygame.display.flip()`
27	`pygame.quit()`

กรณีที่ต้องการให้ผู้ใช้สามารถกดแป้นค้างแทนการกดแป้นเดิมซ้ำ ๆ สามารถกระทำโดยสั่งค่าการรับช่วงค่าเวลาในการกดตามรูปแบบคำสั่งต่อไปนี้

```
pygame.key.set_repeat( delay, interval )
```

โดย delay คือ ค่าหน่วงเวลาในหน่วยมิลลิวินาทีก่อนจะเริ่มการส่งเหตุการณ์ pygame.KEYDOWN หลังจากกดแป้นในครั้งแรก

interval คือ ค่าระยะห่างในหน่วยมิลลิวินาทีในส่งเหตุการณ์ pygame.KEYDOWN ในครั้งต่อ ๆ ไป

ดังนั้น ถ้าต้องการให้เริ่มมีการส่ง pygame.KEYDOWN หลังจากกดค้างไป 1 มิลลิวินาที และหลังจากนั้นถ้ายังกดค้างเอาไว้จะมีการส่งเหตุการณ์ทุก ๆ 10 มิลลิวินาที สามารถเขียนเป็นคำสั่งได้ดังนี้

```
pygame.key.set_repeat( 1, 10 )
```

การเชื่อมประสานกับเมาส์

เมาส์เป็นอุปกรณ์นำเข้าข้อมูลประเภทชี้แล้วกด ด้วยเหตุนี้ในการเชื่อมประสานกับเมาส์จึงเป็นการอ่านค่าตำแหน่งของเมาส์ และสถานะการกดปุ่มของเมาส์ ดังภาพที่ 17.4 ซึ่งการเชื่อมประสานกับเมาส์กระทำได้ 3 ลักษณะ คือ

1. ตรวจสอบเหตุการณ์ที่เกิดกับเมาส์ โดยพิจารณาจากคุณสมบัติ type ซึ่งมี 3 ประเภทคือ

pygame.MOUSEMOTION เมื่อมีการขยับเมาส์

pygame.MOUSEBUTTONUP เมื่อมีการปล่อยปุ่มเมาส์ที่ถูกกด

pygame.MOUSEBUTTONDOWN เมื่อมีการกดที่ปุ่มของเมาส์

2. ตรวจสอบประเภทของปุ่มที่กด ซึ่งจะคืนค่ามาเป็นค่า True หรือ False ของทูเพิล 3 สมาชิกสำหรับแสดงถึงสถานะของการกดเมาส์ปุ่ม 1, 2 หรือ 3 โดยรูปแบบของคำสั่งตรวจสอบเขียนดังนี้

```
(ปุ่ม1, ปุ่ม2, ปุ่ม3) = pygame.mouse.get_pressed()
```

3. ตำแหน่งปัจจุบันของเมาส์ซึ่งคืนค่ากลับมาเป็นทูเพิล 2 สมาชิก สำหรับค่าพิกัดในแกน x และค่าพิกัดในแกน y โดยรูปแบบของคำสั่งอ่านค่าตำแหน่งปัจจุบันของเมาส์เป็นดังนี้

```
(x, y) = pygame.mouse.get_pos()
```

ภาพที่ 17.4 ชื่อเรียกปุ่มเมาส์ซ้าย ขวา และกลาง

คำสั่งสำหรับให้แสดงหรือซ่อนเคอร์เซอร์ของเมาส์เป็นดังนี้

```
pygame.mouse.set_visible( True หรือ False )
```

กรณีที่ต้องการย้ายพิกัดของเมาส์สามารถกระทำโดยใช้คำสั่งย้ายตำแหน่งตามรูปแบบต่อไปนี้

```
pygame.mouse.set_pos( [x, y] )
```

ในบางครั้งที่ผู้เขียนโปรแกรมต้องการเปลี่ยนรูปแบบของเคอร์เซอร์เมาส์ต้องดำเนินการตามขั้นตอนต่อไปนี้

1. สร้างสตริงของเคอร์เซอร์
2. คอมไพล์สตริง ด้วยคำสั่งต่อไปนี้

```
pygame.cursors.compile( ตัวแปรสตริง )
```

3. สั่งเปลี่ยนเคอร์เซอร์ด้วยคำสั่งต่อไปนี้

```
pygame.mouse.set_cursor( (คอลัมน์,แถว),(0,0),*ตัวแปรที่ได้จาก2)
```

ตัวอย่างโปรแกรม 17.3 เป็นตัวอย่างการเชื่อมประสานกับเมาส์โดยเปลี่ยนรูปแบบของเมาส์เมื่อมีการคลิกที่เมาส์ และกลับไปเป็นอีกรูปแบบหนึ่งเมื่อปล่อยการคลิกเมาส์

โปรแกรม 17.3 โปรแกรมเปลี่ยนรูปแบบของเคอร์เซอร์เมื่อมีการกด/ปล่อยเมาส์

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`arrow1str = (  #sized 24x24`
4	`  "XX                      ",`
5	`  "XXX                     ",`
6	`  "XXXX                    ",`
7	`  "XX.XX                   ",`
8	`  "XX..XX                  ",`
9	`  "XX…XX                 ",`
10	`  "XX….XX                ",`
11	`  "XX…..XX               ",`
12	`  "XX…...XX              ",`
13	`  "XX.......XX             ",`
14	`  "XX........XX            ",`
15	`  "XX........XXX           ",`
16	`  "XX......XXXXX           ",`
17	`  "XX.XXX..XX              ",`
18	`  "XXXX XX..XX             ",`
19	`  "XX   XX..XX             ",`
20	`  "     XX..XX             ",`
21	`  "      XX..XX            ",`
22	`  "      XX..XX            ",`
23	`  "       XXXX             ",`
24	`  "       XX               ",`
25	`  "                        ",`
26	`  "                        ",`
27	`  "                        ")`
	→

```
arrow2str = (
  "XX                      ",
  "XXX                     ",
  "XXXX                    ",
  "XXXXX                   ",
  "XXXXXX                  ",
  "XXXXXXX                 ",
  "XXXXXXXX                ",
  "XXXXXXXXX               ",
  "XXXXXXXXXX              ",
  "XXXXXXXXXXX             ",
  "XXXXXXXXXXXX            ",
  "XXXXXXXXXXXXX           ",
  "XXXXXXXXXXXXXX          ",
  "XXXXXXXXXXXXXX          ",
  "XXXXXXXXXX              ",
  "XXXX XXXXXX             ",
  "XX   XXXXXX             ",
  "     XXXXXX             ",
  "      XXXXXX            ",
  "      XXXXXX            ",
  "       XXXX             ",
  "       XX               ",
  "                        ",
  "                        ",
  "                        ")
pygame.init()
screen = pygame.display.set_mode((800,600))
pygame.display.set_caption("Ex17.3")
background = pygame.Color(100, 149, 237)
cursor1 = pygame.cursors.compile(arrow1str,black='X',white='.',xor='o')
cursor2 = pygame.cursors.compile(arrow2str,black='X',white='.',xor='o')
pygame.mouse.set_cursor((24,24),(0,0),*cursor1)
running = True
while running:
    screen.fill(background)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEMOTION:
            (x,y) = pygame.mouse.get_pos()
            print("(",x,",",y,")")
        if event.type == pygame.MOUSEBUTTONDOWN:
            pygame.mouse.set_cursor((24,24),(0,0),*cursor2)
            (button1, button2, button3) = pygame.mouse.get_pressed()
            print("You pressed :: Left=",button1," Middle=",
button2," Right=",button3)
        if event.type == pygame.MOUSEBUTTONUP:
            pygame.mouse.set_cursor((24,24),(0,0),*cursor1)
            print("Mouse Button Released")

    pygame.display.flip()
pygame.quit()
```

การติดต่อกับเกมคอนโทรลเลอร์

เกมคอนโทรลเลอร์ (Game Controller) หรือเกมแพด (Game Pad) เป็นอุปกรณ์ที่ออกแบบเพื่อใช้สำหรับเล่นเกมคอนโซล (Game Console) หรือเครื่องเล่นเกม (ภาพที่ 17.5) บนแป้นของเกมคอนโทรลเลอร์ประกอบด้วยปุ่มทำงานที่ทำหน้าที่ต่าง ๆ เช่น ปุ่มเลื่อนซ้าย ปุ่มเลื่อนขวา ปุ่มเลื่อนขึ้น ปุ่มเลื่อนลง ปุ่มสามเหลี่ยม ปุ่มสี่เหลี่ยม ปุ่มวงกลม ปุ่มกากบาท ปุ่มเลือก ปุ่มเริ่ม คันโยก ปุ่มด้านหน้าซ้ายลย ปุ่มด้านหน้าขวาบน ปุ่มด้านหน้าซ้ายล่าง และปุ่มด้านหน้าขวาล่าง

เป็นต้น ซึ่งเกมคอนโทรลเลอร์แต่ละรุ่นมีการออกแบบที่ไม่เหมือนกัน ทำให้การเขียนโปรแกรมเชื่อมต่อต้องให้ผู้ใช้ตั้งค่าชื่อเรียกของแต่ละปุ่มตรงกับโปรแกรมที่พัฒนาจึงจะทำงานได้ถูกต้อง

คำสั่งสำหรับการนับจำนวนเกมคอนโทรลเลอร์ที่เชื่อมต่อกับคอมพิวเตอร์มีรูปแบบการใช้งานดังนี้

```
จำนวน = pygame.joystick.get_count( )
```

ภาพที่ 17.5 เกมคอนโทรลเลอร์

หลังจากที่ทราบจำนวนเกมคอนโทรลเลอร์ที่ติดตั้ง ผู้เขียนโปรแกรมต้องสั่งให้อุปกรณ์เริ่มทำงานโดยเกมคอนโทรลเลอร์แต่ละตัวจะต้องสั่งดังนี้

```
ตัวแปร = pygame.joystick.Joystick( ลำดับ )
ตัวแปร.init()
```

คำสั่งสำหรับอ่านชื่อผู้ผลิตเกมคอนโทรลเลอร์มีรูปแบบการใช้งานดังนี้

```
ชื่อผู้ผลิต = ตัวแปร.get_name()
```

การเชื่อมต่อกับปุ่มลูกศรซ้าย ขวา บน และล่าง หรือการใช้คันโยกบังคับทิศทางจะต้องอ่านทิศทางของการเคลื่อนที่ในแกนนอนและตั้งโดยใช้รูปแบบของคำสั่งดังนี้

```
ทิศทางแนวนอน = int(ตัวแปร.get_axis(0))*10
ทิศทางแนวตั้ง = int(ตัวแปร.get_axis(1))*10
```

การพิจารณาว่าผู้ใช้กดทิศทางเคลื่อนที่ไปยังทิศทางใด สามารถเขียนเป็นเงื่อนไขตรวจสอบได้ดังตารางที่ 17.1

คำสั่งสำหรับนับจำนวนปุ่มของเกมคอรโทรลเลอร์แต่ละตัว พร้อมทั้งตรวจสอบการกดปุ่มสามารถกระทำโดยใช้คำสั่งนับและตรวจสอบสถานะ ดังนี้

```
จำนวนปุ่ม = ตัวแปร.get_numbuttons()
สถานะการกด = ตัวแปร.get_button( หมายเลขปุ่ม )
```

เหตุการณ์ที่เกิดขึ้นจากเกมคอนโทรลเลอร์ มี 2 ประเภทคือ

pygame.JOYBUTTONDOWN

และ

pygame.JOYBUTTONUP

ตารางที่ 17.1 ความสัมพันธ์ของทิศทางในแนวนอน/แนวตั้งกับทิศทางของปุ่มลูกศร

ทิศทางแนวนอน	ทิศทางแนวตั้ง	ทิศทางของปุ่มลูกศร
<0	=0	ซ้าย
=0	>0	ลง
>0	=0	ขวา
=0	<0	บน
<0	<0	ซ้าย+บน
<0	>0	ซ้าย+ล่าง
>0	<0	ขวา+บน
>0	>0	ขวา+ล่าง

ตัวอย่างโปรแกรมที่ 17.5 เป็นตัวอย่างการอ่านและแสดงผลการทำงานของเกมคอนโทรลเลอร์ตัวที่ 1 (ลำดับที่ 0) บนจอแสดงผลแบบคอนโซล แต่ถ้าโปรแกรมตรวจสอบไม่พบการติดตั้งเกมคอนโทรลเลอร์จะออกจากโปรแกรม

โปรแกรม 17.5 โปรแกรมแสดงสถานะการกดแป้นบนเกมคอนโทรลเลอร์

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`import sys`
4	`def gcShow(gc):`
5	`    gcButtonNames =  ("Tri","Circlr","Cross","Square", "LB", "RB", "LT", "RT", "Select","Start", "LPush", "RPush")`
6	`    tName = "Name : "+gc.get_name()`
7	`    tAxis = "Axes : "+str(gc.get_numaxes())`
8	`    tButton = "Buttons : "+str(gc.get_numbuttons())`
9	`    for i in range(gc.get_numbuttons()):`
10	`        if (gc.get_button(i)):`
11	`            tPressed += gcButtonNames[i]`
12	`            tPressed += " "`
13	`    hori = int(gamer1.get_axis(0)*10)`
14	`    vert = int(gamer1.get_axis(1)*10)`
15	`    text1 = "(" + str(hori) + "," + str(vert) + ")"`
16	`    if hori < 0 and vert == 0:`
17	`        text2 = "Left"`
18	`    elif hori == 0 and vert > 0:`
19	`        text2 = "Down"`
20	`    elif hori > 0 and vert == 0:`
21	`        text2 = "Right"`
22	`    elif hori == 0 and vert < 0:`
23	`        text2 = "Up"`
24	`    elif hori < 0 and vert < 0:`
25	`        text2 = "Left+Up"`
26	`    elif hori < 0 and vert > 0:`
27	`        text2 = "Left+Down"`
	→

```
28          elif hori > 0 and vert < 0:
29              text2 = "Right+Up"
30          elif hori > 0 and vert > 0:
31              text2 = "Right+Down"
32          else:
33              text2 = "Center"
34          print(text1)
35          print(text2)
36          print(tName)
37          print(tAxis)
38          print(tButton)
39      pygame.init()
40      gcCnt = pygame.joystick.get_count()
41      if gcCnt == 0:
42          print("Error.....")
43          sys.exit()
44      fpsClock = pygame.time.Clock()
45      screen = pygame.display.set_mode((800,600))
46      pygame.display.set_caption("Ex17.5")
47      gamer1 = pygame.joystick.Joystick(0)
48      gamer1.init()
49      background = pygame.Color( 0, 0, 0)
50      running = True
51      while running:
52          fpsClock.tick(30)
53          screen.fill(background)
54          gcShow( gamer1 )
55          for event in pygame.event.get():
56              if event.type == pygame.QUIT:
57                  running = False
58              if event.type == pygame.JOYBUTTONDOWN:
59                  gcShow(gamer1 )
60              if event.type == pygame.JOYBUTTONUP:
61                  gcShow(gamer1 )
62          pygame.display.flip()
63      pygame.quit()
```

การใช้ตัวตั้งเวลา

ตัวตั้งเวลาใน PyGame เป็นส่วนของการทำงานเพื่อให้มีการอัพเดตหน้าจอตามอัตราการแสดงภาพ (frame rate) ต่อ 1 วินาที หรือที่เรียกว่า fps (frames per second) ซึ่งมีขั้นตอนการเขียนโค้ด 2 ส่วน คือ

1. สร้างตัวแปรตัวตั้งเวลา โดยใช้คำสั่งในรูปแบบต่อไปนี้

```
ตัวแปรตัวตั้งเวลา = pygame.time.clock()
```

2. กำหนดอัตราการแสดงผล โดยใช้คำสั่งตามรูปแบบด้านล่างนี้

```
ตัวแปรตัวตั้งเวลา.tick( อัตราการอัพเดตหน้าจอ )
```

การสั่งให้โปรแกรมรอการทำงานเพื่อให้หน่วยประมวลผลทำงานอื่นเป็นระยะในหน่วยมิลลิวินาทีสามารถใช้งานได้ 2 คำสั่ง คือ wait( ) และ delay( ) ดังรูปแบบต่อไปนี้ โดยความแตกต่างระว่าง 2 คำสั่งนี้คือ delay( ) เป็นคำสั่งที่ความแม่นยำในการรอสูงกว่าคำสั่ง wait( ) และทั้ง 2 คำสั่งคืนค่ากลับมาเป็นค่าเวลาในหน่วยมิลลิวินาทีที่เกิดการรอ

```
pygame.time.wait( จำนวนเวลาที่ต้องการให้รอในหน่วยมิลลิวินาที )
pygame.time.delay( จำนวนเวลาที่ต้องการให้รอในหน่วยมิลลิวินาที )
```

ตัวอย่างโปรแกรมที่ 17.6 เป็นโปรแกรมสลับพื้นหลังทุก 1 วินาทีจากสีดำเป็นสีขาวสลับกันไป โดยใช้เทคของการนับจำนวนครั้งการอัพเดตหน้าจอ ซึ่งกำหนดอัตราการอัพเดตหน้าจอไว้ที่ 30 ครั้งต่อ 1 วินาที และทุกครั้งที่มีการอัพเดตได้นับจำนวนครั้งการอัพเดต เมื่อไรครบ 30 ครั้งจะเปลี่ยนสถานะของตัวแปร page ให้สลับไปมาระหว่างค่า 0 หรือ 1 เพื่อนำค่า page ใช้สำหรับเป็นเงื่อนไขของการเลือกสีพื้นหลังของจอแสดงผล

โปรแกรม 17.6 โปรแกรมสลับสีพื้นหลังทุก 1 วินาทีโดยอาศัยการนับจำนวนครั้งของการอัพเดตหน้าจอ

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`pygame.init()`
4	`fpsClock = pygame.time.Clock()`
5	`screen = pygame.display.set_mode((800,600))`
6	`pygame.display.set_caption("Ex17.6")`
7	`bg1 = pygame.Color(0,0,0)`
8	`bg2 = pygame.Color(255,255,255)`
9	`cnt = 0`
10	`page = 0`
11	`running = True`
12	`while running:`
13	`    fpsClock.tick(30)`
14	`    if page == 0:`
15	`        screen.fill(bg1)`
16	`    else:`
17	`        screen.fill(bg2)`
18	`    for event in pygame.event.get():`
19	`        if event.type == pygame.QUIT:`
20	`            running = False`
21	`    cnt = cnt+1`
22	`    if cnt == 30:`
23	`        cnt = 0`
24	`        if page == 0:`
25	`            page = 1`
26	`        else:`
27	`            page = 0`
28	`    pygame.display.flip()`
29	`pygame.quit()`

การแสดงรูปภาพ

รูปแบบไฟล์ภาพที่ PyGame รองรับได้แก่ JPG PNG GIFแบบ(non-animated) BMP PCX TGA (แบบ uncompressed) TIF LBM (แบบ and PBM) PBM (แบบ and PGM, PPM) และ XPM คำสั่งสำหรับโหลดรูปภาพเพื่อเก็บในหน่วยความจำที่ถูกอ้างอิงด้วยตัวแปรที่กำหนดสามารถเขียนเป็นชุดคำสั่งได้ดังรูปแบบต่อไปนี้

```
ตัวแปรเก็บภาพ = pygame.image.load("ชื่อภาพ")
```

การนำตัวแปรเก็บภาพไปแสดงผลบนจอแสดงผลมีวิธีการใช้งานคำสั่ง 2 รูปแบบ คือ แสดงทั้งภาพที่ตำแหน่ง (x,y) กับแสดงภาพที่ตำแหน่ง (sx, sy) โดยนำข้อมูลภาพเฉพาะตั้งแต่ตำแหน่ง (x,y) ถึง (x+w, y+h)

```
ตัวแปรรหน้าต่าง.blit(ตัวแปรเก็บภาพ,  (x, y))
ตัวแปรรหน้าต่าง.blit(ตัวแปรเก็บภาพ, (sx,sy),  (x, y, w, h))
```

กรณีที่ต้องการบันทึกภาพหน้าจอแสดงผลลงไฟล์ภาพสามารถเรียกใช้ชุดคำสั่งตามรูปแบบการใช้งานดังนี้

```
pygame.image.save( ตัวแปร, "ชื่อแฟ้มภาพ" )
```

และเมื่อต้องการนำข้อมูลภาพไปใช้สำหรับส่งให้การ์ดแสดงผลผ่านทางไลบรารี OpenGL ซึ่งมีรูปแบบการจัดวางข้อมูลภาพที่ไม่เหมือนกับที่ PyGame จัดเก็บ ผู้เขียนโปรแกรมต้องทำการแปลงประเภทข้อมูลโดยใช้คำสั่งต่อไปนี้

```
ตัวปรสตริงบัฟเฟอร์ = pygame.image.tostring( ตัวแปร, รูปแบบ, flipped=False )
```

โดยที่ รูปแบบ เป็นตัวระบุประเภทของข้อมูลในหน่วยความคำ ซึ่งมีค่าเป็นรูปแบบใดรูปแบบหนึ่งดังนี้ P, RGB, RGBX, RGBA, ARGB, RGBA_PRENULT หรือ ARGB_PREMULT

ตัวอย่างโปรแกรมที่ 3.7 เป็นการสุ่มแสดงภาพรูปกำปั้น แบมือ และ 2 นิ้วดังภาพที่ 3.6 สลับกันไปทุก 2 วินาที ซึ่งขั้นตอนของการสุ่มค่าในภาษาไพธอนต้องเรียกใช้ไลบรารี random และมีคำสั่งสำหรับใช้งานทั่วไป 3 คำสั่ง คือ

1. ตั้งค่าเริ่มการสุ่ม เพื่อให้การสุ่มครั้งแรกไม่เหมือนกัน

```
random.seed(  )
```

2. สุ่มเป็นช่วง ค่าเริ่ม (หรือ 0) แต่ไม่เกิน ค่าสุดท้าย

```
ค่าที่สุ่มได้ = random.randrange( ค่าสุดท้าย )
ค่าที่สุ่มได้ = random.randrange( ค่าเริ่ม, ค่าสุดท้าย [, step] )
```

3. สุ่มเป็นช่วง a ถึง b

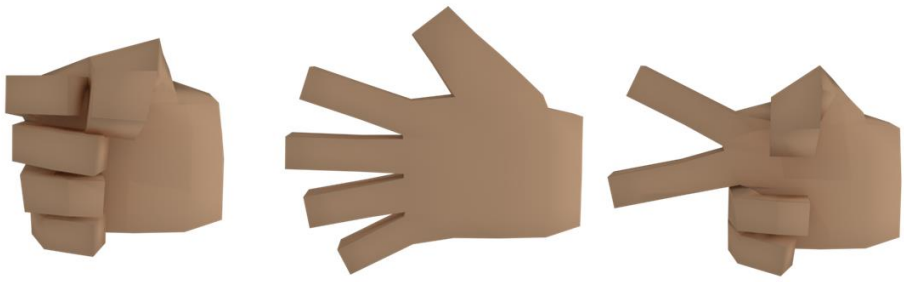
```
ค่าที่สุ่มได้ = random.randint( a, b )
```

สำหรับการสุ่มค่าที่เป็นค่าทศนิยมที่ให้ผลลัพธ์อยู่ในช่วงค่า [0.0 , 1.0) มีคำสั่งใช้งาน 5 คำสั่ง พื้นฐานดังนี้

```
ค่าที่สุ่มได้ = random.random( )  # [0.0 , 1.0)
ค่าที่สุ่มได้ = random.uniform( a, b )
ค่าที่สุ่มได้ = random.betavariate( alpha, beta )
ค่าที่สุ่มได้ = random.gauss( mu, sigma )
ค่าที่สุ่มได้ = random.normalvariate( mu, sigma )
```

ขั้นตอนการทำงานของโปรแกรมที่ 17.7 เป็นดังนี้

1. เปิดโหมด
2. โหลดภาพ
3. เกมลูป
    - 3.1 รับค่า ถ้ากดปิด หรือ ESC ให้จบโปรแกรม
    - 3.2 สุ่มค่า 1..3 กรณีเริ่มใหม่ (showStatus == False)
    - 3.3 ถ้าเป็น 1 แสดงกระดาษ, 2 แสดงค้อน, 3 แสดงกรรไกร
    - 3.4 แสดงผล
    - 3.5 นับเวลาการอัพเดต ถ้าครบ 60 ครั้ง (2 วินาที) ให้เริ่มใหม่



ภาพที่ 17.6 ภาพกำปั้น แบมือ และ 2 นิ้วสำหรับตัวอย่างโปรแกรมที่ 3.7

โปรแกรมที่ 17.7 แสดงภาพ 2 วินาทีสลับวนกันไป

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`import random`
4	`pygame.init()`
5	`fpsClock = pygame.time.Clock()`
6	`screen = pygame.display.set_mode((800,600))`
7	`pygame.display.set_caption("Ex17.7")`
8	`background = pygame.Color( 0, 0, 0)`
9	`imgPaper = pygame.image.load("res/paper.png")`
	→

```
10      imgHammer = pygame.image.load("res/hammer.png")
11      imgScissors = pygame.image.load("res/scissors.png")
12      counter = 0
13      showStatus = False
14      random.seed( )
15      running = True
16      while running:
17          fpsClock.tick(30)
18          screen.fill(background)
19          for event in pygame.event.get():
20              if event.type == pygame.QUIT:
21                  pygame.quit()
22                  sys.exit()
23              if event.type == pygame.KEYDOWN:
24                  if event.key == pygame.K_ESCAPE:
25                      running = False
26          if showStatus == False:
27              showStatus = True
28              imgRandom = random.randint(1,3)
29          if imgRandom == 1:
30              screen.blit(imgPaper,(10,10))
31          elif imgRandom == 2:
32              screen.blit(imgHammer,(10,10))
33          else:
34              screen.blit(imgScissors,(10,10))
35          counter = counter+1
36          if counter == 60:
37              counter = 0
38              showStatus = False
39          pygame.display.flip()
40      pygame.quit()
```

ตัวอย่างโปรแกรมที่ 17.8 เป็นตัวอย่างประยุกต์การแสดงภาพของเคอร์เซอร์ โดยขั้นตอนการทำงานของโปรแกรมเป็นดังนี้

1. เปิดโหมด
2. ปิดการแสดงเคอร์เซอร์ โหลดภาพที่ใช้เป็นเคอร์เซอร์
3. เกมลูป
    - 3.1 ลบฉากหลัง แสดงภาพเคอร์เซอร์
    - 3.2 รับค่า ถ้ากดปิด ให้จบโปรแกรม
    - 3.3 อ่านค่าตำแหน่งเมาส์เพื่อใช้เป็นตำแหน่งของเคอร์เซอร์
    - 3.4 อัพเดตหน้าจอ

โปรแกรมที่ 17.8 การนำภาพมาแสดงเป็นเคอร์เซอร์

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`pygame.init()`
4	`fpsClock = pygame.time.Clock()`
5	`screen = pygame.display.set_mode((800,600))`
6	`pygame.display.set_caption("Ex17.8")`
7	`background = pygame.Color(0,0,0)`
8	`cursor = pygame.image.load('res/cursor.png').convert()`
9	`cursorRect = cursor.get_rect()`

→

```
10  surfaceRect = screen.get_rect()
11  pygame.mouse.set_visible(False)
12  running = True
13  while running:
14      fpsClock.tick(30)
15      screen.fill(background)
16      screen.blit(cursor, cursorRect)
17      for event in pygame.event.get():
18          if event.type == pygame.QUIT:
19              running = False
20      cursorRect.x, cursorRect.y = pygame.mouse.get_pos()
21      pygame.display.flip()
22  pygame.quit()
```

ตัวอย่างโปรแกรมที่ 17.9 เป็นตัวอย่างการแสดงภาพเคลื่อนไหวจากไฟล์ภาพที่จัดเก็บรูปภาพแบบไทล์เบส หรือนำภาพเล็ก ๆ หลายภาพมาต่อกันเป็นภาพใหญ่ ดังเช่นภาพที่ 17.7 (โดยการรวมภาพนั้นเป็นดังตัวอย่างโปรแกรมที่ 17.10) หลังจากนั้นรับการสั่งงานจากผู้ใช้ตามเงื่อนไขต่อไปนี้

กด 1 แสดงเก้าอี้

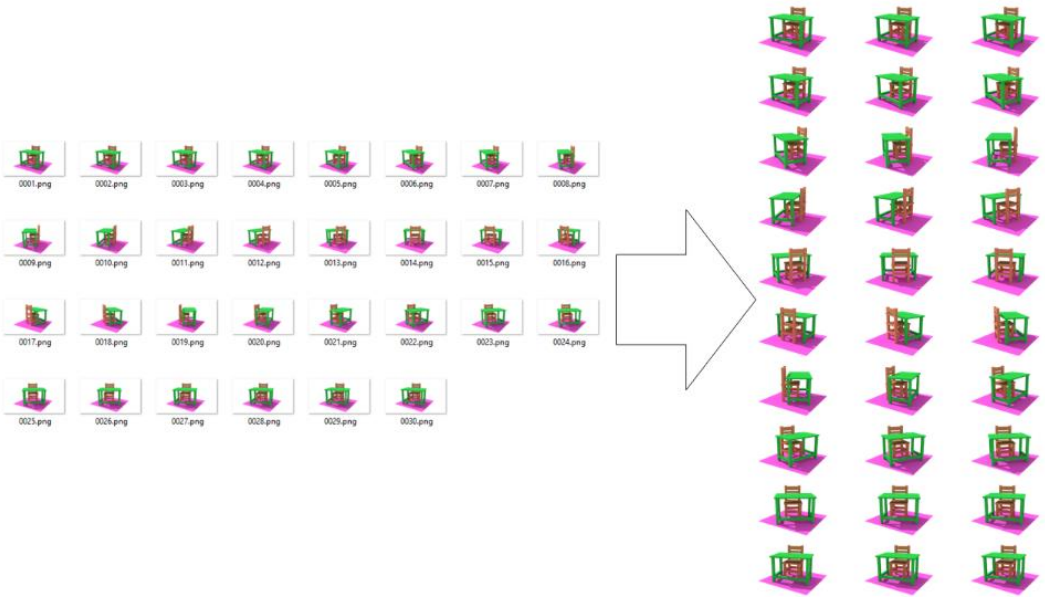
กด 2 แสดงโต๊ะ

กด 3 แสดงเก้าอี้พร้อมกับโต๊ะ

กด ลูกศรซ้าย หมุนทวนเข็มนาฬิกา

กด ลูกศรขวา หมุนตามเข็มนาฬิกา

กด ESC ออกจากโปรแกรม



ภาพที่ 17.7 การนำภาพเล็กหลายภาพรวมเป็นภาพใหญ่ 1 ภาพ

โปรแกรมที่ 17.9 การแสดงภาพเก้าอี้ โต๊ะ และเก้าอี้+โต๊ะหมุนบนจอภาพ

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`pygame.init()`
4	`fpsClock = pygame.time.Clock()`
5	`screen = pygame.display.set_mode((800,600))`
6	`pygame.display.set_caption("Ex17.9")`
7	`background = pygame.Color(0,0,0)`
8	`cursor = pygame.image.load('res/cursor.png').convert()`
9	`cursorRect = cursor.get_rect()`
10	`surfaceRect = screen.get_rect()`
11	`pygame.mouse.set_visible(False)`
12	`running = True`
13	`while running:`
14	`    fpsClock.tick(30)`
15	`    screen.fill(background)`
16	`    screen.blit(cursor, cursorRect)`
17	`    for event in pygame.event.get():`
18	`        if event.type == pygame.QUIT:`
19	`            running = False`
20	`    cursorRect.x, cursorRect.y = pygame.mouse.get_pos()`
21	`    pygame.display.flip()`
22	`pygame.quit()`

โปรแกรมที่ 17.10 โปรแกรมรวมไฟล์ภาพเก้าอี้หลายภาพเป็นภาพเดียว

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`w = 1440`
4	`h = 2700`
5	`pygame.init()`
6	`screen = pygame.Surface((w,h), pygame.SRCALPHA)`
7	`pygame.display.set_caption("Ex17.10")`
8	`img1 = pygame.image.load("chair/0001.png")`
9	`img2 = pygame.image.load("chair/0002.png")`
10	`img3 = pygame.image.load("chair/0003.png")`
11	`img4 = pygame.image.load("chair/0004.png")`
12	`img5 = pygame.image.load("chair/0005.png")`
13	`img6 = pygame.image.load("chair/0006.png")`
14	`img7 = pygame.image.load("chair/0007.png")`
15	`img8 = pygame.image.load("chair/0008.png")`
16	`img9 = pygame.image.load("chair/0009.png")`
17	`img10 = pygame.image.load("chair/0010.png")`
18	`img11 = pygame.image.load("chair/0011.png")`
19	`img12 = pygame.image.load("chair/0012.png")`
20	`img13 = pygame.image.load("chair/0013.png")`
21	`img14 = pygame.image.load("chair/0014.png")`
22	`img15 = pygame.image.load("chair/0015.png")`
23	`img16 = pygame.image.load("chair/0016.png")`
24	`img17 = pygame.image.load("chair/0017.png")`
25	`img18 = pygame.image.load("chair/0018.png")`
26	`img19 = pygame.image.load("chair/0019.png")`
27	`img20 = pygame.image.load("chair/0020.png")`
28	`img21 = pygame.image.load("chair/0021.png")`
29	`img22 = pygame.image.load("chair/0022.png")`
30	`img23 = pygame.image.load("chair/0023.png")`
31	`img24 = pygame.image.load("chair/0024.png")`
32	`img25 = pygame.image.load("chair/0025.png")`
33	`img26 = pygame.image.load("chair/0026.png")`
34	`img27 = pygame.image.load("chair/0027.png")`
35	`img28 = pygame.image.load("chair/0028.png")`
36	`img29 = pygame.image.load("chair/0029.png")`
37	`img30 = pygame.image.load("chair/0030.png")`

→

```
38      surface.blit(img1,(0,0))
39      surface.blit(img2,(480,0))
40      surface.blit(img3,(960,0))
41      surface.blit(img4,(0,270))
42      surface.blit(img5,(480,270))
43      surface.blit(img6,(960,270))
44      surface.blit(img7,(0,540))
45      surface.blit(img8,(480,540))
46      surface.blit(img9,(960,540))
47      surface.blit(img10,(0,810))
48      surface.blit(img11,(480,810))
49      surface.blit(img12,(960,810))
50      surface.blit(img13,(0,1080))
51      surface.blit(img14,(480,1080))
52      surface.blit(img15,(960,1080))
53      surface.blit(img16,(0,1350))
54      surface.blit(img17,(480,1350))
55      surface.blit(img18,(960,1350))
56      surface.blit(img19,(0,1620))
57      surface.blit(img20,(480,1620))
58      surface.blit(img21,(960,1620))
59      surface.blit(img22,(0,1890))
60      surface.blit(img23,(480,1890))
61      surface.blit(img24,(960,1890))
62      surface.blit(img25,(0,2160))
63      surface.blit(img26,(480,2160))
64      surface.blit(img27,(960,2160))
65      surface.blit(img28,(0,2430))
66      surface.blit(img29,(480,2430))
67      surface.blit(img30,(960,2430))
68      pygame.image.save(surface,"chair.png")
69      pygame.quit()
```

การแสดงตัวอักษร

การแสดงตัวอักษรโดยใช้ไลบรารี PyGame เป็นการเรนเดอร์ตัวอักษรประเภททรูไทป์ (Truetype Font) รองรับตัวอักษรรหัสยูนิโค้ด UCS-2 ในช่วงค่า ‘u0001’ ถึง ‘uFFFF’ ซึ่ง PyGame ทำงานบนไลบรารี SDL_ttf อีกทอดหนึ่ง การเรียกใช้กระผ่านทาง pygame.font โดยต้องให้ระบบไลบรารีฟอนต์ทำงานตามคำสั่งต่อไปนี้

```
pygame.font.init()
```

เมื่อเลิกใช้ต้องเรียกคำสั่งต่อไปนี้เพื่อถอนไลบรารีฟอนต์ออกจากหน่วยความจำ

```
pygame.font.quit()
```

การสร้างวัตถุประเภทฟอนต์ก่อนใช้เสมอ ดังรูปแบบของคำสั่งดังนี้

```
ตัวแปรฟอนต์ = pygame.font.Font(ชื่อไฟล์ฟอนต์, ขนาด)
```

กรณีที่ต้องการเรียกใช้ฟอนต์ของระบบให้ใช้ฟังก์ชั่นดังต่อไปนี้แทน pygame.font.Font()

```
ตัวแปรฟอนต์ = pygame.font.SysFont(ชื่อฟอนต์, ขนาด, bold=False, italic=False)
```

การเรนเดอร์ตัวอักษรจะต้องส่งสตริงของข้อความที่ต้องการเรนเดอร์ แฟล็กของการเรนเดอร์เพื่อเกลี่ยสี (antialias) และสีของตัวอักษรดังรูปแบบต่อไปนี้

ตัวแปรพื้นผิว = ตัวแปรฟอนต์.render(ข้อความ, antialias, (r,g,b))

หลังจากได้ตัวแปรพื้นผิวที่เก็บข้อความที่เรนเดอร์เสร็จแล้ว ต้องนำไป blit ลงจอแสดงผลต่อไป

ตัวอย่างโปรแกรมที่ 17.11 เป็นตัวอย่างแสดงข้อความ “Hello, World! สวัสดีชาวโลก” สีขาวที่กลางจอแสดงผล โดยเลือกใช้ฟอนต์ Tahoma ขนาด 36pt

โปรแกรมที่ 17.11 แสดงข้อความที่กลางจอภาพ

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`pygame.init()`
4	`pygame.font.init()`
5	`fpsClock = pygame.time.Clock()`
6	`screen = pygame.display.set_mode((800,600))`
7	`pygame.display.set_caption("Ex17.11")`
8	`background = pygame.Color( 0, 0, 0)`
9	`font = pygame.font.SysFont("Tahoma", 36)`
10	`text = font.render("Hello, World! สวัสดีชาวโลก", True, (255, 255, 255))`
11	`tx = 400-text.get_width()//2`
12	`ty = 300-text.get_height()//2`
13	`running = True`
14	`while running:`
15	`    fpsClock.tick(30)`
16	`    screen.fill(background)`
17	`    screen.blit(text, (tx, ty))`
18	`    for event in pygame.event.get():`
19	`        if event.type == pygame.QUIT:`
20	`            running = False`
21	`    pygame.display.flip()`
22	`pygame.font.quit()`
23	`pygame.quit()`

ตัวอย่างโปรแกรมที่ 17.12 เป็นตัวอย่างที่นำการแสดงสถานะของการกดแป้นบนเกมคอนโทรลเลอร์ให้แดงผลเป็นตัวอักษรบนหน้าจอแทนการแสดงผลในหน้าจอคอนโซล

โปรแกรมที่ 17.12 การนำภาพมาแสดงเป็นเคอร์เซอร์

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`import sys`
4	`def gcShow(gc, font, surface):`
5	`    gcButtonNames =  ("Tri","Circlr","Cross","Square", "LB", "RB", "LT", "RT", "Select","Start", "LPush", "RPush")`
6	`    tName = font.render("Name : "+gc.get_name(),True,(255,255,255))`
7	`    tAxis = font.render("Axes : "+str(gc.get_numaxes()),True,(255,255,255))`
8	`    tButton =  font.render("Buttons :  "+str(gc.get_numbuttons()),True, (255,255,255))`

→

```
9           tPressed = ""
10          for i in range(gc.get_numbuttons()):
11              if (gc.get_button(i)):
12                  tPressed += gcButtonNames[i]
13                  tPressed += " "
14          tPress = font.render(tPressed,True,(128,128,128))
15          hori = int(gamer1.get_axis(0)*10)
16          vert = int(gamer1.get_axis(1)*10)
17          msg = "(" + str(hori) + "," + str(vert) + ")"
18          text1 = font.render(msg, True, (255,255,0))
19          if hori < 0 and vert == 0:
20              text2 = font.render("Left",True,(0,255,0))
21          elif hori == 0 and vert > 0:
22              text2 = font.render("Down",True,(0,255,0))
23          elif hori > 0 and vert == 0:
24              text2 = font.render("Right",True,(0,255,0))
25          elif hori == 0 and vert < 0:
26              text2 = font.render("Up",True,(0,255,0))
27          elif hori < 0 and vert < 0:
28              text2 = font.render("Left+Up",True,(0,128,255))
29          elif hori < 0 and vert > 0:
30              text2 = font.render("Left+Down",True,(0,128,255))
31          elif hori > 0 and vert < 0:
32              text2 = font.render("Right+Up",True,(0,128,255))
33          elif hori > 0 and vert > 0:
34              text2 = font.render("Right+Down",True,(0,128,255))
35          else:
36              text2 = font.render("Center", True, (255,0,255))
37          tw1, th1 = text1.get_size()
38          tw2, th2 = text2.get_size()
39          sw, sh = surface.get_size()
40          tx1 = (sw-tw1)/2
41          ty1 = (sh-th1)/2
42          tx2 = (sw-tw2)/2
43          ty2 = (sh-th2)/2
44          surface.blit(text1,(tx1,ty1))
45          surface.blit(text2,(tx2,ty2-th2))
46          surface.blit(tName,(0,0))
47          surface.blit(tAxis,(0,th2))
48          surface.blit(tButton,(0,th2*2))
49          surface.blit(tPress,(0,th2*3))
50      pygame.init()
51      pygame.font.init()
52      gcCnt = pygame.joystick.get_count()
53      if gcCnt == 0:
54          print("Error.....")
55          sys.exit()
56      fpsClock = pygame.time.Clock()
57      screen = pygame.display.set_mode((800,600))
58      pygame.display.set_caption("Ex17.12")
59      gamer1 = pygame.joystick.Joystick(0)
60      gamer1.init()
61      background = pygame.Color( 0, 0, 0)
62      font = pygame.font.SysFont("Tahoma", 40)
63      running = True
64      while running:
65          fpsClock.tick(30)
66          screen.fill(background)
67          gcShow( gamer1, font, screen )
68          for event in pygame.event.get():
69              if event.type == pygame.QUIT:
70                  running = False
71              if event.type == pygame.JOYBUTTONDOWN:
72                  gcShow(gamer1, font, screen)
73              if event.type == pygame.JOYBUTTONUP:
74                  gcShow(gamer1, font, screen)
75          pygame.display.flip()
76      pygame.font.quit()
77      pygame.quit()
```

เสียงดนตรี

การเล่นเสียงดนตรีด้วยไลบรารี PyGame สามารถกระทำผ่านทาง pygame.mixer ซึ่งโดยปกติรองรับรูปแบบไฟล์เสียงชนิด wav mp3 และ ogg (ในบางระบบปฏิบัติการอาจจะเล่นได้เฉพาะไฟล์ wav ถ้าติดตั้งไลบรารีเสียงไม่ครบ) และกำหนดการเล่นเพื่อให้เล่นจนจบ 1 รอบหรือเล่นวนซ้ำต่อไปเรื่อย ๆ จนกว่าจะสั่งยกเลิกการเล่นเสียงดนตรี โดยผู้เขียนโปรแกรมต้องสั่งให้ตัวเล่นเสียงดนตรีทำงานด้วยการสั่งคำสั่งดังต่อไปนี้

```
pygame.mixer.init()
pygame.mixer.pre_init( 44100, -16, 2, 2048 )
```

การเปิดไฟล์เสียงเข้าสู่ตัวเล่นเสียงดนตรีมีรูปแบบการสั่งงานดังนี้

```
pygame.mixer.music.load("ไฟล์เสียง")
```

การสั่งเล่นไฟล์เสียงที่โหลดเข้าตัวเล่นเสียงดนตรีมีรูปแบบการใช้งานดังนี้

```
pygame.mixer.music.play( )
หรือ กรณีที่ต้องการเล่นแบบวนซ้ำ
pygame.mixer.music.play(-1)
```

คำสั่งสำหรับหยุดเล่นไฟล์เสียงมีรูปแบบการใช้งานดังนี้

```
pygame.mixer.music.stop()
```

สำหรับการสั่งหยุดเล่นเสียงดนตรีชั่วขณะและให้เล่นต่อจากที่สั่งหยุดไว้สามารถทำได้ด้วยคำสั่งต่อไปนี้

```
pygame.mixer.music.pause()
pygame.mixer.music.unpause()
```

กรณีที่ต้องการอ่านค่าระดับความดังของเสียงดนตรีและปรับเปลี่ยนระดับความดังให้ใช้คำสั่งตามรูปแบบต่อไปนี้

```
ค่าความดัง = pygame.mixer.music.get_volume()
pygame.mixer.music.set_volume( ระดับความดังที่ต้องการ )
```

การสั่งให้ตัวเล่นเสียงดนตรีกลับไปเล่นเสียงดนตรีใหม่อีกรอบโดยที่ไม่จำเป็นต้องเล่นเสียงดนตรีนั้นจนจบไฟล์ ใช้คำสั่งดังรูปแบบต่อไปนี้

```
pygame.mixer.music.rewind()
```

กรณีที่ต้องการให้เสียงดนตรีค่อย ๆ เงียบหายไปก่อนจะจบไฟล์เสียงสามารถใช้คำสั่งสำหรับตั้งค่าการเฟดเอาต์ดังรูปแบบต่อไปนี้

```
pygame.mixer.music.fadeout( จำนวนมิลลิวินาที )
```

ในบางครั้งผู้เขียนโปรแกรมต้องการตรวจสอบสถานะการเล่นเสียงเพลงของตัวเล่นเสียงดนตรี สามารถตรวจสอบสถานะ True หรือ False ของการว่างดังคำสั่งรูปแบบต่อไปนี้

```
สถานะ = pygame.mixer.music.get_busy()
```

และเมื่อต้องการทราบค่าตำแหน่งเวลาของเสียงดนตรีที่กำลังเล่นอยู่ ณ ขณะนั้น และตั้งค่าตำแหน่งที่จะเล่นเสียงดนตรีเสียใหม่ สามารถกระทำได้ตามคำสั่งต่อไปนี้ ซึ่งกรณีที่ค่าเวลาเป็น -1 มีความหมายว่า ดนตรีที่สั่งเล่นนั้นจบแล้ว และเวลามีหน่วยเป็นมิลลิวินาที

```
เวลา = pygame.mixer.music.get_pos( )
pygame.mixer.music.set_pos( เวลา )
```

ตัวอย่างโปรแกรมที่ 17.13 เป็นตัวอย่างการเล่นไฟล์เสียง bell1.wav bell2.wav และ bell3.wav ตามที่ผู้ใช้โปรแกรมเลือกกดแป้น 1, 2 และ 3 ตามลำดับ

โปรแกรมที่ 17.13 เล่นไฟล์เสียงตามแป้นที่กด

บรรทัด	โค้ด
1	`# -*- coding: utf-8 -*-`
2	`import pygame`
3	`pygame.init()`
4	`pygame.font.init()`
5	`pygame.mixer.init()`
6	`pygame.mixer.pre_init(44100, -16, 2, 2048)`
7	`fpsClock = pygame.time.Clock()`
8	`surface = pygame.display.set_mode((800,600))`
9	`pygame.display.set_caption("Ex17.13")`
10	`background = pygame.Color( 0, 0, 0)`
11	`font = pygame.font.SysFont("Tahoma", 20)`
12	`text =  font. render( " Press 1, 2 or 3 to play sound No. 1,2 or 3", True, (192,255,40))`
13	`running = True`
14	`while running:`
15	`    fpsClock.tick(30)`
16	`    surface.fill(background)`
17	`    surface.blit(text,(10, 10))`
18	`    for event in pygame.event.get():`
19	`        if event.type == pygame.QUIT:`
20	`            running = False`
21	`        if event.type == pygame.KEYDOWN:`
22	`            if event.key == pygame.K_1:`
23	`                pygame.mixer.music.stop()`
24	`                pygame.mixer.music.load("./res/sounds/bell1.wav")`
25	`                pygame.mixer.music.play(0)`
26	`            elif event.key == pygame.K_2:`
	→

```
27                      pygame.mixer.music.stop()
28                      pygame.mixer.music.load("./res/sounds/bell2.wav")
29                      pygame.mixer.music.play(0)
30                  elif event.key == pygame.K_3:
31                      pygame.mixer.music.stop()
32                      pygame.mixer.music.load("./res/sounds/bell3.wav")
33                      pygame.mixer.music.play(0)
34          pygame.display.flip()
35      pygame.font.quit()
36      pygame.quit()
```

บทสรุป

การปฏิสัมพันธ์ คือ วิธีการที่ผู้ใช้โปรแกรมสื่อสารกับโปรแกรมผ่านทางอุปกรณ์นำเข้า และอุปกรณ์แสดงผล โดยอุปกรณ์หลักสำหรับระบบคอมพิวเตอร์กราฟิกส์ คือ เมาส์ แป้นพิมพ์ เกมคอนโทรลเลอร์และจอแสดงผล

ไลบรารี PyGame มีฟังก์ชั่นการทำงานที่ครอบคลุมการควบคุมหน้าต่าง การเชื่อมประสานกับเมาส์ การเชื่อมประสานกับแป้นพิมพ์ การเชื่อมประสานกับเกมคอนโทรลเลอร์ สามารถติดต่อกับตัวตั้งเวลา การเรียกใช้ไฟล์ภาพ การแสดงผลตัวอักษรจากฟอนต์แบบทรูไทป์ และการเล่นไฟล์เสียง ทำให้ผู้พัฒนาโปรแกรมมีความสะดวกในการเขียนโปรแกรมสำหรับสร้างปฏิสัมพันธ์กับผู้ใช้

เอกสารอ้างอิง

Angel Edward. and Shreiner. (2012). **Interactive Computer Graphics: A top-down approach with shader-based OpenGL**. (6th ed.). Boston: Pearson Education.

AL Sweigart. (2017). **Invent Your Own Computer Games With Python 4th Edition**. San Francisco : No starch press, Inc..

Alejandro Rodas de Pas, and Joseph Howse. (2015). **Python Game Programming By Example: A pragmatic guide for developing your own games with Python**. UK, Birmingham: Packt Publishing Ltd..

Balley Mike. and Cunningham Steve. (2012**). Graphics Shaders Theory and Practice**. (2nd ed.). Boca Raton: CRC Press.

Guha Sumanta. (2015). **Computer Graphics Through OpenGL® From Theory to Experiments comprehensive coverage of OpenGL 4.**3. (2nd ed.). Boca Raton: CRC Press.

Harrison Kinsley, and Will McGugan. (2015). **Beginning Python Games Development With PyGame 2nd Edition**. USA, New York: Springer Science..

Paul Vincent Craven. (2016). **Program Arcade Games: With Python and Pygame 4th Edition.** USA, New York: Springer Science.

Rodríguez Jacobo. (2013). **GLSL Essentials: Enrich your 3D scenes with the power of GLSL!**. Birmingham: Packt Publishing.

Shirley Peter and Marschner Steve.(2009). **Fundamentals of Computer Graphics**. (3rd ed.). Boca Raton: CRC Press.

Shreiner Dave, Sellers Graham, Kessenich John and Licea-Kane Bill. (2013). **OpenGL(R) Programming Guide**. (8th ed). Boston: Pearson Education.

Sloan Kelly. (2016). **Python, PyGame and Raspberry Pi Game Development**. USA, New York: Springer Science.

Wolf David. (2013). **OpenGL 4 Shading Language Coodbook**. (2nd ed). Birmingham: Packt Publishing.