Pygame - Dev Notes - Animation - Colour Scale

A few notes on modifying colours in animations with Pygame.

Contents

- Intro
- Colour Scales

Intro

We may use animation with Pygame to create many different effects within our games, including dynamically updating colours and colour scales.

Colour scales

We can use the RGB colour scale to animate colour changes in shapes, e.g.

```
# rgb colours for rect
rectRed = random.randint(0, 255)
rectGreen = random.randint(0, 255)
rectBlue = random.randint(0, 255)

# create game loop
while True:
    # set clock
    msElapsed = clock.tick(max_fps)
    #print(msElapsed)
    # 'processing' inputs (events)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gameExit()
    #print(elapsedSecs)
    if rectRed >= 255:
        rectRed = random.randint(0, 255)
    else:
        rectRed +=1
    if rectGreen >= 255:
        rectGreen = random.randint(0, 255)
    else:
        rectGreen +=1
    if rectBlue >= 255:
        rectBlue = random.randint(0, 255)
    else:
        rectBlue +=1
    # draw
    window.fill(WHITE)
    pygame.draw.rect(window, (rectRed, rectGreen, rectBlue), (50, 50, winWidth / 2,
winHeight / 2))

    # update the display window...
    pygame.display.update()
```

So, we can modify a rectangle's colour as we execute each iteration of the *game loop*. We add some conditional statements to check if our RGB values are about to go over 255. If yes, then we assign them a random value using the standard RGB scale, 0 to 255.

The rectangle slowly changes colour by 1 value per execution of the *game loop*, simply adding 1 to each R, G, and B value.