How to design and create levels in PyGame

You can add levels to the game to make it more engaging and interesting. Here are instructions on how to design and create levels in PyGame.

You can add levels to the game to make it more engaging and interesting. Here is **a tutorial on how to design and create levels in PyGame** .

Levels are a fundamental aspect of most video games. They provide structure, challenge and variety, and can even greatly enhance the player's experience. PyGame, a popular Python library in game development, provides various features that you can easily create multiple levels. A well-designed level is easy to attract players, promoting a sense of conquest to receive well-deserved rewards.

Plan and design levels

Before creating levels, you need to plan and design them. Level planning helps create a well-structured game, for the right difficulty. To do this, you need to consider factors such as the gameplay mechanics, the story, and the overall structure of the game.

For example, if you are designing a platformer game, you need to think about the placement of the platform, enemies, and obstacles. You also have to consider player movements such as jumping, running and level design accordingly.

Create two simple levels

Before you start, make sure you have pip installed on your system. Use this command to install the pygame library:

```
pip install pygame
```

Now create two simple levels with a platform and a player. Here is the code for the first level:

```
import pygame pygame.init() # Xác ??nh các ch? s?
 SCREEN_WIDTH = 800 SCREEN_HEIGHT = 600 PLATFORM_WIDTH = 100 PLATFORM_HEIGHT = 20
?
o màn hình screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT)) # T
?
o platform platform = pygame.Rect(350, 500, PLATFORM_WIDTH, PLATFORM_HEIGHT) # T
?o ng??i ch?
i player = pygame.Rect(375, 450, PLAYER_WIDTH, PLAYER_HEIGHT) # Game loop running
????for event in pygame.event.get(): ????????if event.type == pygame.QUIT:
???????????running = False ????????elif event.type == pygame.KEYDOWN:
???????????if event.key == pygame.K_LEFT: ???????????????
player.move_ip(-10, 0) ???????????elif event.key == pygame.K_RIGHT:
????????????????player.move_ip(10, 0) ????# Tô màu ?en cho màn hình ????
screen.fill((0, 0, 0)) ????# V? platform ????
pygame.draw.rect(screen, PLATFORM_COLOR, platform) ????# V? ng??i ch?i ????
pygame.draw.rect(screen, PLAYER_COLOR, player) ????# Update màn hình ????
pygame.display.flip() pygame.quit()
```

Then create a second level by just changing the player's platform and position. Here is the code for the second level:

```
import pygame pygame.init() # Xác ??nh các ch? s?
 SCREEN_WIDTH = 800 SCREEN_HEIGHT = 600 PLATFORM_WIDTH = 150 PLATFORM_HEIGHT = 20
?
o màn hình screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT)) # T
?
o platform platform = pygame.Rect(200, 500, PLATFORM_WIDTH, PLATFORM_HEIGHT) # T
?o ng??i ch?
i player = pygame.Rect(225, 450, PLAYER_WIDTH, PLAYER_HEIGHT) # Game loop running
????for event in pygame.event.get(): ????????if event.type == pygame.QUIT:
???????????running = False ????????elif event.type == pygame.KEYDOWN:
???????????if event.key == pygame.K_LEFT: ???????????????
player.move_ip(-10, 0) ???????????elif event.key == pygame.K_RIGHT:
????????????????player.move_ip(10, 0) ????# Tô màu ?en cho màn hình ????
screen.fill((0, 0, 0)) ????# V? platform ????
pygame.draw.rect(screen, PLATFORM_COLOR, platform) ????# V? ng??i ch?i ????
pygame.draw.rect(screen, PLAYER_COLOR, player) ????# Update màn hình ????
pygame.display.flip() pygame.quit()
```

This code is similar to the first level, but the player's platform and position will change.

Connect different levels

To connect the different levels, you need to edit the game loop to switch between levels when the player crosses a table. You can do this by creating a variable to keep track of the current level and using the conditional

command to switch between levels. You can also do collision detection to check if the player has reached the end of a level.

Here is the edited game loop:

```
current_level = 1 # V? platform và ng??i ch?i d?a trên c?p ?? hi?n t?
i if current_level == 1: ????
pygame.draw.rect(screen, PLATFORM_COLOR, platform1) ????
pygame.draw.rect(screen, PLAYER_COLOR, player) ????
# Check if the player has reached the end of the level ????
if player.colliderect(platform1) == False: ????????
current_level = 2 elif current_level == 2: ????
pygame.draw.rect(screen, PLATFORM_COLOR, platform2) ????
pygame.draw.rect(screen, PLAYER_COLOR, player) ????
# Check if the player has reached the end of the level ????
if player.colliderect(platform2) == True: ????????running = False
```

Sharing game data between levels

You need to save game data to share it between levels. Do this by creating a separate Python module to store game data and import it into each level.

Here are the specific steps:

1. Create a new Python module named **game_data.py** .
2. Define a global variable in the **game_data.py** module to store the player's position.
3. At each level, import the **game_data** module .
4. To retrieve the player's position at each level, use the **game_data.player_pos** variable instead of creating a new player object.
5. To update the player's position, edit the **game_data.player_pos** value in each iteration.

Here is the edited code to share player's location between levels.

In the game_data.py module , add the following code:

```
player_pos = (0, 0)
```

First level:

```
import pygame import game_data # T?o ng??i ch?
i player = pygame.Rect(game_data.player_pos[0], game_data.player_pos[1], PLAYER_W
????# V? ng??i ch?i ????pygame.draw.rect(screen, PLAYER_COLOR, player) ????
# Update v? trí c?a ng??i ch?i trong mô ?un game_data ????
game_data.player_pos = (player.x, player.y)
```

Above is how to design and create levels in Pygame. Hope the article is useful to you.

You finished reading the article "**How to design and create levels in PyGame**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.