How to download custom fonts and text effects in Pygame

You can personalize the game with text that represents the game design and genre for Pygame. Below are detailed instructions.

The aesthetic factor plays an important role in game programming. Small details can affect how a game captures the player's attention and immerses them in the game world. Custom fonts and text effects let you inject personality and personal style into your game's interface, dialogue, and HUD elements.

Whether you're designing an adventure game, puzzle game or any other font or text effect, you can transform your project from basic to engaging.

Create simple 2D games

Before diving into the world of custom font and text effects, create a basic 2D game foundation. In this example, design a game with player movement using Pygame features.

To get started, set up the game window. Pygae provides the **pygame.display.set_mode()** function to create a game window. You can also title the window with **pygame.display.set_caption()** .

```
import pygame pygame.init() # Thi?t l?p hi?n th?
 width, height = 800, 600 screen = pygame.display.set_mode((width, height)) pygam
```

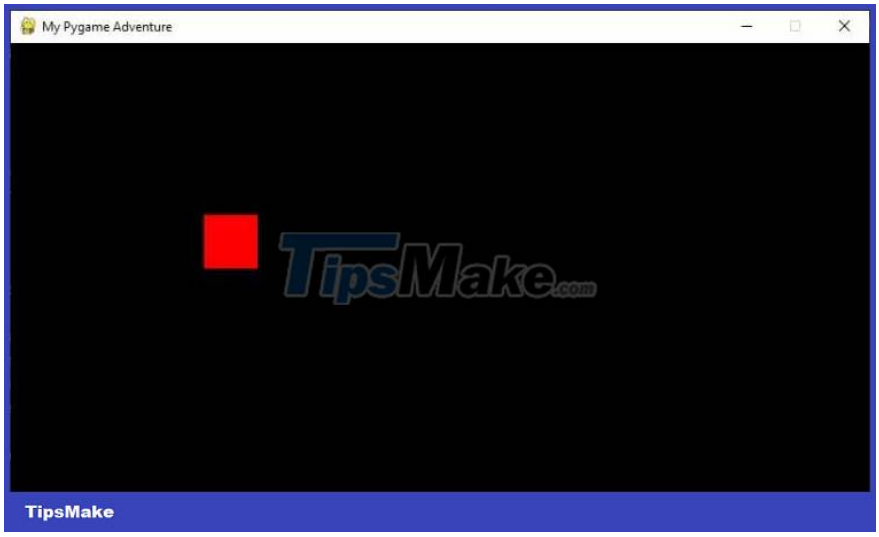
Now initialize the player object with class **pygame.Rect()** . This class represents a rectangle, which you can use to render characters and objects in 2D games.

```
# Kh?i t?o ng??i ch?
i player = pygame.Rect(50, 50, 50, 50) player_color = (255, 0, 0)
```

The game loop is the heart of the game, where all the action happens. It continuously checks player input, updates the game state, and redraws the screen.

```
# Vòng l?
p game chính running = True while running: for event in pygame.event.get(): if ev
?n ??ng ng??i ch?
i keys = pygame.key.get_pressed() if keys[pygame.K_LEFT]: player.x -= 5 if keys[p
? ng??i ch?
i pygame.draw.rect(screen, player_color, player) # Update màn hình pygame.displa
```

Result:



Add text to the game

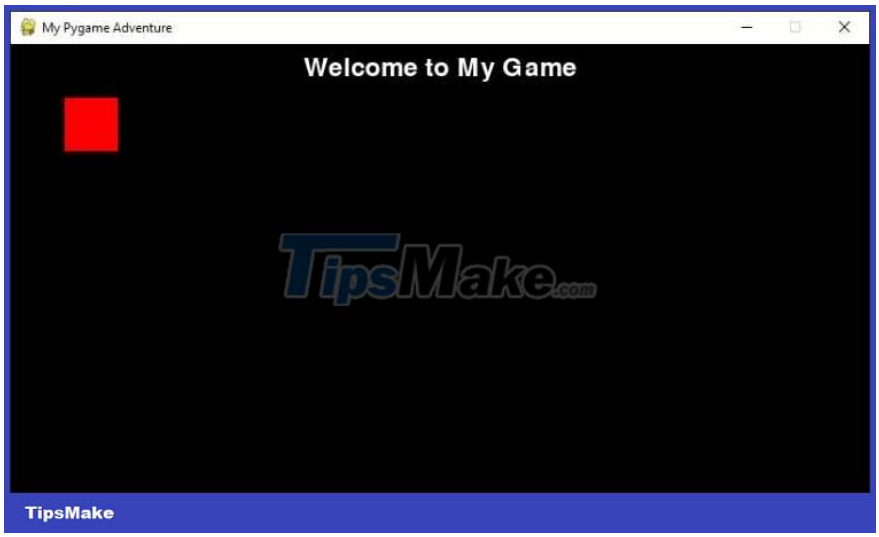
To display text, start by loading the font using the **pygame.font.Font()** class . You can specify the font file and font size:

```
# T?i font font = pygame.font.Font(None, 36)
```

With the font described, you can now render and display text on the screen. **The font object's render()** method takes text, anti-aliasing settings, and color as arguments. You can then use the **blit()** function to draw the text displayed on the screen.

```
# Vòng l?p game chính while running: # V? ng??i ch?i # K?t xu?t và hi?
n text text = font.render("Welcome to My Game", True, (255, 255, 255)) screen.bli
```

The result will look like this:



More custom fonts

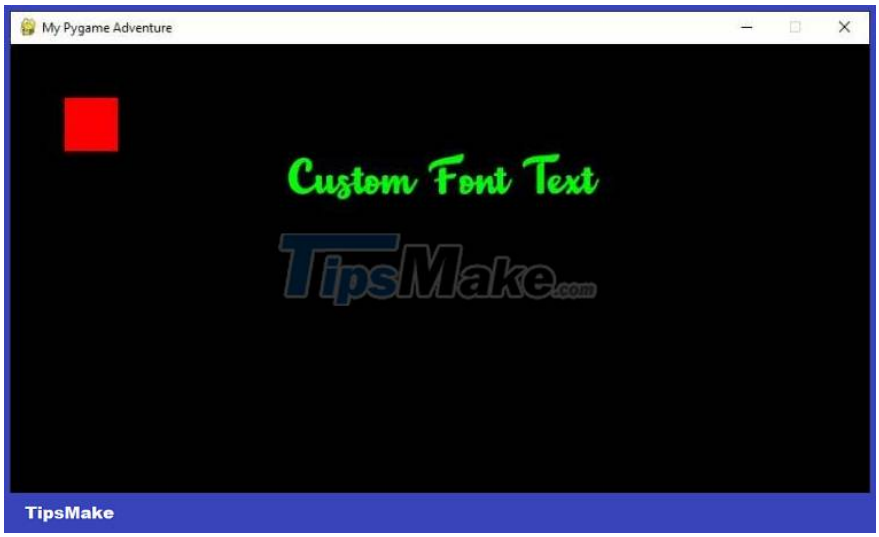
While the default font has worked well, using a custom font can greatly improve the visual appeal of a game. To add custom fonts, follow these steps:

First, you need a custom TrueType font file (TTF) that matches the game's aesthetic. There are many online resources where you can find fonts for free or for a fee.

Place the font file in the same folder as the game script. This ensures Pygame can locate and load that font.

```
# T?i font tùy bi?
n custom_font = pygame.font.Font("custom_font.ttf", 48) # Vòng l?
p game chính while running: # V? ng??i ch?i #K?t xu?t và hi?
n text font tùy bi?
n custom_text = custom_font.render("Custom Font Text", True, (0, 255, 0)) screen
```

You will see the text displayed in the selected font:



Create text effects

Adding text effects to your game can transform static text into dynamic and attention-grabbing elements. One of the simplest yet effective text effects is vibrating hieuejuwns.

The vibrating text effect involves making text appear to expand and contract rhythmically. Here's how you can implement this effect:

```
# Vòng l?p game chính while running: # V? ng??i ch?i # T?o v?n b?n v?i hi?
u ?
ng rung pulsating_text = custom_font.render("Pulsating Text", True, (0, 0, 255))
? s? xung theo th?
i gian pulsation_factor = 1 + abs((pygame.time.get_ticks() % 1000) - 500) / 500 v
? l? v?n b?n d?a trên h? s?
 xung pulsating_text = pygame.transform.scale( pulsating_text, (width, height) )
? trí v?n b?n ?? c?n gi?
a trên màn hình text_x = width // 2 - pulsating_text.get_width() // 2 text_y = 20
```

Calculate the clock factor based on the current time using **pygame.time.get_ticks()** . By adjusting the coefficient, you control the degree of expansion and contraction, thereby achieving a vibrating effect.

Besides the pulse effect, you can try out many other text effects to add more flair to the game:

1. **Typewriter effect** : Displays text letter by letter, simulating the sound of a typewriter.
2. **Fade Text Effect** : Gradually fade the text inward or outward to create a smooth transition.
3. **Jiggle effect** : Causes text to jolt or shake slightly to create a sense of urgency or excitement.
4. **Glow effect** . Add a subtle glow effect to the text to make it stand out in dark environments.

To implement these effects, you can combine techniques such as changing the alpha (transparency) channel, editing position, and applying color transitions.

Don't be afraid to get creative and try combining effects to find the perfect option to suit your game style.

Above are ways **to download custom fonts and add text effects for Pygame** . Hope the article is useful to you.