```python
# Pygame configuration module

# Define color constants
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)
BROWN = (139, 69, 19)
YELLOW = (255, 255, 0)
RED = (255, 0, 0)
PURPLE = (191, 64, 191)

# Game window dimensions
WINDOW_WIDTH = 800
WINDOW_HEIGHT = 600

# Window title (caption)
# Update the window title as needed
TITLE = "Pygame Shapes Using Dictionaries"

# Frame rate (frames per second)
FPS = 60
```

```python
# External module containing functions for drawing various shapes
# shapes.py

import pygame

def draw_circle(screen, shape):
    pygame.draw.circle(screen, shape['color'], shape['position'], shape['radius'])

def draw_rect(screen, shape):
    pygame.draw.rect(screen, shape['color'], (shape['position'][0], shape['position'][1], shape['width'], shape['height']))

def draw_line(screen, shape):
    pygame.draw.line(screen, shape['color'], shape['start_pos'], shape['end_pos'], shape['width'])


```

```python
# Pygame game template

import pygame
import sys
import config  # Import the config module
import random
import shapes  # Import the shapes module

def init_game():
    pygame.init()
    screen = pygame.display.set_mode((config.WINDOW_WIDTH, config.WINDOW_HEIGHT))  # Use constants from config
    pygame.display.set_caption(config.TITLE)
    return screen

def handle_events():
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            return False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                return False
    return True

def main():
    screen = init_game()
    clock = pygame.time.Clock()  # Initialize the clock object

    shapes_list = []  # List to hold shapes

    running = True
    while running:
        running = handle_events()
        screen.fill(config.WHITE)  # Use color from config

        # Generate a random number to represent the various shapes
        shape_type = random.randrange(3)

        # Create a new shape and add it to the list
        if shape_type == 0:
            # Circle: (type, color, position, radius)
            new_shape = {
                'type': 'circle',
                'color': (random.randrange(255), random.randrange(255), random.randrange(255)),
                'position': (random.randrange(config.WINDOW_WIDTH), random.randrange(config.WINDOW_HEIGHT)),
                'radius': 50
```

```
            }
        elif shape_type == 1:
            # Rectangle: (type, color, position, width, height)
            new_shape = {
                'type': 'rectangle',
                'color': (random.randrange(255), random.randrange(255), random.randrange(255)),
                'position': (random.randrange(config.WINDOW_WIDTH - 100), random.randrange(config.WINDOW_HEIGHT - 100)),
                'width': 100,
                'height': 100
            }
        elif shape_type == 2:
            # Line: (type, color, start_pos, end_pos, width)
            new_shape = {
                'type': 'line',
                'color': (random.randrange(255), random.randrange(255), random.randrange(255)),
                'start_pos': (random.randrange(config.WINDOW_WIDTH), random.randrange(config.WINDOW_HEIGHT)),
                'end_pos': (random.randrange(config.WINDOW_WIDTH), random.randrange(config.WINDOW_HEIGHT)),
                'width': 10
            }

        # Add the new shape to the list
        shapes_list.append(new_shape)

        # Draw all shapes from the list using the appropriate function from the shapes module
        for shape in shapes_list:
            if shape['type'] == 'circle':
                shapes.draw_circle(screen, shape)
            elif shape['type'] == 'rectangle':
                shapes.draw_rect(screen, shape)
            elif shape['type'] == 'line':
                shapes.draw_line(screen, shape)

        # UPDATE the screen with what we've drawn
        pygame.display.flip()

        # Limit frame rate to certain number of frames per second (FPS)
        clock.tick(config.FPS)

    pygame.quit()
    sys.exit()

if __name__ == "__main__":
    main()
```