```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

def initialize():
    glClearColor(0.5, 0.5, 0.5, 1.0)  # Background color: gray
    glEnable(GL_DEPTH_TEST)          # Enable depth testing
    glEnable(GL_LIGHTING)            # Enable lighting
    glEnable(GL_LIGHT0)              # Enable light source 0
    glEnable(GL_LIGHT1)              # Enable light source 1
    glEnable(GL_LIGHT2)              # Enable spotlight for cone top

    # Light 0: Green ambient and general lighting
    # Setup light 0 (main light)
    glLightfv(GL_LIGHT0, GL_AMBIENT, GLfloat_4(0.7, 0.7, 0.1, 1.0))  # Slightly
increased ambient light
    glLightfv(GL_LIGHT0, GL_DIFFUSE, GLfloat_4(0.7, 0.7, 0.5, 1.0))  # Slightly
increased diffuse light
    glLightfv(GL_LIGHT0, GL_SPECULAR, GLfloat_4(0.5, 0.5, 0.4, 1.0))  # Slightly
brighter specular light
    glLightfv(GL_LIGHT0, GL_POSITION, GLfloat_4(0.4, 1.5, 2.0, 1.0))  # Same
position

    # Light 1: Red ambient and general lighting
    glLightfv(GL_LIGHT1, GL_AMBIENT, GLfloat_4(0.5, 0.5, 0.5, 1.0))  # Increased
ambient light
    glLightfv(GL_LIGHT1, GL_DIFFUSE, GLfloat_4(0.3, 0.3, 0.4, 1.0))  # Slightly
increased diffuse light
    glLightfv(GL_LIGHT1, GL_SPECULAR, GLfloat_4(0.2, 0.2, 0.3, 1.0))  # Slightly
brighter specular light
    glLightfv(GL_LIGHT1, GL_POSITION, GLfloat_4(0.4, 1.5, 2.0, 1.0))  # Same
position

    # Light 2: Spotlight for cone tip
    glLightfv(GL_LIGHT2, GL_POSITION, [0.0, 1.2, 0.0, 1.0])    # Positioned at the
tip of the cone
    glLightfv(GL_LIGHT2, GL_SPOT_DIRECTION, [0.0, -1.0, 0.0])  # Direction
pointing downwards
    glLightf(GL_LIGHT2, GL_SPOT_CUTOFF, 15.0)                  # Focused spotlight
angle
    glLightfv(GL_LIGHT2, GL_DIFFUSE, [1.0, 1.0, 0.8, 1.0])     # Bright spotlight
effect
    glLightfv(GL_LIGHT2, GL_SPECULAR, [1.0, 1.0, 0.8, 1.0])    # Highlight effect
    glLightf(GL_LIGHT2, GL_SPOT_EXPONENT, 50.0)                # Concentrated
spotlight

    # Global ambient light (consistent with second code)
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, [0.2, 0.2, 0.2, 1.0])  # Slightly
brighter ambient lighting


def setMaterial(color):
```

```
    # Material properties for both shapes
    glMaterialfv(GL_FRONT, GL_AMBIENT, [color[0] * 0.2, color[1] * 0.2, color[2] *
0.2, 1.0])
    glMaterialfv(GL_FRONT, GL_DIFFUSE, color + [1.0])
    glMaterialfv(GL_FRONT, GL_SPECULAR, [1.0, 1.0, 1.0, 1.0])
    glMaterialf(GL_FRONT, GL_SHININESS, 80.0)  # Higher shininess for tighter
specular highlights

def drawScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)  # Clear screen and depth
buffer
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(40, 1.0, 1.0, 10.0)  # Perspective setup

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    gluLookAt(0, 0, 6, 0, 0, 0, 0, 1, 0)  # Camera position and orientation

    # Draw the sphere
    glPushMatrix()
    setMaterial([0.9, 0.9, 0.7])  # Light yellow material
    glTranslatef(0.0, -0.5, 0.0)  # Position sphere lower
    glutSolidSphere(0.8, 50, 50)  # Sphere radius = 0.8
    glPopMatrix()

    # Draw the cone
    glPushMatrix()
    setMaterial([0.9, 0.9, 0.7])  # Same material as sphere
    glTranslatef(0.0, 0.0, 0.0)   # Position cone to align with sphere's top
    glRotatef(-90, 1, 0, 0)       # Rotate cone to point upwards
    glutSolidCone(0.8, 1.3, 50, 50)  # Cone base radius = 0.8, height = 1.3
    glPopMatrix()

    glutSwapBuffers()  # Swap buffers for double buffering

def main():
    glutInit()
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
    glutInitWindowSize(500, 500)
    glutCreateWindow("Cone and Sphere with Lights")
    initialize()
    glutDisplayFunc(drawScene)
    glutMainLoop()

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

# 3D Scene with Sphere and Cone Illumination

**Code Breakdown**

This Python code, leveraging OpenGL libraries, constructs a 3D scene featuring a sphere and a cone illuminated by three distinct light sources. Let's delve into the key functions:

## 1. `initialize()`

- **Background Setup:** Sets the backdrop to a gray hue.
- **Depth Testing:** Enables depth testing to ensure correct object layering.
- **Lighting Activation:** Activates the overall lighting system and three specific light sources (0, 1, and 2).
- **Light Configuration:**
  - **Light 0 (Main Light):** Adjusts ambient, diffuse, and specular light properties, positioning it for general illumination.
  - **Light 1 (Red Ambient):** Modifies ambient, diffuse, and specular light properties, placing it in the same position as Light 0.
  - **Light 2 (Spotlight):** Positions the spotlight at the cone's tip, directs it downwards, and defines its focus and intensity.

## 2. `setMaterial()`

- **Material Properties:** Assigns material properties (ambient, diffuse, specular, and shininess) to both the sphere and the cone, influencing their interaction with light.

## 3. `drawScene()`

- **Scene Clearing:** Clears the screen and depth buffer.
- **Perspective Setup:** Configures the camera's perspective using `gluPerspective`.
- **Camera Positioning:** Sets the camera's position and orientation using `gluLookAt`.
- **Object Drawing:**
  - **Sphere:** Positions and draws a solid sphere with specified radius and detail level.
  - **Cone:** Positions, rotates, and draws a solid cone with defined base radius, height, and detail level.

## 4. `main()`

- **Initialization:** Initializes the OpenGL window, sets display mode, and specifies window size.
- **Function Registration:** Registers the `drawScene` function to be called for rendering the scene.
- **Main Loop:** Enters the main loop to handle events and continuously redraw the scene.

**Overall Functionality** This code effectively demonstrates the application of OpenGL's lighting and material properties to create a visually appealing 3D scene. By manipulating light sources, object materials, and camera perspective, the program achieves realistic lighting effects and depth perception.