Light Module: Dynamic Spotlights and Radial Light Masks (Pygame)

**This module provides dynamic lighting effects using radial and elliptical gradients. It supports both elliptical spotlights (directional beams) and circular masks (omnidirectional glows or fades), which can be composited into 2D games for realistic lighting, shadow casting, or fog-of-war effects.**

**Dependencies**

**import pygame**

**import math**

**Class: SpotLight**

**Purpose:**

**Generates a spotlight beam effect — a conical or elliptical gradient mask that can be rotated and pointed at a target. This simulates focused light sources such as torches, flashlights, or searchlights.**

**Constructor:**

**SpotLight(angle: float = None, beam\_surface\_size: tuple = None, display\_surface: tuple = None)**

**Parameters:**

* **angle: The beam angle (in degrees). Defaults to 60° if not provided.**
* **beam\_surface\_size: Tuple (width, height) for beam dimensions. Defaults to 2/3 screen width x 1/2 screen height.**
* **display\_surface: The main display surface size (must be provided to anchor beam scale).**

**Internal Notes on Beam Geometry**

**The beam is rendered using a polygonal approximation of an ellipse sector. For each of the steps:**

* **It creates a layered polygon mask by sweeping across an angular arc (angle) from +angle/2 to -angle/2.**
* **The points are calculated from the parametric form of an ellipse:**

**x=acos⁡(θ),y=bsin⁡(θ)x = a \cos(\theta), \quad y = b \sin(\theta)x=acos(θ),y=bsin(θ)**

**where a=widtha = \text{width}a=width, b=heightb = \text{height}b=height, and θ∈[−α2,α2]\theta \in [-\frac{\alpha}{2}, \frac{\alpha}{2}]θ∈[−2α​,2α​]**

**Method: create\_beam(...)**

**create\_beam(alpha=90, steps=90, debug=False)**

**Creates the spotlight gradient as a surface with per-pixel alpha. Each layer is a darker ellipse, simulating light falloff.**

**Parameters:**

* **alpha: Maximum opacity at the beam’s outer edges (0–100%).**
* **steps: Number of fading layers (more = smoother gradient).**
* **debug: If True, fills the surface with yellow for testing.**

**Returns:**

* **A pygame.Surface with a transparent-to-black beam.**

**Method: draw(...)**

**draw(surface, pos, target, rotation=0)**

**Renders and rotates the beam towards a target point, giving directional behavior.**

**Parameters:**

* **surface: The pre-generated beam surface.**
* **pos: Origin (light source position, pygame.math.Vector2).**
* **target: The point to aim at.**
* **rotation: Optional manual angle offset.**

**Returns:**

* **A tuple (rotated\_image, rotated\_rect) ready for blit.**

**Method: \_draw\_ellipse\_points(...)**

**Private method that constructs the beam polygon by generating points along an angular sweep. Uses trigonometry to shape the gradient.**

**Method: \_calculate\_angle(...)**

**Internal utility to compute the angle (in degrees) between the light’s position and a target.**

**Function: circle\_light\_mask(...)**

**Purpose:**

**Creates a radial circular mask with a gradient from the center outward. Perfect for torches, lamps, or any 360° light source.**

**circle\_light\_mask(radius, steps, alpha)**

**Parameters:**

* **radius: Radius of the circle in pixels.**
* **steps: Number of fading rings.**
* **alpha: Maximum opacity at the outer edge (0–100%).**

**Returns:**

* **A pygame.Surface containing a radial gradient.**

**Mathematical Breakdown:**

**Each step draws a smaller circle using:**

**Radius of each circle layer (r):  
r = R - (R \* i / steps)  
Where:**

* **R is the maximum radius,**
* **i is the current step in the gradient loop,**
* **steps is the total number of layers.**

**Alpha (opacity) per layer (a):  
a = (i / steps) \* ((α / 100) \* 255)  
Where:**

* **α is the maximum opacity percentage (0–100),**
* **255 is the max alpha value in RGBA color format.**

**This builds an alpha ring from the outside in.**

**Customization Tips**

**Custom Spotlight Shapes:**

* **Modify beam\_surface\_size to stretch the beam horizontally or vertically.**
* **Tighter angle → Narrower beam.**
* **Higher steps → Smoother fade.**
* **Increase alpha → Stronger darkness, heavier shadow feel.**

**Rotation:**

* **Use draw() with a target vector to dynamically rotate spotlight in-game.**

**Baking Light Masks:**

**To improve performance:**

* **Pre-render light masks and spotlight beams into textures using create\_beam() or circle\_light\_mask().**
* **Store them in a cache or surface atlas.**
* **Use blit() at runtime instead of regenerating per frame.**

**Real-Time Example:**

**spot = SpotLight(display\_surface=(800, 600))**

**beam = spot.create\_beam(alpha=80, steps=50)**

**image, rect = spot.draw(beam, pygame.math.Vector2(300, 300), pygame.math.Vector2(mouse\_x, mouse\_y))**

**screen.blit(image, rect)**

**Or for a static flashlight mask:**

**torch\_mask = circle\_light\_mask(200, 50, 90)**

**screen.blit(torch\_mask, (x - 200, y - 200), special\_flags=pygame.BLEND\_RGBA\_SUB)**

**Notes**

* **All masks are additive/subtractive visual effects — consider using BLEND\_RGBA\_SUB or BLEND\_RGBA\_MIN for subtractive lighting.**
* **Use multiple light sources to simulate overlapping lights.**
* **Elliptical beams can be skewed to simulate motion blur or dynamic flashlights.**