Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavrepalanchowk

Project Report
On
“COMP 342”

Submitted by:
Aaditya K.C (05)
Jagat K.C (07)
Saurav Bhujel (23)

Level:UNG CS(III/I)

Submitted to: Dhiraj Shrestha
Department of Computer Science and Engineering
Submission Date: 1/16/2026

Day–Night Transition Simulation Using PyOpenGL

Abstract

The Day–Night Transition Simulation is a computer graphics project that models the natural cycle of day and night using mathematical and rendering techniques. The simulation represents the apparent motion of the sun and moon through rotational movement along semi-circular paths, where time is mapped to angular displacement to ensure smooth transitions. Variations in lighting, sky color, and shadow behavior are dynamically calculated based on the height of the sun.

In addition to the daily cycle, the system supports **seasonal changes**, allowing the user to switch between summer and winter. Seasonal variation affects sunrise and sunset timings, daylight duration, sky color tones, and environmental elements such as snow. By integrating trigonometric formulas, interpolation methods, and time-based animation, the project demonstrates how real-world astronomical and environmental changes can be effectively visualized using 2D computer graphics and OpenGL.

1. Introduction

Natural phenomena such as the day–night cycle and seasonal variation are governed by predictable physical and astronomical principles. In computer graphics, these phenomena can be recreated using mathematical models that control motion, lighting, and color transitions. Simulating such behavior helps in understanding core graphics concepts including animation, transformations, interpolation, and lighting models.

This project presents a **Day–Night Transition Simulation** developed using **Python and PyOpenGL**, which visually demonstrates a complete 24-hour cycle along with seasonal changes. The sun and moon are rotated along orbital paths using trigonometric functions, while time is continuously updated to produce smooth visual transitions. Lighting intensity, sky color, and shadows dynamically change based on the simulated time and sun position.

The simulation also incorporates **seasonal variation**, enabling the user to switch between summer and winter modes. This affects the length of daytime and nighttime, sunrise and sunset timings, and environmental appearance. Through this approach, the project provides a clear and practical demonstration of how mathematical formulas and computer graphics techniques can be combined to simulate real-world temporal and environmental changes.

2. Objective of the Project

The main objectives of this project are:

- To simulate a complete 24-hour day–night cycle
- To model sun and moon rotation using trigonometry
- To calculate dynamic lighting and brightness
- To generate realistic shadows based on sun position
- To visualize seasonal changes in daylight duration
- To demonstrate mathematical modeling in computer graphics

3. Technologies Used

- **Programming Language:** Python
- **Graphics Library:** PyOpenGL (OpenGL + GLUT)
- **Mathematical Tools:** Trigonometric functions (sin, cos), interpolation

4. Time Modeling and Progression

4.1 Representation of Time

Time in the simulation is represented as a **continuous variable**, not as discrete hours. A full day is mapped to a **24-hour cycle**, which is further divided into day and night periods.

Instead of rotating the Earth, the simulation rotates the **sun and moon along a semi-circular path** using angles.

4.2 Time to Angle Conversion

For smooth animation, time is converted into an angular value.

Day progress is calculated as:

DAY_PROGRESS = (current_hour − DAY_START) / DAY_DURATION

Sun angle is then calculated as:

sun_angle = DAY_PROGRESS × π

This converts time into an angle between 0 and π radians.

- 0 radians → sunrise
- π/2 radians → noon
- π radians → sunset

4.3 Time Advancement Formula

Time advances every frame using:

current_hour = current_hour + (frame_time × TIME_SCALE)

Where:

- frame_time = time between frames
- TIME_SCALE = speed of simulation

This ensures smooth and frame-independent animation.

5. Sun Rotation and Position Calculation

5.1 Rotational Motion of the Sun

The sun moves along a semi-circular arc from east to west. This motion is simulated using circular rotation equations.

Sun position is calculated using:

sun_x = center_x + radius × cos(sun_angle)
sun_y = center_y + radius × sin(sun_angle)

Here:

- center_x, center_y → center of orbit
- radius → distance of sun from horizon
- sun_angle → angle obtained from time

This creates the illusion of the sun rising, reaching the highest point, and setting.

5.2 Sun Height and Illumination

The vertical position of the sun (sun height) is given by:

sun_height = sin(sun_angle)

This value is used to control:

- Brightness
- Shadow length
- Sky color

6. Moon Rotation and Position Calculation

The moon follows the **same rotational model** as the sun but appears during nighttime.

Moon angle is calculated as:

moon_angle = ((current_hour − NIGHT_START) mod 24 / NIGHT_DURATION) × π

Moon position is calculated using:

moon_x = center_x + radius × cos(moon_angle)
moon_y = center_y + radius × sin(moon_angle)

This ensures smooth moonrise and moonset during night.

7. Brightness and Lighting Calculation

7.1 Light Intensity Formula

Brightness is calculated using a sine-based function:

brightness = max(0, sin(angle))^1.5

This produces:

- Zero brightness at sunrise and sunset
- Maximum brightness at noon
- Smooth light falloff

7.2 Sky Color Interpolation

Sky color transitions between night and day using linear interpolation:

sky_color = night_color + (day_color − night_color) × sin(sun_angle)

This formula is applied separately to red, green, and blue components.

8. Gradient Sky Rendering

The sky is rendered using multiple horizontal strips. Each strip's color is adjusted using:

gradient_color = base_color − (strip_index / total_strips) × base_color × brightness

This creates a realistic vertical gradient in the sky.

9. Shadow Generation Mechanism

9.1 Shadow Visibility Condition

Shadows are rendered only when:

sun_height > 0.05

This avoids unrealistic shadows during sunrise and sunset.

9.2 Shadow Direction Formula

Shadow direction is calculated opposite to the sun:

$dx = sun\_x - object\_x$
$dy = sun\_y - object\_y$

$distance = \sqrt{(dx^2 + dy^2)} + 1$

$shadow\_dir\_x = -dx / distance$
$shadow\_dir\_y = -dy / distance$

9.3 Shadow Length Calculation

Shadow length depends on sun height:

$shadow\_length = 300 \times (1 - sun\_height)$

- Longer shadows in morning and evening
- Short shadows at noon

10. Seasonal Day Length Calculation

Summer Season

- DAY_START = 6 AM
- DAY_DURATION = 12 hours

Winter Season

- DAY_START = 8 AM
- DAY_DURATION = 8 hours

Changing seasons modifies all time-based formulas automatically.

11. Observations and Results

- Smooth rotation of sun and moon observed
- Natural lighting transitions achieved
- Correct shadow direction and length
- Clear difference between summer and winter daylight
- Stable real-time animation

12. Conclusion

This project successfully demonstrates how mathematical formulas can be used to simulate a realistic day–night cycle in computer graphics. By mapping time to angular motion and using trigonometric functions for rotation, the simulation achieves smooth celestial movement, dynamic lighting, and realistic shadows. The project provides a strong practical understanding of animation, transformation, and lighting principles in computer graphics.

13. References

- PyOpenGL Documentation
- OpenGL Programming Guide
- LearnOpenGL
- Computer Graphics – Hearn & Baker