Mini Report — EPL426 OpenGL Scene

**Overview**

This project renders a small animated solar-system vignette with a textured Earth and Moon, a glowing Sun, a prop-driven airplane in orbit, and a twinkling starfield. It’s built with legacy OpenGL/GLUT + GLU and uses **stb\_image** to load textures. The scene demonstrates:

* dual-pass texturing to blend **day** and **night** Earth maps,
* time-driven animation (orbits and propellers),
* simple atmospheric day/night using a sine curve,
* multiple light sources (Sun and airplane “nav” light),
* additive, depth-friendly rendering for stars.

Repository URL: https://github.com/PantelisKanaris/Open\_GL\_Project

**Basic Controls**

**Keyboard**

* 1 — Quick side camera preset (looks along +X from the left).
* 2 – Quick front camera preset(Looks along the +Z from the front close to the center planet)
* p — Increase/Decrease **planet (sun/moon) orbit speed**.  
  Tip: hold **Down Arrow** first to make this a *decrease* instead of increase.
* a — Increase/Decrease **airplane orbit speed** (same “Down Arrow to decrease” rule).
* s — Increase/Decrease **propeller rotation speed**.
* r — **Reset** speeds to defaults.
* + — **Zoom out** (camera Z increases).
* \_ (underscore) — **Zoom in** (camera Z decreases).
* Esc — Exit.

**Special keys**

* F1 — Toggle **fullscreen**.
* **Up Arrow** — Set “increment” mode (the p, a, s keys nudge values **up**).
* **Down Arrow** — Set “decrement” mode (the p, a, s keys nudge values **down**).

**Notes**

* Zoom is implemented by moving the camera along its **Z** axis (m\_camera.m\_pos.z()).

**Day/Night Cycle (Sine Wave)**

Two related pieces create a convincing day/night feel:

1. **Sky color transition**  
   ChangeColourOfBackground() maps the planet’s orbital angle to a smooth **sine curve**:

* Normalize angle → radians
* Compute t = (-sin(radians) + 1)/2 so it cycles **night → day → night**.
* Linearly interpolate between a **night** color (0,0,0) and a **day** sky (0.25, 0.45, 0.7).

**Result:** the background grades gradually from black to blue and back, in sync with the Sun’s apparent motion.

1. **Earth day/night blending**  
   GetDayFactor() uses the Sun’s **Y** position (derived from the same orbit) to compute a day factor in [0,1], then applies **smoothstep** to soften the transition. In CreatePlanetUsingTexture():

* **Pass 1 (Day):** bind **day map**, enable lighting, write depth.
* **Pass 2 (Night):** bind **night map**, set **additive blending** glBlendFunc(GL\_ONE, GL\_ONE), **don’t write depth** and use glDepthFunc(GL\_EQUAL) so we only add light **exactly** where the day sphere rendered.

This dual-pass approach avoids seams and z-fighting, and lets the night lights “glow” over the same geometry without affecting depth.

**Textures & Assets**

* Image loading via **stb\_image** (stb\_image.h) with stbi\_set\_flip\_vertically\_on\_load(1) to match OpenGL’s texture coordinate convention.
* **Earth:** Textures/earth\_2k.jpg (day), Textures/earth\_2k\_night.jpg (night).
* **Moon:** Textures/2k\_moon.jpg.
* **Sun:** Textures/2k\_sun.jpg (for the hot core; outer glow is untextured translucent shells).

**Lighting Model**

**Global:**

* glEnable(GL\_LIGHTING) and glEnable(GL\_COLOR\_MATERIAL) (ambient+diffuse tracking) with glEnable(GL\_NORMALIZE) to keep normals correct under scaling.

**Sun light — GL\_LIGHT1**

* Position follows the orbit:
* Brightness scales with **sun Y** (higher = “noon”, lower = “midnight”).  
  A brightnessMultiplier remaps Y to [0.05, ~5.0] and modulates **ambient**, **diffuse (warm tint)**, and **specular** components.

**Airplane light — GL\_LIGHT2**

* A cool white/blue “navigation light” that follows the airplane’s orbit.  
  Ambient is low, diffuse/specular are high to give a crisp highlight.

**Sun visual**   
The **textured Sun** (CreateSunWithTexture) is drawn emissively (lighting disabled for the core), with several translucent shells for a gentle glow. The **actual illumination** comes from GL\_LIGHT1, so the Sun both looks bright and also lights the Earth/Moon realistically.

**Stars & Atmospherics**

* InitializeStars() seeds 1,000 stars with position, base alpha, per-star **phase**, **twinkle amplitude**, and **size**.
* RenderStars():
  + Renders by size buckets as **GL\_POINTS** with additive-style blending SRC\_ALPHA, ONE.
  + Alpha modulates by sin(phase + time\*2) and is scaled by **night** factor, so stars fade out during the day.

**Animation & Timing**

* Update() uses GLUT elapsed time to compute deltaTime, then:
  + m\_PlanetAngle (orbit for Sun/Moon),
  + m\_AirplaneAngle,
  + m\_ProperllerAngle,
  + and the star twinkle clock m\_timeStars,  
    all wrap with fmod(..., 360.f) for continuous motion.
* Keyboard adjusts **speeds** in real time; **Down Arrow** flips the sign of adjustments for quick increases/decreases.