3D Flight Simulator: Shaders, Lighting, and Physics

Mundhir Almaamari(5124FG01-7) Max Mayer (5123FG30)
OUFFA ABDELLAH (5123FG18-0)

January 25, 2025

Introduction

- Purpose: Create an immersive 3D flight simulation.
- Features:
  - Realistic shaders and lighting.
  - Dynamic environment with terrain and sky.
  - Comprehensive physics system.
  - Custom 3D aircraft design.

Shaders and Lighting: Vertex and Fragment Shaders

- Vertex Shader:

$$v_{position} = modelMatrix \cdot vec4(position, 1.0) \tag{1}$$

Transforms the vertex position from local to world coordinates.

- Fragment Shader:

$$color = mix(bottomColor, topColor, \max(\text{normalize}(v_{worldPosition}).y, 0.0)) \tag{2}$$

Computes a gradient based on the vertical component, enhancing visual effects like the sky.

Shaders and Lighting: Light Sources

- Phong Lighting Model:

$$I = k_a I_a + k_d I_d (\mathbf{L} \cdot \mathbf{N}) + k_s I_s (\mathbf{R} \cdot \mathbf{V})^n \tag{3}$$

  Combines ambient, diffuse, and specular components for realistic lighting effects.
- Parameters:
  - k_a, k_d, k_s: Material coefficients.
  - I_a, I_d, I_s: Light intensities.
  - $\mathbf{L}, \mathbf{N}, \mathbf{R}, \mathbf{V}$: Light, normal, reflection, and view vectors.

Dynamic Night Mode Transition

- Night factor $f(t)$:

$$f(t) = 0.5 - 0.5\cos\left(\frac{\pi t}{T}\right) \tag{4}$$

  Smoothly interpolates lighting and colors to transition between day and night.
- Stars become visible during night mode with fading effects, and lighting intensity is reduced to simulate darkness.
- The transition uses shader uniforms to dynamically adjust visual elements based on the time factor.

Graphics and Terrain Rendering

- Procedural terrain generation:

$$h(x, z) = \sum_{i=1}^{n} A_i \sin(f_i x) \cos(f_i z) \tag{5}$$

  Multi-octave noise functions create realistic terrains with varying frequencies and amplitudes.
- The terrain dynamically adjusts for detail based on proximity to the camera, optimizing performance.

Aircraft Construction

- The aircraft is built using modular geometries:
  - Fuselage: Created with `LatheGeometry` for a cylindrical body.

$$\text{fuselagePoints} = \text{generateCurve}(r(t), h(t)) \tag{6}$$

  - Wings: Constructed with `BoxGeometry`, providing lift surfaces.
  - Engines: Designed with `LatheGeometry`, featuring dynamic exhaust trails.
  - Stabilizers: Tail stabilizers built with `ExtrudeGeometry` for aerodynamics.
- Components are grouped using `Three.Group()` and added to the scene.

$$\text{aircraft} = \text{group}(\text{fuselage}, \text{wings}, \text{engines}, \text{stabilizers}) \tag{7}$$

Physics System: Forces

- Lift Force:

$$F_{\text{lift}} = \frac{1}{2}\rho v^2 S C_L \sin(\alpha) \tag{8}$$

Generates upward force proportional to air density, velocity, wing area, and the angle of attack.

- Drag Force:

$$F_{\text{drag}} = \frac{1}{2}\rho v^2 S C_D \tag{9}$$

Opposes motion and depends on velocity squared, drag coefficient, and surface area.

- Velocity Update:

$$\mathbf{v}_{\text{new}} = \mathbf{v}_{\text{old}} + \frac{\mathbf{F}_{\text{total}}}{m}\Delta t \tag{10}$$

Updates velocity considering the net force acting on the airplane and the time step.

Camera System

- Camera follows the aircraft using smooth interpolation:

$$\mathbf{p}_{\text{camera}} = \text{lerp}(\mathbf{p}_{\text{current}}, \mathbf{p}_{\text{target}}, \alpha) \tag{11}$$

- The interpolation factor $\alpha = 0.1$ ensures smooth transitions without abrupt movements.
- Provides a dynamic view that enhances user immersion by following the aircraft's orientation and movement.

Conclusion and Demo

- This simulation combines advanced computer graphics and physics to create an engaging and interactive experience.
- Key features include:
  - Realistic shaders and lighting models.
  - A physics-based flight model with accurate forces and motion.
  - Custom aircraft model with modular design.
- Demo