RACECAR: (Raceca)r (R)enderer
Team 6: Charles Wang, Anthony Ge, Saahil Gupta, Aaron Jiang
GitHub repo: https://github.com/upgrade-central-tech/racecar

Goals and motivations: We aim to develop a real-time Vulkan car renderer and create the most *scenic* drives possible. In our opinion, cars are great subjects in graphics demos! Aesthetically, they're visually boosted by supporting numerous physical settings, lighting conditions, and materials. In particular, we're inspired by driving games like the *Forza Horizon* series and *Slow Roads*, as well as media like *Cars* and *Top Gear*.

Features: Since RACECAR is a highly visual project, we'll mainly be focusing on graphics and rendering.

1. Complex material rendering: glinty/glittery, anisotropic, and clearcoat materials that cars are known to have, in addition to complex paint jobs and textures.
2. Hardware raytraced reflections and shadows.
3. An interactive environment featuring weather like rain and snow.
4. Interactive day/night skies with naturalistic clouds.

To demonstrate the flexibility of our renderer, the demo will feature the following scenes: a snowy environment, a sunset road, and rain puddles along the terrain.

Schedule

Milestone 1 (11/12)	• Set up base renderer
	  ◦ glTF import (meshes, environment maps, textures)
	  ◦ Environment mapping, direct lighting
	  ◦ Slang shader pipeline
	• Implement the glints with point lights, then add a clear coat
	• Procedural dirt/snow

Milestone 2 (11/24)	• Implement glints (Kniephof 2025, Unity implementation) and spherical harmonics • Add clearcoat, sheen iridescence, and other fancy material options. • Atmospheric effects: sky, fog, clouds (Realistic Real-time Sky Dome Rendering in Gran Turismo 7) • Snow/mud tessellation • Ray traced shadows and reflections with Vulkan Ray Tracing
Milestone 3 (12/1)	• Post-processing - bloom, color grading • Ray traced reflections where heuristically simple • UI polish to access and change material properties
Final	• Render demos, complete performance analysis, create writeup • Stretch goals: car controls, iridescent glints, particles

API and platform usage: Vulkan, Slang/SPIR-V, potentially WebGPU for initial demos (faster iteration time)

Third-party libraries: SDL3, vk-bootstrap, VMA, volk, GLM, ImGUI, stb_image, tinygltf