Neutron Game Engine — Design Document

# Introduction

Neutron is a C++ game engine aimed at indie developers who value rapid prototyping, simple workflows, and high-performance rendering without AAA-level complexity. The goal is to provide developers with powerful yet easy-to-use tools, solving common pain points in game development.

# Core Goals and Developer-Centric Features

## Rapid Iteration

Minimize time-to-feedback with live asset and script hot-reloading, allowing developers to tweak gameplay without full rebuilds.

## Visual Logic Support

Path to visual scripting or node-graph systems for prototyping without deep coding knowledge.

## Pre-built Templates & Assets

Provide starter templates for common genres (e.g., platformers, shooters) to skip boilerplate.

## Cross-Platform Support

Target Windows, macOS, Linux, and mobile with automated asset importers and platform-agnostic deploy scripts.

## Developer-Friendly API

High-level abstractions with an ECS model and a scene graph to reduce boilerplate and ease composition over inheritance.

# Architecture Highlights

## Entity-Component System (ECS)

Data-driven architecture where entities are IDs with detached components, improving modularity and performance.

## Scene Graph / Node System

Hierarchical node system similar to Godot, enabling clear scene and entity management.

## Simplified Rendering Pipeline

Out-of-the-box 2D pipeline and lightweight 3D renderer with modern shader support optimized for low-end and high-end devices.

## Asset Pipeline

Automatic asset importing with instant preview and hot-reloading for images, models, audio, and animations.

## Extensibility & Scripting

Future-ready design to embed scripting languages like Lua and allow C++ module extensions.

## Developer Tooling

A lightweight IDE-like environment with scene editor, asset manager, and debugging overlays.

# Development Phases

## Phase 1 – Research & Architecture

Gather requirements and design the engine based on indie developer needs.

## Phase 2 – Core Engine Prototype

Implement core loop, window management, rendering, ECS, node systems, and basic input/audio.

## Phase 3 – Tools & Assets

Develop asset pipelines, hot-reloading, and a basic editor UI with debugging overlays.

## Phase 4 – Advanced Systems

Add physics, UI toolkit, networking, scripting, and optimize the renderer with batching and shaders.

## Phase 5 – Polish & Extend

Refine APIs, fix bottlenecks, add high-level features like particles and animation systems, and prepare starter templates.

# Summary

Neutron aims to be a problem-solving C++ engine for indie developers focused on rapid prototyping, simplified rendering, and developer-friendly workflows. With modular architecture, ECS, scene graph, asset pipelines, and extensibility, Neutron empowers small teams to bring their creative visions to life with minimal friction.