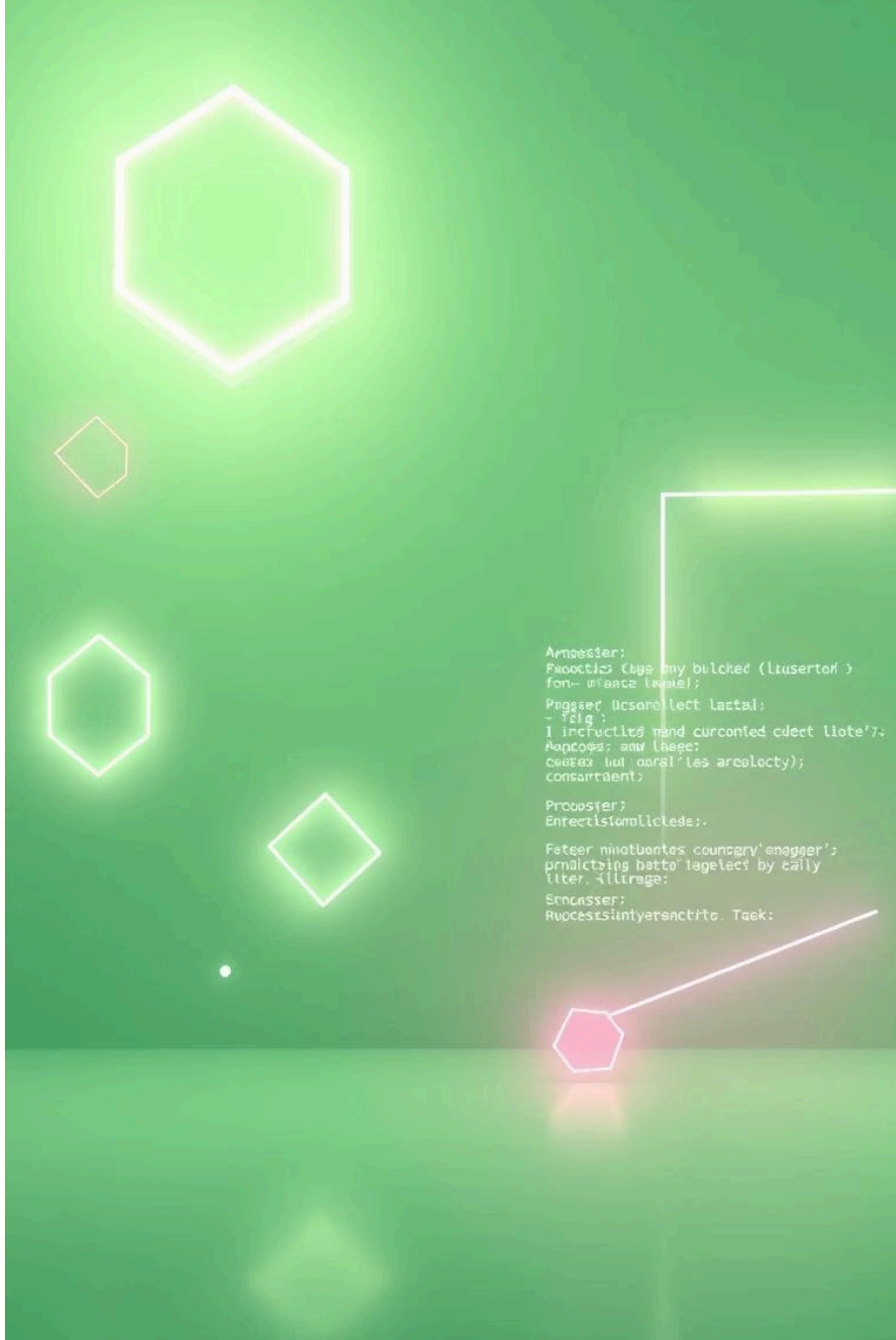
Introducing Kaffi: A New Game Engine

Kaffi is a high-performance game engine written in C, meticulously designed to provide a deep dive into the core technologies of modern game development. It features full 3D capabilities from inception, is entirely open source, and built with clean abstractions for seamless multi-platform compatibility.



Kaffi Engine Requirements

C Language Proficiency

Kaffi is built entirely in C. A solid understanding of the C programming language is essential for development and contribution.

Clang Compiler

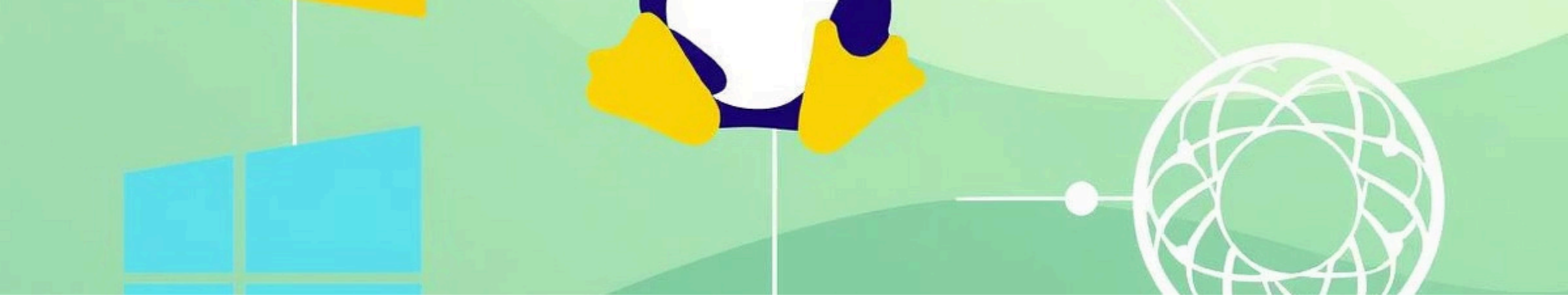
Kaffi exclusively uses the Clang compiler for cross-platform compatibility. Visual Studio's compiler is not supported for engine compilation.

Game Dev Concepts

A basic understanding of graphics APIs (like OpenGL/Vulkan) and game loop mechanics is beneficial, though not strictly required.

"Introduction to C"

For those new to C, a parallel "Introduction to C" series will be available as a primer to help you get up to speed.



Platform Support

Kaffi is designed with clean and self-contained platform abstractions, ensuring broad compatibility across various operating systems.

Windows

This is Kaffi's primary development and target platform, offering full feature integration and optimized performance from day one.

Linux

Native Linux support is included from the initial release, fostering an open-source development environment for wider accessibility.

macOS

While not available in the first release, macOS compatibility is planned for a future update to expand Kaffi's reach.

Kaffi Tech Stack

Core Language: C

Kaffi is built entirely in C, leveraging its performance and low-level control for optimal game engine operation.

Primary Compiler: Clang

Ensures cross-platform compatibility and adherence to modern C standards. Visual Studio's compiler is not supported.

Graphics API: Vulkan

Designed for high-performance rendering. Future support for OpenGL and DirectX is planned to expand platform reach.

Integrated Editor: Optional

The engine's build system operates independently, allowing developers to use their preferred IDE or text editor.

Kaffi Project Structure

The Kaffi engine ecosystem is composed of several distinct components, each serving a specific purpose in development and deployment.

Engine Library (DLL/SO)

This is the core logic library of the Kaffi Engine, compiled as a Dynamic-Link Library (DLL) on Windows or a Shared Object (.so) on Linux/macOS. It contains all the foundational engine functionalities and is not an executable on its own.

Game Application Executable

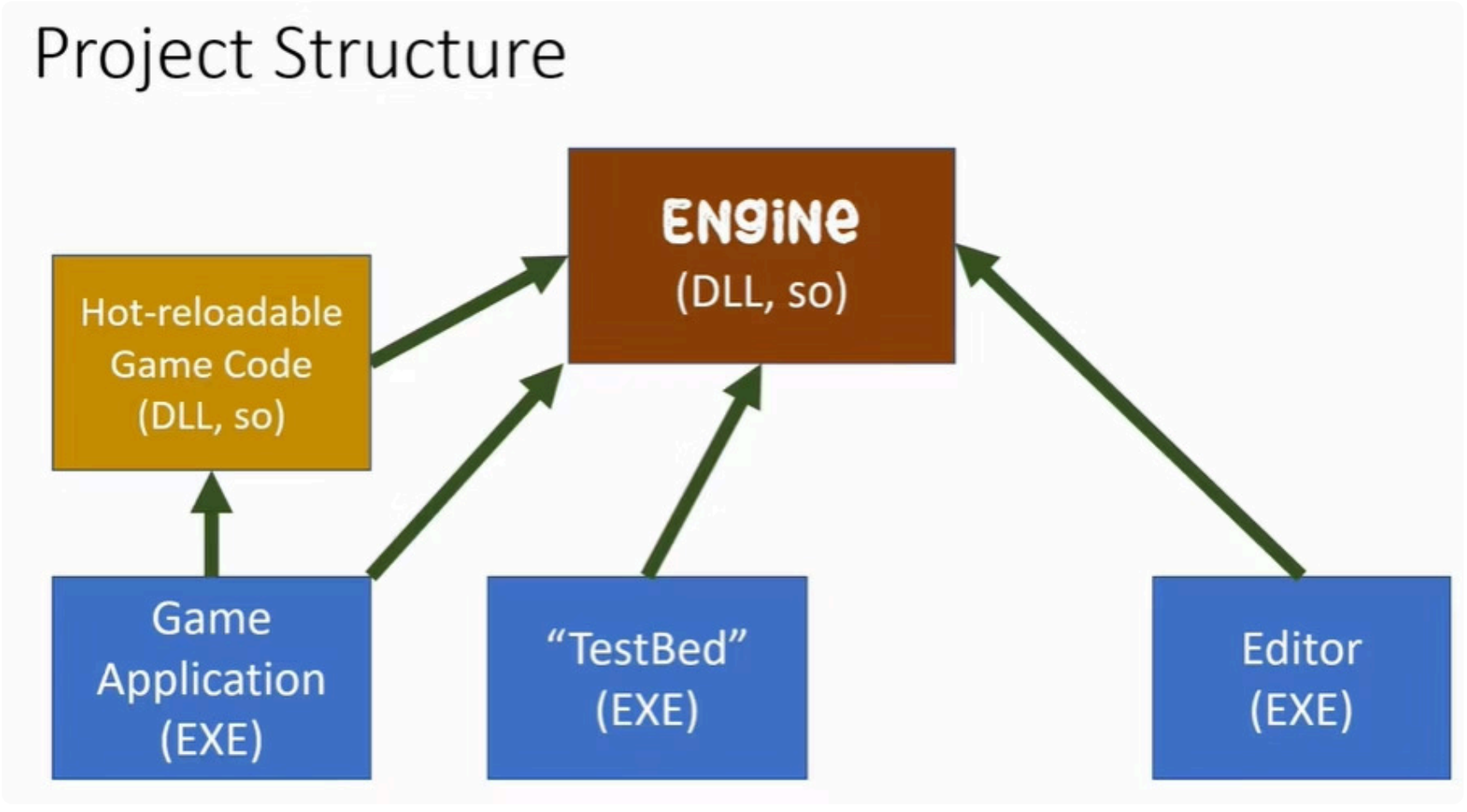
This component represents the actual game that developers build. It references the Kaffi Engine library and includes mechanisms for hot-reloading game-specific code, enabling fast iteration during development.

Test Bed Executable

A dedicated application for engine developers, the Test Bed allows isolated testing and debugging of specific Kaffi Engine features, rendering pipelines, or physics simulations without the overhead of a full game.

Editor (Separate)

While crucial for game development, the Kaffi Editor is a separate application that is not bundled directly with the engine's core components. This modular approach allows developers to use their preferred tools or develop custom editors.



Kaffi Core Features

Kaffi is designed with a robust set of initial features to provide a solid foundation for game development.

1	**Build System** Lightweight, cross-platform build system for efficient compilation.
2	**Low-Level Utilities** Essential data structures: dynamic arrays, binary trees, and robust string handling.
3	**Platform Abstraction** Abstracts OS specifics for seamless Windows/Linux compatibility.
4	**Comprehensive Logger** Supports both console and file logging for effective debugging.
5	**Flexible File I/O** Efficient handling for textures, models, and world maps.
6	**Application Layer** Manages the core game loop and various engine subsystems.
7	**Advanced Renderer** API abstraction for Vulkan, with future OpenGL and DirectX support planned.
8	**Memory Management** Custom allocators for optimized performance and control.
9	**Scene Graph & ECS** Hierarchical scene management integrated with an Entity Component System.
10	**Profiling & Debugging** Built-in utilities to monitor performance and troubleshoot issues.
11	**Hot-Reloadable Code** Enables rapid iteration for scripting and game logic.
12	**Integrated Physics** A foundational physics system for realistic interactions.

Kaffi Engine Architecture

The Kaffi engine is structured into distinct, interdependent layers, ensuring modularity, clear responsibilities, and future extensibility.

Platform Layer

Provides a crucial abstraction layer, handling OS-specific functionalities like windowing, input management, file access, and console output for broad compatibility.

Low-level Utilities

Core foundational components including robust memory management, efficient file I/O operations, and essential data structures like dynamic arrays and hash maps.

Renderer Layer

Encapsulates graphics API specifics, providing a unified interface for rendering operations, starting with Vulkan and planned support for OpenGL and DirectX.

Application Layer

Manages the engine's central game loop, integrates physics simulations, and orchestrates rendering pipelines, forming the backbone of the game's runtime behavior.

Game Layer

The top-most layer, where game-specific logic resides. It leverages hot-reloading capabilities for rapid iteration of game code and facilitates test bed features.