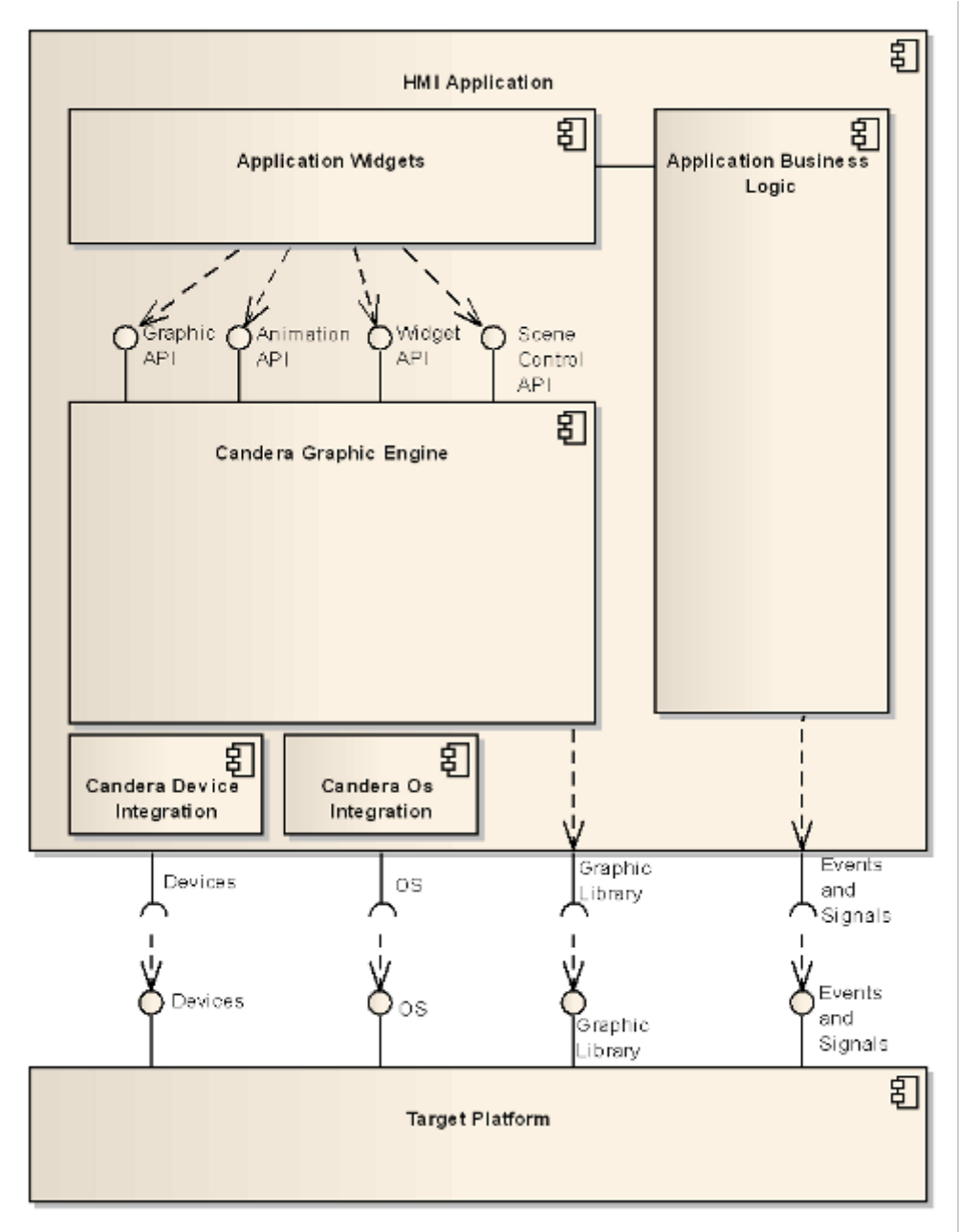
Candera Architecture Overview

Introduction

Candera Graphic Engine enables embedded applications to render and control externally designed content, especially 3D content. Applications typically do not manipulate content directly but provide widgets which basically translate high-level events into graphical representation by e.g. moving or rotate visual objects.

The figure below shows a typical architecture of a Candera-based HMI application:



Such an application consists of a 'Business Logic' component which communicates with the system environment, e.g. via field buses such as CAN, processes the signals and forwards them to application widgets on a high abstraction level (e.g. current speed in km/h). The application specific widgets visualize this information by manipulating content that is rendered via Candera.

- Candera requires small integration layers for graphical output devices, hardware accelerated or software realised basic mathematical functions, access to system memory and other platform related functionality which is typically implemented in scope of the hosting application.
- Candera provides a set of interfaces which need to be connected with platform functionality for this purpose.
- Candera is intended to drive different graphic systems, such as OpenGL ES 2.0 which are also required to be provided by the hosting platform.

Candera-based HMI applications typically define a run-time environment potentially including thread contexts and signal paths in which Candera is embedded. Candera does not define its own execution container to avoid threading and timing issues that are not transparent to the application and to provide the degree of freedom to design the application fitting best into the overall system.

Candera Graphic Engine only supports single-thread. Therefore, Candera implementation is not thread-safe to avoid potential unnecessary overhead in execution time and code-size. Hence the hosting application is required to execute any functionality on Candera from within the same thread.

Revision #8
Created 2023-02-20 06:39:44 UTC by Tsuyoshi.Kato
Updated 2026-08-24 07:34:40 UTC by Kanai Tomoaki