




1 Augmented Reality UNIX C++ Engine for Enhanced 2 Visual Guidance in Woodworking

3 **Andrea Settimi** ¹, **Hong-Bin Yang** ¹, **Julien Gamarro** ², and **Yves**
4 **Weinand** ¹

5 ¹ IBOIS EPFL, Switzerland ² Independent Researcher, Switzerland  Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright,
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#)).

6 Summary

7 Statement of need

8 Functionalities

9 Layer-stack flow

10 The layer stack is primarily responsible for managing the flow control of the AR engine.
11 Designed as a modular system, each layer encapsulates the code for a specific domain of the
12 AR application, such as camera processing, object tracking, UI, and rendering. The general
13 order and expansion of these layers can be configured in the top-level main file `ACApp.cpp`.

14 Each layer in the stack inherits from a superclass interface defined in `Layer.h`, which includes
15 event-like methods triggered at various points during frame processing (e.g., `OnFrameAwake()`,
16 `OnFrameStart()`, etc). These methods are invoked by the main `Run()` function in the sin-
17 gleton application loop from `Application.h`. This design allows application tasks to be
18 containerized and executed sequentially while facilitating data exchange between specific layers
19 through the `AIAC_APP` macro, enabling the retrieval of any particular layer data. Exchange
20 between layers can also take place in a more structured way with the integrated event system
21 (`ApplicationEvent.h`), which is capable of queuing events from layers and trigger them in
22 the next main loop.

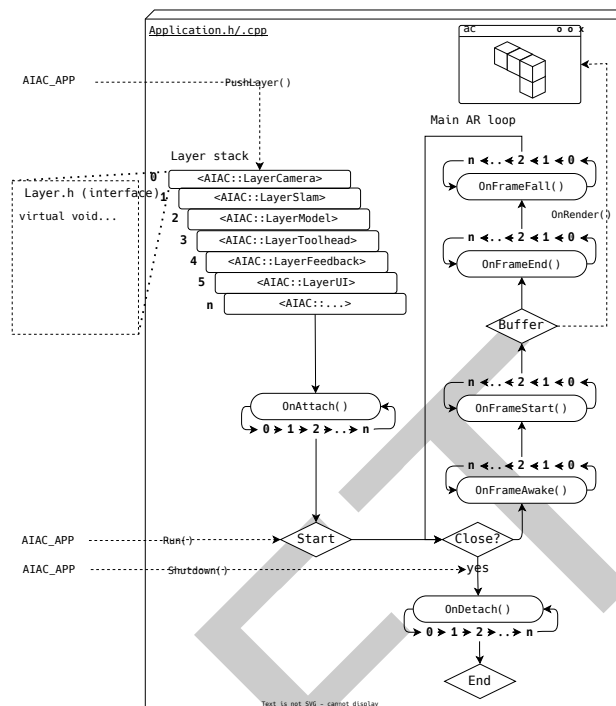


Figure 1: Illustration of the layer-stack design and the main loop for the AR engine.

Geometry framework

Computed Feedback System

The LayerFeedback.h module handles the computation of all essential data required to deliver visual guidance to the user during the fabrication process. This system occupies one of the final positions in the stack, positioned just before the LayerUI. To compute feedback, information is primarily retrieved from two preceding layers:

1. LayerModel.h: Contains the execution model and the geometries associated with the currently active hole or cut.
2. LayerToolhead.h: Provides similar information, but specific to the current toolhead attached to the tool.

Feedback is computed in tool-specific sets, categorized by tool families such as drilling (HoldeFeedback.h), circular cutting (CutCircularSawFeedback.h), and chainsaw cutting (CutChainSawFeedback.h). Each feedback category inherits from a interface class (AIAC/Feedback/FabFeedback.h), which provides top-level control functions such as Update(), Activate(), and Deactivate(). Each tool's visual guidance might consist of multiple visual cues, most of which are built on the template FeedbackVisualizer.h. These internal components (e.g. CutBladeThicknessVisualizer.h or CutPlaneVisualizer.h) manage their own geometric visual cues calculation and representation stored as graphic object (G0) instances in the belonging superclass member vector. Thus, visualization of these G0 elements, hence of the feedback itself, can be selectively enabled or entirely toggled on/off using the Activate()/Deactivate() functions.

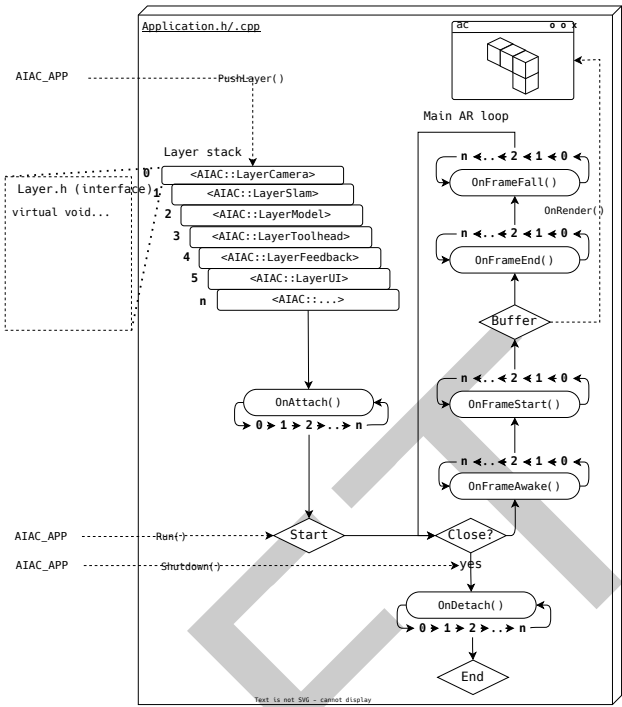


Figure 2: Illustration of the layer-stack design and the main loop for the AR engine.

44 **AR rendering**

45 **Acknowledgements**

46 **References**