Resource Summary Report

Generated by NIF on Sep 16, 2026

Fast Light Toolkit

RRID:SCR_008514
Type: Tool

Proper Citation

Fast Light Toolkit (RRID:SCR_008514)

Resource Information

URL: http://www.fltk.org

Proper Citation: Fast Light Toolkit (RRID:SCR_008514)

Description: This document defines the processes and standards that all FLTK developers must follow when developing and documenting FLTK, and how trouble reports are handled and releases are generated. The purpose of defining formal processes and standards is to organize and focus our development efforts, ensure that all developers communicate and develop software with a common vocabulary/style, and make it possible for us to generate and release a high-quality GUI toolkit which can be used with a high degree of confidence. Much of this file describes the existing practices that have been used up through FLTK 1.1.x, however I have also added some new processes/standards to use for future code and releases. The fltk-dev mailing list and fltk.development newsgroup are the primary means of communication between developers. All major design changes must be discussed prior to implementation. Specific Goals The specific goals of the FLTK are as follows: Develop a C++ GUI toolkit based upon sound object-oriented design principles and experience. (*) Minimize CPU usage (fast). (*) Minimize memory usage (light). (*) Support multiple operating systems and windowing environments, including UNIX/Linux, MacOS X, Microsoft Windows, and X11, using the native graphics interfaces. (*) Support OpenGL rendering in environments that provide it. (*) Provide a graphical development environment for designing GUI interfaces, classes, and simple programs. (*) Support UTF-8 text. Support printer rendering in environments that provide it. Support schemes, styles, themes, skinning, etc. to alter the appearance of widgets in the toolkit easily and efficiently. The purpose is to allow applications to tailor their appearance to the underlying OS or based upon personal/user preferences. Support newer C++ language features, such as templating via the Standard Template Library (STL), and certain Standard C++ library interfaces, such as streams. However, FLTK will not depend upon such features and interfaces to minimize portability issues. Support intelligent layout of widgets. Many of these goals are satisfied by FLTK 1.1.x (*), and many complex applications have been written using FLTK on a wide range of platforms and devices. Development of the remaining features is proceding for FLTK 2.0 with a new, namespace-based API. While 2.0 offers some limited 1.x source compatibility, the

changes to the underlying widget classes are significant enough to prevent full compatibility. Software Development Practices Documentation All widgets are documented using the Doxygen software; Doxygen comments are placed in the header file for the class comments and any inline methods, while non-inline methods should have their comments placed in the corresponding source file. The purpose of this separation is to place the comments near the implementation to reduce the possibility of the documentation getting out of sync with the code. All widgets must have a corresponding test program which exercises all widget functionality and can be used to generate image(s) for the documentation. Complex widgets must have a written tutorial, either as full text or an outline for later publication The final manuals are formatted using the HTMLDOC software. Sponsor. Easy Software Products

Synonyms: FLTK

Resource Type: software resource

Resource Name: Fast Light Toolkit

Resource ID: SCR_008514

Alternate IDs: nif-0000-30565

Record Creation Time: 20220129T080247+0000

Record Last Update: 20260912T195705+0000

Ratings and Alerts

No rating or validation information has been found for Fast Light Toolkit.

No alerts have been found for Fast Light Toolkit.

Data and Source Information

Source: SciCrunch Registry

Usage and Citation Metrics

We found 4 mentions in open access literature.

Listed below are recent publications. The full list is available at NIF .

Lin HY, et al. (2021) Collaborative Complete Coverage and Path Planning for Multi-Robot Exploration. Sensors (Basel, Switzerland), 21(11).

Winkler R, et al. (2015) An evolving computational platform for biological mass spectrometry: workflows, statistics and data mining with MASSyPup64. PeerJ, 3, e1401.

Khomtchouk BB, et al. (2014) HeatmapGenerator: high performance RNAseq and microarray visualization software suite to examine differential gene expression levels

using an R and C++ hybrid computational pipeline. Source code for biology and medicine, 9(1), 30.

Balteau E, et al. (2010) Improved shimming for fMRI specifically optimizing the local BOLD sensitivity. NeuroImage, 49(1), 327.