GLACI: Arbitrary Code Instrumentation Tool for OpenGL

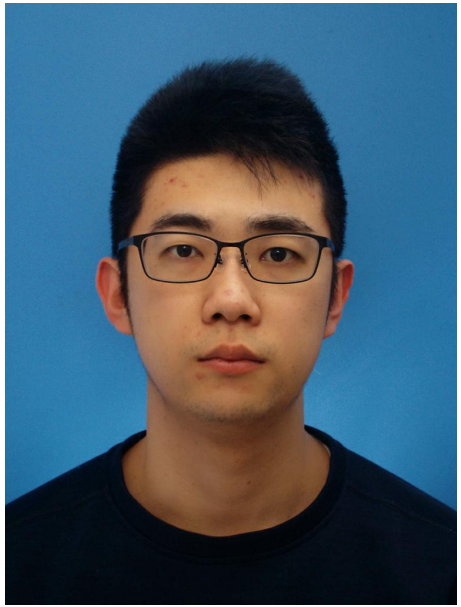
Authors: Shotaro Tsuboi, Yixiao Li, Yutaka Matsubara, Hiroaki Takada

TSUBOI Shotaro
Graduate School of Informatics, Nagoya University
Nagoya, Japan
email: s_tsuboi@ertl.jp

Resume Of Presenter

TSUBOI Shotaro (坪井正太郎)
Ph.D. Student (2nd year)
Graduate School of Informatics, Nagoya University

- Research area
  - Embedded systems
  - Real-time GPU systems
  - QoS insurance



Background

- Grows of high-performance meter clusters
- **Mixed criticality apps on single system**
- GPU is needed for complex rendoring
  - OpenGL: Graphics API most used

https://cigs.canon/article/20200423_6370.html

https://car.watch.impress.co.jp/docs/event_repo/2019aee/1186062.html

Metrics

FPS: Frames Per Second

- Critical apps must maintain certain level FPS
  - Speed, Gear, etc.
- Non-critical apps need performance
  - Maps, Video games, etc.
  - Best effort

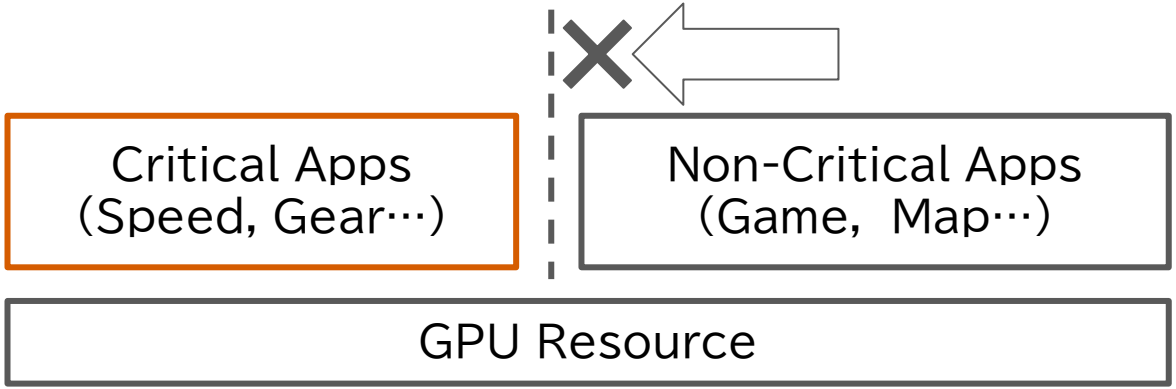
https://www.cctvsg.net/frame-rate/

Challenge

- Lack of tracing and resource management techniques
- Interference from Non-critical apps

We need a tool to

- detect performance issues
- record traces for analysis
- adjust GPU resource allocation



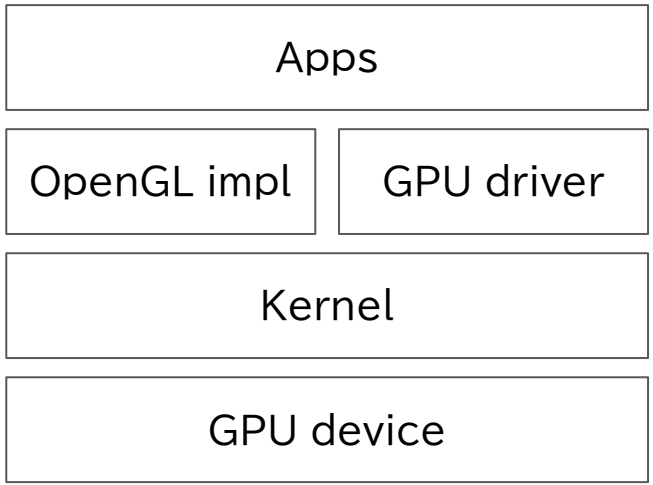
Related Work/Problem: QoS controlling

- Main targets are GPGPU programs

Some tools need to modify

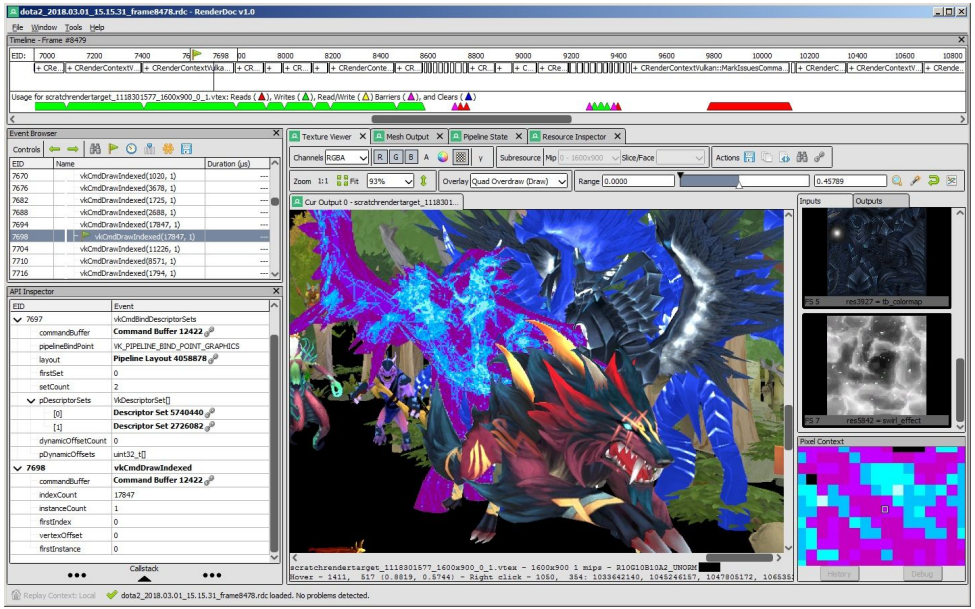
- Source code
  - Graphical apps are typically distributed as binaries
- GPU driver
  - Depends on a specific GPU model and kernel version
- OpenGL library stack
  - Unmodifiable proprietary OpenGL libraries

Real-time scheduling for 3D rendering on automotive embedded systems



Related Work/Problem: Tracing

- RenderDoc, Apitrace
  - Vendor free
  - Hooks all API calls
  - For test environment
- Intel GPA, NVIDIA Nsight
  - Vendor specific
  - Difficult to extend



These are GUI based tools.
Enables rich analysis,
but lacks generality and takes huge overhead.

Proposed Method: GLACI Overview

GLACI: An open-source instrumentation tool for OpenGL

- Enables arbitrary code injection before/after API calls
- No source code modification required
- Designed to be generic, portable, and efficient

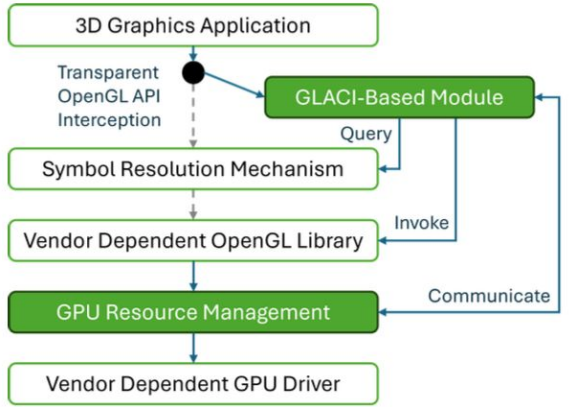
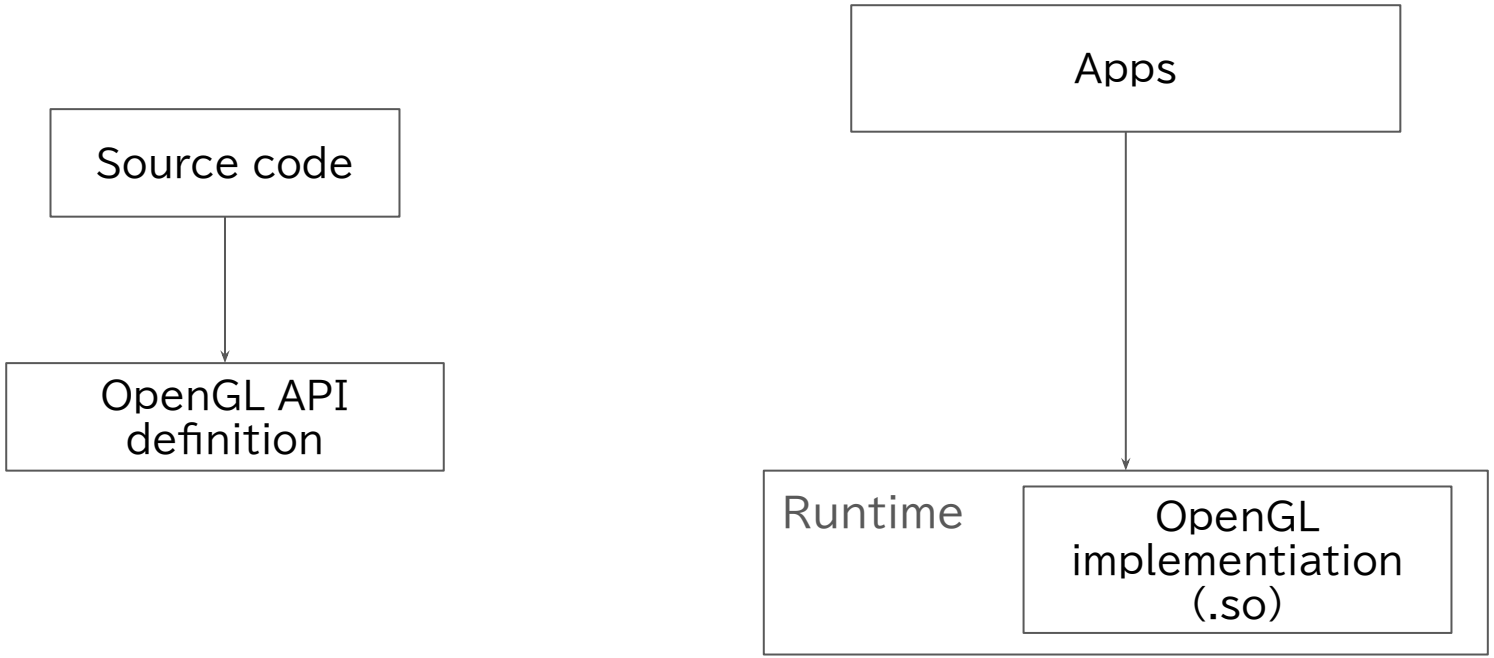


Figure 1. The overview of common functions in a GLACI-based module.

How GLACI Works

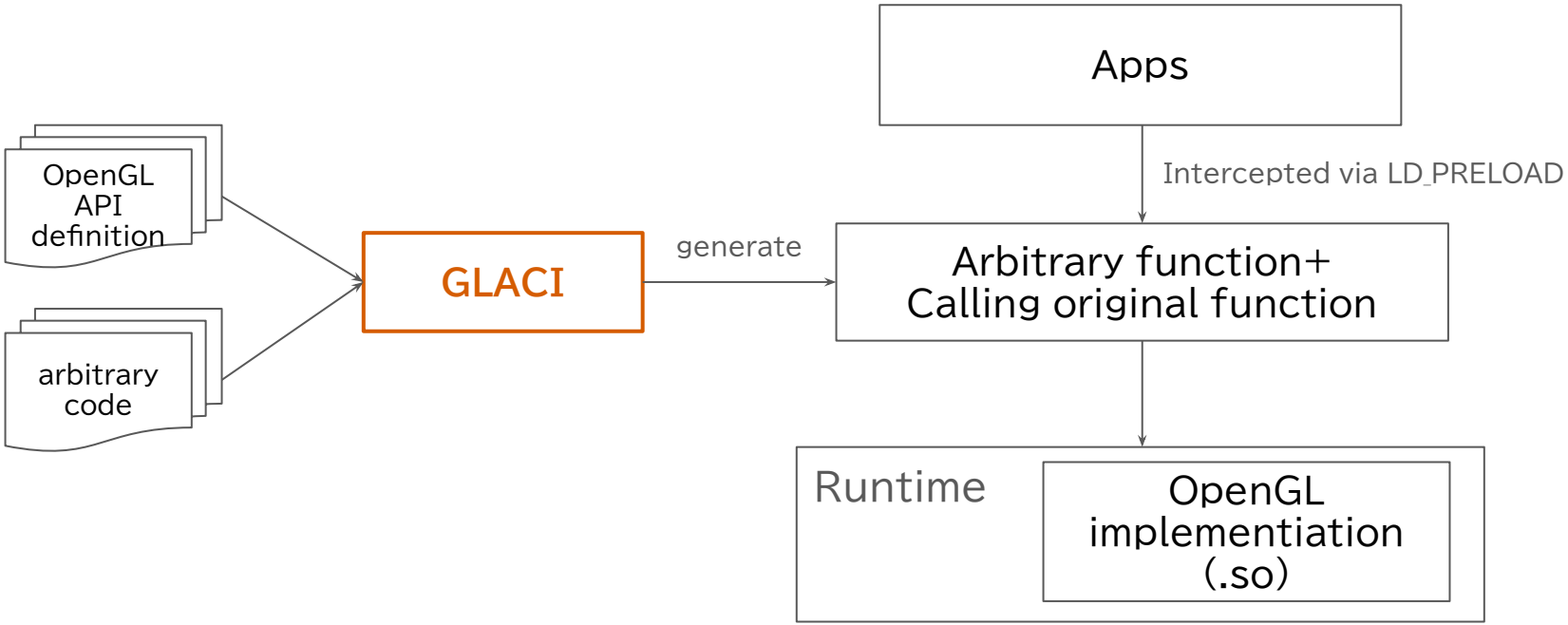
OpenGL implementation is provided as a shared lib.



How GLACI Works

OpenGL implementation is provided as a shared lib.

Uses LD_PRELOAD to transparently intercept API calls.



How GLACI Works

```
def _print_enter(f: func.Func) -> str:
    return f'std::cerr << "{f.name} Enter" << std::
        endl;'

def _print_leave(f: func.Func) -> str:
    return f'std::cerr << "{f.name} Leave" << std::
        endl;'

debug_hooks = func.Hooks(
    header="#include <iostream>",
    hook_funcs=[
        lambda f: func.Hook(
            is_target=lambda f: "glDraw" in f.name,
            before_run=_print_enter(f),
            after_run=_print_leave(f),
        ),
    ],
)
```

Figure 8. Example of hook in the instrumentation script.

```
extern "C" PUBLIC
void glDrawBuffer(GLenum buf) {
    std::cerr << "glDrawBuffer Enter" << std::endl;
    (*original_glDrawBuffer)(buf);
    std::cerr << "glDrawBuffer Leave" << std::endl;
}

extern "C" PUBLIC
void glDrawBuffers(GLsizei n, const GLenum *bufs) {
    std::cerr << "glDrawBuffers Enter" << std::endl;
    (*original_glDrawBuffers)(n, bufs);
    std::cerr << "glDrawBuffers Leave" << std::endl;
}

... // other instrumented *glDraw* functions
```

Figure 9. Example of generated wrapper functions.

Prototype: GPU Resource Manager

- FPS Monitoring & Alarms
  - Detect anomalies and trigger tracing on demand
- QoS-Aware Resource Control
  - Dynamically throttle low-QoS apps
  - Adjust rendering time for FPS limiting
- On-Demand API Tracing
  - Dynamically attach/detach probes using eBPF

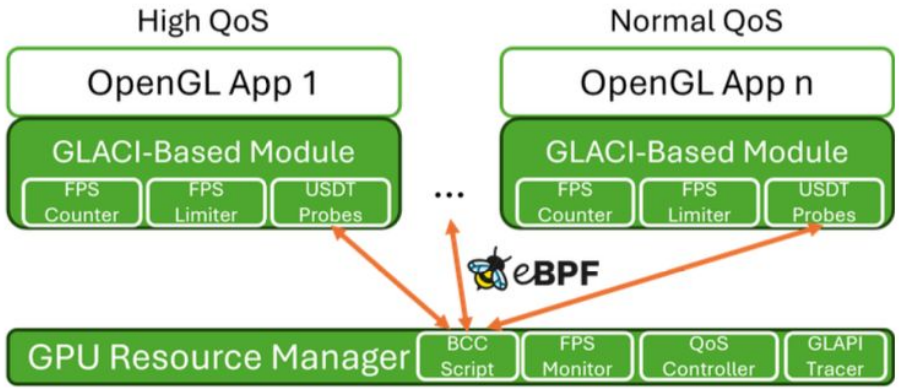


Figure 10. The overview of GPU resource manager prototype.

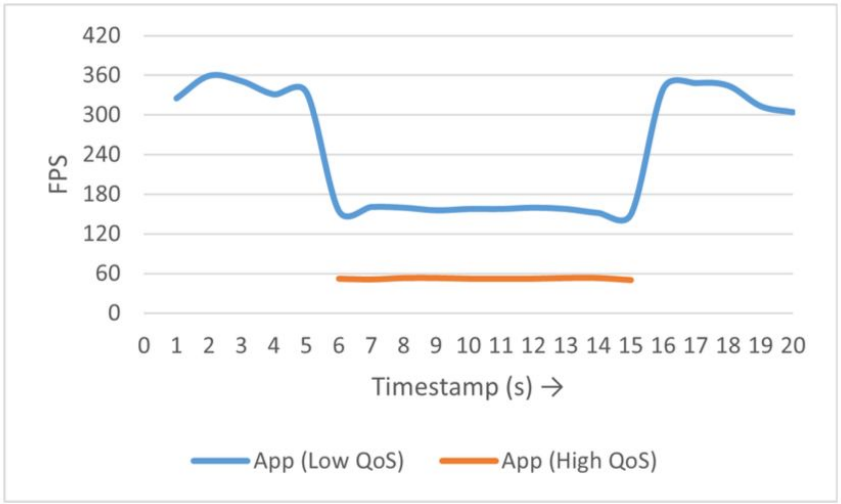
Evaluation

- Two evaluations
  - QoS control demonstration
  - Measuring overhead
- Two different platforms

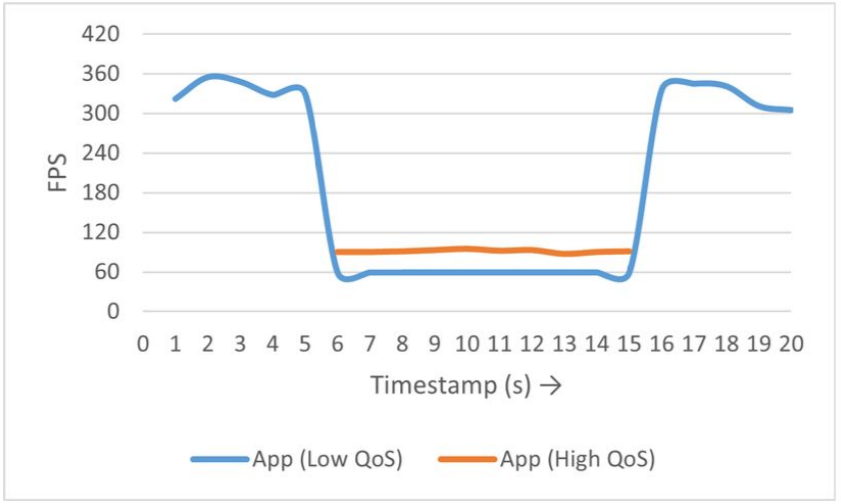
Platforms		
	Intel NUC (Open-source OpenGL)	NVIDIA Jetson (Proprietary OpenGL)
CPU	Core i5-1240	6-core Arm Cortex-A78AE v8.2
GPU	Intel Iris Xe	1024-core NVIDIA Tegra Orin
RAM	32GB	8GB
OS	Ubuntu 22.04	Ubuntu 22.04

Evaluation: QoS control demonstration

- Target FPS is 60FPS
- With GLACI, both apps meet requirements



(a) Default settings without GLACI



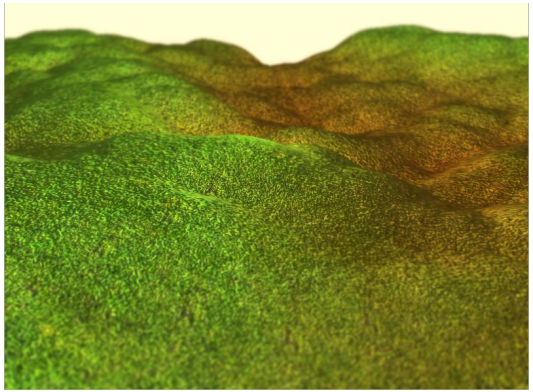
(b) Prototype of GPU resource manager and GLACI-based module

Evaluation: Measuring overhead

- 1-2FPS degradation for simple task
  - The slight performance drop is inherent to core instrumentation logic
- Enough to be used in the production

TABLE II. OVERHEAD OF THE PROTOTYPE

	`glmark2 (terrain)` Average FPS	
	Intel NUC	NVIDIA Jetson
No GLACI Loaded	222	119
FPS Monitor & Alarm	221	118
QoS-based FPS Limiter	220	118
API Tracing (NOP)	218	117
API Tracing (Logging)	212	116



glmark2(terrain)

Conclusion

GLACI: A general, portable, low-overhead framework for OpenGL API instrumentation.

- Bypasses source modification by LD_PRELOAD
- Supports real-world use cases: monitoring, QoS tuning, dynamic tracing
- Same code works across diverse platforms

Future Work

- Adaptive QoS control via real-time GPU metrics
  - (e.g., latency, frame time)
- Support for kernel-level GPU tracing
  - Integration with Gallium3D and open GPU drivers?