Porting OpenGL Implementation: PortableGL

Google Summer of Code Program 2026 Project Proposal

Name: Temirlan Emilbekov
Discord Handle: 1087702479673774101 (CurveDoor)
Discourse Username: PuplyPulpy
Email: tamiramil@proton.me
University: International University
City, Country: Bishkek, Kyrgyzstan
Phone Number: *TBR*
Emergency Contact: *TBR*
Telegram: @emilbektemir

The document was originally written in Markdown, this might cause some issues while converting

Project Abstract

PortableGL is an OpenGL 3.x subset implementation in a single-header C99 library, actively maintained by Robert Winkler. The main goal of the project is to bring a modern programmable graphics pipeline to RTEMS by integrating PGL with RTEMS frame buffer driver, providing a familiar OpenGL-like API for developing and porting graphical applications. The result will include an RSB recipe, a compatibility layer, documentation and test suite validated across multiple BSPs.

Project Scope

Medium (approx. 175 hours)

Project Description

Current graphics support in RTEMS is limited to basic image decoding (libpng, libjpeg), the legacy Microwindows/Nano-X environment, and decade-old RTEMS Graphics Toolkit, not counting the frame buffer driver. Microwindows provides windowing capabilities, but lacks support for a modern programmable graphics pipeline. One can see some gap between low-level pixel processing and the high-level applications needs. PortableGL bridges this gap by providing an API equivalent to OpenGL 3.x that can work with existing libraries such as FreeType or libpng/libjpeg/libtiff but offers significantly greater rendering power for complex scenes.

The project aims to provide hardware-agnostic (given that the BSP has the frame buffer driver support), CPU-only, lightweight, dependency-free OpenGL implementation support. The project should provide the familiar OpenGL interface to facilitate native graphics applications development, as well as porting of existing ones.

It was decided to port the PortableGL library as the most realistic starting point for OpenGL support. The fact that the implementation is a single-header (for user) library, written in pure C99 with minimal dependencies (anyway covered by RTEMS) makes it a good candidate.

The compatibility layer (next: CL) will be a single-header library that initializes the framebuffer, configures the PortableGL library parameters using the framebuffer driver interface, and maps functions and introduces new ones for abstracting setup, completion, and other procedures requiring RTEMS-specific instructions to provide OpenGL-like interface.

The project will be backed by comprehensive documentation and a set of validation test cases to ensure the library won't die a quiet death. I plan to develop a dedicated "Getting Started" guide, covering the entire flow, in addition to API documentation. The validation block will include a suite of test cases designed to ensure the implementation works across different BSPs with different video output interfaces.

Technical Approach

The solution will consist of three components:

- An RSB recipe that downloads and installs the header files for both PortableGL itself and the compatibility layer;
- A compatibility layer, which is a single-header library too that connects PortableGL and the RTEMS framebuffer driver: it initializes the framebuffer via the RTEMS driver interface, configures PGL macros according to the BSP, and encapsulates setup and shutdown procedures.
- A set of tests and demo applications that verify integration with multiple BSPs.

Why I Chose This Project

The reasons, among other ones, include the fact that the topic of graphics is quite close to me as a person who is interested in video game development as well. I also just like it and find it cool.

Plan Of Maintenance

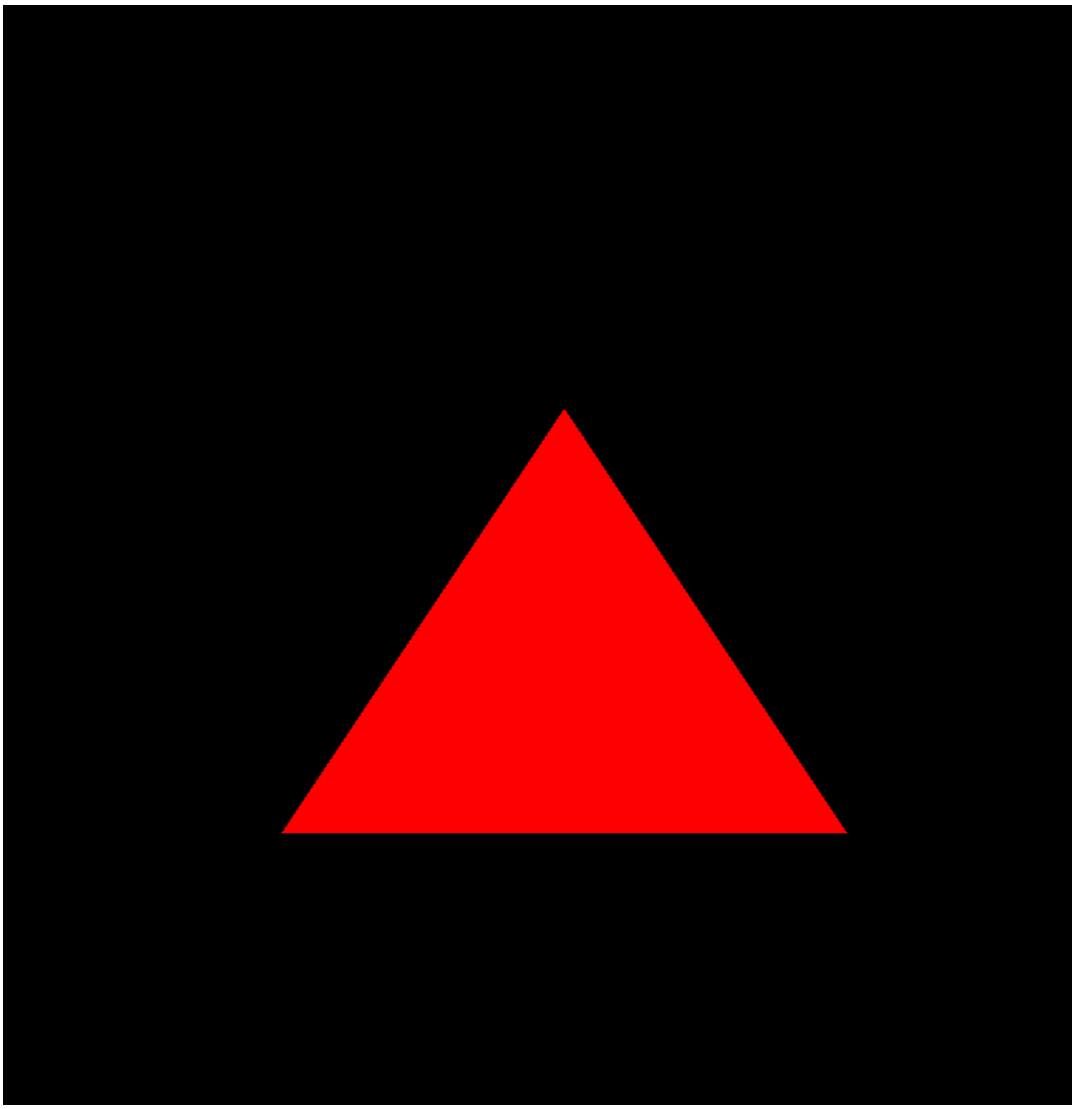
Of course, life happens, but I intend to stand by the project.
Post-project maintenance plan would include the following points:

- **BSP & Driver Support.** If kernel changes break the CL, I will release a patch for the adapter;
- **Debugging based on community feedback;**
- **Regression Testing.** Periodic launch of demo apps/test suite on supported BSPs when new major RTEMS versions are released.

Preliminary Work

I have done some initial work and research within submission period, which include:

- [x] Modify, build and run the “Hello World” application;
- [x] Started with modifying a couple of PortableGL tests to build and run on i386/pc386 BSP with graphics output via frame buffer driver. As a result the triangle is drawn on the screen via QEMU emulator;



- [x] Created initial RSB recipe, which was reviewed and merged to the upstream.
  Tested on BSPs: i386/pc386, arm/beagleboneblack, arm/raspberrypi;

- [x] Conducted a study of the source code of the library and the RTEMS graphics stack. As a result, this led me to contribute to the library upstream.

Project Deliverables

- GSoC Timeline: *GSoC Timeline*;
- **June 7 (coding begins):**
  - Working RTEMS environment is set up and verified;
  - RTEMS Source Builder fork is created with access granted to mentors, backup mentors, RTEMS organization administrator, and other community members;
  - CL repository is created with access granted to appropriate members.
- **July 12-16 (Midterm Evaluation):**
  - A working demo of the "Rotating Cube" on QEMU;
  - The code is integrated into the build tree;
  - Initial test suite passing on i386 and ARM BSPs.
- **August 16-23 (Final Evaluation):**
  - Complete test suite with documented results across tested BSPs;
  - Documentation published and reviewed;
  - Final completion of RTEMS-PortableGL compatibility layer.
- **August 31 (Final Results Announced):**
  - Create a gallery (a directory in the CL repo) of demos across all tested BSPs;
  - Clean up documentation and sources.
- **Post GSOC - What will I do?**
  - Maintain the library in the context of RTEMS;
  - Perhaps, improve RTEMS graphics stack, e.g. expand the frame buffer driver support.

Proposed Schedule

March 16 - March 31 (Application Period)

- Communicate with RTEMS community and potential mentors to clarify project details; maintain contact with PortableGL author.
- Research the code base of the frame buffer driver and PortableGL.

March 26 - April 22 (Acceptance Waiting Period)

- Prototype potential solutions and continue source code exploration.
- Continue communicating with the community.
- Will be partially focused on local hackathon organization and management.

April 23 - May 23 (Community Bonding Period)

- Ensure the RTEMS environment and required tool chain is set up.

- Observe RTEMS style guides and standards.
- Finalize technical discussions with mentors.

May 24 - June 26 (First Half)

- Weeks 1-4. Initial porting:
    - Set up pixel formats (through frame buffer driver and PGL macros);
    - Create initialization and closure functions;
    - Add the CL library to the build set;
    - Set up PGL macros to fit the given BSP.
- Weeks 5-6. Test suite:
    - Adapt list of native library tests;
    - Implement the rotating cube demo.

During weeks 1-4, integration is verified through ad-hoc tests and visual output checks. These are then formalized in Weeks 5-6.

June 27 - August 23 (Second Half)

- Weeks 7-9. Expanding test suite and polishing:
    - Expand test suite: basic OpenGL primitives, texture mapping, edge cases;
    - Complete test suite with text rendering test-cases (FreeType integration);
    - Debug critical issues, polish and finalize CL.
- Weeks 10-12 / Parallel:
    - Document the project (API reference, guides);
    - Fix bugs based on mentor feedback;
    - Prepare demo gallery across tested BSPs (i386, ARM).

Completion Criteria

The project will be considered completed if the following is reached:

- PortableGL is integrated into the RSB and can be built for architectures: ARM, i386 (support the frame buffer driver);
- The CL library abstracts the RTEMS-specific instructions and provides an OpenGL-like entry point;
- Test suite covers basic OpenGL primitives and texture mapping; Demos work as expected;
- The documentation is published on the RTEMS Docs.

Future Improvements

The list of future improvements in support of the library could include:

- Rendering optimization on algorithmic CPU (e.g. SIMD) and RAM (per-line rendering?) levels.

- Architectural improvements. This could include concurrency and thread management improvements addressing different scenarios. E.g. what if multiple tasks attempt to access the frame buffer?
- Further integrations of RTEMS graphical stack and/or input & event management.

As a more realistic stretch goal if the project is done earlier than scheduled, I would consider implementing dirty rectangle optimization first. Only the changed regions are copied to the display. This will require tracking modified regions, and modifying the flush procedure accordingly. Requires no additional dependencies and addresses the CPU performance challenges.

Continued Involvement

I would like to continue improving the support of the library in the future and maintain the graphics stack in RTEMS, perhaps being involved in porting Mesa3D. It is possible that my interest may also extend to other areas of RTEMS. I don't rule out also the possibility that I might want to mentor others in this area in context of GSoC and in common in the future.

Other Commitments

I have final examinations at my university from April 27 to May 15. These exams will not create a large load.
Additionally, I have seasonal family commitments related to agricultural work (fieldwork) throughout the summer. It takes about 2 days per week. However, it will not cause any problems with allocating an average of 20-30 hours per week.

There are no legal restrictions or copyright claims from my university or anyone else regarding the code I develop for GSoC.

Eligibility

I solemnly confirm my eligibility to participate in the GSoC 2026 program in accordance with the rules:
https://summerofcode.withgoogle.com/rules/

Major Challenges foreseen

- **BSP compatibility**
  Although the frame buffer driver provides the same interface regardless of hardware, not all capabilities have been tested across BSPs. For example, pixel format and memory alignment may vary — while the driver exposes hardware parameters and PGL allows defining corresponding macros, the actual behavior on untested BSPs remains uncertain.
- **CPU performance**
  Getting acceptable or adequate framerates on low-power CPUs can be challenging, since PortableGL is a CPU-based renderer, especially with complex scenes.

- **Memory constraints**
  Managing framebuffer allocation alongside PGL's internal buffers may require more careful tuning per BSP, particularly on memory-constrained targets.

References

- PortableGL | GitHub
- RTEMS BSP and Driver Guide | Docs
- RTEMS 3rd Party Packages | Docs
- RTEMS BSP | Docs
- RTEMS Frame Buffer Docs
- VESA based frame buffer utilizing real mode interrupt 10h | commit
- POSIX API Configuration | Docs

Relevant Background Experience

I have experience of working with CAD APIs — netDxf, BricsCAD (see Coswalt, BricsCAD Plugins). Since it requires working with under-documented APIs, it also requires doing reverse-engineering and source-code analysis. I'd say it taught something. I'm into gamedev-related stuff as well and love math (have some math-related projects. see linal, ascii-water) so dealing with graphics won't be something alien for me. I lead a community myself and that's made me understand the importance of open communication. I believe it's quite enough to be called qualified for bringing this problem to the finish: engineering stubbornness, passion and communication.

Personal

I'm Temirlan Emilbekov, a 2nd-year Computer Software Engineering student at Ala-Too International University, Bishkek, Kyrgyzstan. I believe the interest in embedded development is the expected result of the question "Nahhh, how it actually works", asked from time to time, which led me lower and lower till here.

I was just looking for something interesting and something bigger than me. Then I found GSoC and after researching the options I loved RTEMS. It was important the challenge to be low-level and hard enough. I also wanted to take part in something related to Aerospace, since I love aircraft and space. So the RTEMS was the only right answer.

It's not in our tradition to apply the word "special" to ourselves but I could mention my experience as a founder and a leader of a student game dev club in my college. We organized (and organize) one of the first gamedev-related events in Bishkek, collaborating with other communities, universities and local professionals.

Experience

Free Software Experience/Contributions:

- Contribution to PortableGL upstream.

Language Skill Set

- **Better than intermediate:** C#
- **Intermediate:** C/C++, Bash
- **Familiar:** Python

Reference Links and Web URLs:

- GitHub
- LinkedIn