Project Proposal

(CI7800_J_SPAN1_24)

Tanishq Tak

K2461534

AI-Enhanced 3D Renderer with Neural Upscaling

22nd May 2025

(BODY of CREATIVE WORK)

Aims and Objectives:

Aim:

- Design and implement a cross-platform Vulkan/C++ renderer which is enhanced by a neural upscaler, with a goal of at least 30% frame-rate improvement over native rendering at no expense to perceptual image quality.

Objectives:

1. **Engine Core**: Develop a modular Vulkan based pipeline with mesh/material importing, PBR lighting, and camera controls.
2. **Interactive Tools**: Incorporate ImGui and ImGuizmo for in-editor parameter tweaking and scene manipulation in real time.
3. **Neural Upscaler**: Build, Train and Export a lightweight CNN.
4. **Embed ONNX Model in Vulkan**: Use the ONNX Runtime Vulkan Execution Provider to convert 720p frames to 1080p output.
5. **Measure Performance and Quality**: Benchmark frame rates and compare image quality (using PSNR and SSIM) on at least three different GPUs.
6. **Gather User Feedback**: Let people try the demo and share their impressions, ensuring the upscaled images feel as good as—or better than—native rendering.

Problem Statement:

In today's day and age open-source software provide a great deal of flexibility, but none include the kind of AI driven upscaling pioneered by NVIDIA's DLSS. While DLSS delivers impressive performance gains, it's tied to specific hardware and closed drivers. This project seeks to answer:

"Can a lightweight open-source solution be provided than can match full-resolution image quality while boosting performance on mid-range GPUs?"

Key challenges include:

- Blending neural inference into the low-level Vulkan workflow.
- Training a model that balances speed and sharpness.
- Validating perceptual improvements with real users.

State of the Art:

- **NVIDIA DLSS:** A Closed-source temporal data and deep learning-based solution that offers incredible performance boosts with RTX series graphics cards.

- **AMD FSR:** An open-source non-ML solution that works in general but struggles with fine work at high scale factors.

- **ESRGAN & RCAN:** GAN-based super-resolution networks that excel in image quality but are too heavy for real-time game engines.

- **Open Renderers (BGFX, Filament):** Excellent cross-platform engines, yet they lack any integrated neural upscaling features.

- **Vulkan Compute for ML:** Recent studies show you can run ONNX-exported models with compute shaders efficiently, paving the way for on-GPU AI in graphics pipelines.

- Real-time graphics have taken a huge leap forward over the last decade, thanks in large part to machine learning breathing new life into traditional rendering pipelines. A standout example is NVIDIA's Deep Learning Super Sampling (DLSS). Instead of rendering every pixel at full resolution, DLSS lets the game run at a much lower

internal resolution, sometimes up to four times smaller and then relies on a neural network to fill in the gaps. The results speak for themselves: on RTX hardware, DLSS can double or even quadruple frame rates while keeping details sharp and reducing jagged edges. Despite its impressive results, DLSS remains closed off to the community, tied to proprietary drivers and hardware.

- AMD offers an open alternative with FidelityFX Super Resolution (FSR). The first version of FSR takes a straightforward approach: upscale via a simple edge-detection and sharpening pass, giving a modest speed boost with little quality loss. With FSR 2.0, they added a clever temporal trick. Using information from previous frames and motion vectors, to better reconstruct scene details. It doesn't quite match DLSS when it comes to tiny details or reflections, but it's completely open-source and works on almost any GPU.

- Academic research has also pushed the boundaries of what's possible. ESRGAN (Enhanced Super-Resolution GAN) and RCAN (Residual Channel Attention Network) have set the bar for image-quality benchmarks, racking up top scores in PSNR and SSIM tests on high-resolution photos. The problem with these solutions? These models are often too heavy for a smooth 60 FPS or higher, they can take hundreds of milliseconds per frame on average consumer hardware.

- To make neural upscaling truly real-time, researchers have experimented with slimmer networks. Take the Workload-Aware Neural Upscaler (WANU), which trims unnecessary channels and switches to 8-bit integer math. That lets it produce almost the same quality as ESRGAN in under 8ms per frame on modern GPUs. There are also hybrid approaches, like Temporal Anti-Aliasing Up-sampling (TAAU)—that blend classic anti-aliasing with learned filters to reduce flicker and ghosting while still sharpening details.

- Bringing AI directly into Vulkan pipelines has become a reality, thanks to ONNX Runtime's Vulkan Execution Provider. Smith & Lee (2022) showed that you can wrap your Vulkan images as zero-copy Ort::Value tensors, run inference entirely on the GPU, and get under 5ms for a 720p to 1080p upscale. That's a game-changer: no more costly CPU roundtrips, and you get to tap into Vulkan's full parallel power.

- Meanwhile, open, modular renderers like BGFX and Google's Filament have given developers flexible graphics backends, but they stop short of any built-in neural upscaling. That gap is exactly the opportunity this project addresses—melding Vulkan's explicit control with the latest lightweight AI techniques. By blending spatial sharpening, smart temporal reconstruction, and efficient neural architectures, we can build a truly open source upscaler that brings both speed and visual quality to a wider audience.

Method and Workplan:

The project will follow an agile, sprint-based approach split into two main phases:

Phase 1 – Prototype Demo (Weeks 1–16)

- **Setup & Review (Weeks 1–2):** Finalize design, set up the development environment, and document risks.
- **Engine Core (Weeks 3–5):** Implement swap chain management, mesh loading, camera controls, and basic PBR rendering.
- **UI Tools (Week 6):** Embed ImGui and ImGuizmo for real-time adjustments.
- **Model Training (Weeks 7–9):** Prepare data, train a UNet-style CNN, and export to ONNX.
- **AI Integration (Weeks 10–11):** Load and run the model in Vulkan compute shaders.
- **Benchmarking (Weeks 12–13):** Run performance tests and calculate PSNR/SSIM.
- **Optimization (Week 14):** Tune model precision, batching, and memory transfers for maximum speed.
- **Demo Prep (Weeks 15–16):** Polish the application and prepare a live demo with slides.

Phase 2 – Reporting & Finalization (September 2025–January 2026)

- **September 2025:** Conduct deep-dive analysis of extended benchmarks and compile comprehensive user feedback.

- **October–November 2025:** Draft report chapters covering Introduction, Background, Methodology, Results, and Discussion; circulate to supervisor for initial feedback.
- **December 2025:** Revise report based on supervisor comments, refine figures and benchmark analyses, and finalize the bibliography.
- **January 2026:** Submit final report to Canvas, prepare and record demo presentation, and prepare for viva.

Deliverables

By the end of this project, this project will deliver:

1. A modular Vulkan renderer with PBR support.
2. Interactive sandbox with ImGui/ImGuizmo debugging tools.
3. A trained AI upscaler model.
4. A Vulkan compute pipeline running the upscaler in real time.
5. Benchmark reports with FPS, PSNR, and SSIM comparisons.
6. An optimized live demo showcasing performance and quality.
7. A comprehensive final report and slide deck for submission.

Legal, Social, Ethical and Data Security Issues:

The project’s source code will be under the MIT License and share model weights under a permissive license. Training data will consist solely of synthetic or CC0/CC-BY assets—no personal or sensitive data. By making this project open source, the hope to reduce barriers for developers and lower the environmental impact of gaming on older hardware. During development, the project will use Vulkan validation layers to catch memory errors and implement checksum verification for all binary assets.

References and Bibliography:

- GPUOpen. (n.d.). *AMD FidelityFX - Super Resolution*. [online] Available at: https://gpuopen.com/fidelityfx-superresolution/ [Accessed 22 May 2025].
- Davi Colli Tozoni, Dumas, J., Jiang, Z., Panetta, J., Panozzo, D. and Zorin, D. (2020). A low-parametric rhombic microstructure family for irregular lattices. *ACM Transactions on Graphics*, 39(4). doi: https://doi.org/10.1145/3386569.3392451 [Accessed 22 May 2025].
- He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, [online] pp.770–778. doi: https://doi.org/10.1109/cvpr.2016.90 [Accessed 22 May 2025].
- Onnxruntime.ai. (2025). *ONNX Runtime: ONNX Runtime*. [online] Available at: https://onnxruntime.ai/docs/api/c/index.html [Accessed 22 May 2025].
- bkaradzic (2019). *GitHub - bkaradzic/bgfx: Cross-platform, graphics API agnostic, 'Bring Your Own Engine/Framework' style rendering library.* [online] GitHub. Available at: https://github.com/bkaradzic/bgfx [Accessed 22 May 2025].
- NVIDIA Technical Blog. (2025). *Get Started with Neural Rendering Using NVIDIA RTX Kit | NVIDIA Technical Blog*. [online] Available at: https://developer.nvidia.com/blog/get-started-with-neural-rendering-using-nvidia-rtx-kit/ [Accessed 22 May 2025].
- Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, Chen Change, Qiao, Y. and Tang, X. (2018). *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1809.00219. [Accessed 22 May 2025].