PS-1
Seminar II

Pathik Shah (2021A7PS2085G), Joel Tony (2021A7PS2077G), Aayush Kumar Singh(2021A7PS0430H)

Coditation Systems Pvt. Ltd.

Coditation Systems

- Coditation was founded by Chetan Saundankar in 2017 with a vision of creating a boutique firm specializing in application of emerging technologies
- A team of skilled software architects, engineers, and data scientists who excel at solving complex problems and delivering high-quality solutions.
- Expertise in Graphics Programming, Realtime Apps and WebRTC.
- Mature data science team to build cutting-edge ML and AI products.
- Specialized in creating exceptional user experiences across UIs, Micro-frontends, and Mobile Apps.

Outline

About Coditation Systems

1. Introduction
2. Initial Divisions
   a. DevSync
   b. Compiler Infrastructure
   c. WebRTC
   d. WebGPU
3. WebGPU
   a. Basic Implementation
   b. gLTF Model Rendering
   c. Challenges
4. Point Clouds
   a. Open 3D
   b. Three js

Project Boxxy

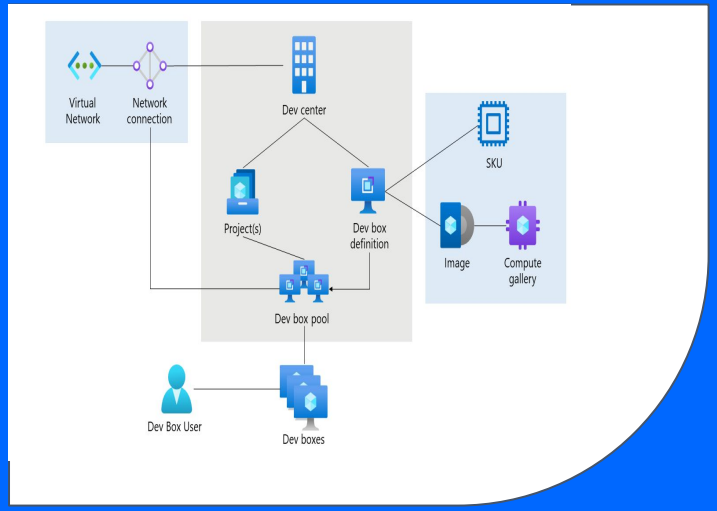
For developers and researchers working on resource-intensive tasks, Project Boxxy aims to provide affordable and accessible GPU resources. High-end GPUs are currently prohibitively expensive for both individuals and businesses. We mainly aim to solve this problem by providing a sandbox environment for the creation and deployment of graphic intensive scenarios.

The project also aims to support point cloud rendering of 3D models and scenarios in the future

Initial Divisions

DevSync

- Enable **hot reloading** of graphics viewport
- Seamless development experience for remote graphics developers
- Integration of development tools and frameworks for live code updates
- Implementation of **hot module replacement** for improved efficiency
- Facilitating remote collaboration on graphics payloads

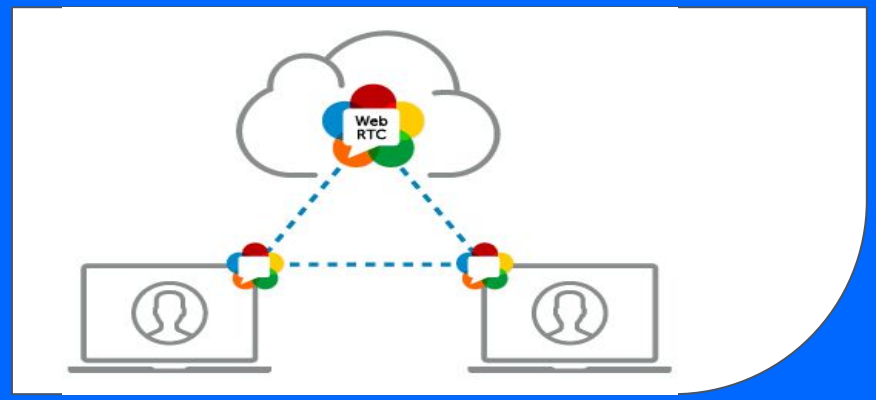


COMPILER INFRASTRUCTURE

- Explored linking libraries in large-scale projects using different build systems.
- C++ compilation process deepdive
- Read about task-based and artifact-based **Build Systems.**
- Chosen approach: **CMake** for managing the C++ project and **Bazel** for **remote caching** as the project scales in future.
- Gained insight into linking libraries with **CMakeLists** and integrating package managers like **vcpkg** as **git submodules**.

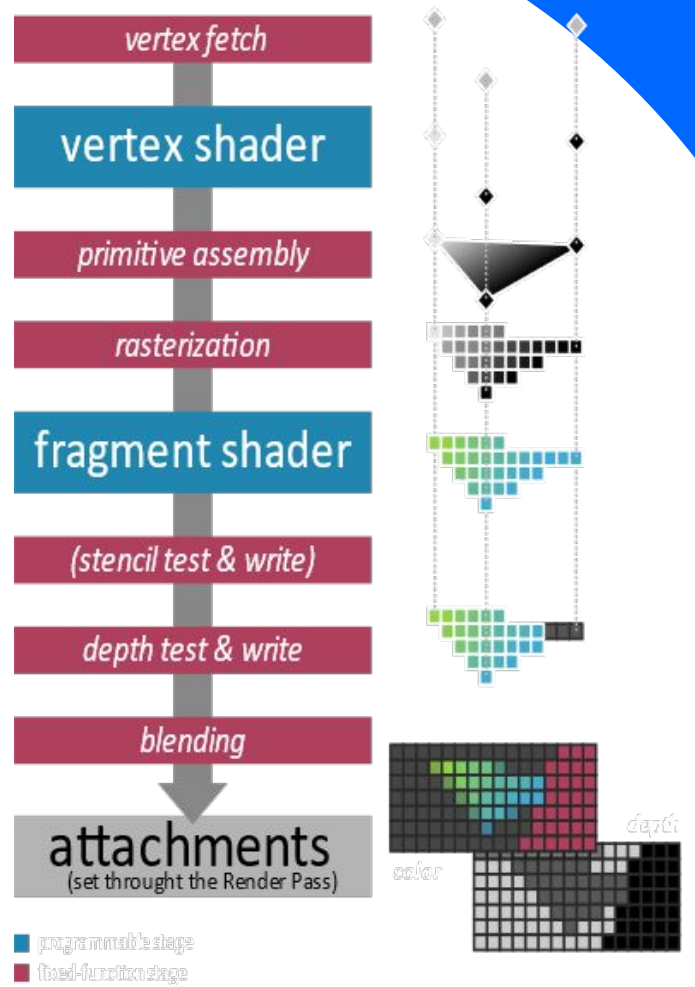
WebRTC

- The WebRTC team had to figure out how to stream the graphics rendered to the web
- They also had to make sure that the stream would support a variety of the most popular browsers that the current consumers use like Google Chrome, Mozilla Firefox, etc.
- To ensure secure and efficient streaming of the graphics data



WebGPU

- **Graphics Programming Fundamentals:** Covers vertex buffers, vertex shaders, and fragment shaders.
- **Rendering Pipeline**: Vertex Fetch,Vertex Shading, Rasterization, Blending
- **Shading Languages:** Introduces **WGSL** and supports GLSL and SPIR-V.
- **WebGPU:** Cutting-edge low-level graphics API succeeding WebGL, ensuring platform agnosticism.
- **Core Engine Features:** glTF(gl Transmission Format) loader and covers **pipeline layout**, **render architecture**, **swap chains**, and **texture views**.



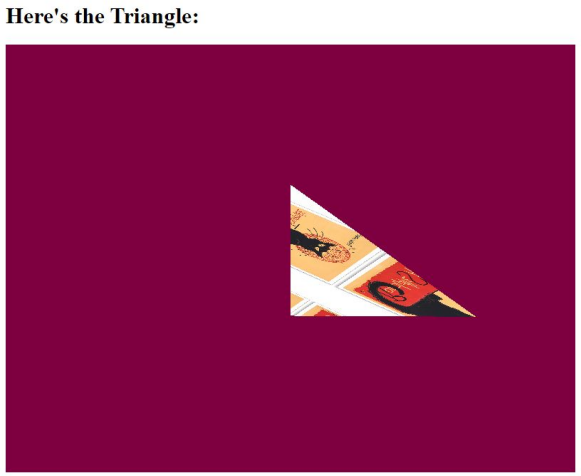
- **WebGPU Backends:** Backend implementations by Chromium (Dawn (C++)) and Mozilla (WGPU-Native(Rust)).

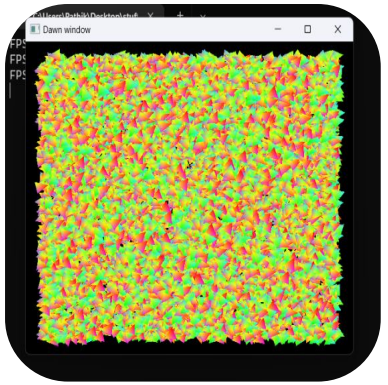
(Chromium)

wgpu-native (Firefox)

Basic Implementations

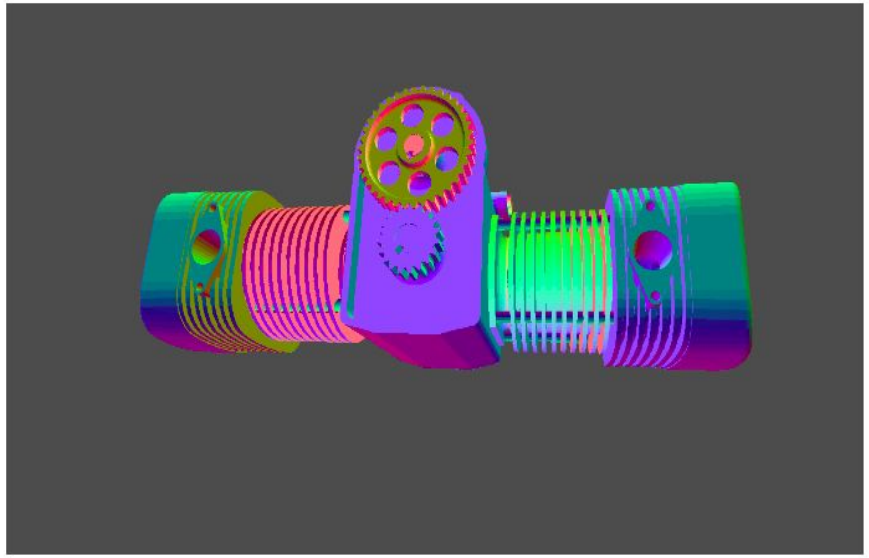
- Triangles and Squares using **OpenGL** for understanding fundamentals.
- Hello-triangle on our machines using **WebGPU-Native**
- Textured triangles using **WebGpu-Web**.
- A starter project involving **glTF loader**.





glTF Model

- **GLTF (GL Transmission Format):** Open standard file format for 3D scenes and models, designed to be lightweight and efficient for use in real-time applications.
- Developed by the **Khronos Group**(consortium that developed the OpenGL and Vulkan).
- Compact binary format that can be easily loaded and rendered by graphics engines and web browsers.
- **Platform-independent**

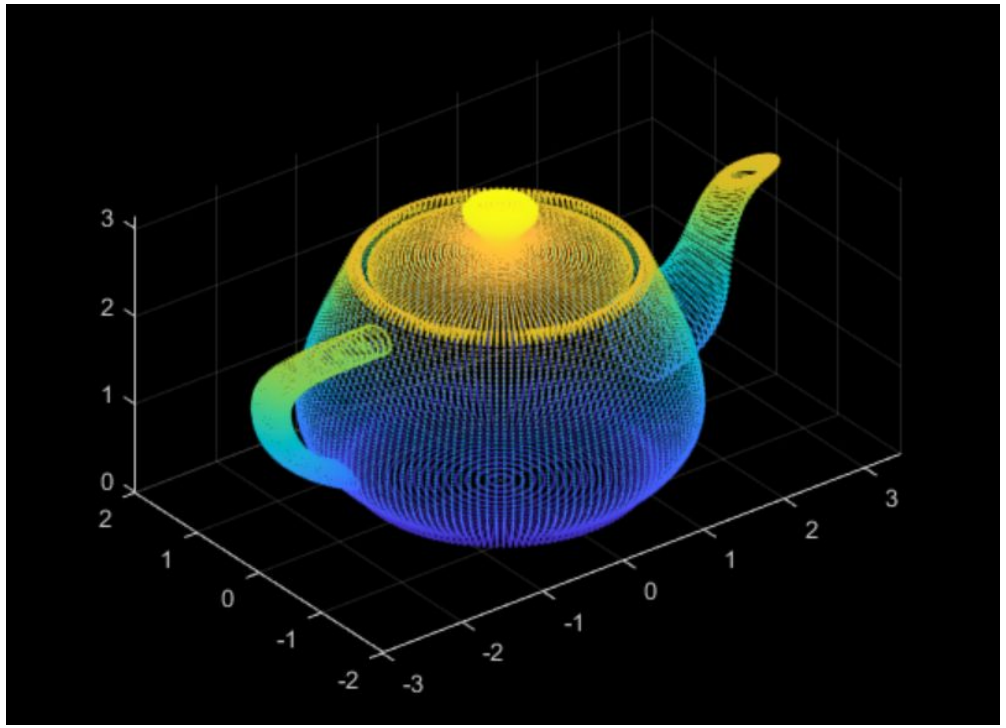


Challenges Faced during WebGPU phase

- Unable to compile **Dawn-chromium** webgpu implementation. We shifted our main project from Dawn to **WGPU-native** by Mozilla due to same reason. Also, for using Dawn we started using precompiled binaries.
- Another issue was constantly **changing spec** of WebGPU. We had to verify some parts from the official documentation.

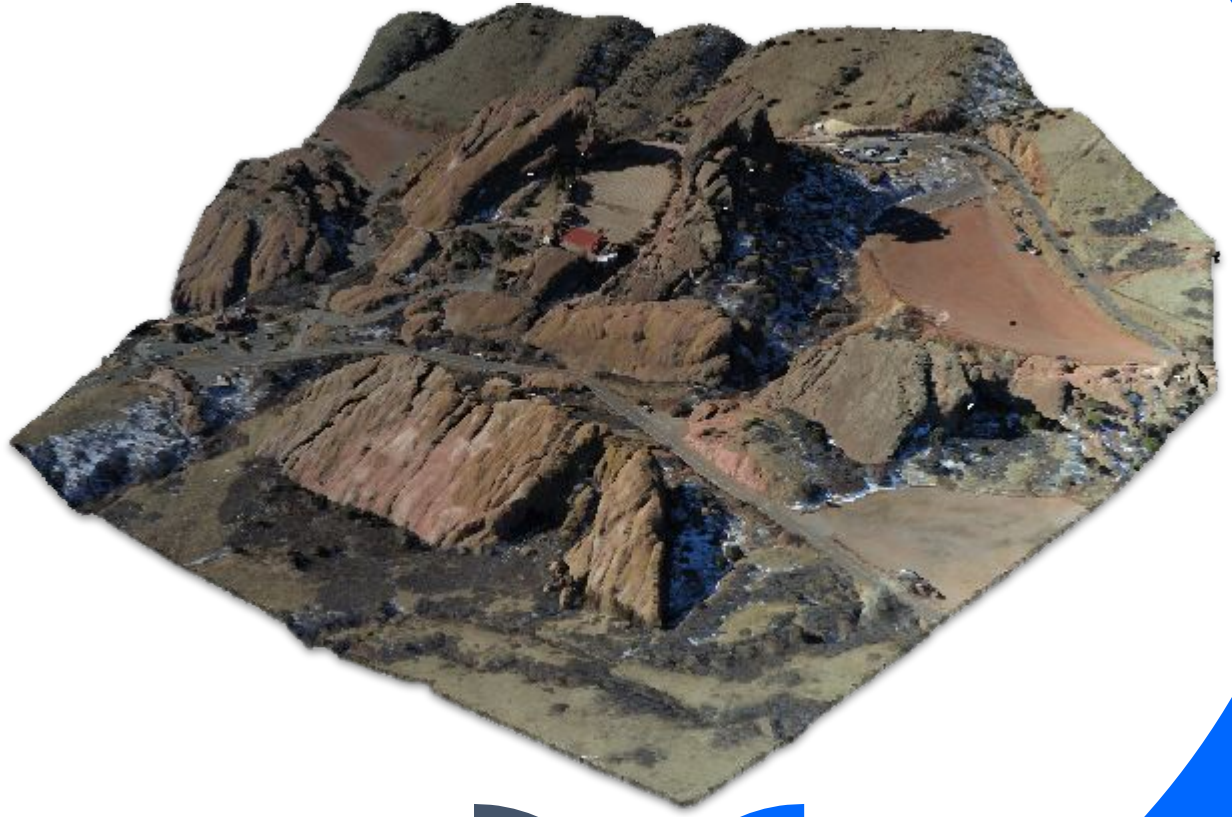
Point Clouds

- Collection of data points in 3D space
- Generated by 3D scanners & LiDAR (Light Detection & Ranging)



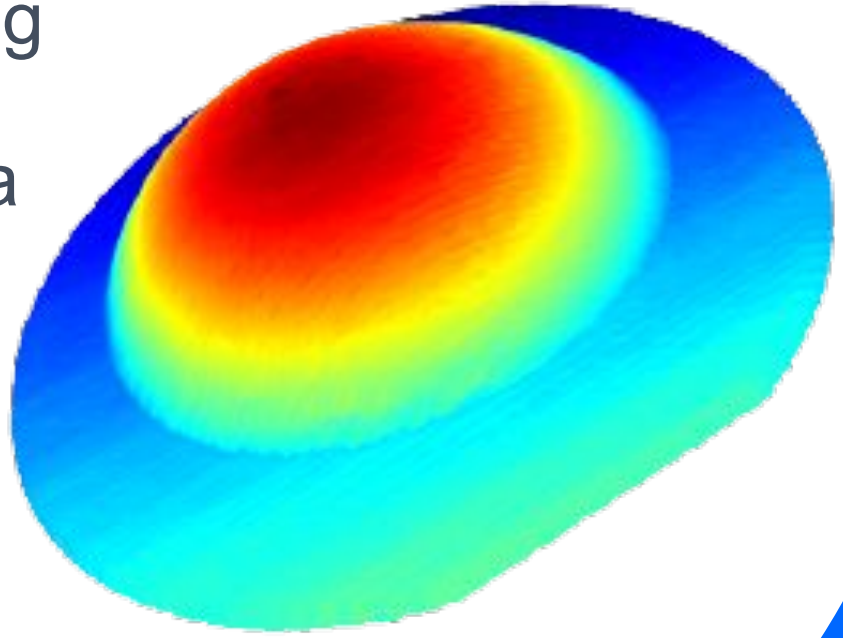
Applications of point clouds

- 3D modelling
- Landscape planning
- Medical imaging
- Reverse engineering
- Autonomous vehicles



Point Clouds in Open3D

- Open-source library for 3D data processing
- Written in **C++**. **Python API** available
- Rendered a **heatmap representation** of a **tumour**
- Data was in **proprietary binary format**, obtained from **3D scanner**



Three.js

- Javascript library which acts as a **WebGL wrapper**
- Handles entire rendering pipeline (geometries, lighting, shading)
- Investigated point cloud rendering using the PCD format

Implementation Plan

- Utilise **Three.js's PCD loader** or **Potree** (WebGL-based point cloud renderer)
- Consider **porting to WebGPU** for high performance 3D graphics
- Create a **native WebGPU** version to optimize rendering process and enhance performance

Questions?

Thank you