Technical Report: Cinematic Navigation HW04

Approach Summary

We implemented an end-to-end autonomous cinematographer that starts from a Gaussian splatting PLY scene and ends with a SparkJS-rendered panorama synced to a soundtrack. The Python pipeline performs four stages: scene analysis, beat extraction, camera planning, and renderer orchestration. Rendering happens in headless Chrome via SparkJS + THREE.js, guaranteeing that the deliverable leverages SparkJS as required.

Novelty Highlights

- **Beat-Snapped Shot Scheduler**: Each shot's start/end timestamps snap exactly to detected beat indices, ensuring edits always land on-music. This schedule feeds SparkJS directly through JSON keyframes.
- **Shot Templates with Guarantees**: The planner always includes at least one wide lateral establishing move and one zoom-in shot toward the highest-score cluster, then alternates orbit and elevated dolly patterns for coverage variety.
- **Indoor-Safe Navigation**: Safety volumes derived from the splat bounding box clamp every keyframe and raise the camera whenever the view ray would pierce walls, keeping indoor shots plausible.
- **Playwright Spark Harness**: Instead of screen recording, the renderer drives SparkJS deterministically in headless WebKit (Playwright), saving per-frame PNGs before muxing with FFmpeg. This keeps the workflow scriptable and reproducible even on CI hosts without full GPU stacks.

Challenges & Solutions

- **Large PLY Parsing**: Gaussian splat files can exceed millions of points. To keep analysis tractable, the explorer samples up to 150k points with a fixed RNG seed and computes PCA/cluster statistics on the sample.
- **Beat Robustness**: Ambiguous tempos can derail shot timing. We compute an autocorrelation-based confidence score and expose it in `beat_times.json` so users can swap tracks or override tempo if necessary.
- **WebGL in Headless Mode**: Some environments disable GPU acceleration in headless Chromium. The renderer adds `--use-gl=desktop` and `--enable-webgl` flags while serving assets locally to guarantee SparkJS can stream the `.ply` file without additional hosting.

Results & Evaluation

- Generated video duration lands within the 60–110 second requirement (bounded by both music length and configuration limits).
- Camera motion stays smooth thanks to `smoothstep` interpolation across all keyframes, and

transitions only occur on beats.
- Outputs include inspection artifacts (scene analysis, beats, shots, runtime config) to aid debugging or grading.

Future Improvements

1. **Semantic Object Selection**: Plugging in an object detector (e.g., Segment Anything + CLIP) would allow the zoom/orbit shots to target semantically meaningful artifacts instead of geometric clusters.
2. **Obstacle-Aware Trajectories**: Integrating ray-marching or signed distance approximations from the splat cloud would prevent the camera from cutting through geometry.
3. **Realtime Preview UI**: A lightweight web UI running the same SparkJS scene could enable manual overrides before committing to the render farm.
4. **Secondary Object Tour**: Re-running the planner with shot templates focused solely on detected objects would yield the optional Video 2 deliverable automatically.