DEMO REEL BREAKDOWN

By Aman Sachan

➔ MONTE CARLO PATH TRACER: (00:04) (C++, OPENGL)

- CPU based Path Tracer with a lot of features including:
- Volumetric Rendering; Multiple Importance Sampling; Multi-Threading;
- BVH Acceleration (9800% speed up);
- Handled materials with Micro-facet surfaces and Fresnel reflectance models;
- Realistic modeling of light sources and Thin Lens camera models;

➔ CUDA FLOCKING SIMULATION (00:25): (CUDA, C++)

- Implemented Craig Reynold's crowd simulation algorithm to model flocking behavior
- Can Visualize 1.6 million particles running at 60 FPS on a notebook GTX 1070
- Performance and design analysis on amansachan.com

➔ INTERESTING LEVEL GENERATOR (00:43): (JAVASCRIPT, WEBGL, GLSL, THREEJS)

- A procedural multi-layer dungeon generator that generates levels based on a voronoi-like graph after it has been heavily modified by various filters to create interesting level layouts
- These filters were in-place to remove things like intersections between paths and rearranging paths for interesting level design (inter-looping paths but not too many loops that it's a jumbled mess)
- Also Implemented: a Realistic Fog shader; Biome and Elevation dependent Terrain Shader on the GPU
- Also Implemented a controllable Crumbling Pathway aesthetic via instancing

➔ REAL TIME IMPLICIT SURFACES (01:05): (JAVASCRIPT, WEBGL, GLSL, THREEJS)

- Generated metaballs in real time using the marching cubes algorithm
- Resolution of ~1700 dynamic triangles at 60 FPS on a GTX 1070
- Triangles are created by evaluating a Density Field and then using that same density field to compute vertex normals

➔ ART OF COLLISIONS (01:24): (C++, MAYA API, MEL)

- Implemented a particle based rigid-body simulator based on the paper, " Unified particle physics for real-time applications ", by Macklin, Muller, Chentanez, and Kim
- Jointly implemented Shape Matching Constraints and Position Based Dynamics
- Implemented the conversion of arbitrary meshes into particle groups
- **TECHNIQUES USED:**
  - Intersection tests via ray marching were used to create particle filled representations of meshes.
  - Sphere instancing used to represent resultant particles.
  - Parallel constraint solver extended from position-based dynamics:
    - Shape Matching -- maintain particle configurations inside rigid bodies
    - Distance Constraint -- maintain Particle Separation/closeness
  - Semi-implicit Euler Integration for solving applied forces.
  - Callback functionality piggybacks on maya's animation controller.
  - Redefining flags as output variables allowed for dynamic re-evaluation of the compute function, and hence updating of the simulation.