Archery Environment 3D Simulation

CSCI5229 Final Project - Fall 2025 - **Richard Roberson**

- **Project Scope:** A complete 3D archery simulation set in a procedurally generated forest island with a surrounding mountain ring.
- **Grad Feature:** Custom GLSL **Normal-Mapped Terrain Shader** for high-detail surface rendering.
- **Core Features:**
  - Physics-based projectile motion & collision.
  - Algorithmic tree generation.
  - Dynamic Day/Night Cycle with atmospheric fog.

Normal-Mapped Terrain Shader

- **Goal:** Enhance terrain detail by using a normal map texture to simulate bumps and rocks without increasing polygon count.
- **Vertex Shader (Tangent Space):** Computes the TBN matrix from geometric normal N_0 (from finite diff method)

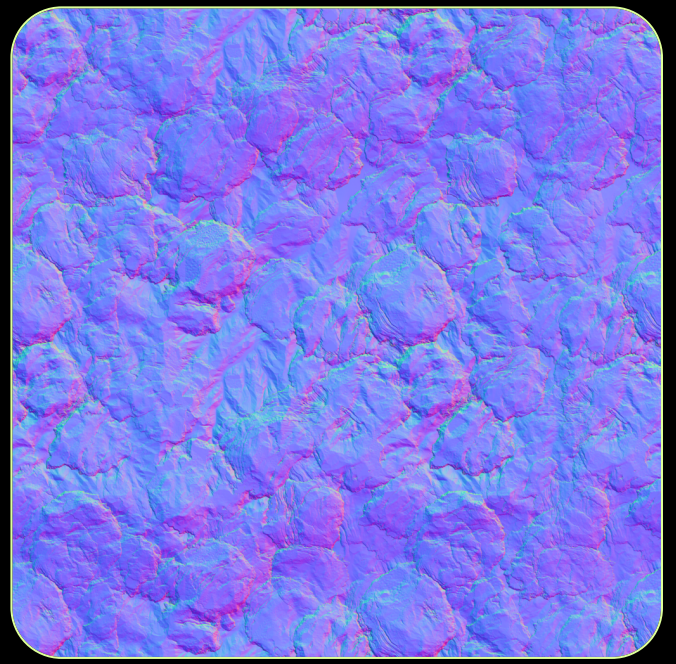
$$T = \text{normalize}(\text{Up} \times N_0); B = N_0 \times T$$

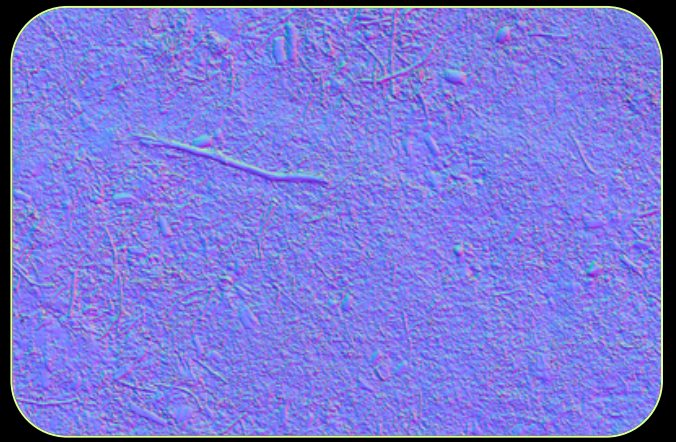
- **Fragment Shader:** Samples the normal map (n_{tex}) and transforms it to Eye Space

$$N' = \text{normalize}(TBN \cdot (2n_{\text{tex}} - 1))$$

- **Lighting Model:** Combines fixed-pipeline style Ambient, Diffuse, and Specular components with distance-based Fog mixing.

$$I_{\text{final}} = I_{\text{amb}} + I_{\text{diff}}(L \cdot N') + I_{\text{spec}}(R \cdot V)^{\text{shininess}}$$





Procedural Terrain & Mountains

- **Forest Floor:** Generated via summation of sine and cosine waves to subtle create rolling hills.

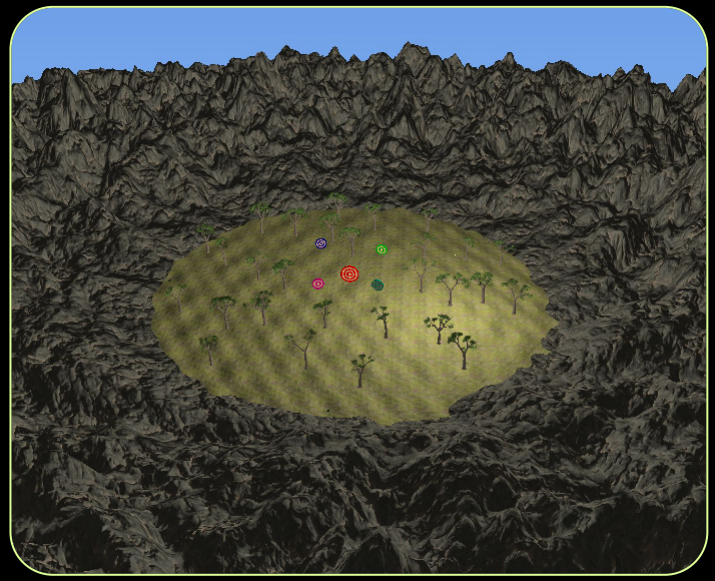
$$h(x, z) = s \cdot [0.3 \sin(0.5x) \cos(0.5z) + \dots]$$

- **Mountain Ring:** Uses a radial "bowl" function modulated by Fractal Brownian Motion (fBm) to clamp the play area.

$$H_{\text{mtn}} = \sin(\pi s) \cdot (0.5 + 0.5 \cdot \text{Noise}(x, z)) \cdot \text{scale}$$

- **Normals (Finite Difference):** Compute exact slopes from neighbors using helper functions for smooth lighting. When using normal maps, these normals are adjusted through the shaders.

$$\vec{T} = (2\Delta, h_R - h_L, 0)$$
$$\vec{B} = (0, h_U - h_D, 2\Delta)$$
$$\vec{N} = \text{normalize}(\vec{B} \times \vec{T})$$



Physics & Collision Detection

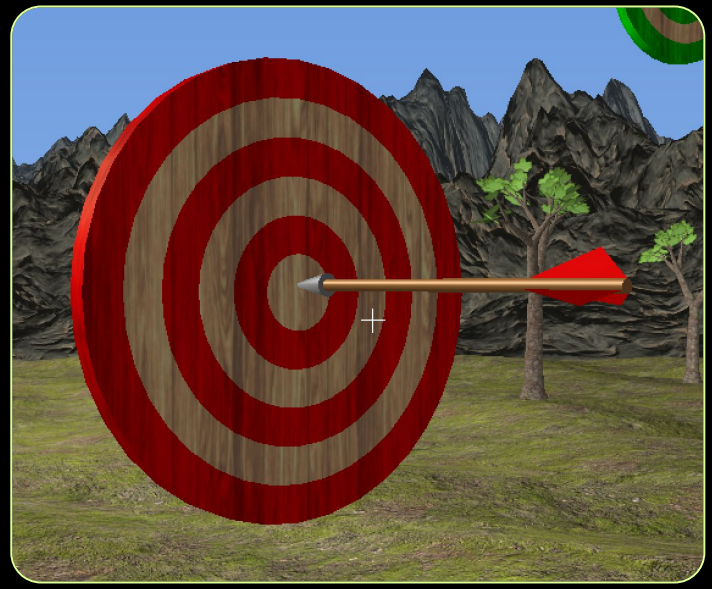
- **Projectile Physics:** Symplectic Euler integration for gravity and velocity. Arrow orientation (Yaw/Pitch) is derived from the velocity vector.

$$P_{\text{new}} = P_{\text{old}} + V\Delta t; V_{y,\text{new}} = V_{y,\text{old}} - g\Delta t$$

- **Sweep Test (Ray-Casting):** Prevents "tunneling" at high speeds by checking the ray between previous (P_{prev}) and current (P_{curr}) tip positions.

$$t = \frac{-(N \cdot P_{prev} + d)}{N \cdot (P_{curr} - P_{prev})}$$

- **"Sticky" Arrows:**
  On collision, the hit point is projected into the target's local basis (U, V, N). This allows the arrow to move and rotate perfectly with the animated target.



Algorithmic Tree Generation

- **Recursive Structure:** Trees are built using recursive function calls (fractals).

  Parameters control depth, length, radius, and branching angles.

- **Tapered Frustum Geometry:**

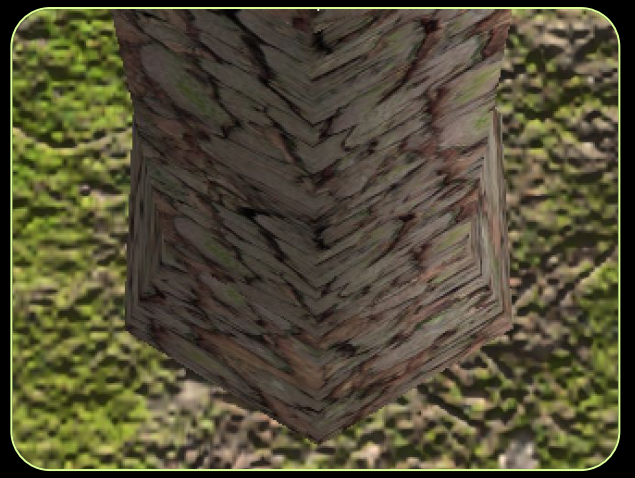
  Each branch is an octagon shaped cylinder that tapers from radius r_0 to r_1

- **Normal Calculation:**

  Normals must account for the branch slope to light correctly.

$$N_{\text{raw}} = (\cos\theta,\ \frac{r_0 - r_1}{L},\ \sin\theta)$$

- **Leaf Rendering:** Uses alpha-blended "billboards" that rotate slightly to face the camera, creating illusion of fuller trees with lower vertex counts.



Optimization & Performance

- **Two-Pass Tree Rendering:**
  1. **Opaque Pass:** Draws trunks/branches with GL_CULL_FACE enabled.
  2. **Transparent Pass:** Draws leaves with depth buffer read-only.

  Both passes share the same random seeds for perfect alignment.
- **Display Lists:** Static geometry (Terrain strips, Mountain rings) are precomputed and cached on the GPU.
- **Texture Quality:** Enabled Anisotropic Filtering for sharp textures at grazing angles.