EVGENY MAKAROV, DEVTECH NVIDIA

NVIDIA TURF EFFECTS:

MASSIVE GRASS RENDERING WITH DYNAMIC SIMULATION
OUTLINE

- Real-time grass rendering
- Turf Effects: grass rendering
- Turf Effects: dynamic simulation
- Grass authoring
- Q&A
REAL-TIME GRASS RENDERING
PARTICLE BASED GRASS

- Billboards
- Grass objects

Top view
PARTICLE BASED GRASS

Pros
- Good density
- Plausible rendering results
- Good performance

Cons
- View dependent
- Complicated physical interaction
GEOMETRY BASED GRASS
GEOMETRY BASED GRASS

- **Pros**
  - High density
  - Looks good at any scale
  - Simplified physical interaction
  - Two-sided lighting and self-shadowing

- **Cons**
  - High rendering complexity
TURF EFFECTS: GEOMETRY BASED GRASS
BREAK GRASS INTO A SET OF PATCHES
RENDERING A GRASS BLADE

Vertex shader

Hull shader

Domain shader

(u1, v1)

u

v
PATCH INITIALIZATION

- Application provides a set of seed points
- Seed point
  - Position
  - Expansion
  - Asset ID
SEED PARAMETERS

Dispersal
(coverage radius)

Bunching
(radial alignment)

LOW  HIGH
GRASS PATCH RENDERING

- Build a list of visible patches
- Library renders data for a single asset type at once
- Render $\text{PatchesNum} \times \text{AssetsNum}$

- Need to minimize API state changes
  - Use single API call per patch per asset
    
    ```c
    for (n = 0; n < \text{patchPerAssetCount}; n++)
    DrawInstanced(\ldots, \text{patchID});
    ```

- ~18.0 millions DrawInstanced() calls/sec

* i7 4770K 3.5GHz, GeForce GTX 960 driver 347.52
OCCLUSION CULLING

- Per patch occlusion culling on GPU
- Use application provided depth with occluders information
- Rasterize per patch conservative boundary boxes with read-only depth
OCCLUSION CULLING

```c
void DrawInstanced( UINT VertexCountPerInstance,
                    UINT InstanceCount,
                    UINT StartVertexLocation,
                    UINT StartInstanceLocation );
```
RENDERING WITH OCCLUSION CULLING

- Per asset:
  - Write per patch blade count
  - Call DrawInstancedIndirect() to render each patch

- Culled patches would still issue draw calls on CPU
VARYING DENSITY

- Reduce blade count for distant patches
- Analytically defined
- Use fixed number of discrete LODs with smooth transitions
VARYING DENSITY

Close up patch view

Distant patch view
(<10% of original blades)
DYNAMIC SIMULATION

- Grass interacts with dynamic scene objects
DYNAMIC SIMULATION (2)

- Simulate collision with per-blade control points
- Use mass-spring model
EXTERNAL FORCES
EXTERNAL FORCES

RENDER VOLUME
EXTERNAL FORCES

\[ F = k\Delta \]

\[ F_{\text{combined}} \]
SIMULATION SUMMARY

- Encode dynamic objects in a form of deformation heightfield
  - Patch-space distance to occluder
  - Per-pixel movement speed

- Landscape is also a heightfield

- Per asset simulation parameters
  - Stiffness
  - Restitution

- Do not resolve collisions between individual grass blades
GRASS AUTHORING

- Seed parameters
  - Control blades distribution

- Asset parameters
  - Control individual blade look
ASSET PARAMETERS

- **Width**
  - float4(w0, w1, w2, w3)

Diagram:
- Tip
- w3
- w2
- w1
- w0

Root

- (1,1,1,1)
- (1,1,1,0)
- (0,1,1,1)
VARYING SMOOTHNESS
ASSET PARAMETERS (2)

- Edge offset
- Density falloff
- Shape variations

A set of “barebones” used for particular grass type

2 user-defined CPs
3 user-defined CPs
SUMMARY

- Turf Effects: real-time grass simulation and rendering
- Currently uses tessellation (D3D 11.0 or OpenGL 4.0)
- Can rely on existing content
  - Convert existing particle-based grass into geometry
THANK YOU

JOIN THE CONVERSATION
#GTC15