GPU-Generated Procedural Wind Animations for Trees

- Renaldas Zioma
Main goal: Enable the simulation and visualization of large open environments with massive amounts of vegetation

1. To offload motion simulation of trees from CPU to GPU.
2. To leverage hardware instancing to reduce render calls.
3. To allow seamless integration with GPU-based fluid simulation for an evolving wind field.
Simulate with motion equations applied to each branch.

- Too time-consuming for, e.g., games
- Less-than-natural look because of underlying problem complexity

Observation of trees show that their movement appears random simulate with stochastic process.

In this case, noise functions.
Wind

Vector field

\[ \mathbf{v} = G(\mathbf{x}, t) \]

\( \mathbf{x} \) is the position in the vector field, \( t \) is the time and \( \mathbf{v} \) is the wind vector at the given position.
Conceptual structure of a tree

- Main trunk
- Branches - rigid segment that can rotate around the joint connecting it to parent branch

"A shallow tree, two or three nodes deep is usually enough to represent visually plausible tree motion"
Animating the trunk

Trunk movement is a result of forces applied to branches. Need to calculate for all branches to calculate for trunk. Hard to parallelize.

Instead include branches affect as higher frequency noise in noise function.
Animating the trunk

Apart from drag forces there are a number of phenomena affecting the tree movement:

- Inertia
- Uneven distribution of branches
- Turbulence in wind field

All these forces combined into one stochastic process:

$$m \ddot{a}(t) + c \dot{v}(t) + k x(t) = f(t)$$

where $m$ is mass, and $c$ and $k$ are damping and stiffness coefficients, respectively.
\[
\cos(x \times \pi) \times \cos(x \times 3 \times \pi) \times \cos(x \times 5 \times \pi) \times \cos(x \times 7 \times \pi) + \sin(x \times 25 \times \pi) \times 0.1
\]
Animating the branches

Branch is on the wind-facing side of the tree.

Branch is on the opposite side of the tree.

Branch is perpendicular to the wind direction.
Branch data

- Branch origin
- Main axis
- Perpendicular Tangent
- Stiffness Factor
- Parent Index

- Branch Indices
- Branch weights
Rotations

Traverse all branches related to the vertex
- Combine periodic functions
- Convert to quaternions
- Concatenate rotations

Quaternions
- Compact and fast
- Slower in practice
Rendering the Tree

- Bone Blending
- Each Vertex
  - Current Branch
  - Parent Branch
- Ratio of Influence
  - Offline
  - Run-time

- DirectX 9
  - Unnecessary overhead
  - Transformation recalculated for each vertex
- DirectX 10
  - stream-out functionality
Analysis & Comparison

Pros:
- Leverages GPU power
- Fully compatible with GPU instancing
- Not limited to simple wind movement
- Simulation step highly parallelizable

Cons:
- DX 9 overhead
- Not strictly physically correct

Table 6-1. Performance of DirectX 9 and DirectX 10 Implementations of the Algorithm

<table>
<thead>
<tr>
<th>Instances</th>
<th>Branches</th>
<th>DirectX 9 Implementation (Milliseconds)</th>
<th>DirectX 10 Implementation (Milliseconds)</th>
<th>Two-Bone Skinning Without Branch Simulation (Milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2,400</td>
<td>1.46</td>
<td>1.34</td>
<td>0.86</td>
</tr>
<tr>
<td>100</td>
<td>8,000</td>
<td>3.90</td>
<td>3.82</td>
<td>2.32</td>
</tr>
<tr>
<td>256</td>
<td>20,480</td>
<td>9.81</td>
<td>9.68</td>
<td>5.82</td>
</tr>
<tr>
<td>1,000</td>
<td>80,000</td>
<td>38.21</td>
<td>37.61</td>
<td>22.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21.61</td>
</tr>
</tbody>
</table>