Pasto Interactivo

David Pérez Cárdenas

7 de mayo de 2025

1. Creación de la brizna de pasto

1.1. Geometría de la brizna

La brizna de pasto se modela como una serie de vértices que forman un plano vertical con ligera curvatura, simulando la forma natural del pasto. Cada par de vértices define un segmento de la brizna, y la curvatura se logra mediante la variación de las coordenadas en el eje Z.

Listing 1: Definición de vértices de la brizna de pasto

```
constexpr Vertex CubeVerts[] =
{
    {float3{-w * (1 - 0.0f), 0.0f, k * 0.0f * 0.0f}, float2{0.0f, 0.0f}},
    {float3{w * (1 - 0.0f), 0.0f, k * 0.0f * 0.0f}, float2{1.0f, 0.0f}},
    {float3{-w * (1 - 0.1f), 0.1f, k * 0.1f * 0.1f}, float2{0.0f, 0.1f}},
    {float3{w * (1 - 0.1f), 0.1f, k * 0.1f * 0.1f}, float2{1.0f, 0.1f}},
    // ...
    {float3{-w * (1 - 1.0f), 1.0f, k * 1.0f * 1.0f}, float2{0.0f, 1.0f}},
    {float3{w * (1 - 1.0f), 1.0f, k * 1.0f * 1.0f}, float2{1.0f, 1.0f}},
};
```

1.2. Índices para la malla

Listing 2: Definición de índices de la brizna de pasto

```
constexpr Uint32 Indices[] =
{
    0, 1, 2,    1, 3, 2,
    2, 3, 4,    3, 5, 4,
    // ...
    18, 19, 20, 19, 21, 20
};
```

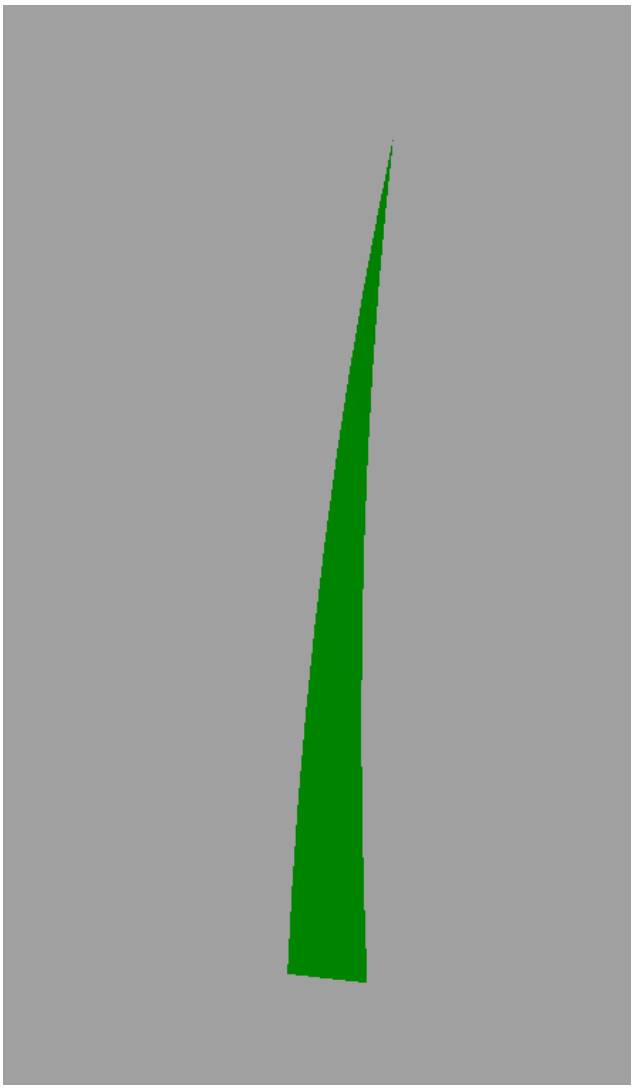

Figura 1: Brizna de pasto

2. Generación de la cuadrícula de pasto 100x100

Listing 3: Generación de la cuadrícula de pasto

```
float scaleFactor = 0.25f;
float spacing = 1.0f;
float4x4 scaleMatrix = float4x4::Scale(scaleFactor, scaleFactor, scaleFactor);

int gridSize = 100;
float offset = (gridSize - 1) * spacing / 2.0f;

for (int row = 0; row < gridSize; ++row)
{
    for (int col = 0; col < gridSize; ++col)
    {
```

```
            float x = col * spacing - offset;
            float z = row * spacing - offset;

            float4x4 translation = float4x4::Translation(x, 0, z);
            InstanceData.push_back(m_RotationMatrix * translation * scaleMatrix);
        }
    }
```

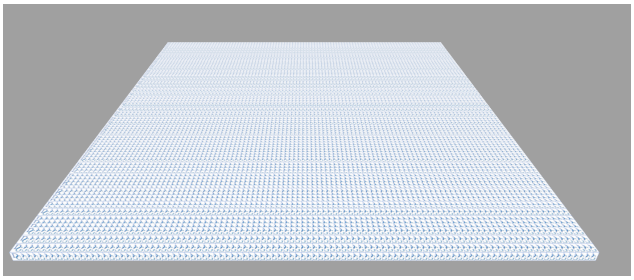

Figura 2: Cuadricula 100 x 100

3. Simulación de deformación del pasto por interacción

3.1. Métodos propuestos

- **Animación por clústeres**: Se calcula un vector de traslación para cada grupo de briznas en la CPU y se pasa al *shader* de vértices. Esto permite animar regiones del pasto de forma coherente, aunque puede aumentar las llamadas de dibujo.
  Referencia: Rendering Countless Blades of Waving Grass - GPU Gems, Capítulo 7

- **Animación por vértice**: La deformación se calcula directamente en el *shader* de vértices, lo que permite animaciones más detalladas y eficientes para representar miles de briznas en tiempo real.
  Referencia: Rendering Countless Blades of Waving Grass - GPU Gems, Capítulo 7

- **Simulación basada en física**: Utiliza las ecuaciones de Navier-Stokes para simular el viento de manera realista. Esta técnica modela cómo el aire afecta al movimiento del pasto, ideal para entornos con viento controlable y efectos suaves.
  Referencia: Real-Time Simulation on Grass Swaying with Controllable Wind Dynamics - MDPI