# Instant Animated Grass

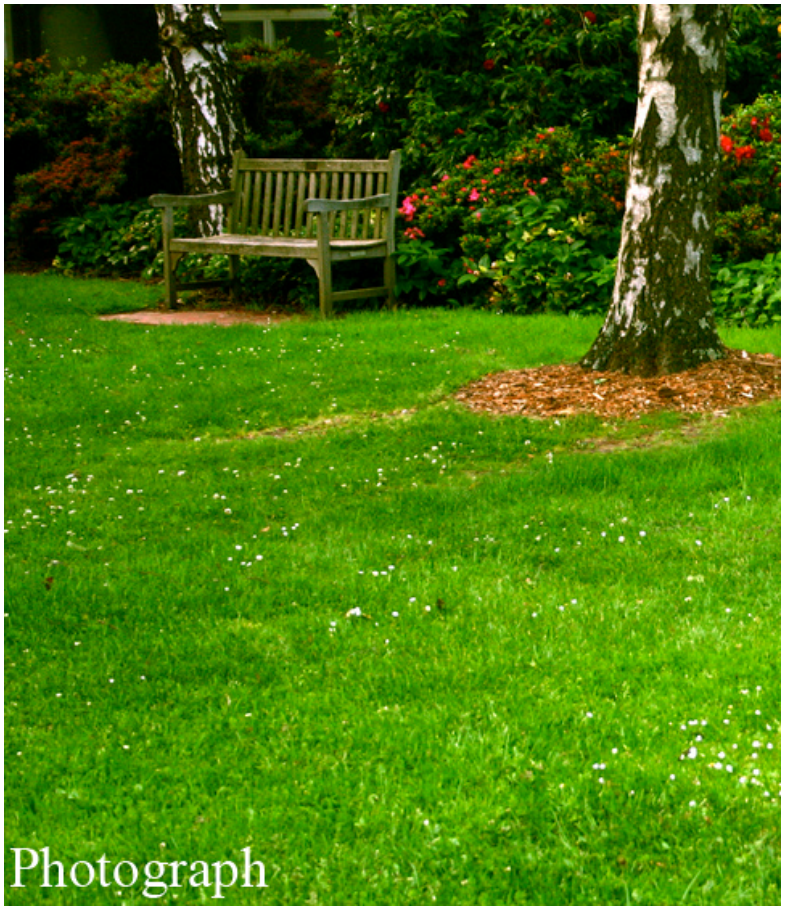
**Ralf Habel** , Michael Wimmer, Stefan Jeschke

Institute of Computer Graphics and Algorithms

**Vienna University of Technology**

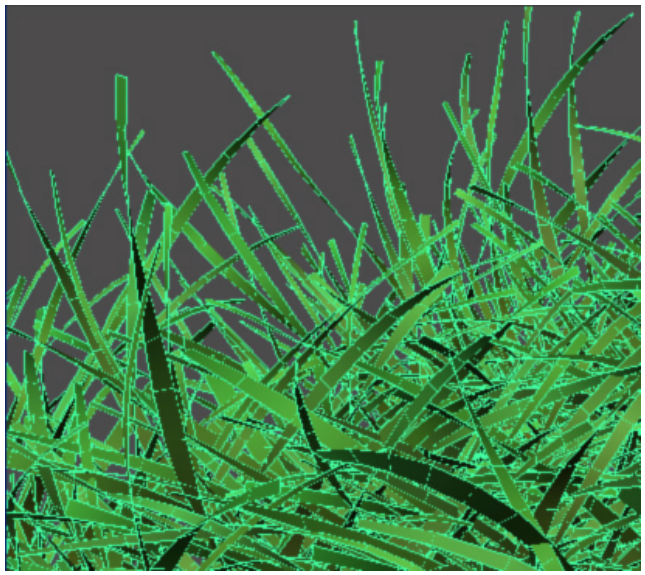
# Motivation

- Render dense short grass in real-time
- Lawns, meadows
- 1st person viewpoint

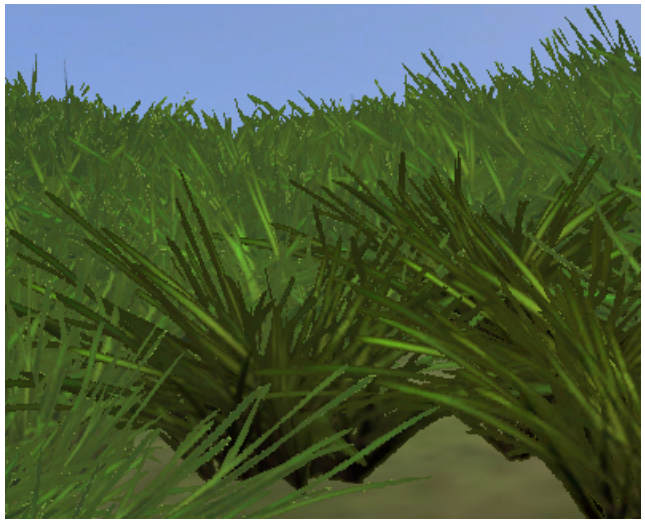


# Challenges

- Full polygon representation too expensive



- Billboard representation
  - Massive overdraw
  - Spatial aliasing



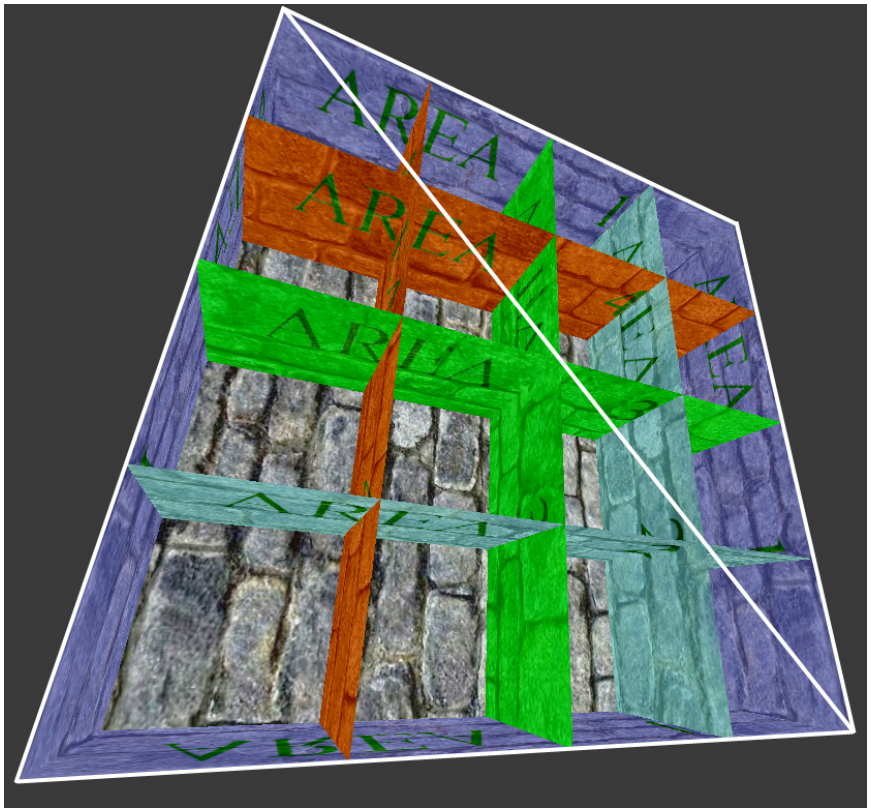
# Properties

- Important visual properties of grass
  - Parallax
  - Occlusion
  - Animation


- High frequency structure
  - No need for accuracy

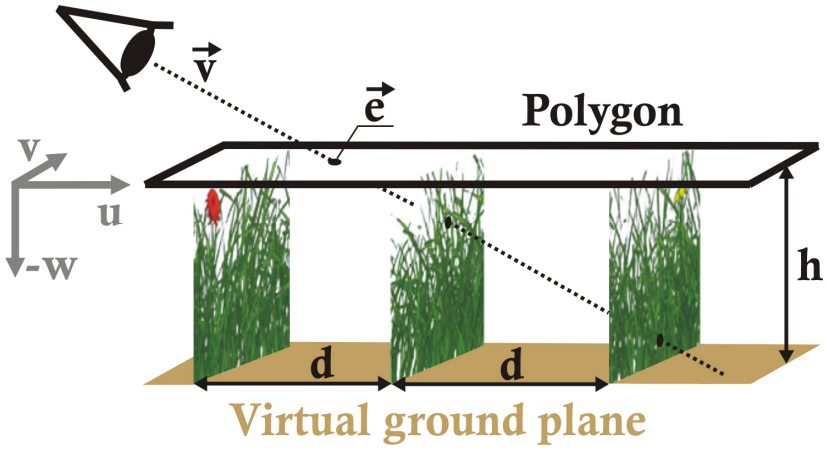
# Grass representation

- Dense regular grid of textured billboards
  - Provides approximate parallax and occlusion
- Generate billboard grid and ground plane in the fragment shader
- Polygons act as a carrier

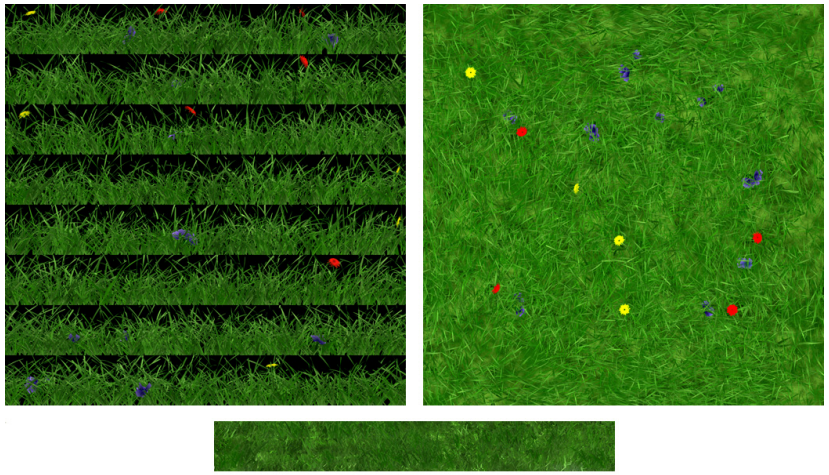


# Grass Ray Tracer

- Initialize texture offsets, increments/decrements and first grid planes
  - Dependent on entry point and sign of view direction

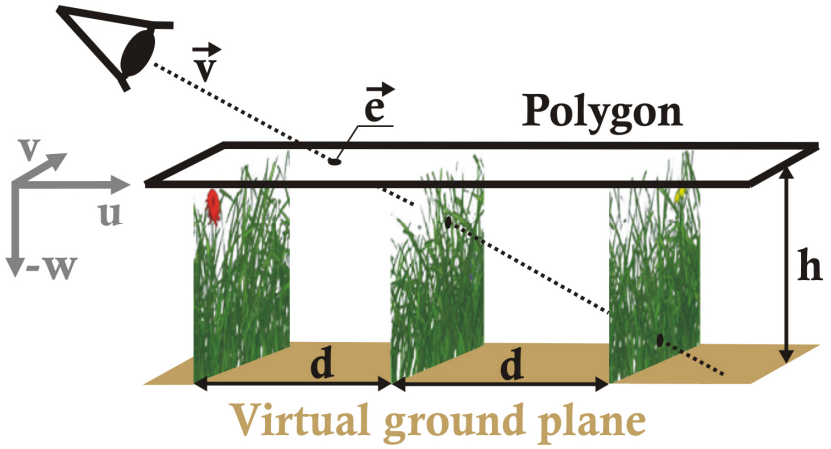


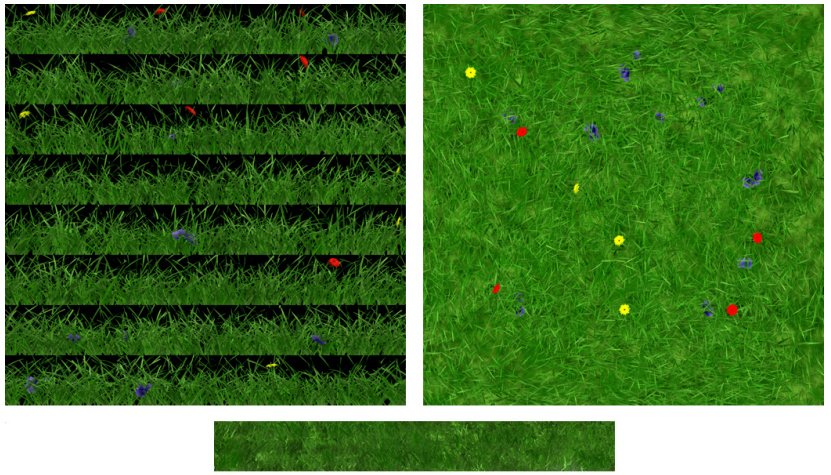

- Iterate through different grass billboard textures
- Same texture is seen from both sides



# Grass Ray Tracer

- Initialize texture offsets...
- Ray tracing loop
  - Intersect with next axis aligned planes (grass and ground)
  - Increment/decrement hit grid plane
  - Blend current color according to $\alpha$ (“over-operator)



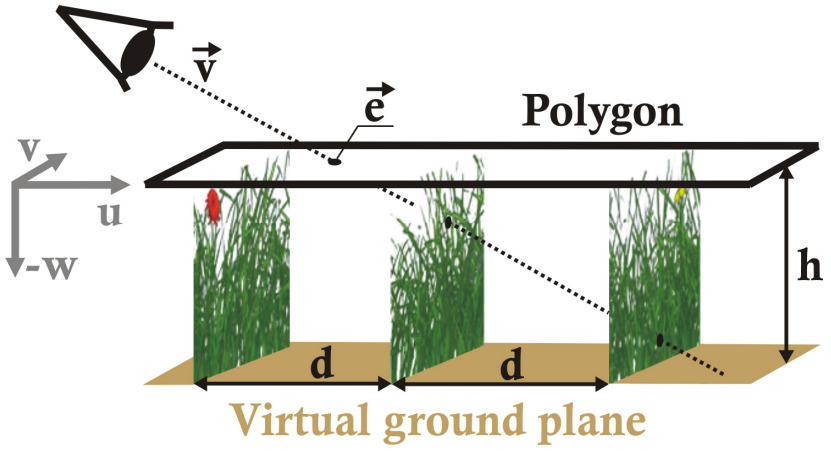
- 4-5 fixed iterations
- Early loop exit may be faster on certain hardware

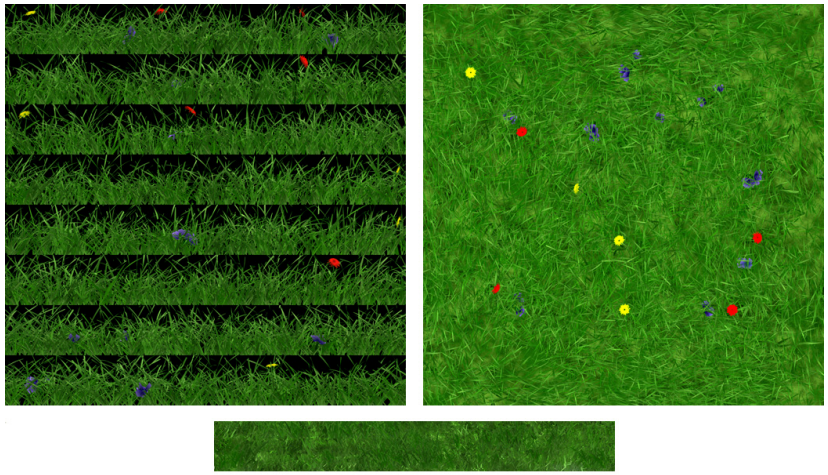
# Grass Ray Tracer

- Initialize texture offsets…
- Ray tracing loop...

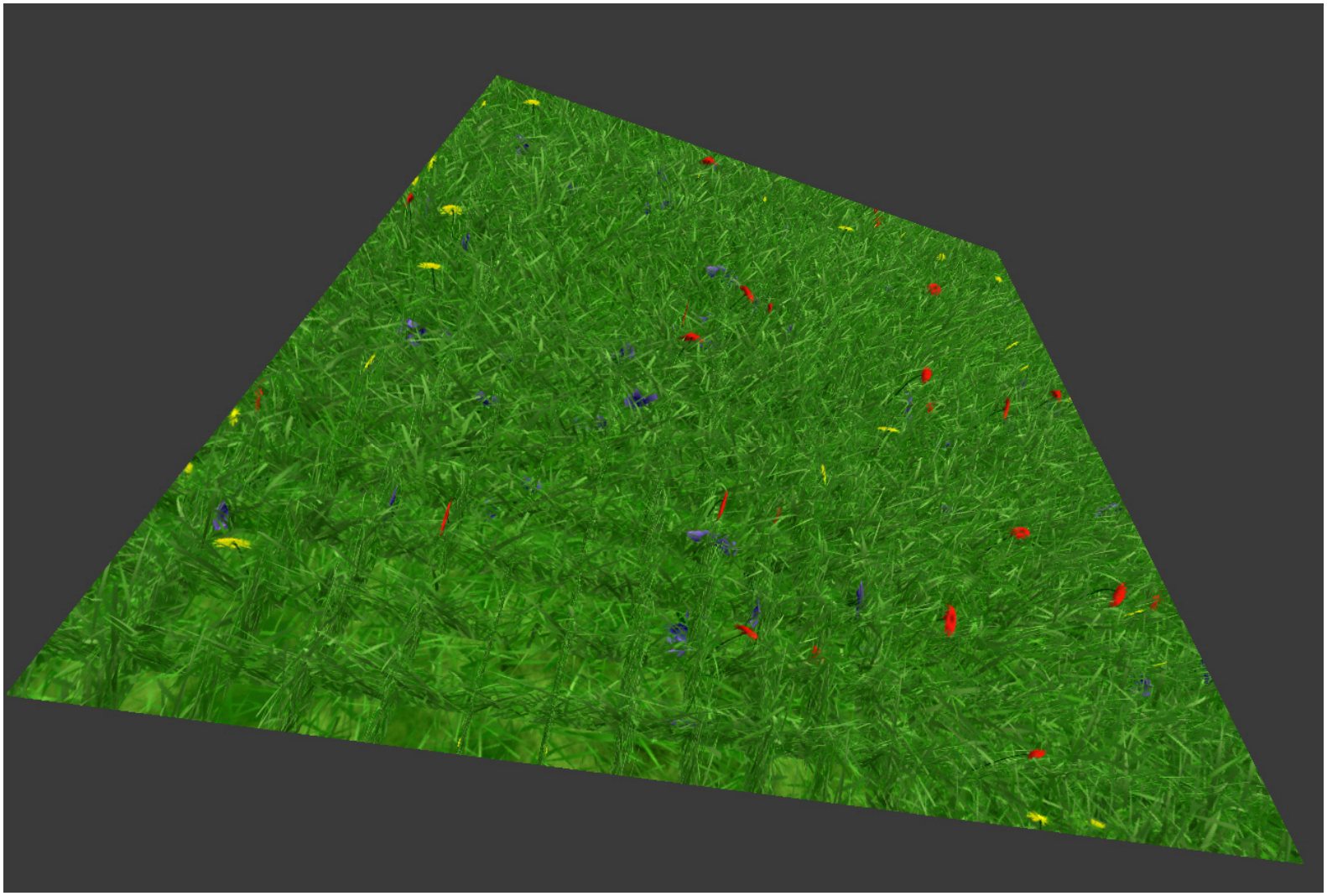
- Fill remaining transparency
  - Fully opaque grass texture

  or
  - Average color



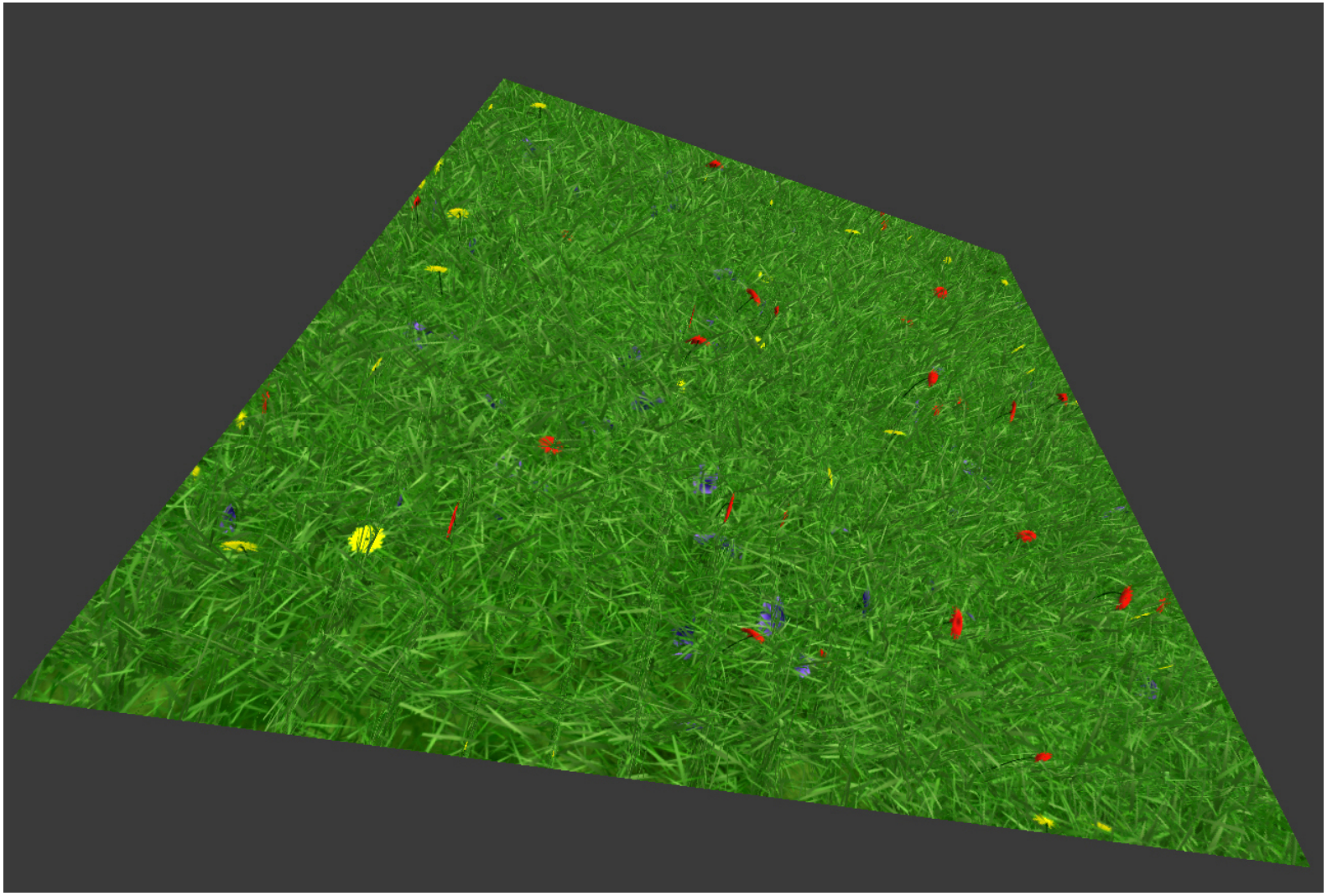



# Result



# Result

Additional horizontal plane at half the grass depth.



# Visibility interactions

- Correct z-buffer required
  - Avoid clipping at carrier polygon
  - Remember depth from ray entry to the point where a threshold opacity is reached
  - Transform into view space, add to carrier polygon depth

  - $\alpha$-testing instead of $\alpha$-blending
  - No modification to render pipeline

# Instant Animated Grass

# Overview

+ Confined to a shader
+ No additional geometry required
+ Front to back compositing
  + Accurate $\alpha$-blending
  + Reduced or no overdraw
+ Performance is not dependent on the number of billboards, but pixels covered and tracing depth

- No silhouettes
- Viewpoint cannot move into the grass

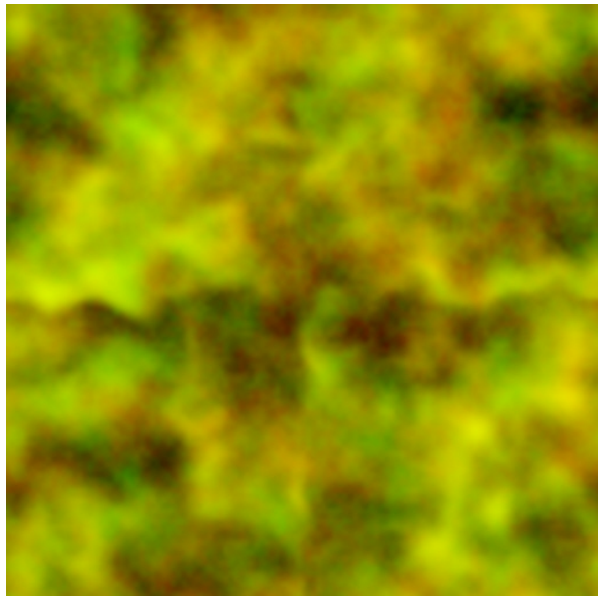
# Animating Grass

- Realistic simulation of grass requires two components
  - Gusts of wind cause large areas of grass to bend in the same direction
  - Wind turbulence near the ground causes smaller but erratic movements

# Animating Grass

- Simulate this behavior with texture lookup distortion
  - Translate noise map over terrain
  - Offset texture lookups horizontally, scale with height so grass stays fixed on the ground

    (local shear-transformation)
  - ➢ Transport high frequencies



- Noise map
  - low frequencies with high amplitude
  - high frequencies with low amplitude

# Instant Animated Grass

Video

# Performance

- Half the screen covered
- Full view of terrain
- 5 ray tracing loop iterations

- 3.2 GHz, 1024x786

| | Polys | Raytraced |
|---|---|---|
| NVIDIA 7900 GT : | ~90 fps | ~140 fps |
| NVIDIA 8800 GTS : | ~120 fps ($\alpha$-tested) | ~300 fps |



# Instant Animated Grass

- Future work:
  - Derive wind textures from physical simulation of grass
  - Adapt higher order surface approximations for silhouettes
  - DirectX 10 geometry shader adaptation



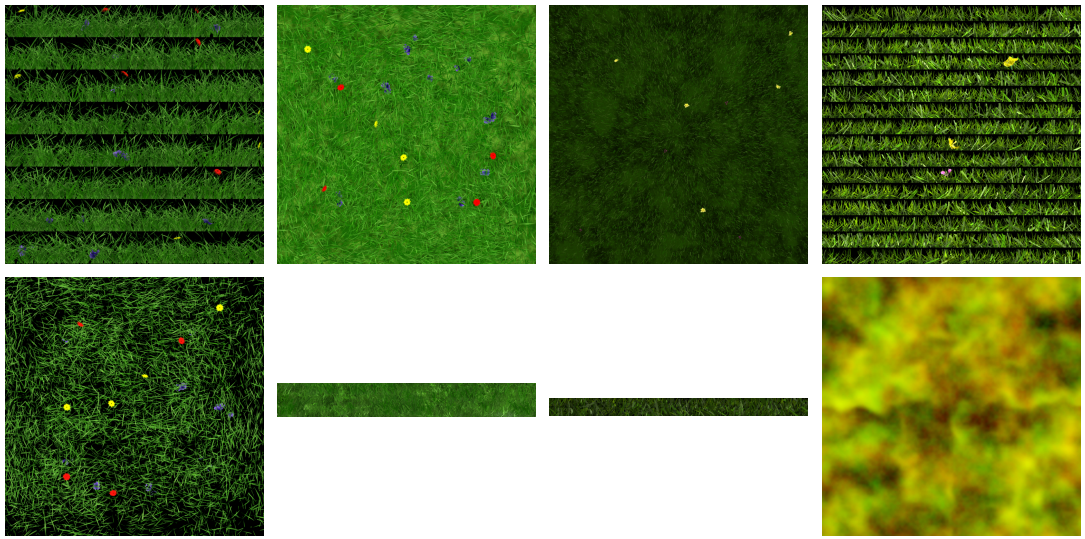
# Instant Animated Grass

## Resources:

HLSL implementation and used textures at:

http://www.cg.tuwien.ac.at/research/publications/2007/Habel_2007_IAG/

```
float3 rayHitpointX = rayEntry + eyeDir
float3 rayHitpointY = rayEntry + eyeDir

//Check if we hit the ground. If so, ca

if ((rayHitpointX.z <= -GRASSDEPTH) &&
{
    float distanceZ = (-GRASSDEPTH)/eyeD

    float3 rayHitpointZ = rayEntry + ey
    float2 orthoLookupZ = float2(rayHit

    color = (color)+((1.0-color.w) * te
    if(zFlag ==1) zOffset = distanceZ;
    zFlag = 0; //Early exit here if fa
}
else
{

    float2 orthoLookup; //Will contain
```




Thanks
for your attention!