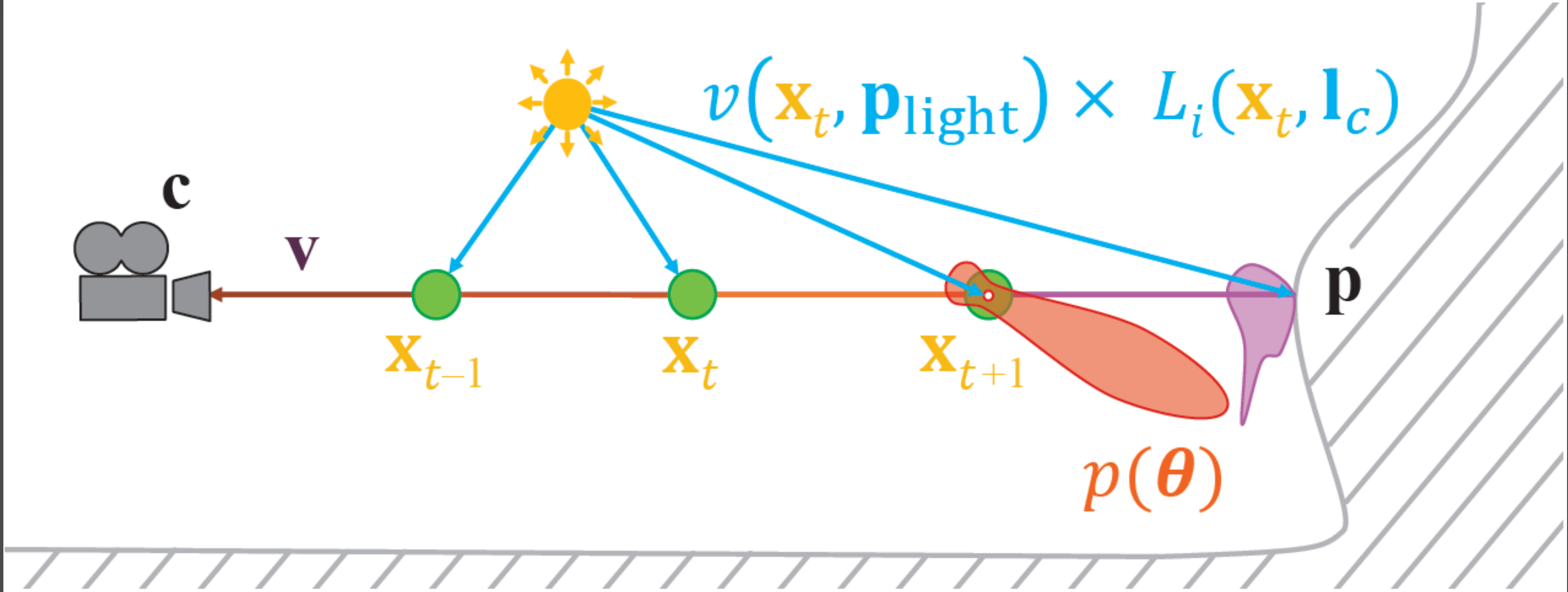
Shaders

Luca Türk, Samuel Knöthig, Sven Lüpke

Physically Based Volumetric Fog and Atmosphere

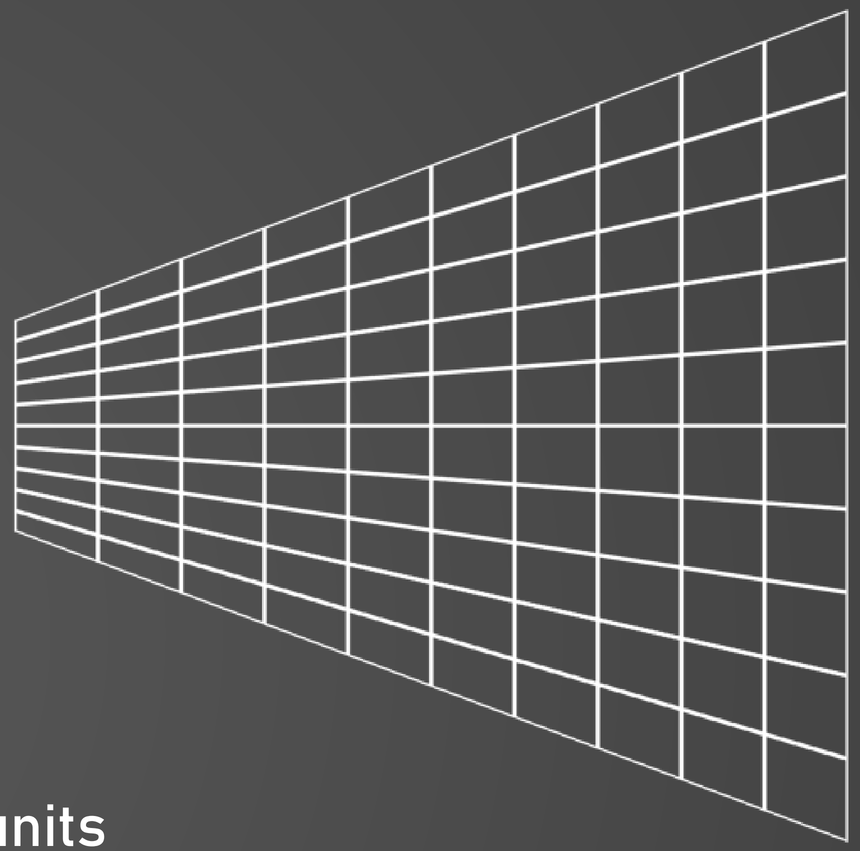
Sven Lüpke

Light Scattering Theory [Akine-Möller18]



$$L_i(c, -v) = T_r(c, p)L_o(p, v) + \int_{t=0}^{\|p-c\|} T_r(c, c - vt)L_{scat}(c - vt, v)\sigma\, dt$$

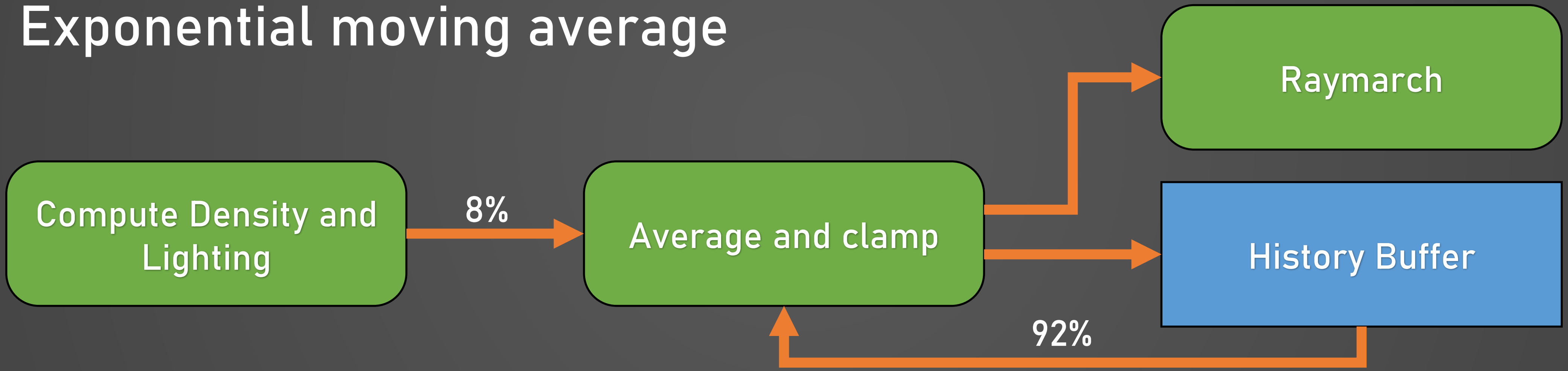
Implementation



- Frustum oriented volume [Wronski14]
  - 240x135x96
  - In-scattering in RGB
  - Scattering coefficient in Alpha
- Exponential depth distribution [Sousa16]
- Animated perlin noise
- Covers scene up to 900 units from camera
  - Analytic pixel shader fog [Quilez10] beyond 900 units
- 4 passes:
  - Compute density and lighting at each sample position (Compute Shader)
  - Temporal Filter (Compute Shader)
  - Raymarch through volume (Compute Shader)
  - Upscale Fog and apply to scene using depth (Pixel Shader)

Temporal Supersampling

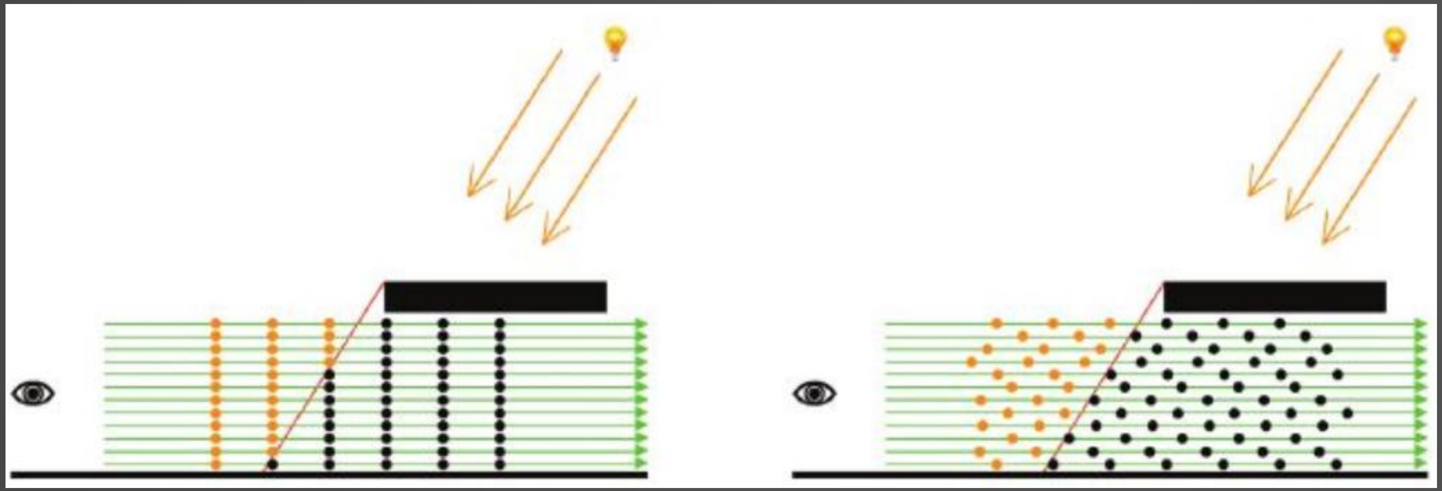
- Use previous frames to increase sample count
- Jitter sample positions using Halton sequence(2, 3, 5) [Karis14]
- Exponential moving average

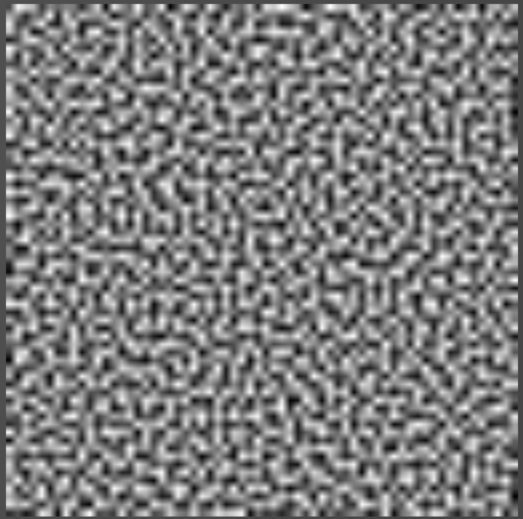


- Need to compute position in history buffer if camera moves
- Neighborhood clamping [Salvi16] to avoid ghosting on moving lights

Dithering

- Offset sample depth using blue noise [Vos14][Bauer19]
  - ➔Better sample distribution
  - ➔Helps against flickering caused be neighborhood clamping

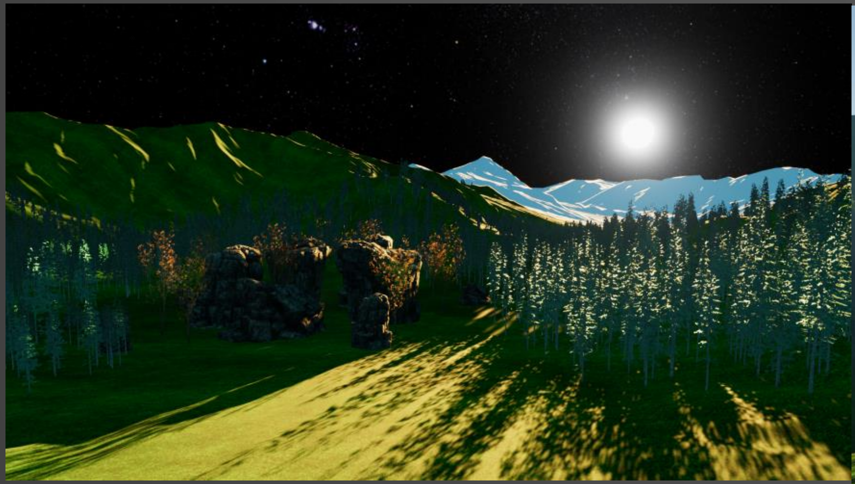




- Dither sample positions during upscaling as well [Bauer19]

Atmosphere

- Uses stripped down version of the fog shaders
  - No shadows, dithering and temporal supersampling
- Volume resolution: 32x32x32
- Scattering coefficient set to $(0.25, 0.5, 1)*10^{-4}$



No fog or atmosphere

Atmosphere only

Fog and Atmosphere

Performance

- Fog + Atmosphere
- Measured on a GTX 1050
- Without point lights: 3.5 ms
- With 8 point lights: 4.5 ms

Grass Shader – Live Demo

Vegetation

Luca Türk

Grass Shader – General

- Generate grass geometry in tessellation and geomety stage
  - Allow artist to pass properties such as height, color distribution and wind displacement
- Collide grass blades with player and other specified objects

Grass Shader – Tessellation

- Create additional ground verticies for geometry shader to work on, effectively increasing grass density

- Optimizations: Tessellation fractors decrease with distance to player, and input verticies are frustum culled

Grass Shader – Geometry

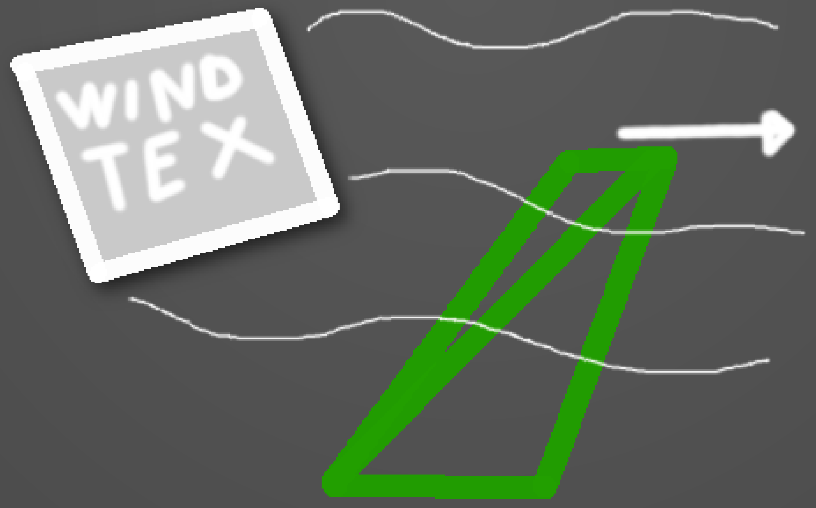
Initially, a single grass blade was added in the centre of the input triangle.
But, a lot of tessellation required for the dense grass look.
==> Add multiple blades for each triangle

Grass Shader – Geometry

Grass blades are constructed out of four verticies, which allows for less pointy looking blades, by pushing down the top verticies.

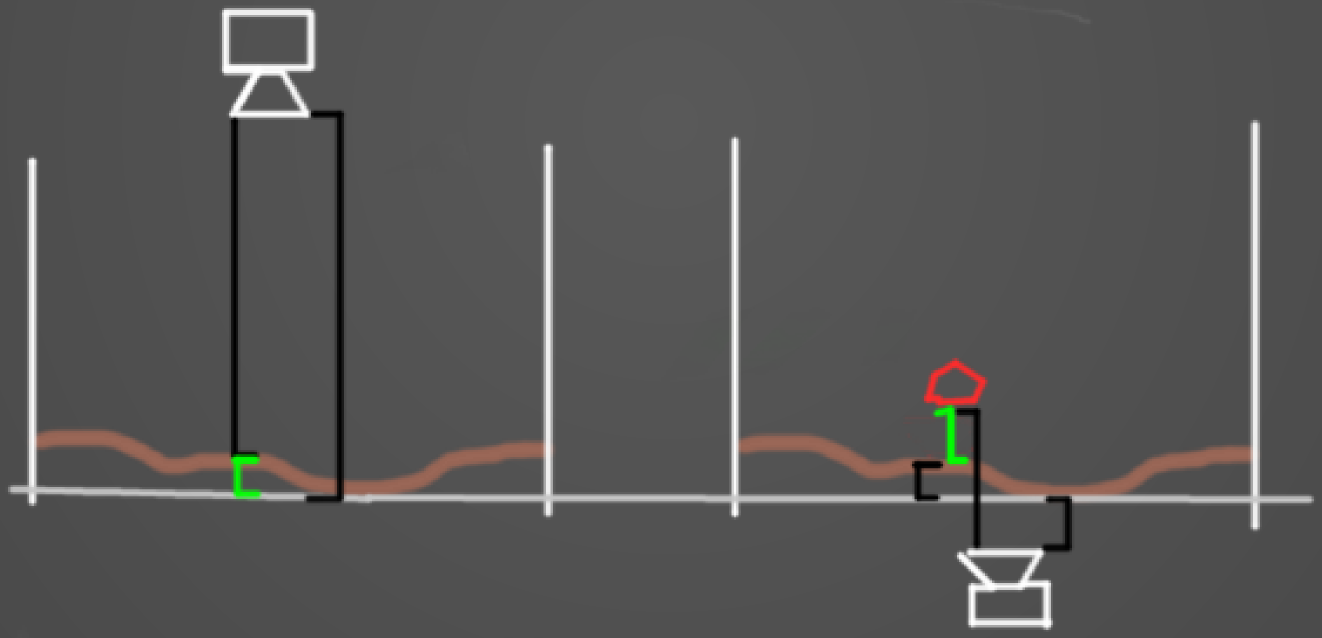
Grass Shader – "External" Influences

Grass Blades are influenced by wind, achieved by a displacement texture applied to the top vertex positions.



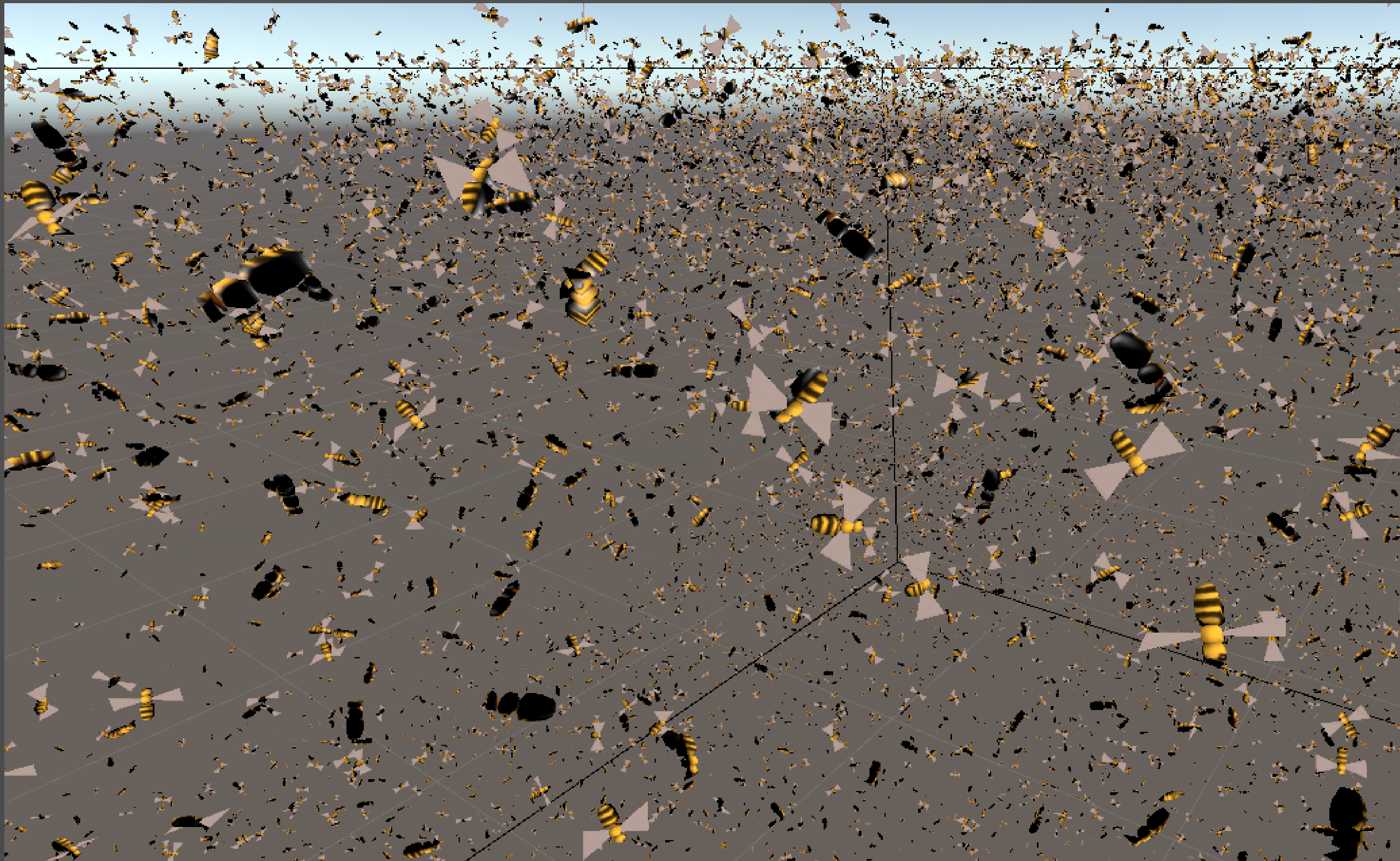
Grass Shader – "External" Influences

Additionally grass will collide with players or other objects, and bend away.



Boids (Bird-like objects)

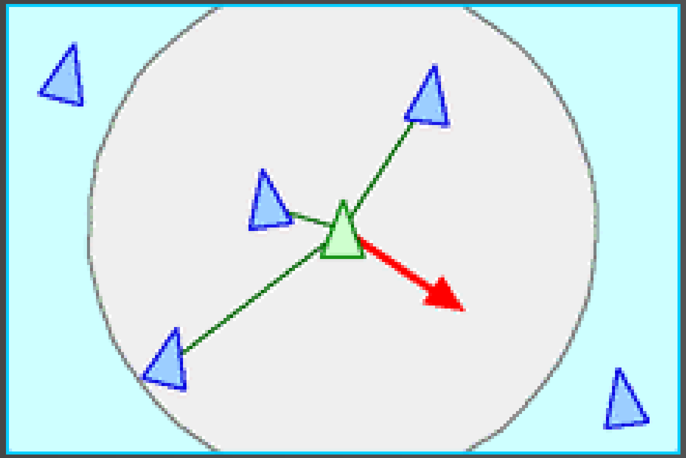
Samuel Knöthig



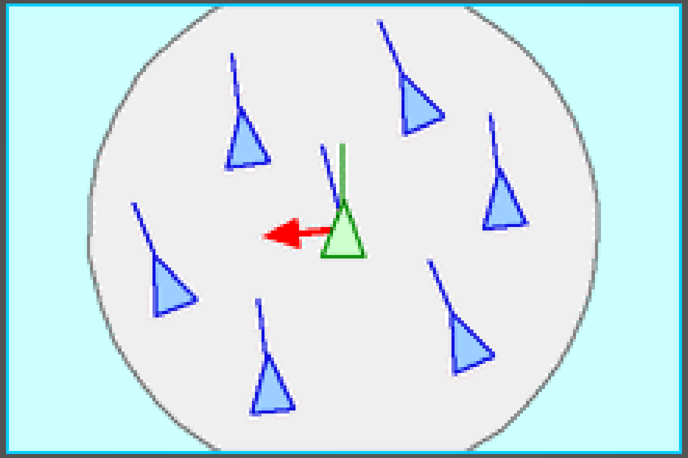
Boids (Bird-like objects)

- Simulate birds by calculating the desired movement-vector of each separate particle
- Calculated via a compute shader
- Drawn using DrawMeshInstancedIndirect with appropiate shader

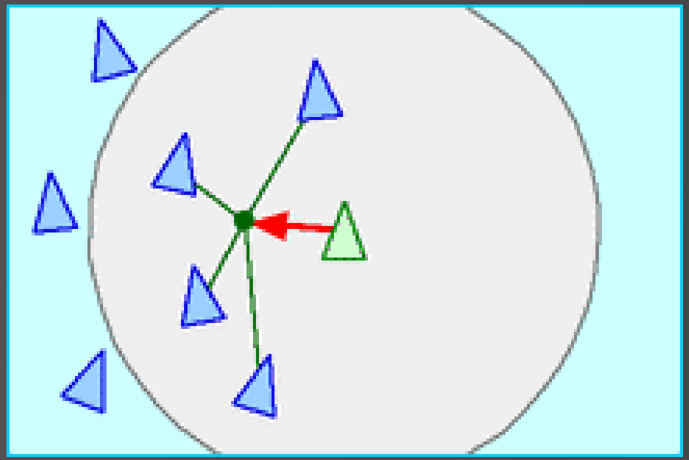
Boids (Theory)



Separation



Alignment



Cohesion

Boids (Separation)

- Avoid crashing into other boids and obstacles
  - Steer away from other boids and the walls of the boundry-box
  - Increase impact of desire with decreasing distance

$$separation_i = \sum_{b=-i} (pos_i - pos_b)^7$$

Boids (Alignment)

- Try to match speed with other boids
- Creates a more "stable" look

$$alignment_i = \sum_{b=-i} heading_b$$

Boids (Cohesion)

- Fly towards other boids
- But not too close

$$cohesion_i = \sum_{b=-i} \frac{1}{(pos_b - pos_i)^3} * (||pos_b - pos_i|| > 1)$$

Boids (Additional stuff)

- Finetune formulas with (a lot) of constants
- Rotate boids towards their flying direction
- Put everything into a prefab with some constants exposed (e.g. size)
- Draw meshes at the position of the particles

References

- [Wronski14] B. Wronski: Volumetric Fog: Unified compute shader based solution to atmospheric scattering, SIGGRAPH 2014
- [Hillaire15] S. Hillaire: Physically Based and Unified Volumetric Rendering in Frostbite, SIGGRAPH 2015
- [Bauer19] F. Bauer: Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution, SIGGRAPH 2019
- [Vos14] N. Vos: Volumetric Light Effects in Killzone: Shadow Fall, GPU Pro 5, CRC Press 2014
- [Akine-Möller18] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, S. Hillaire, Real-Time Rendering, Fourth Edition, CRC Press 2018
- [Hillaire16] S. Hillaire: Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite, SIGGRAPH 2016
- [Karis14] B. Karis: High Quality Temporal Supersampling, SIGGRAPH 2014
- [Salvi16] M. Salvi: An Excursion in Temporal Supersampling, GDC16
- [Jimemez16] J. Jimenez: Filmic SMAA, Sharp Morphological and Temporal Antialiasing, SIGGRAPH 2016
- [Sousa16] T. Sousa, J. Geffroy: The Devil is in the Details: idTech 666, SIGGRAPH 2016
- [Quilez10] https://www.iquilezles.org/www/articles/fog/fog.htm