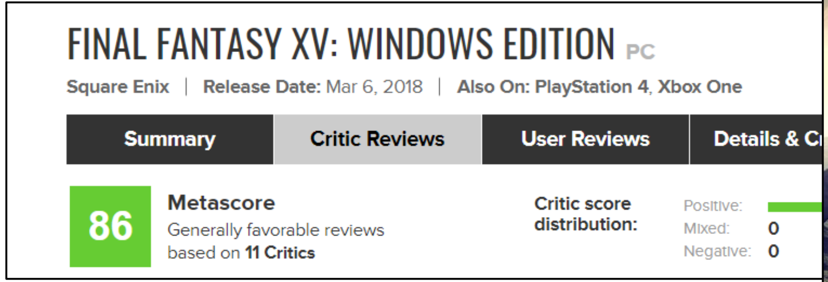
NVIDIA GameWorks Technologies in 'FINAL FANTASY XV', Behind the Scenes

Evgeny Makarov, Senior Developer Technology Engineer, NVIDIA
Masaya Takeshige, Senior Developer Technology Engineer, NVIDIA

3/22/2018
www.nvidia.com/GDC

FINAL FANTASY XV WINDOWS EDITION

- Single-player, Action role-playing
- Developed by Square Enix Business Division 2
- Luminous Engine
- Released March 7th, 2018


FINAL FANTASY XV: WINDOWS EDITION PC

Square Enix | Release Date: Mar 6, 2018 | Also On: PlayStation 4, Xbox One

86 Metascore
Generally favorable reviews
based on 11 Critics

Critic score distribution:
Positive:
Mixed: 0
Negative: 0

GDC

NVIDIA

Integrated Features

- HBAO+
- Hair Works
- Turf Effects
- Flow
- VXAO
- Shadow Works
- Ansel
- NVIDIA Highlights

Integration Case Studies

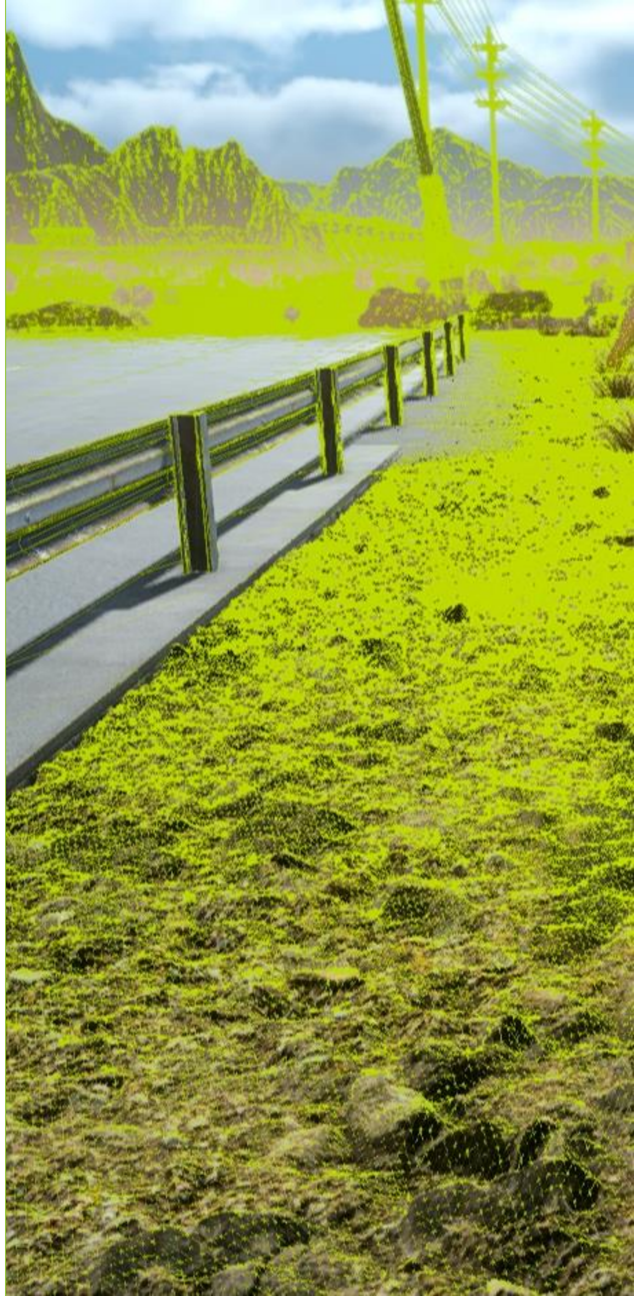
Terrain Tessellation

Terrain Tessellation

1. Apply HW tessellation to terrain primitives

1. Add displacement

1. Fix cracks



Primitives Tessellation

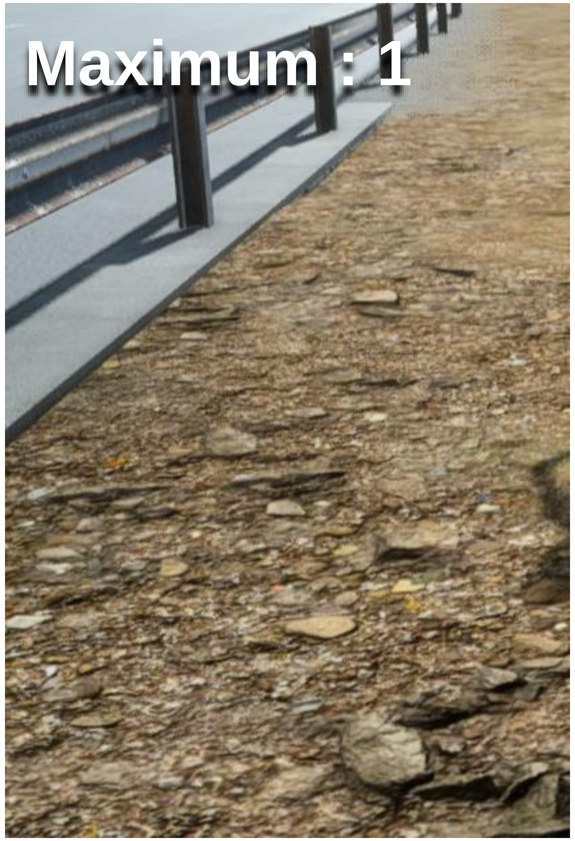
Terrain Tessellation

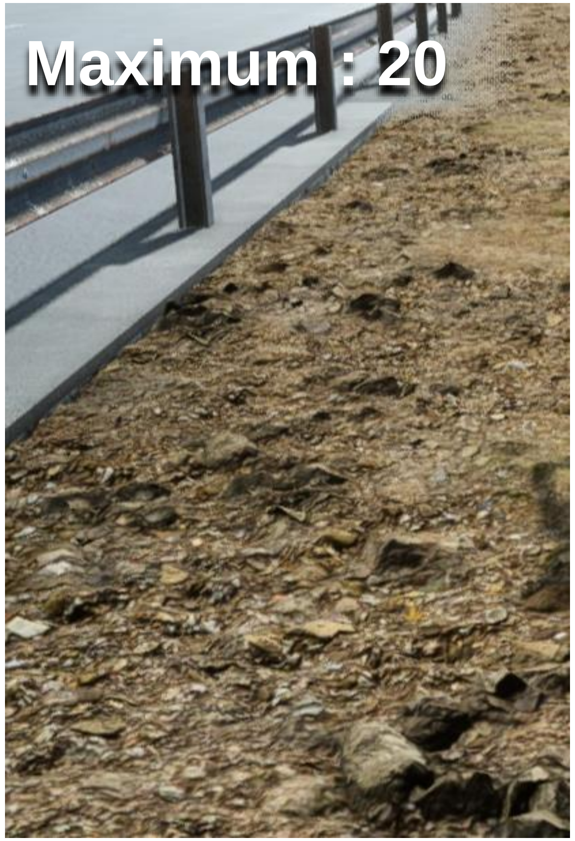
- Triangle tessellation is straightforward
  - Better use quads
    - More control
    - Better tessellation patterns

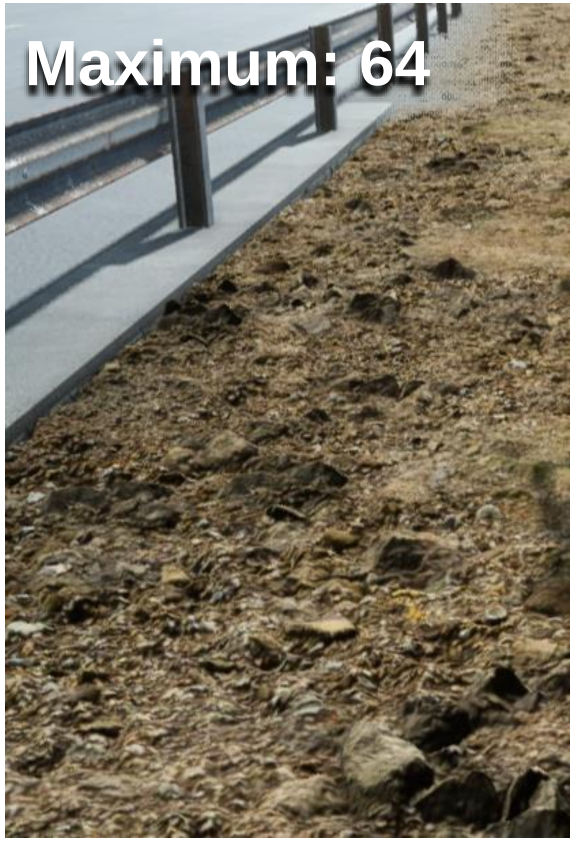
- Were able to generate index buffer for quads at runtime
  - Makes integration simpler

- Try “integer” tessellation first
  - Clamp maximum tessellation factors early
  - Use various clamp factors with presets

← performance →







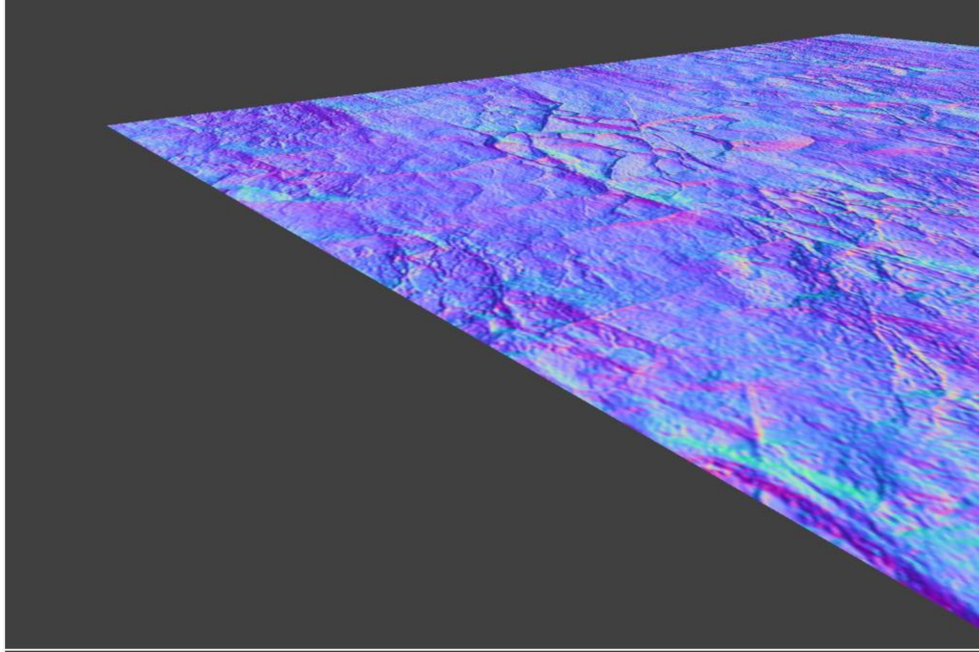
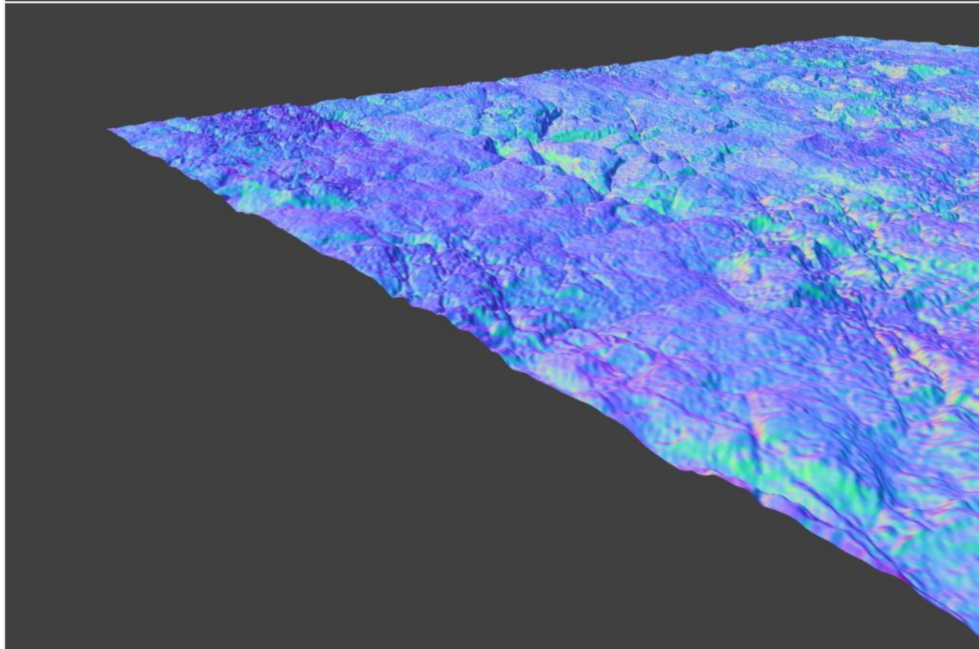
← quality →

Displacement Data

Terrain Tessellation

- Ideally use existing displacement maps
  - We didn't have any :(
- Use normal maps instead
  - Convert normal maps to displacement maps
  - Assign proper world scale

GDC

GameWorks: Materials & Textures

Terrain Tessellation

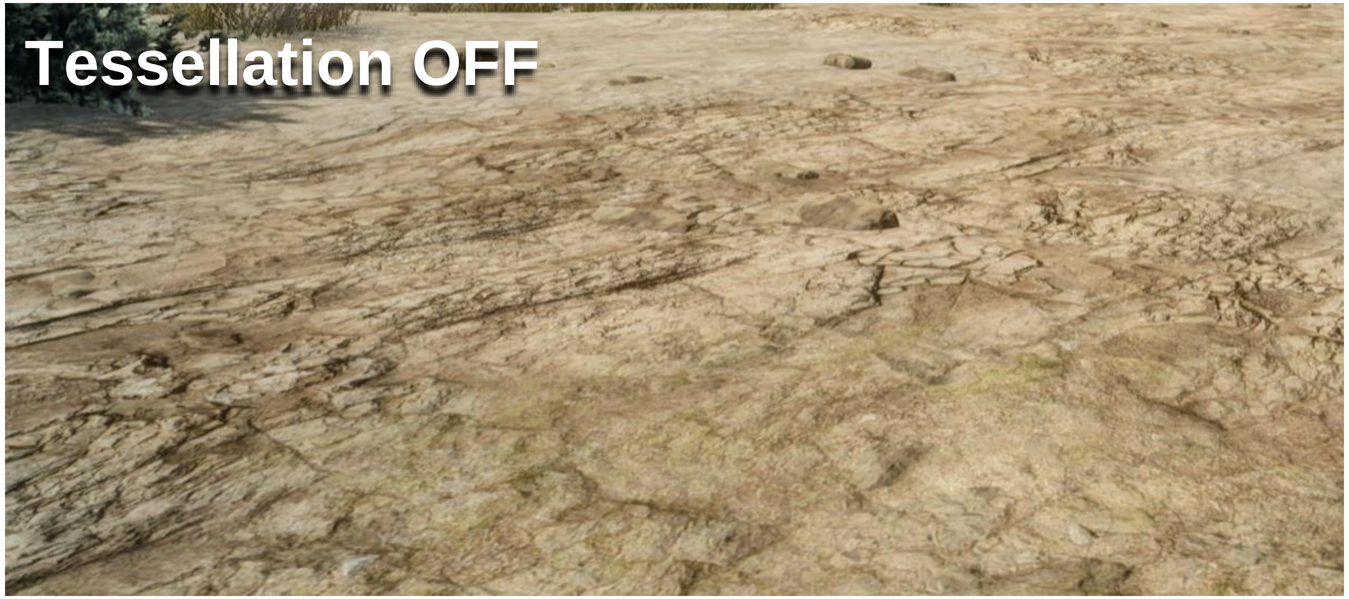
GameWorks: Materials & Textures is a set of tools targeted at 3D and graphics artists that leverages the power of Deep Learning and NVIDIA CUDA

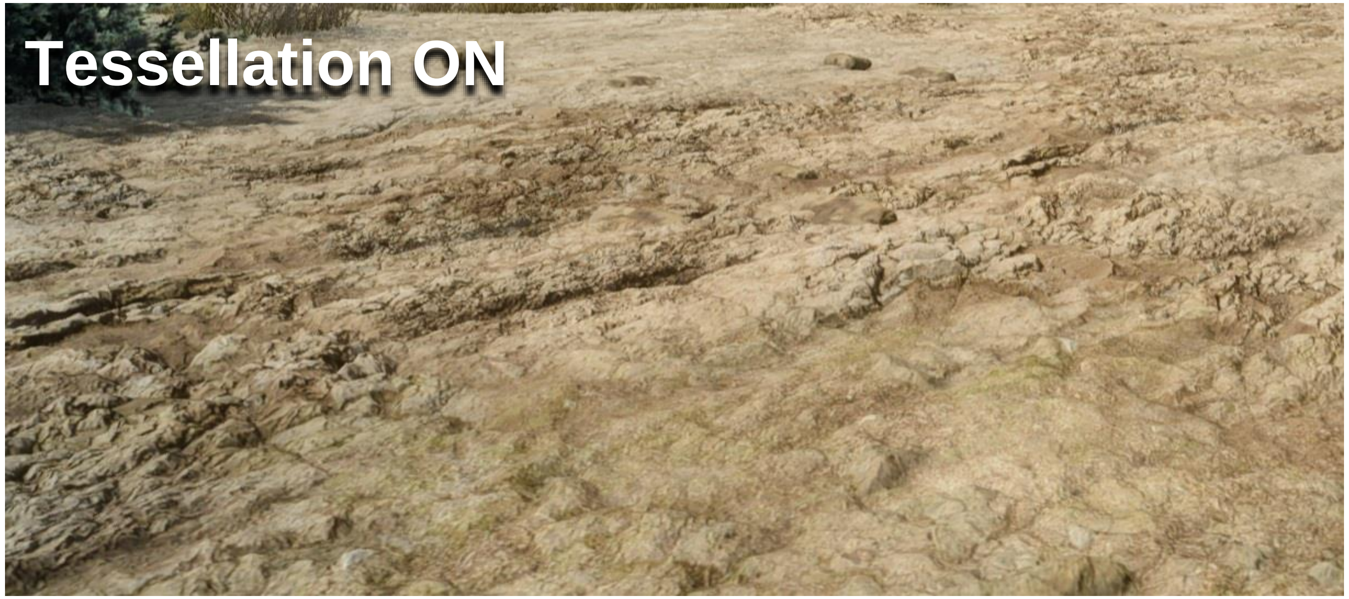
Super-Resolution

Photo To Material

Texture Multiplier

Normals To Displacement

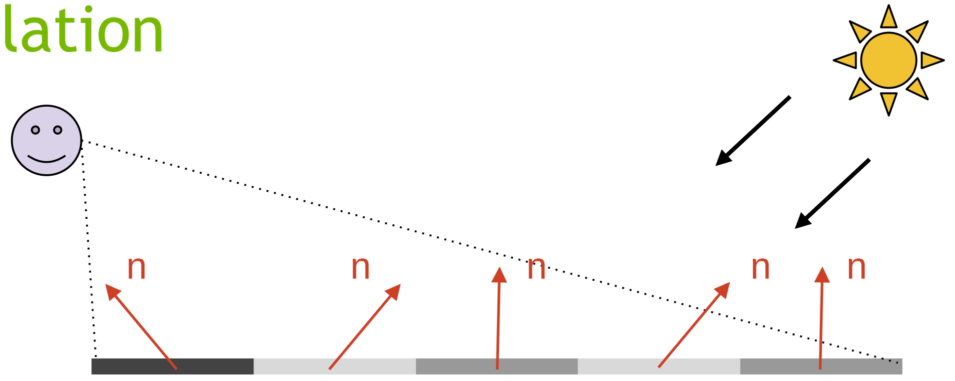

Tessellation OFF


Tessellation ON

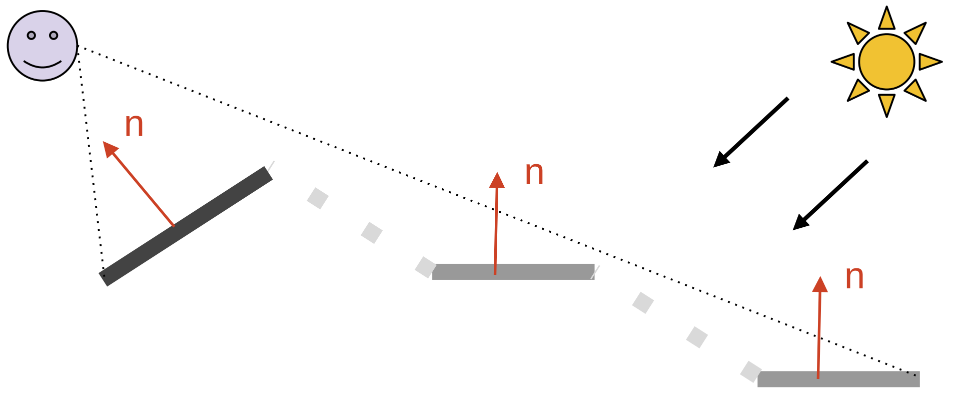
Lighting Perception

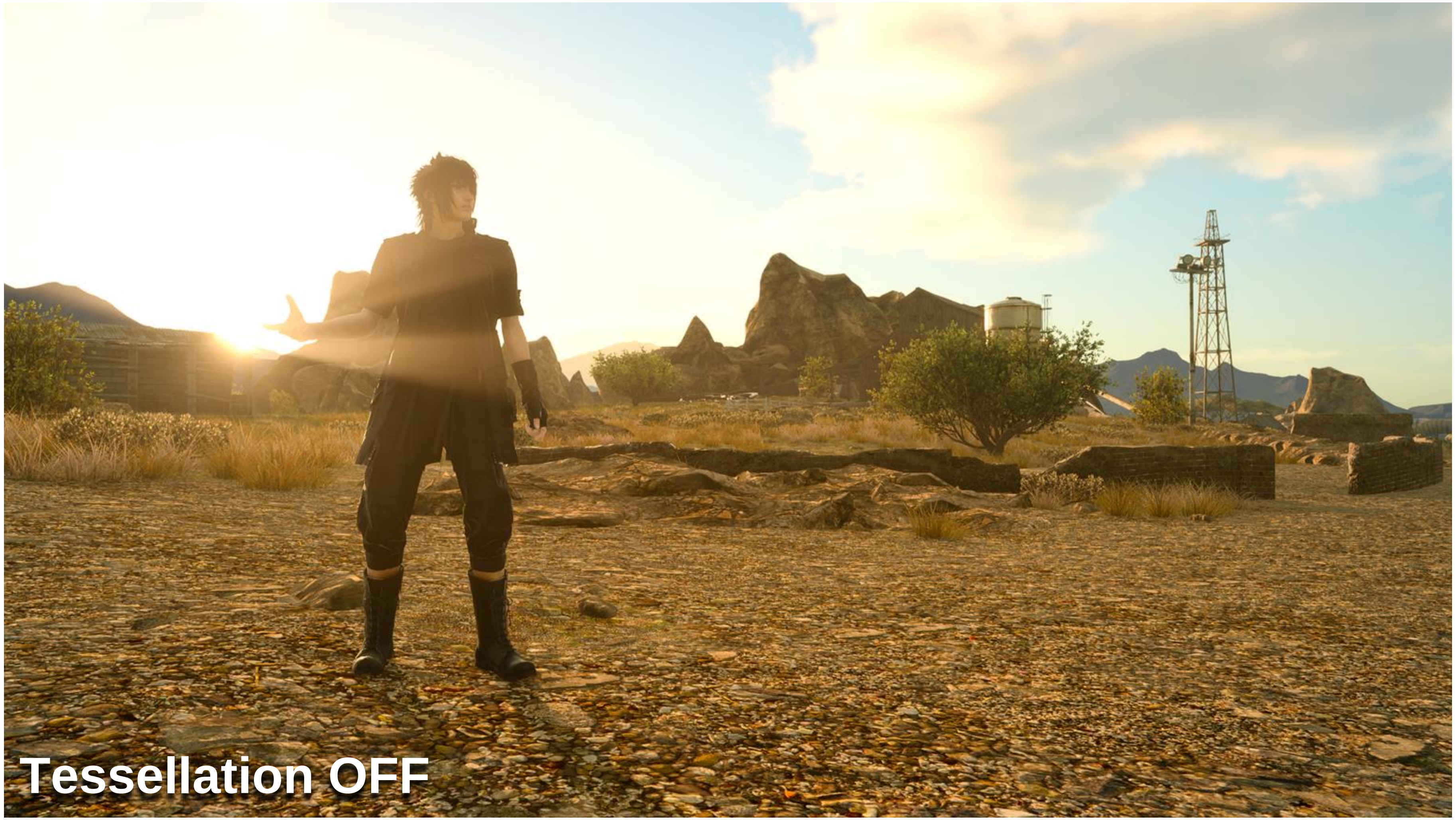
Terrain Tessellation

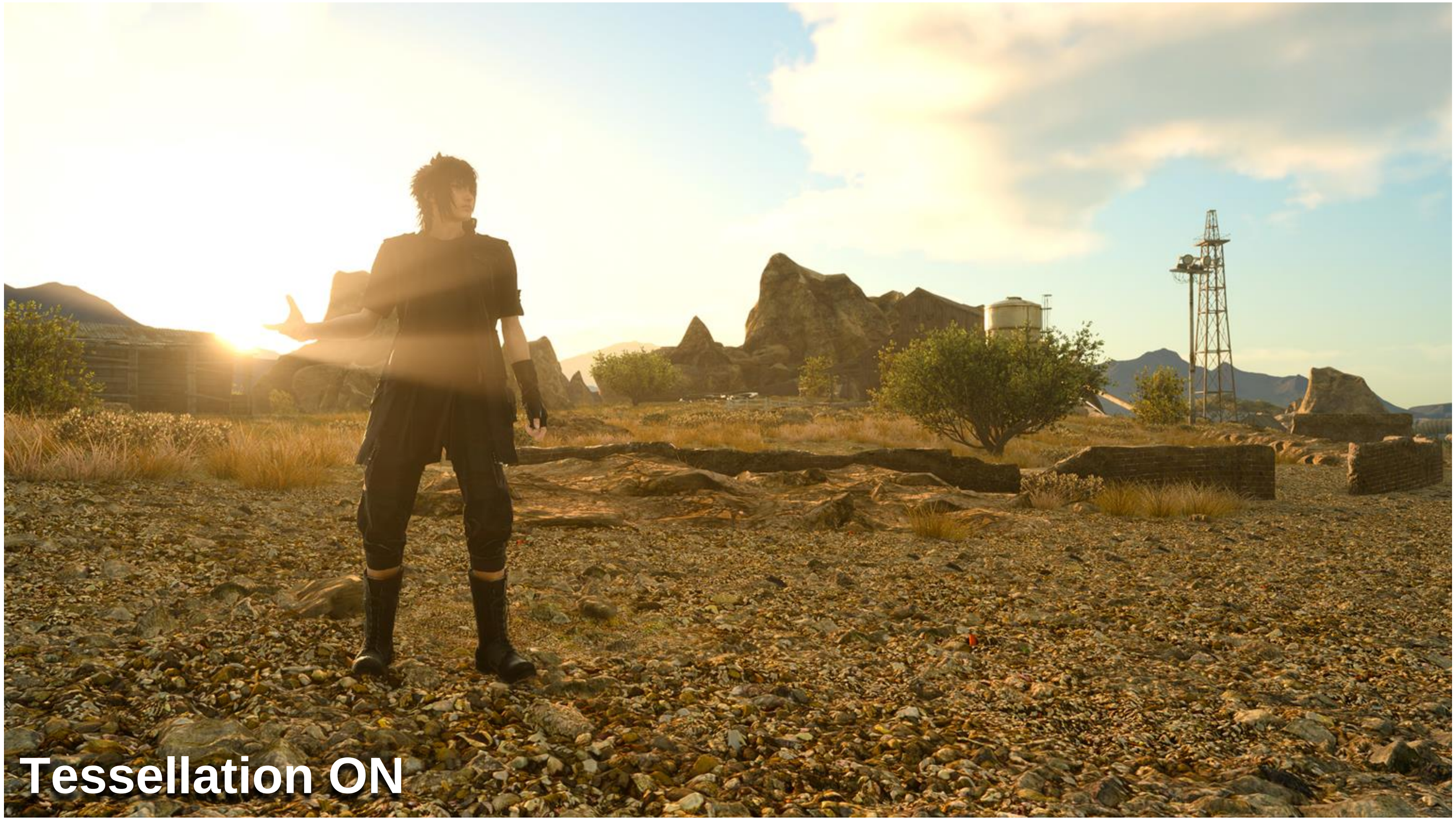
- Without displacement diffuse lighting remains constant for every view angle

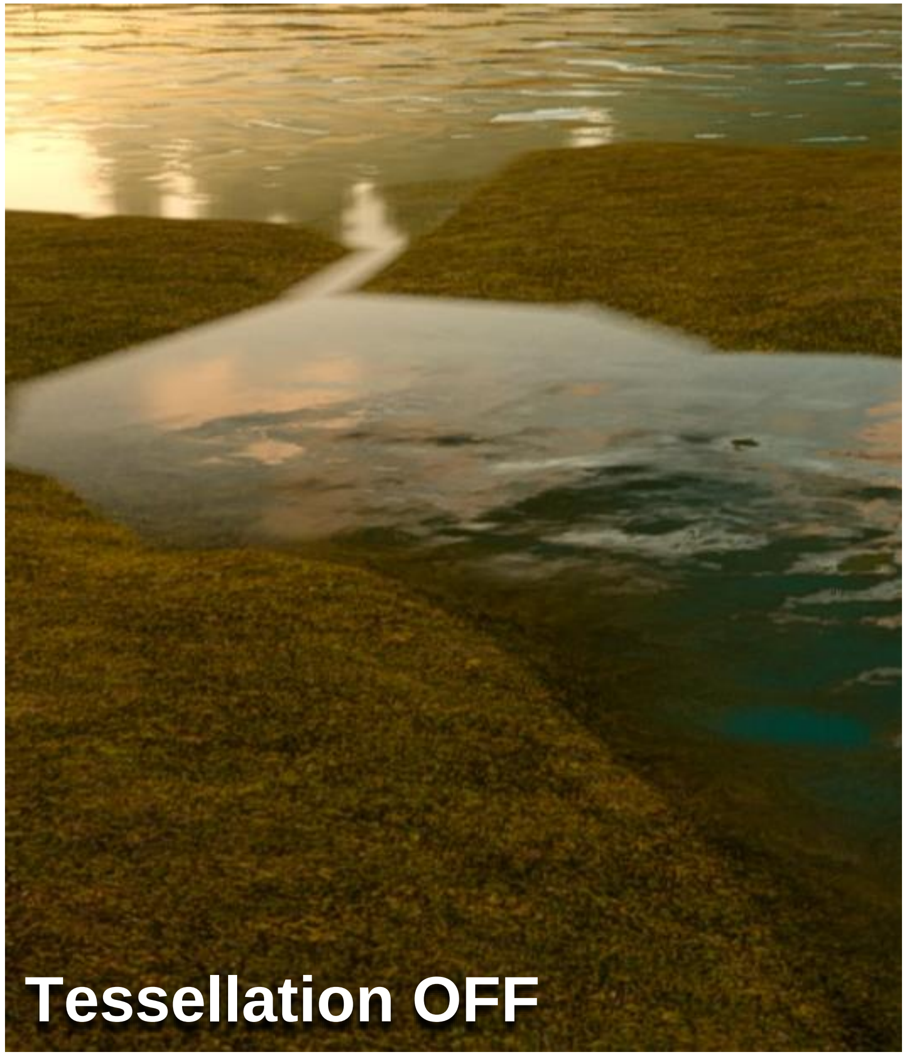


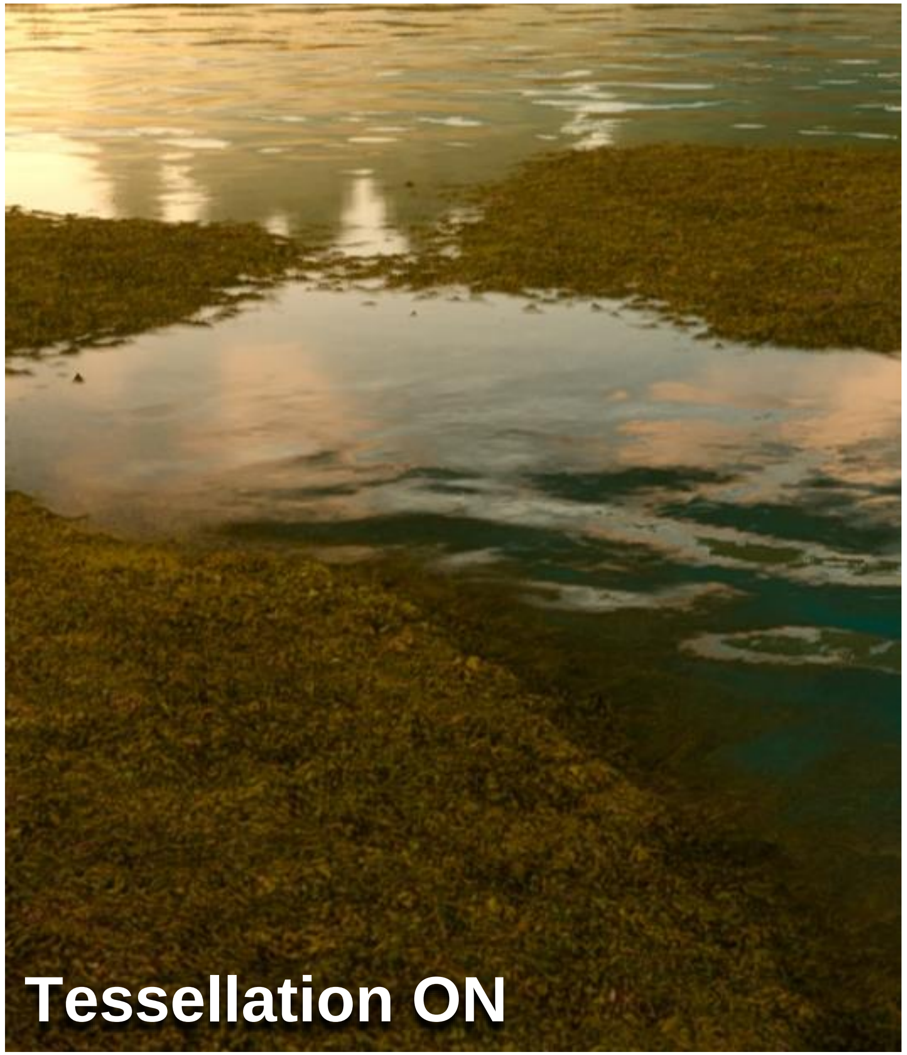
- With displacement applied we should observe less lighting if view and light vectors are opposite to each other and vice versa

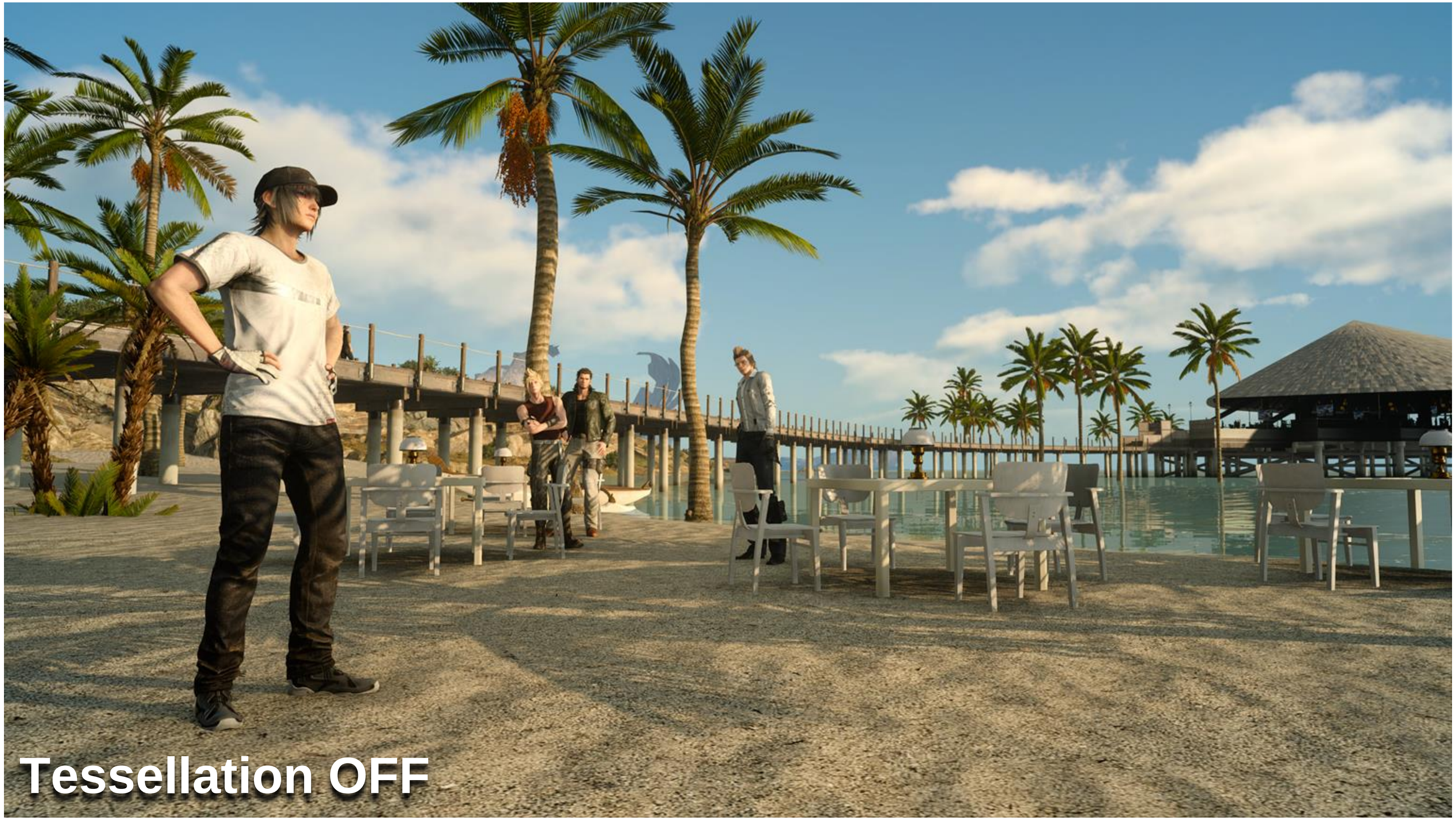


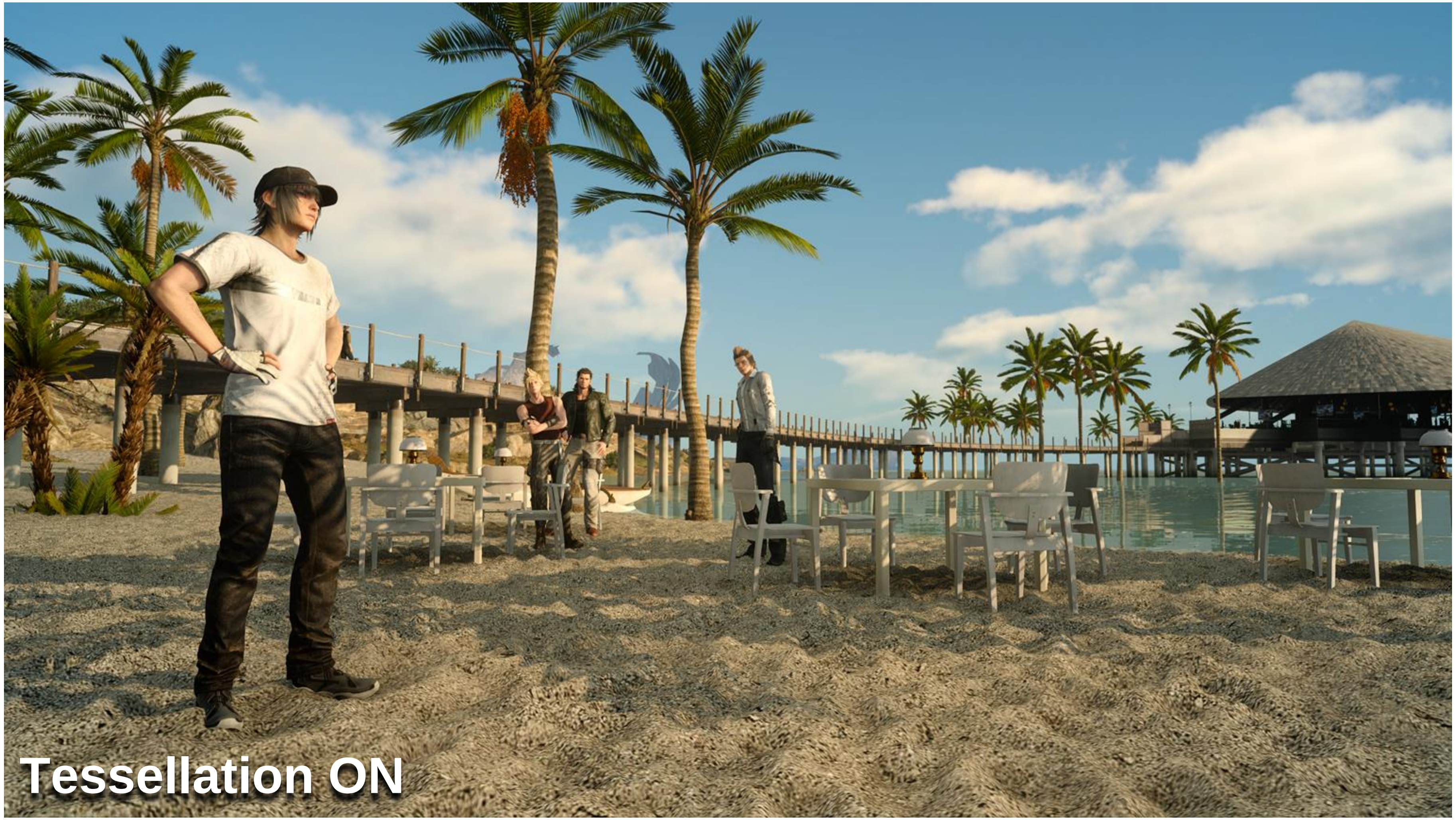

Tessellation OFF


Tessellation ON


Tessellation OFF


Tessellation ON


Tessellation OFF


Tessellation ON

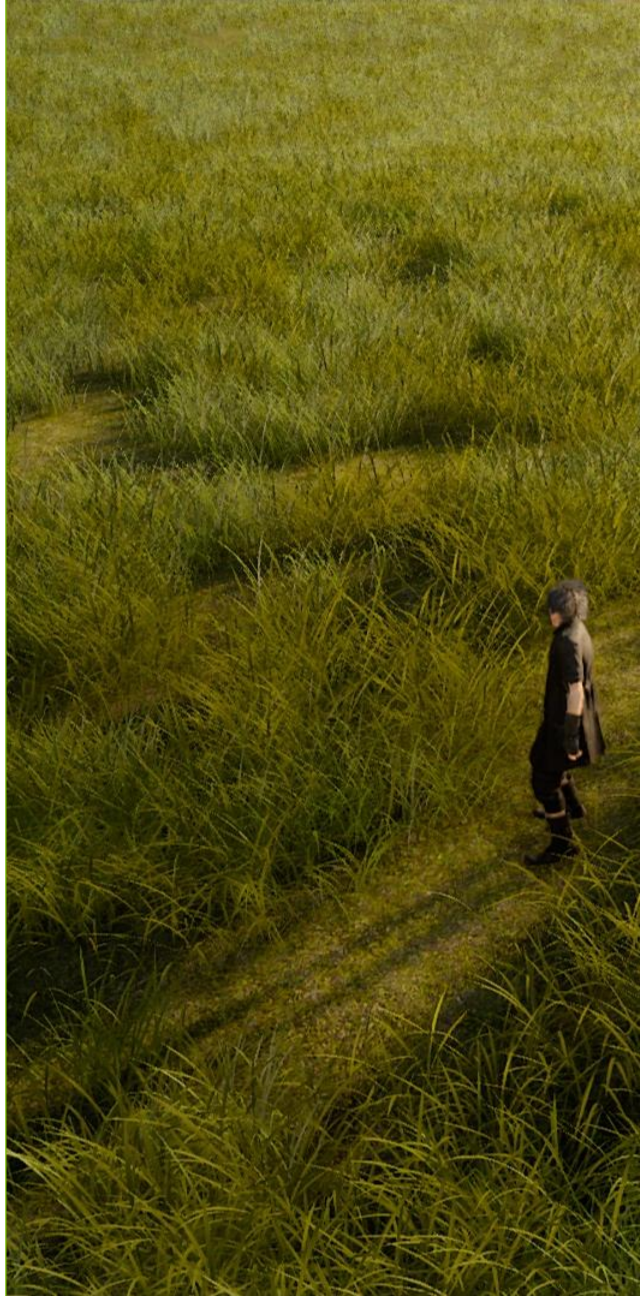
Turf Effects

Turf Effects

- Use existing foliage content for Turf data generation
  - Original foliage distribution and scales
  - Single mesh/asset forms several grass batches
  - Account for terrain slopes

- Tries to preserve original look and feel
  - Predictable quality and performance

- Special test map with all grass variations
  - Tweak once, apply everywhere



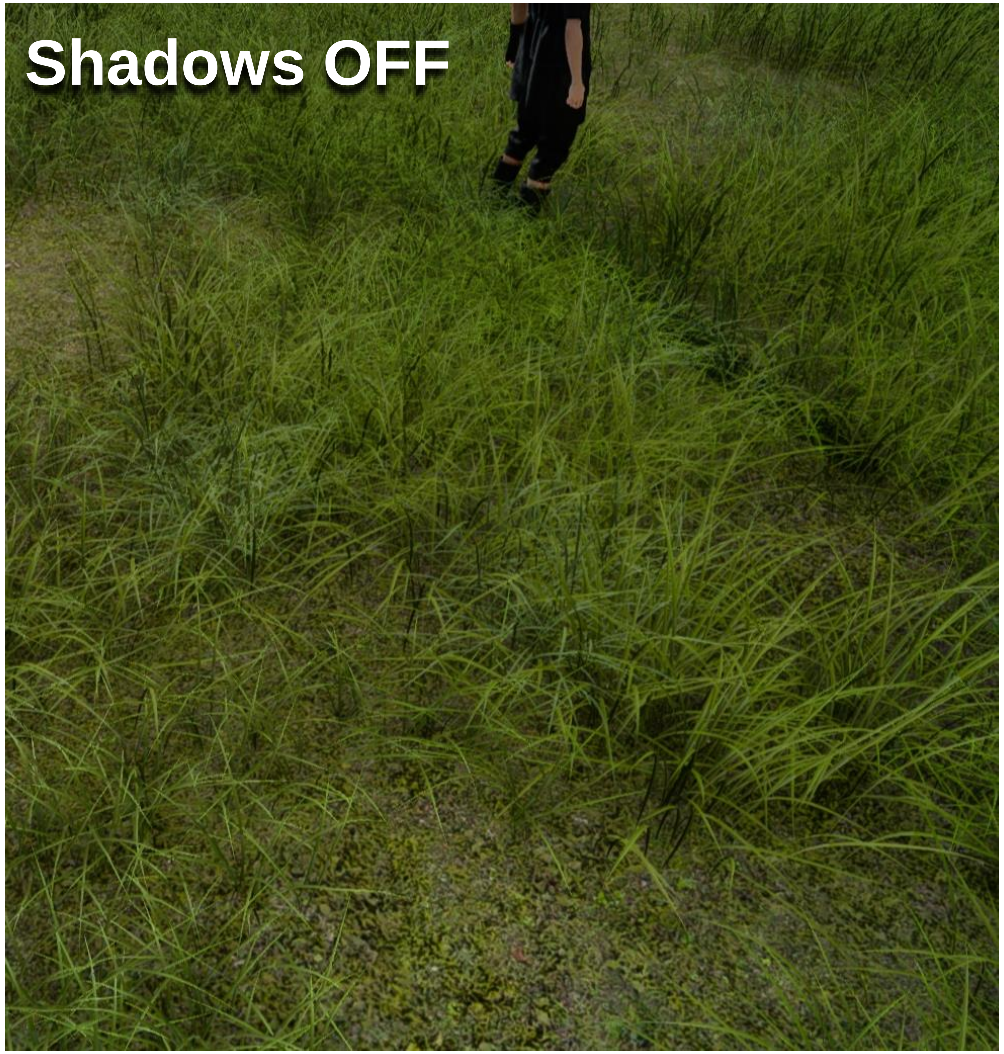
GDC

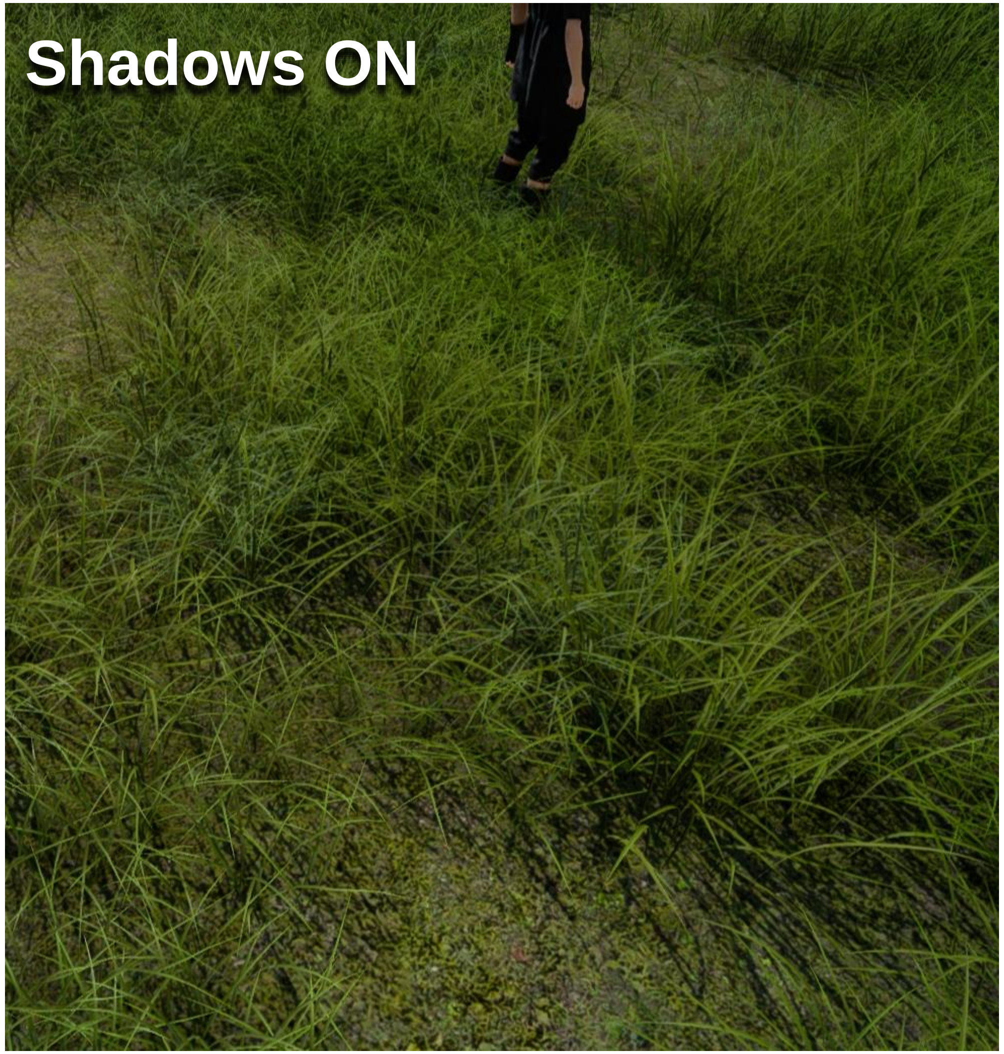
Rendering

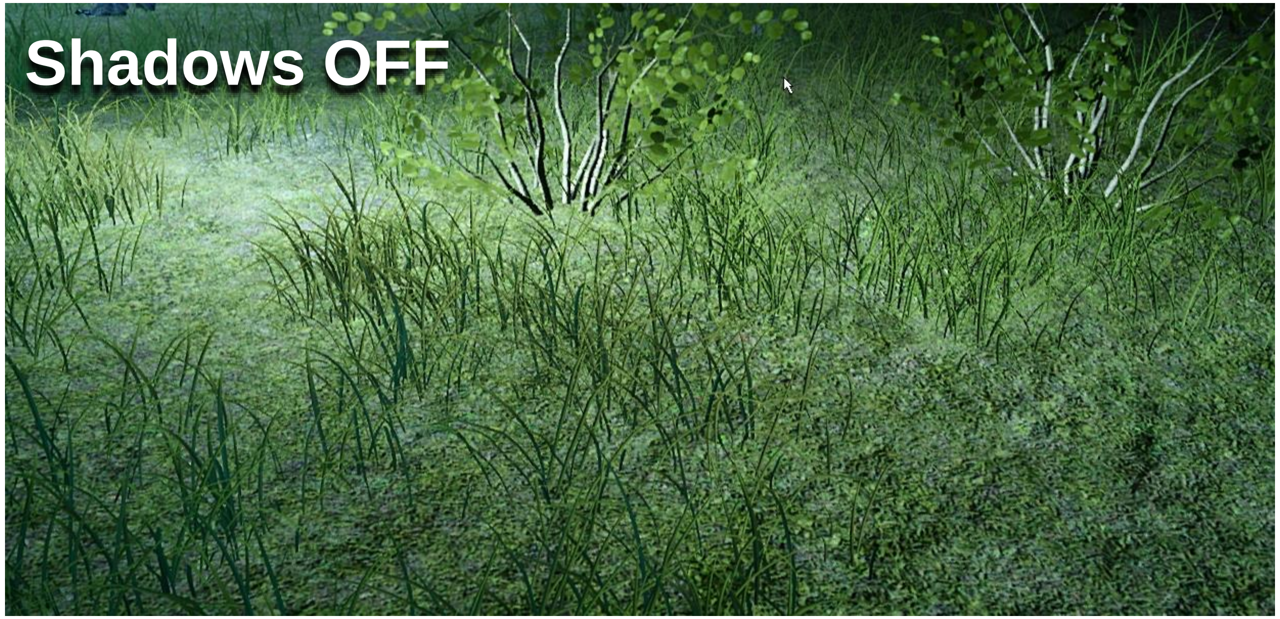
Turf Effects

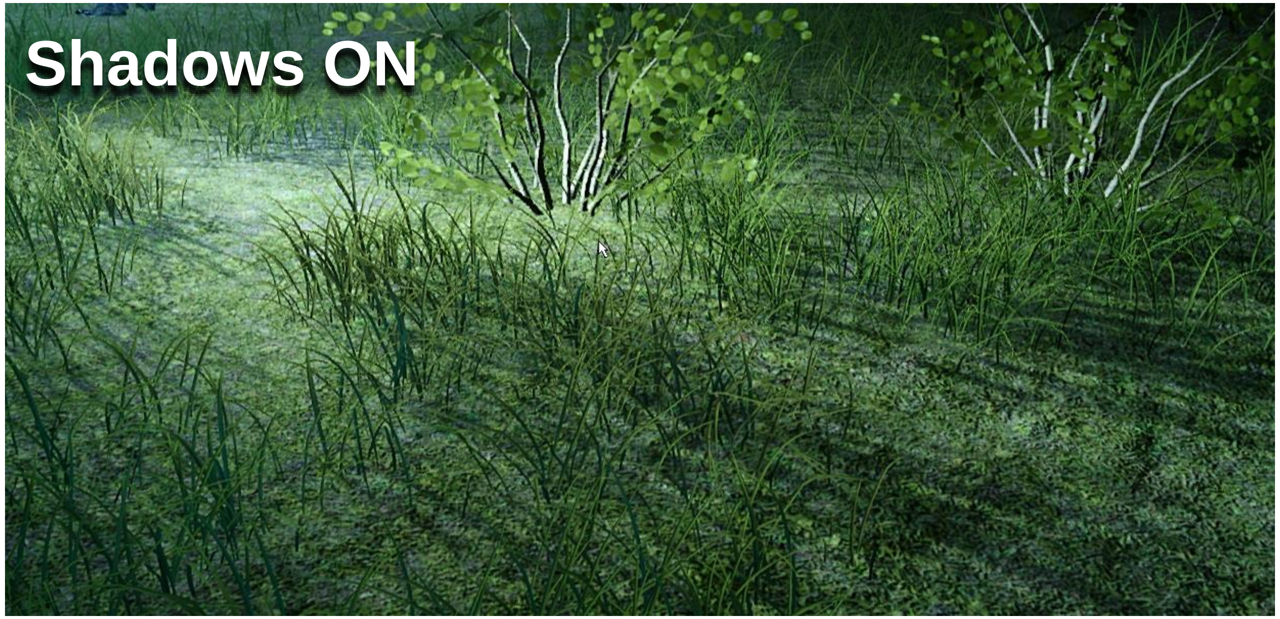
- Deferred shading with physically based material system
  - Fill GBuffer and enjoy the results

- All assets cast and receive shadows
  - Two nearest cascades for directional lights
  - Shadows from the flashlight at night time


Shadows OFF


Shadows ON

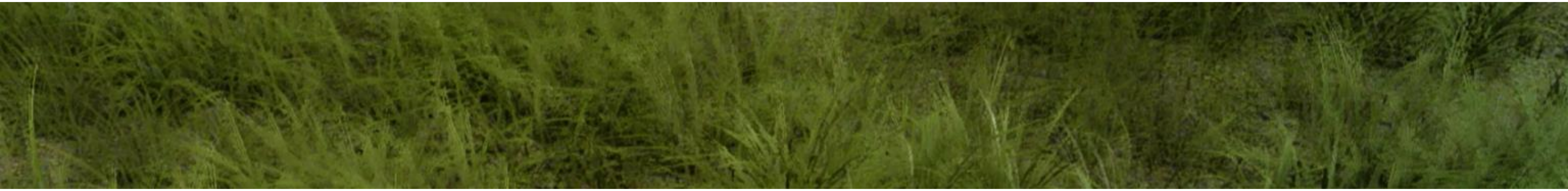

Shadows OFF

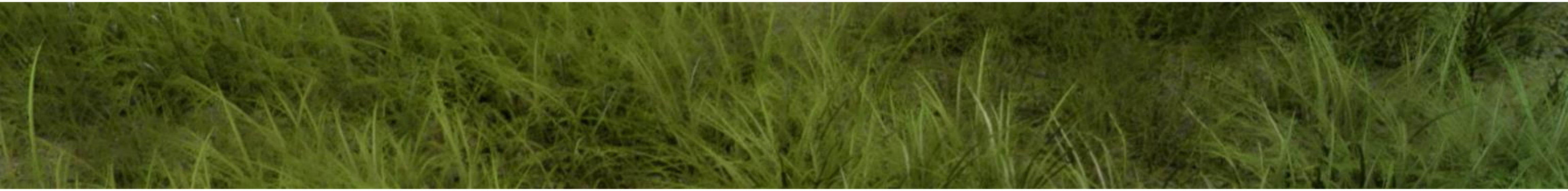

Shadows ON

Rendering

Turf Effects

- Temporal AA
  - Motion vectors were added to the library during integration
  - Used approximated velocities from the control shapes





Rendering

Turf Effects

- Per patch occlusion culling
  - Test conservative boundary boxes against depth buffer from previous passes
  - Use DrawIndirect()

- Finer grained occlusion culling is WIP

Physical Simulation

Turf Effects

- Procedural wind-driven animation plus interaction
- Render local heightfield for blades placement and simulation
- Use existing physical meshes for interaction

- Persistent deformation
  - Use separate buffer to store dynamic patch data(positions, velocities, deformation)
  - Time-based relaxation

The Numbers

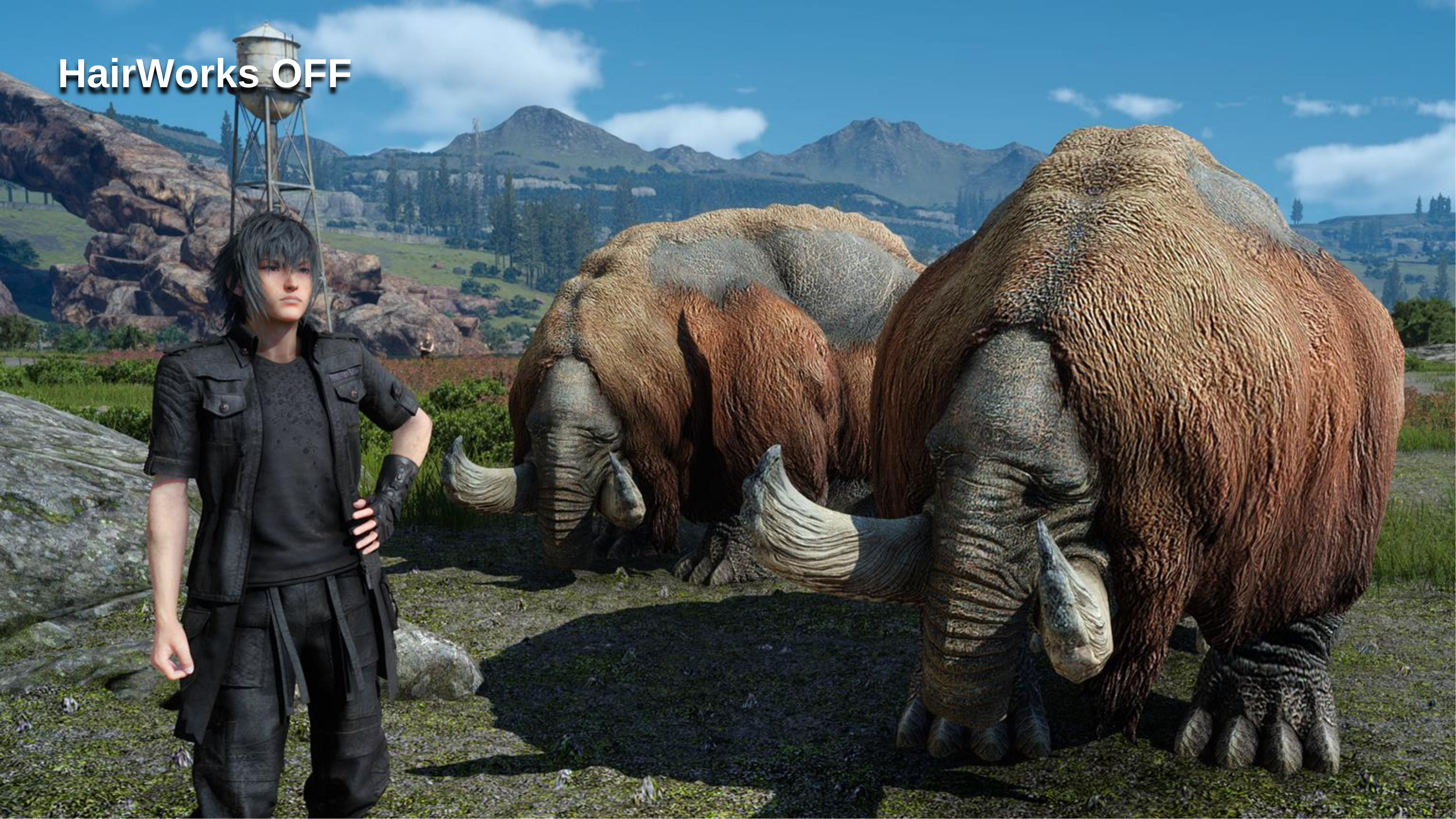
Turf Effects

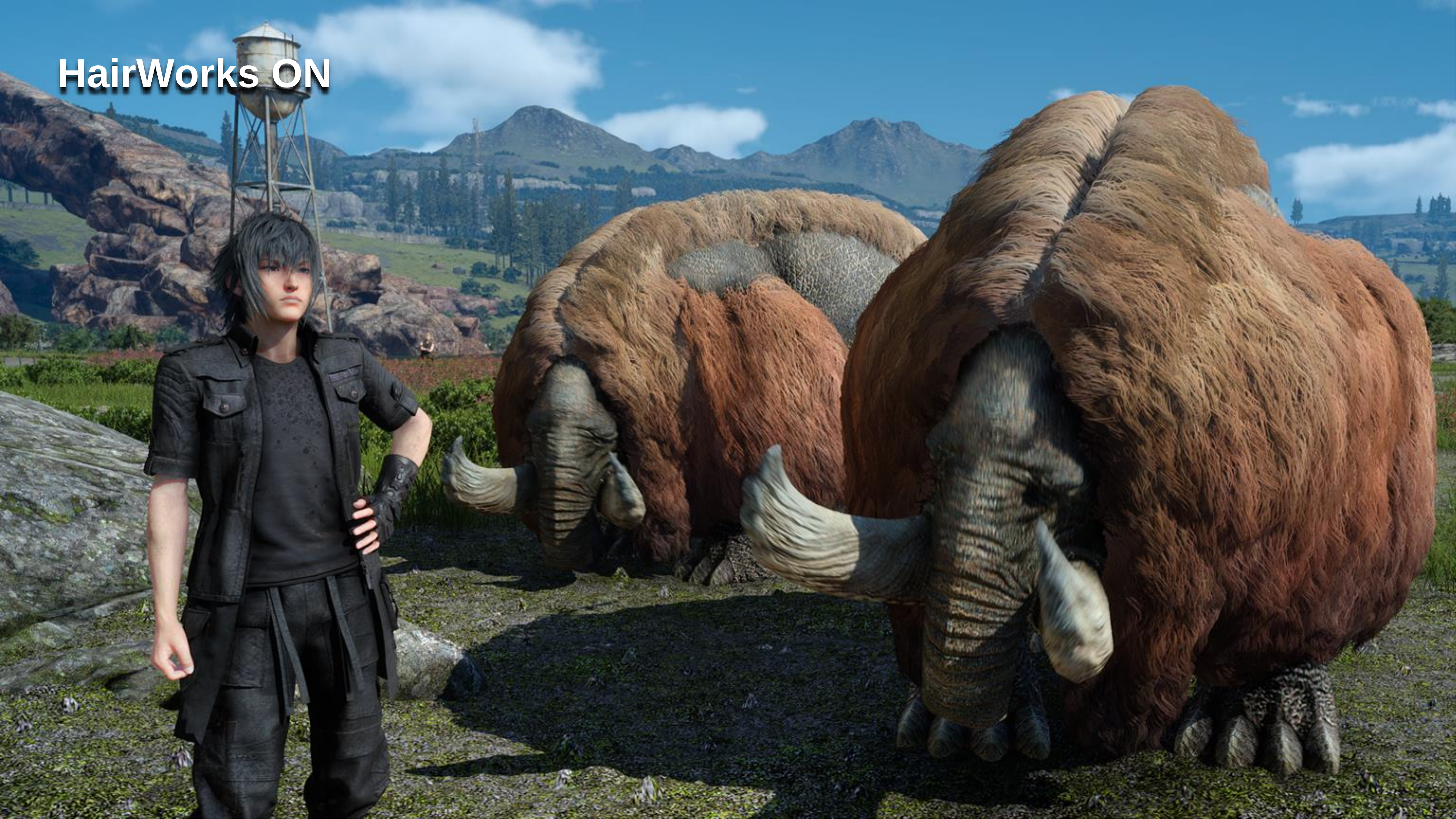
- Single grass grid covering 250 000 square meters

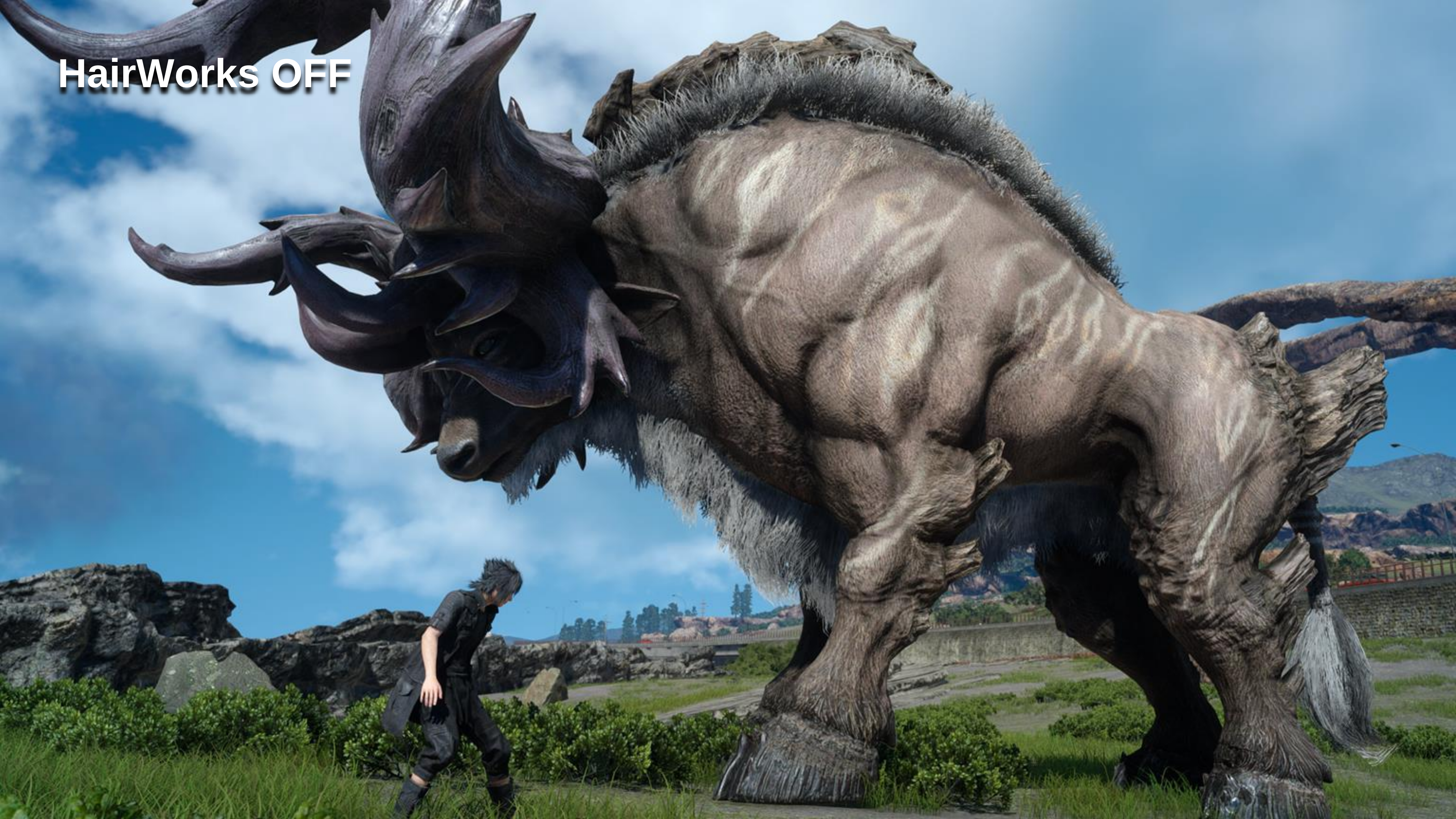
- 200 x 200 grid of patches
  - 40 000 patches
  - 2500 grass blades per patch
  - Up to 100 000 000 of grass blades

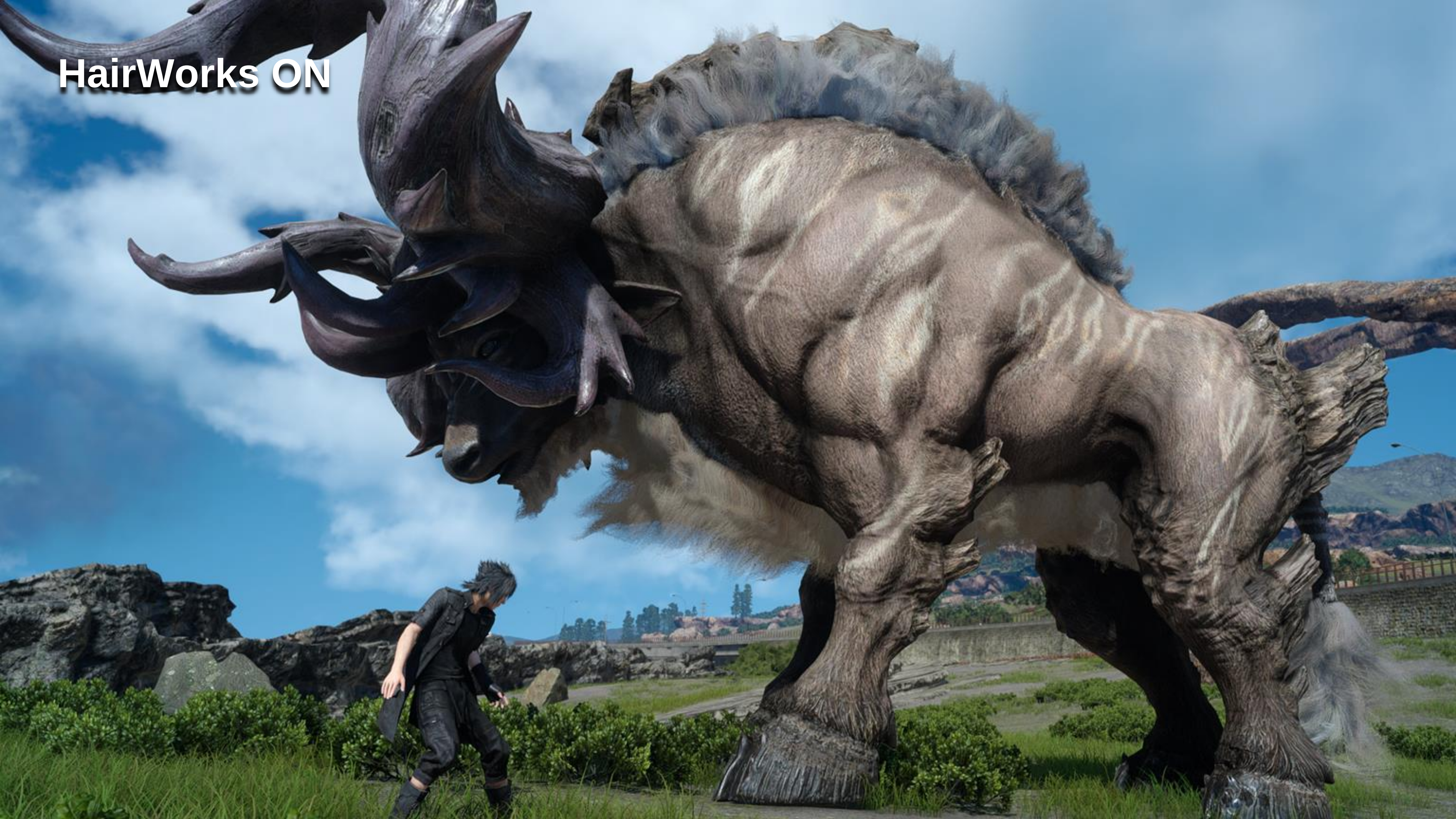
- 16 different assets for the whole world

Hair Works


HairWorks OFF


HairWorks ON


HairWorks OFF


HairWorks ON

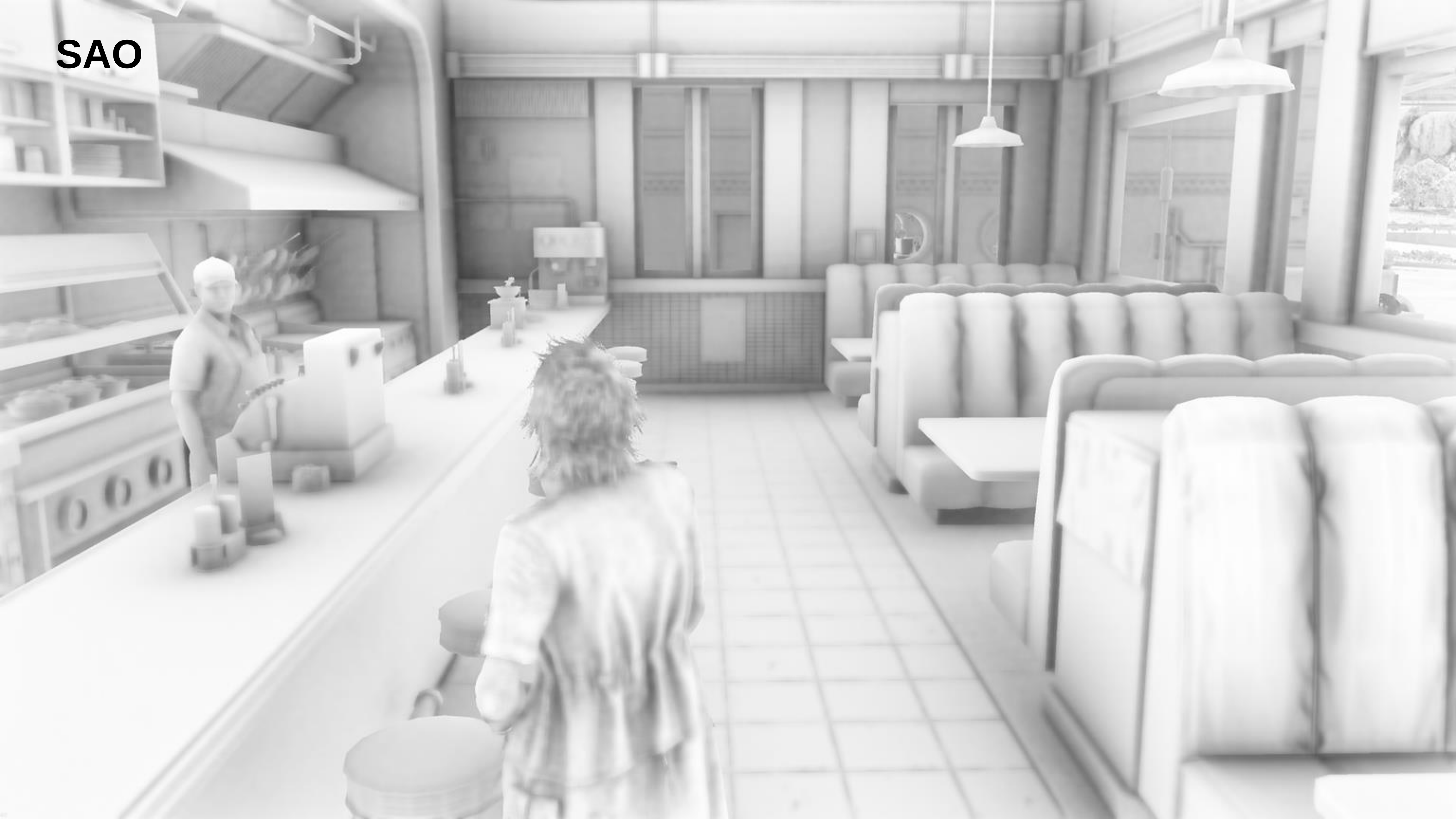
Hair Works

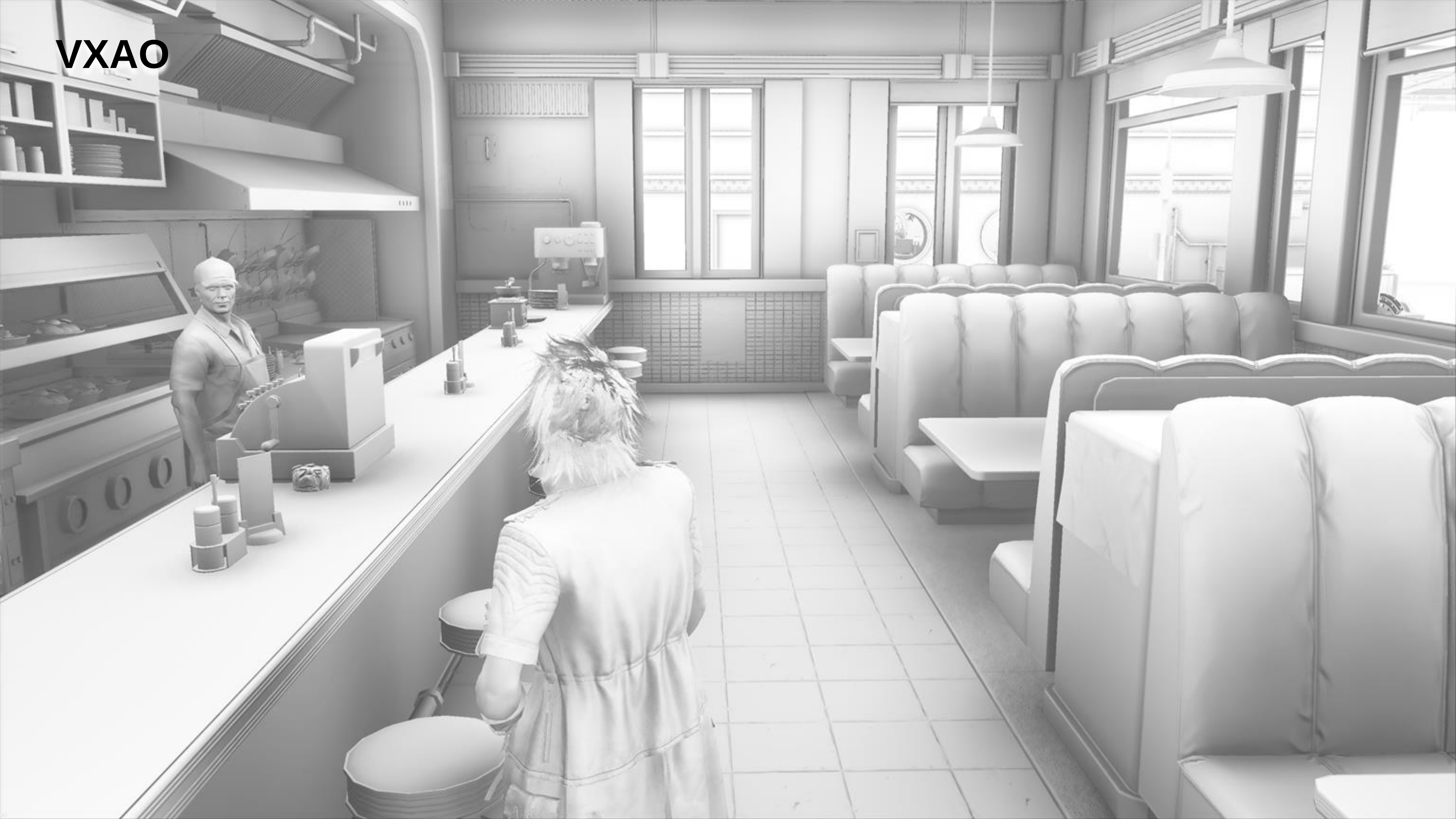
Integration notes

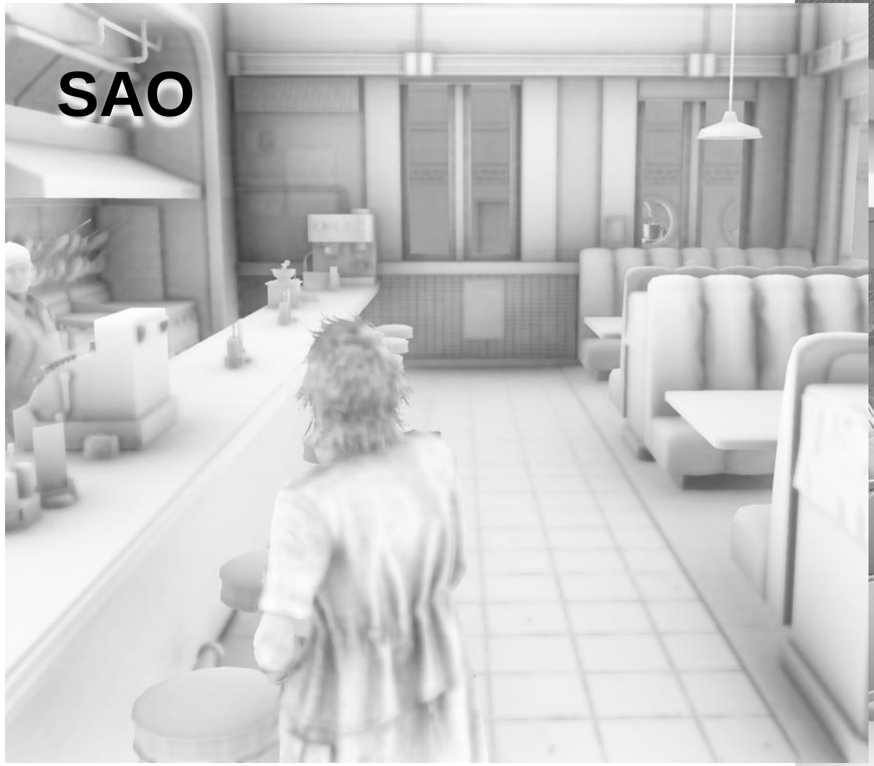
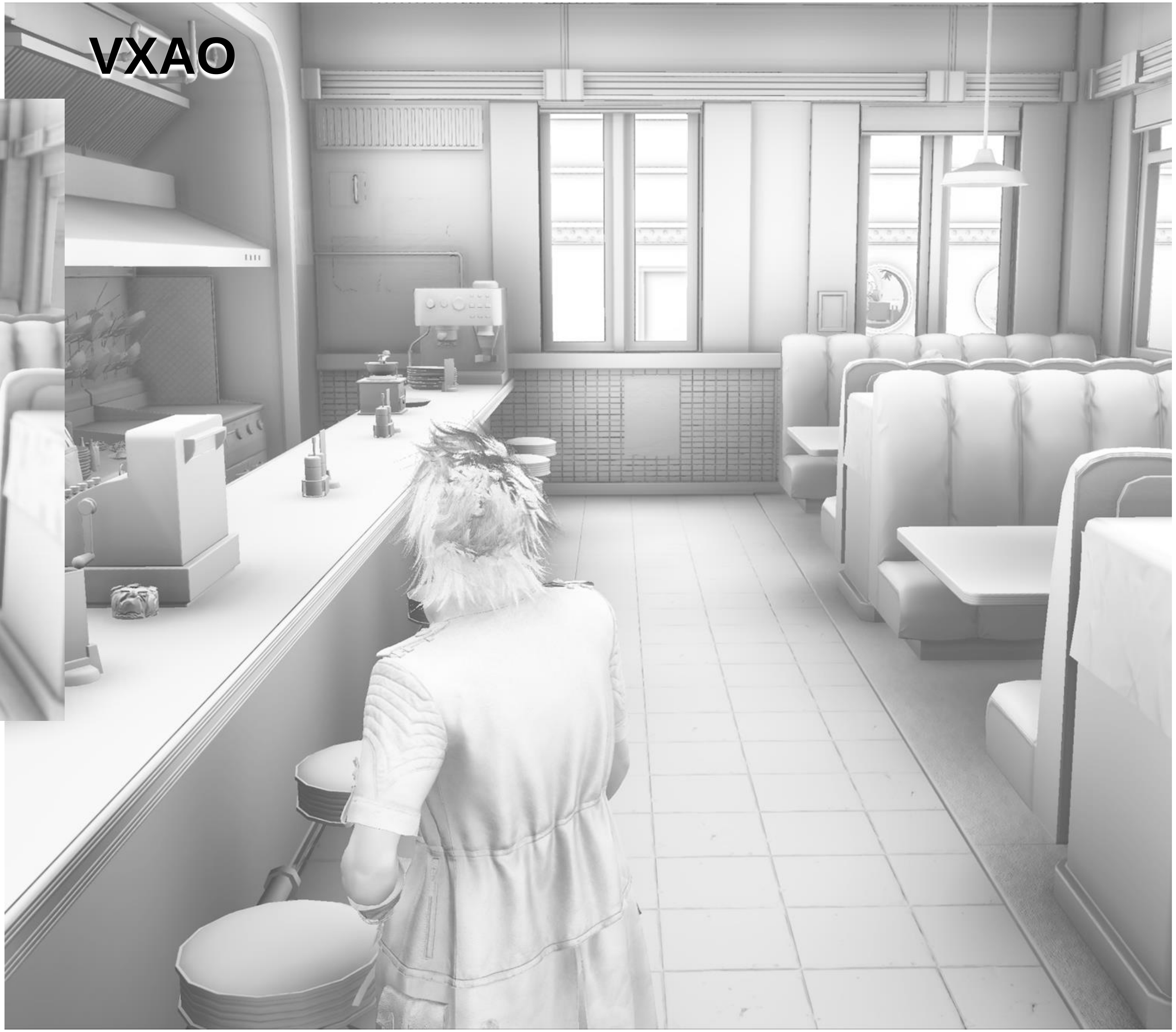
- HW Render pass was moved from forward pass to G-buffer pass.
  - Needed to fill velocity buffer.
- Motion vector was calculated from hair strand’s control point.
- Shading/Lighting was left to the Luminous Engine. It just filled G-Buffer.

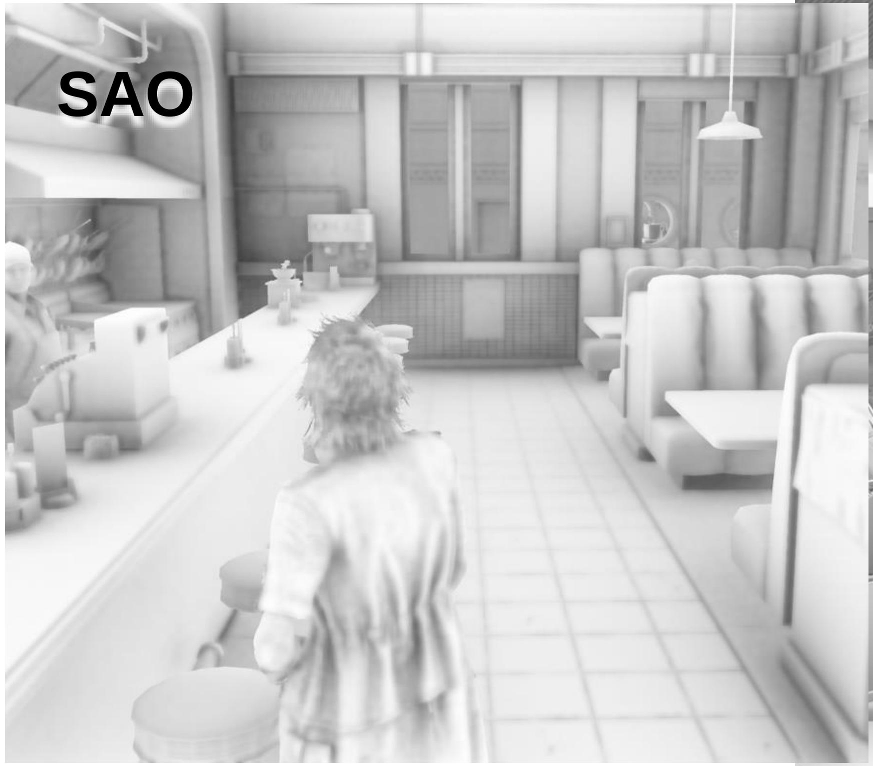
VXAO

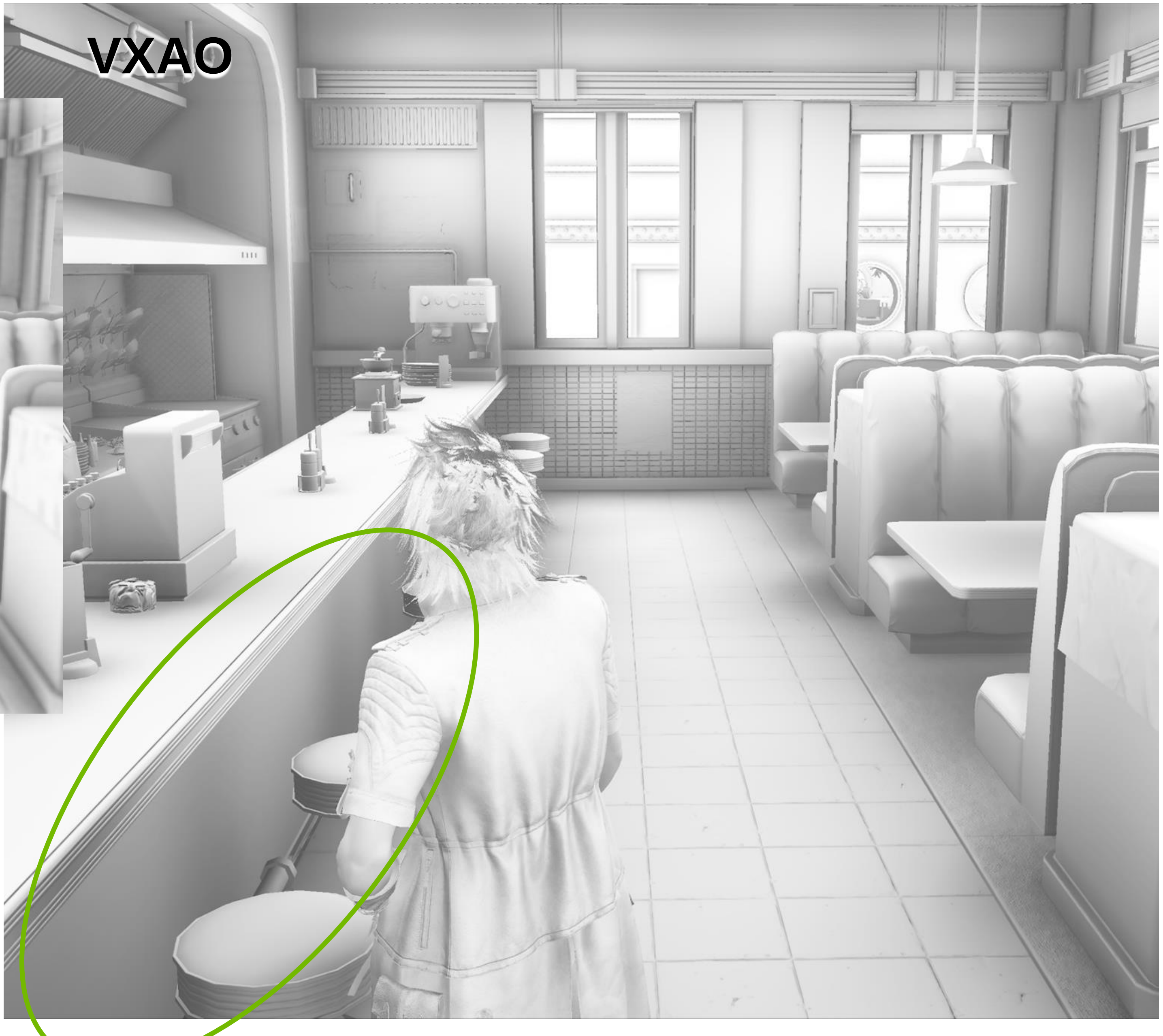
SAO

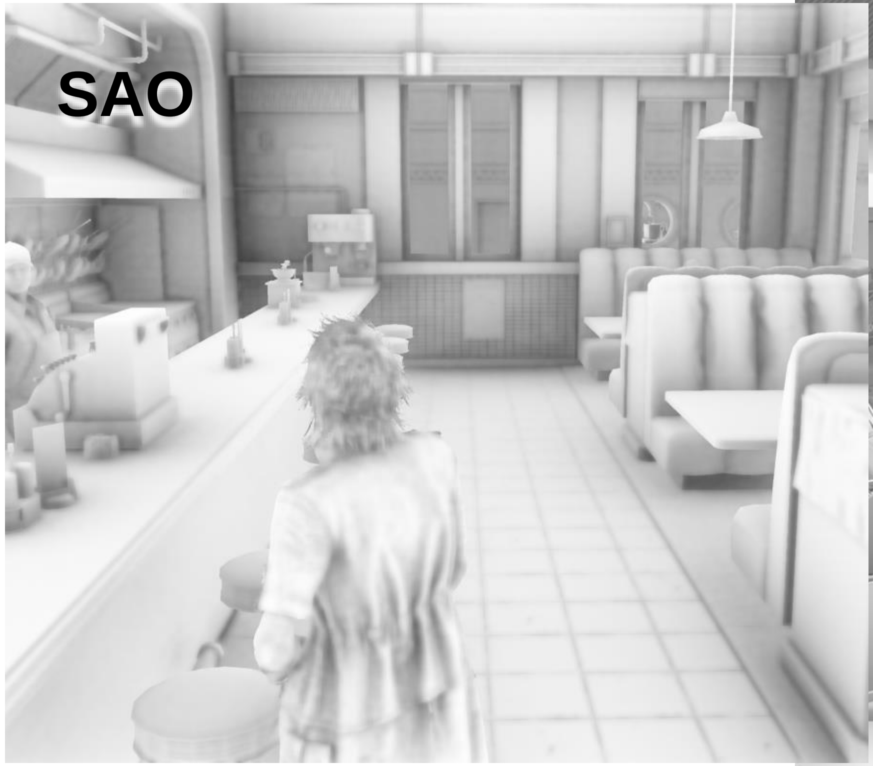


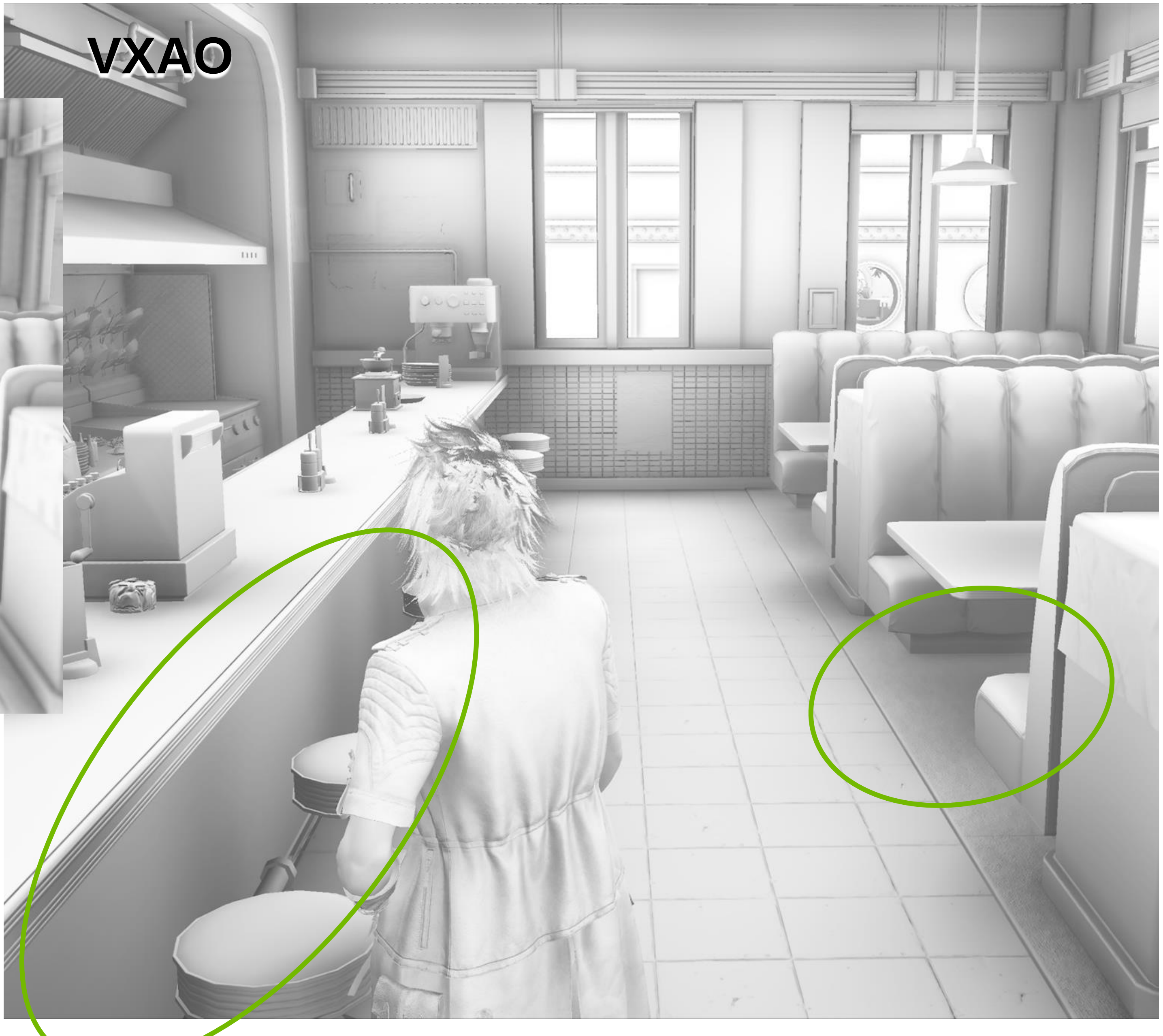

VXAO


SAO

VXAO

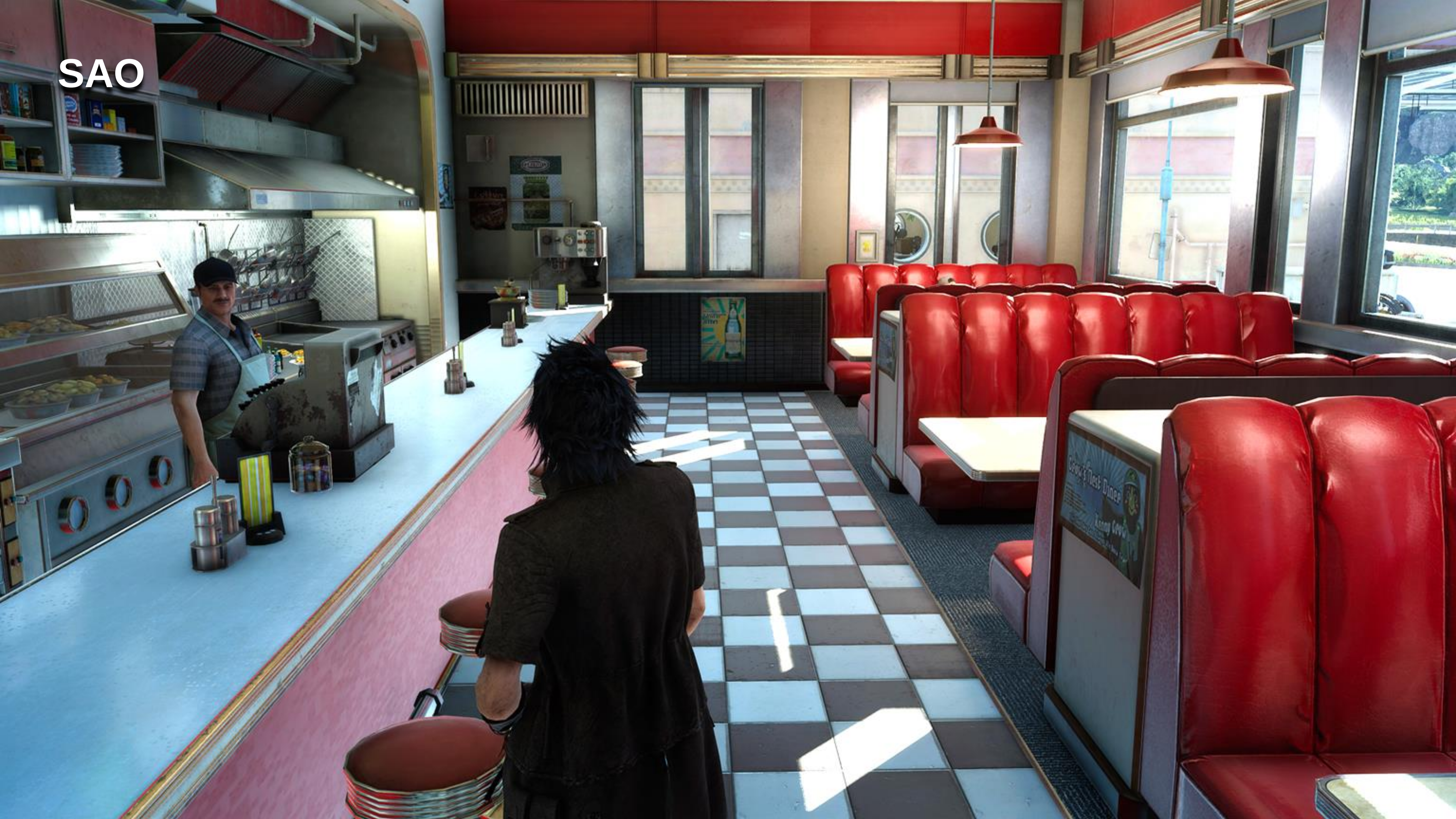

SAO

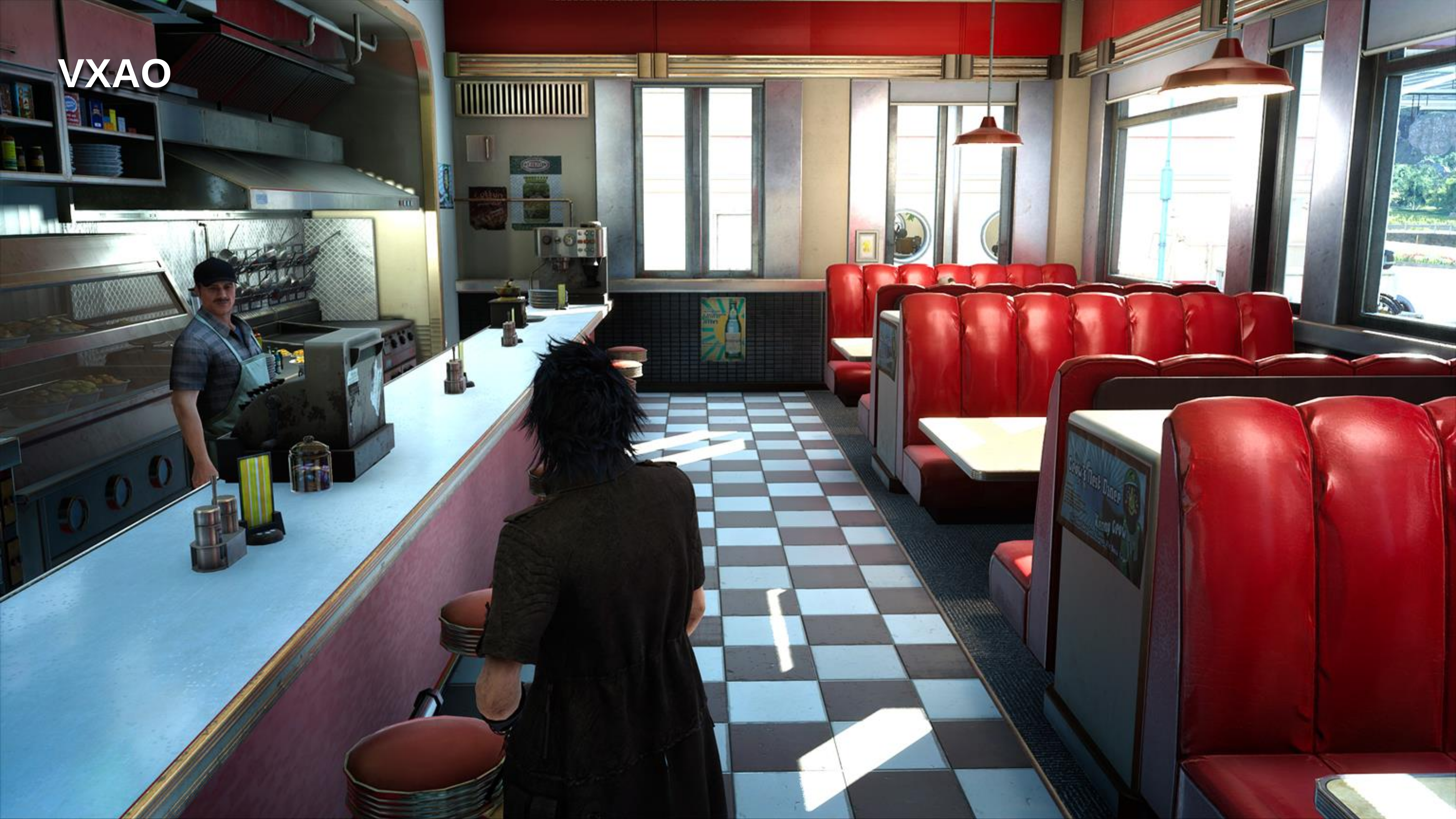

VXAO

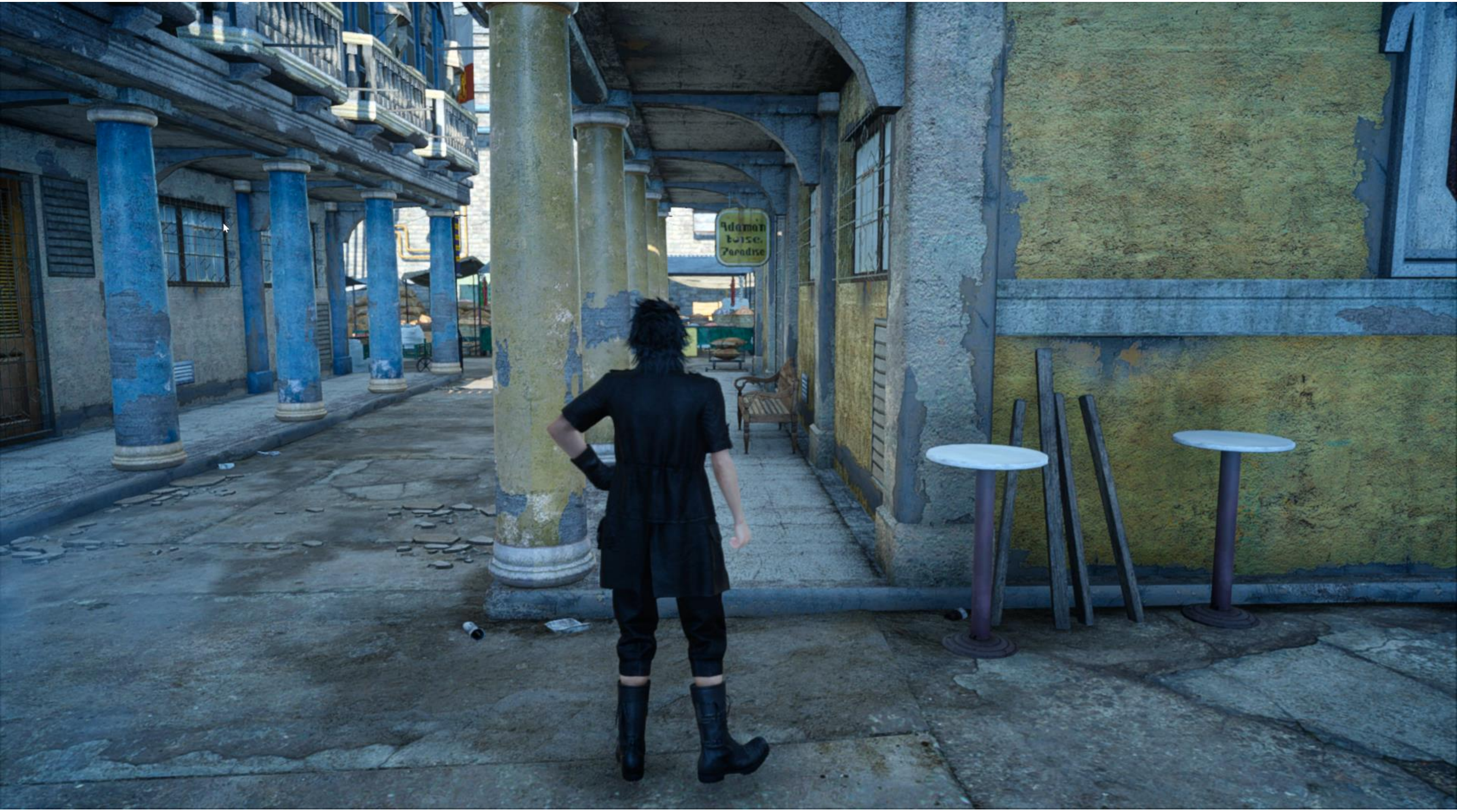

SAO


VXAO

SAO




VXAO

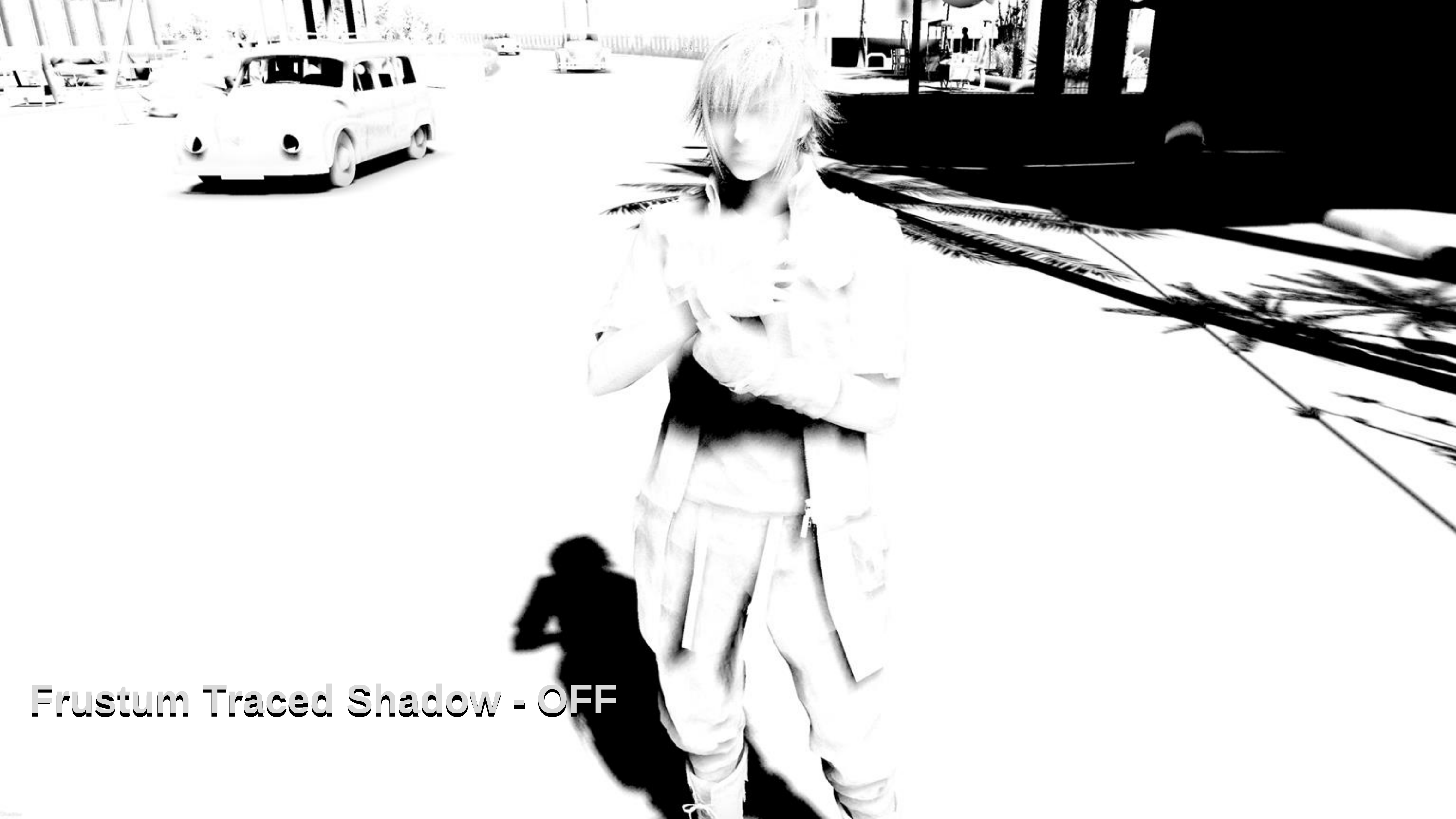

Adaman toise Paradise

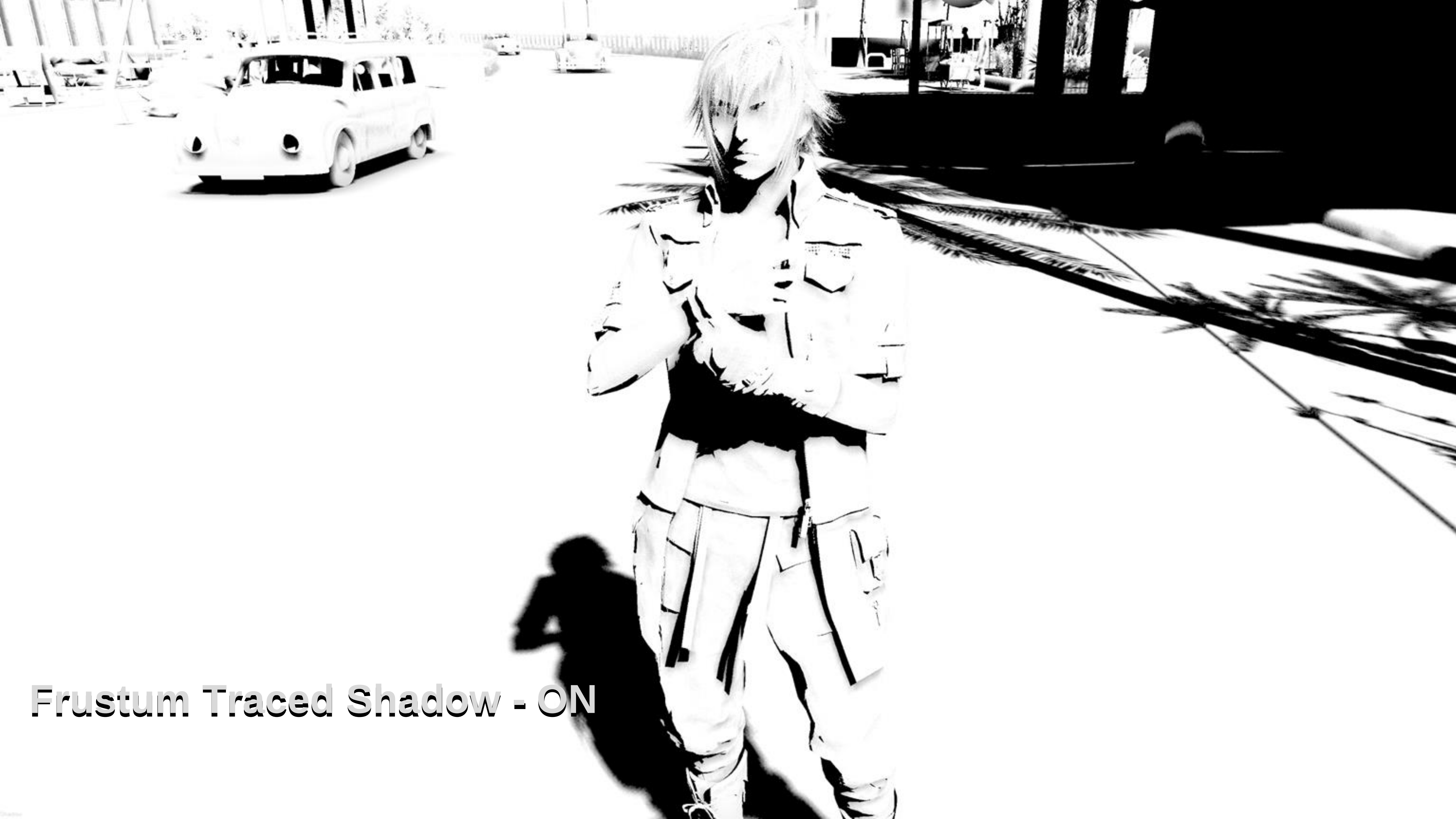
VXAO

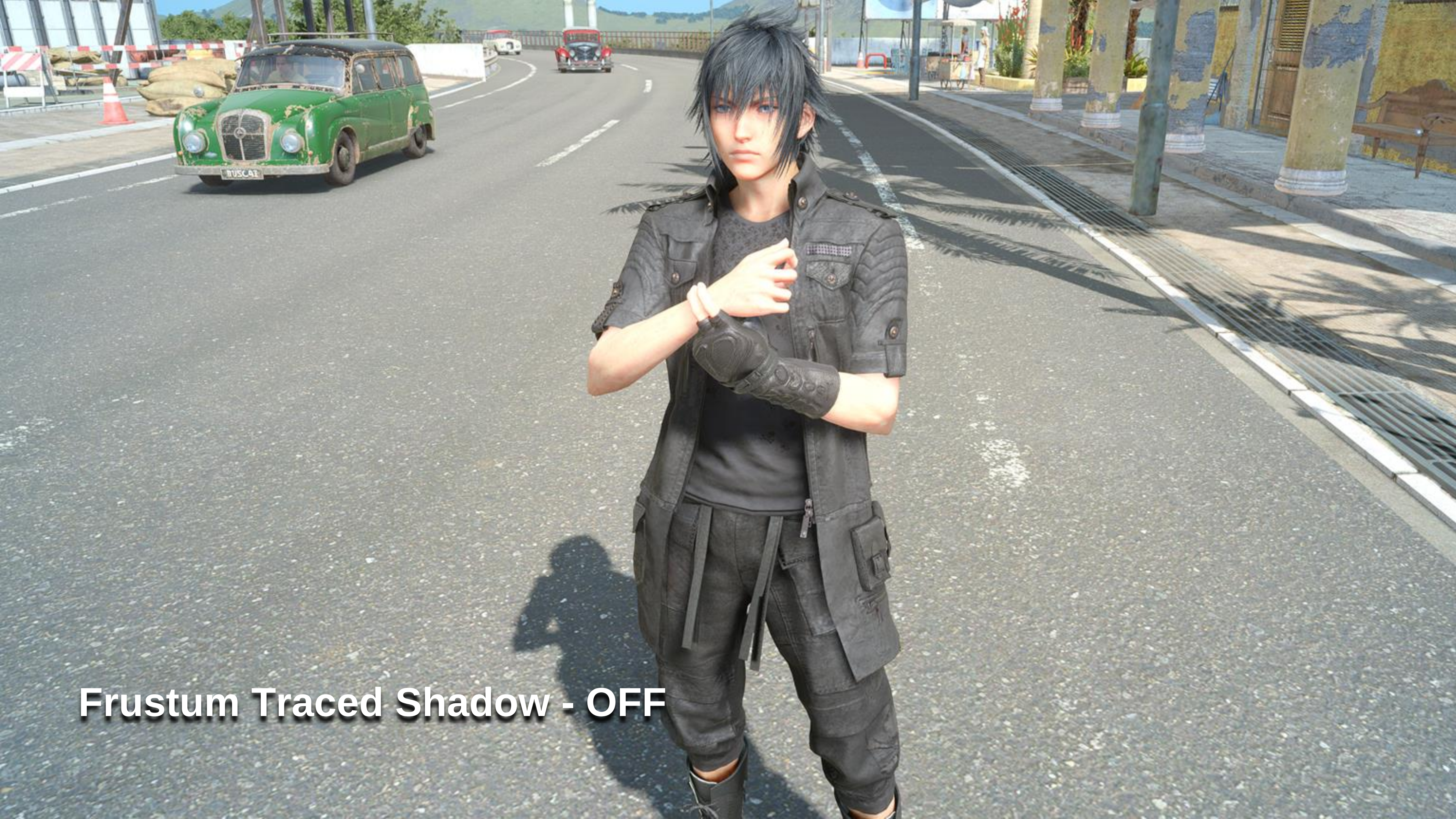
Integration notes

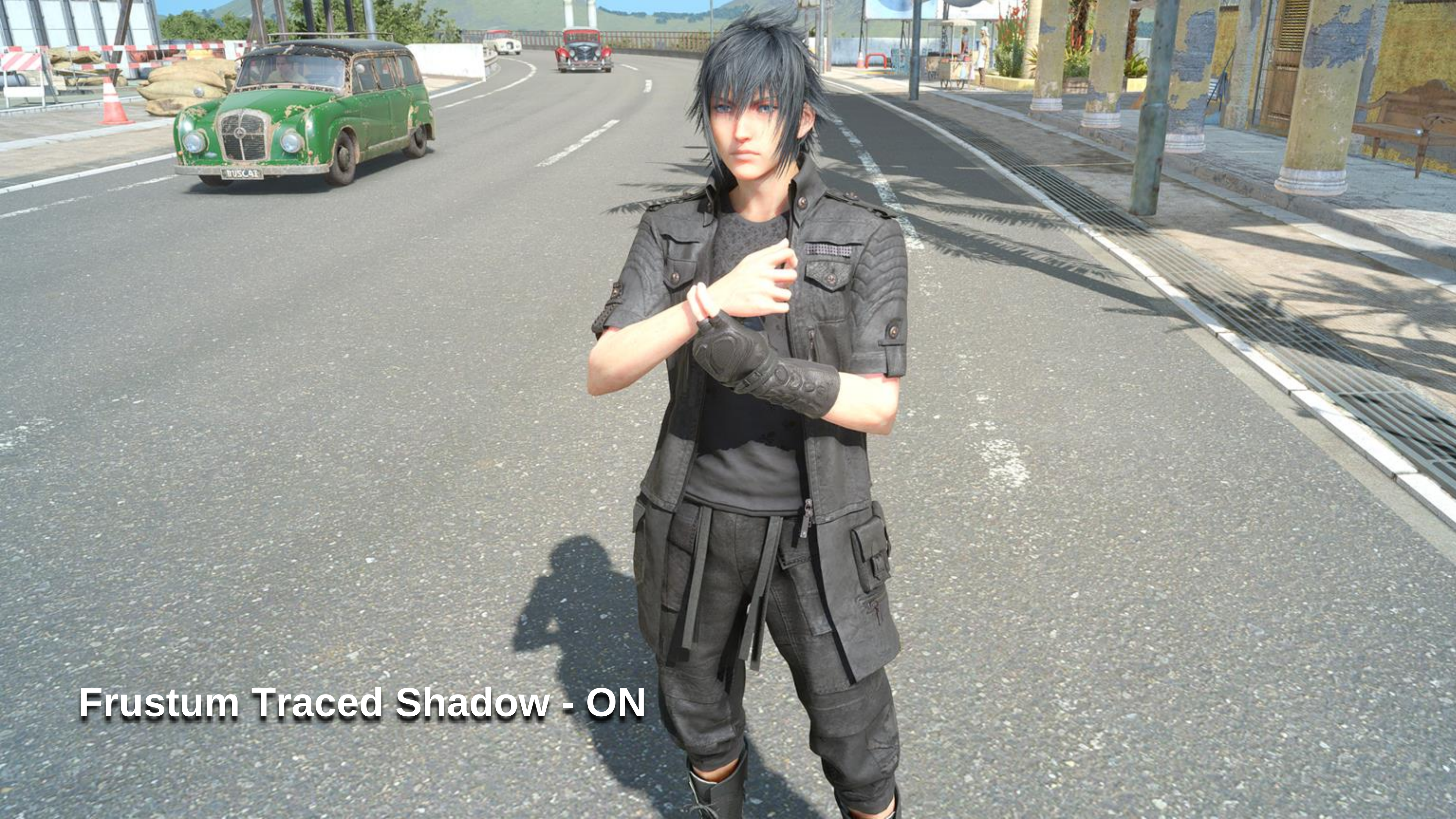
- The result of Cone Tracing is blended with SAO.
  - VXAO SDK has build-in SAO pass which is a subset of HBAO, and blended with the result of Cone Tracing.
  - In FFXV, it is also possible to blend VXAO with the Luminous Engine’s SAO.
- In FFXV, height field, HairWorks strands and foliage are not drawn in the Voxelization pass.
  - These are not likely to produce complex AOs, however those have high drawing costs.
  - These are omitted on the premise of using any of SSAO together.
- There is no special omission in Cone Tracing pass, but you can skip Cone Tracing itself or change the Cone Tracing parameters with stencil testing, if necessary.

Shadow Works


Frustum Traced Shadow - OFF

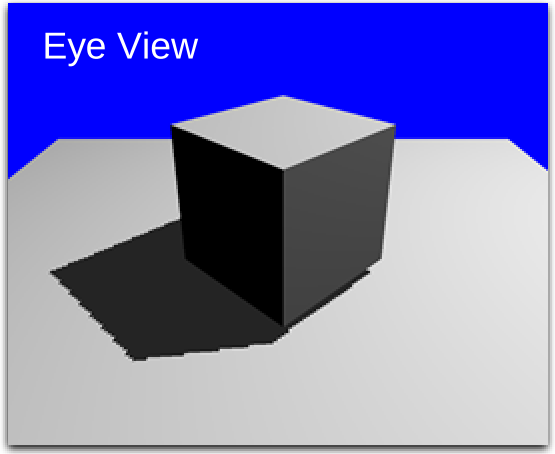

Frustum Traced Shadow - ON


Frustum Traced Shadow - OFF


Frustum Traced Shadow - ON

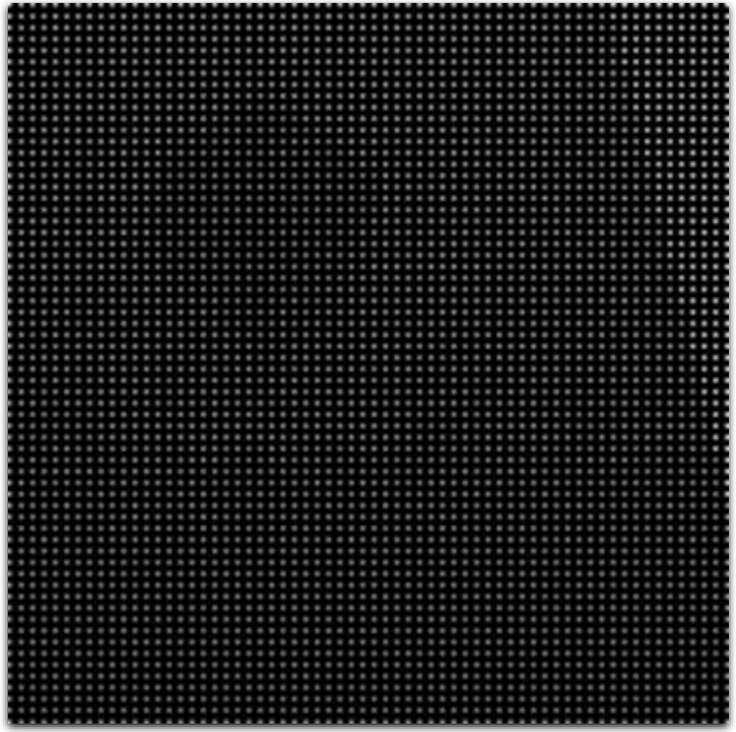
Frustum Traced Shadow

Irregular-z buffer

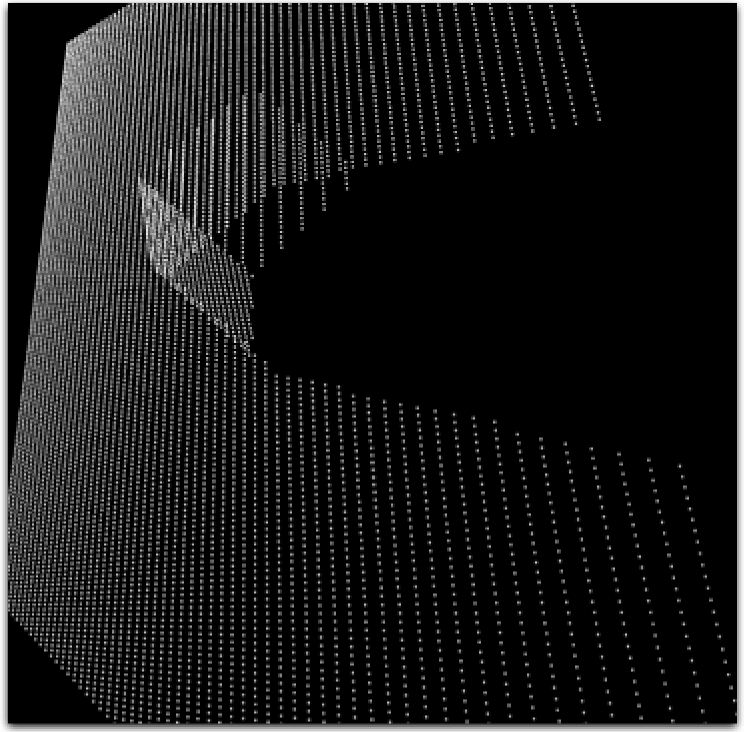


Light View
(ordinal shadow map)

Shadow map
Stores nearest depth in light space.
Format : Depth Texture



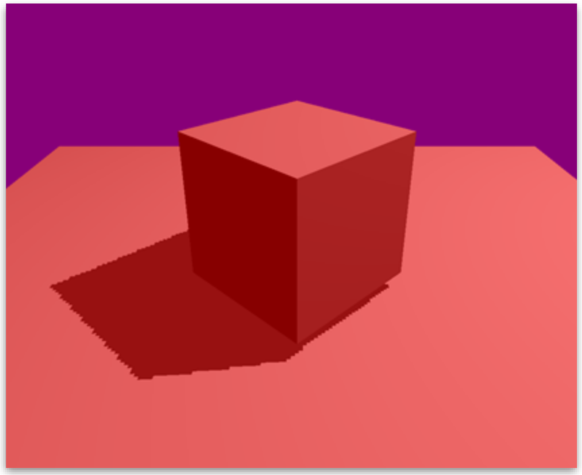
Irregular-z buffer
Stores screen space position in light space.
Format: Linked List

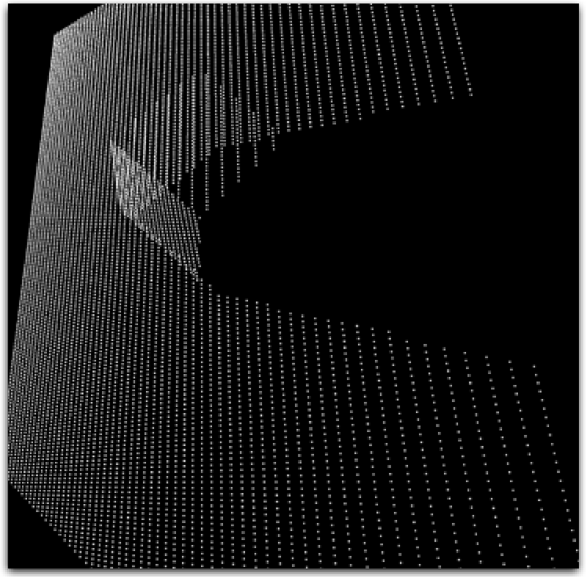


Frustum Traced Shadow

Self Shadowing

- Frustum traced shadow needs to store screen pixel positions into an Irregular-z buffer.

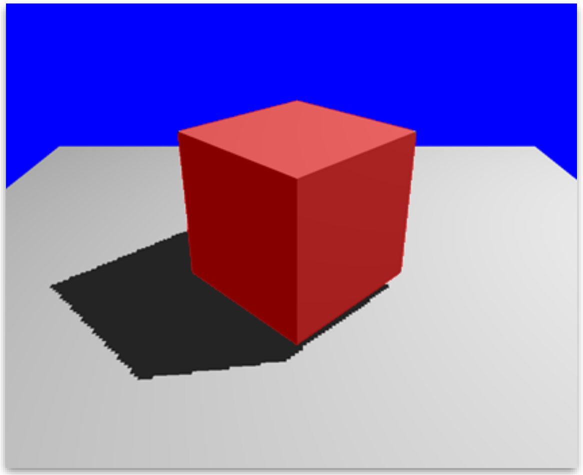


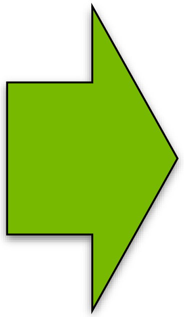


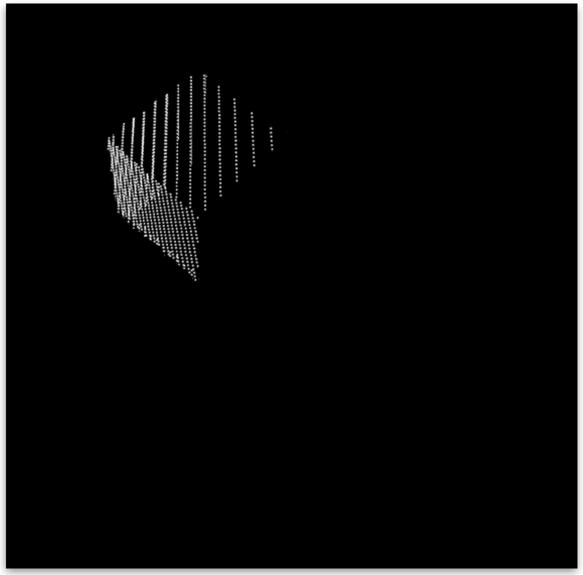
Frustum Traced Shadow

Self Shadowing

- Frustum traced shadow needs to store screen pixels positions into an Irregular z-buffer.
- If you think about self shadow of the cube, you only need to store screen space pixel positions rendered the cube into an irregular-z buffer.



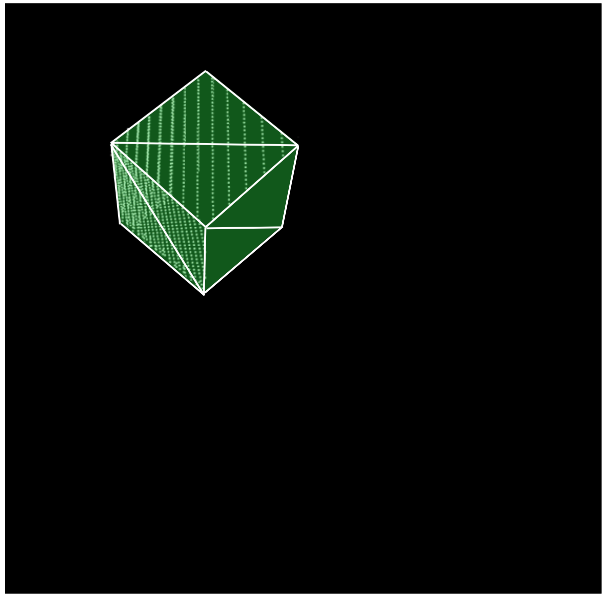




Frustum Traced Shadow

Self Shadowing

- Next, Frustum Trace Shadow needs to render shadow caster primitives in the light space, to test with screen space pixels stored in the the Irregular-z buffer.
- It only needs to render the cube, in case of self shadowing.



Shadow Works (Frustum Traced Shadow)

Integration notes

- Frustum Traced shadow was used for the player character’s self shadowing.
- Only pixels where the player character was rendered were stored in Irregular-Z buffer.
- Only the player character was rendered in Frustum Trace path.
- No filter was applied for the result, since blocker and receiver should be close.

Flow

Flow

Flow

Any Questions?

Special Thanks

- Lars Nordskog
  - General performance issues
- Bryan Dudash
  - HBAO+, Ansel, SPH,
    General Project Management
- Eveny Makarov
  - Turf Effect, Terrain Tessellation
- Jiho Choi
  - Hair Works
- Dane Johnston
  - General Project Management
- Johnny Castello
  - On-site Hair Works Training
- Spahyra Amaro
  - Geforce Game Producer
- Taich Hirayanagi
  - Developer relations
- Masaya Takeshige
  - HBAO+, VXAO, Flow
    Hair Works, Shadow Works,
    Perf and General Issues