The Aircraft of Theseus
Shipping X-Plane for 30 years

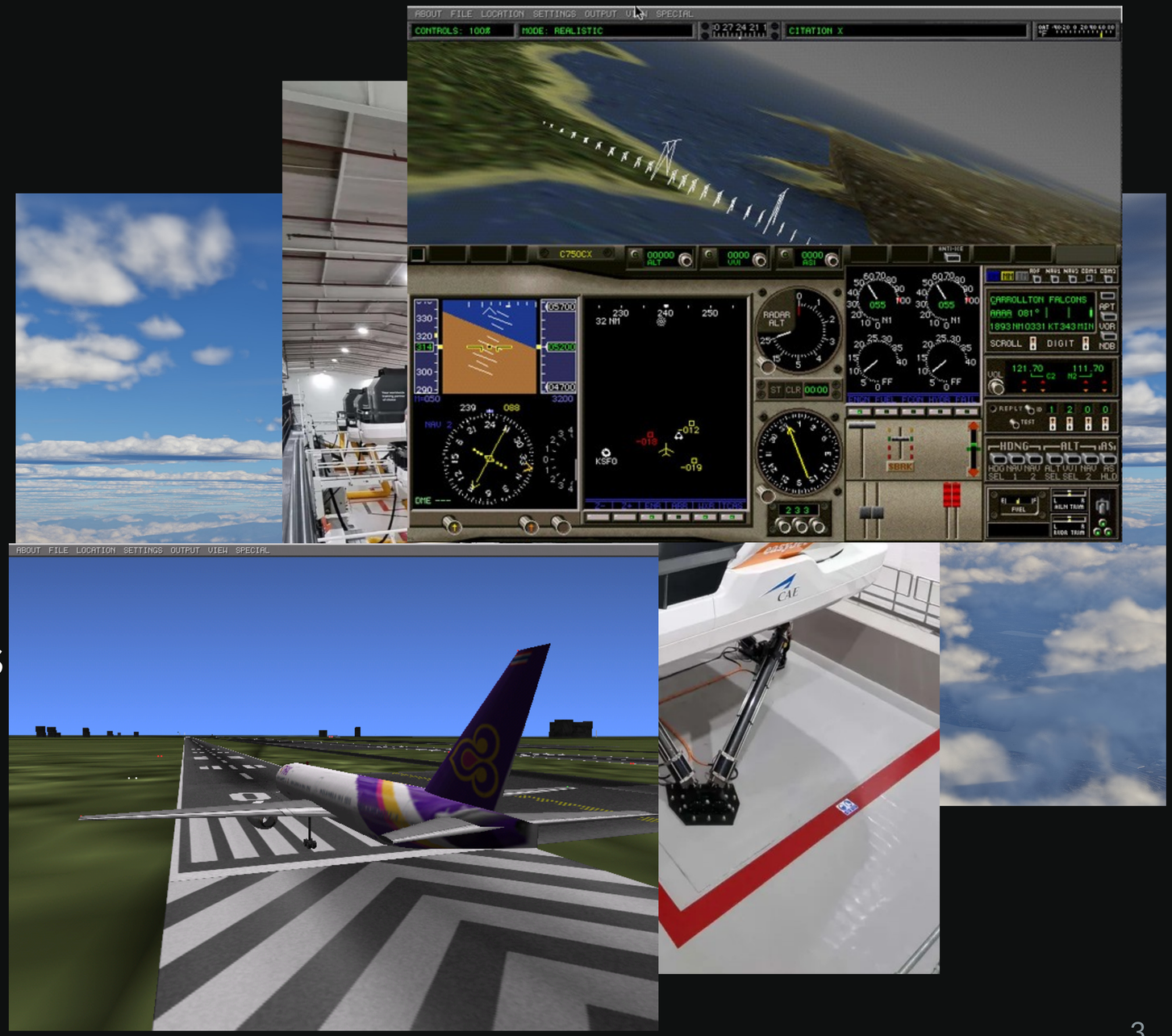
XPLANE Laminar Research

Introduction - Laminar Research

- Small, privately owned, company
  - Founded in 1995
- Fully remote
- Exclusively working on X-Plane

Introduction - X-Plane

- FAA approved flight simulator
- But not Level-D certified
- Continuously developed
- Current version is X-Plane 12
- Long release cycles
- Available on all major platforms

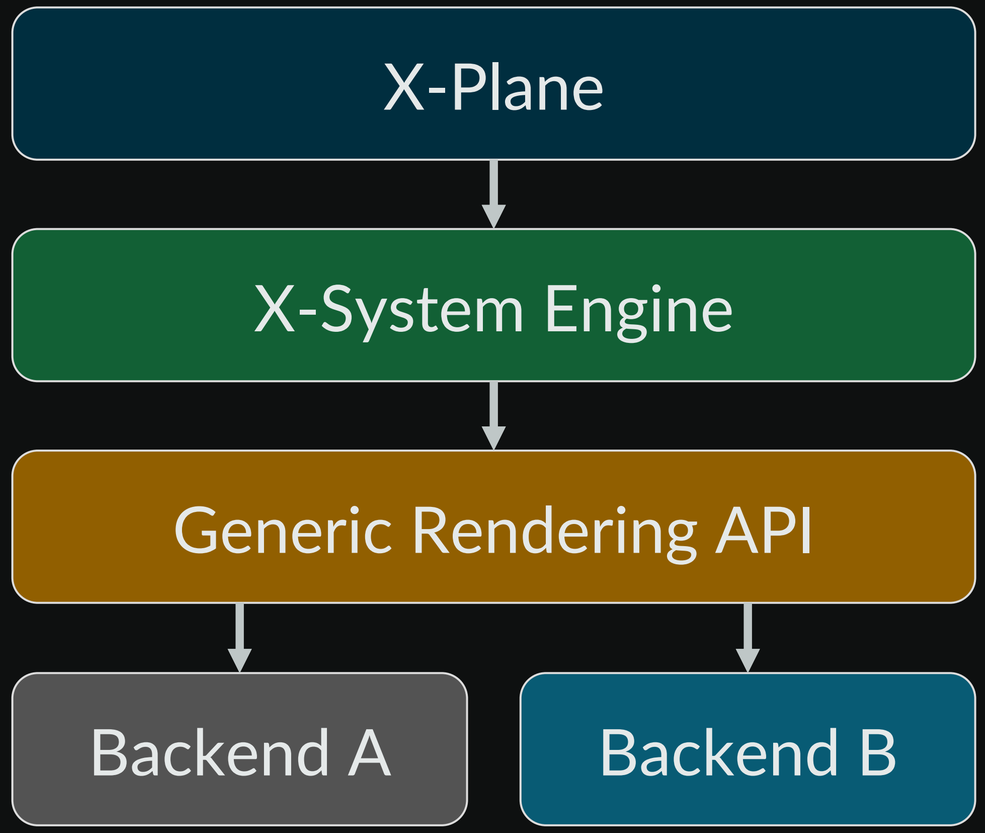


Overview

- Introduction
  - Who are Laminar Research and what is an X-Plane?
  - You are here 📍
- The dark ages of X-Plane 6
- Modernizing our engine in X-Plane 11
- Shipping X-Plane 12
- What’s next?

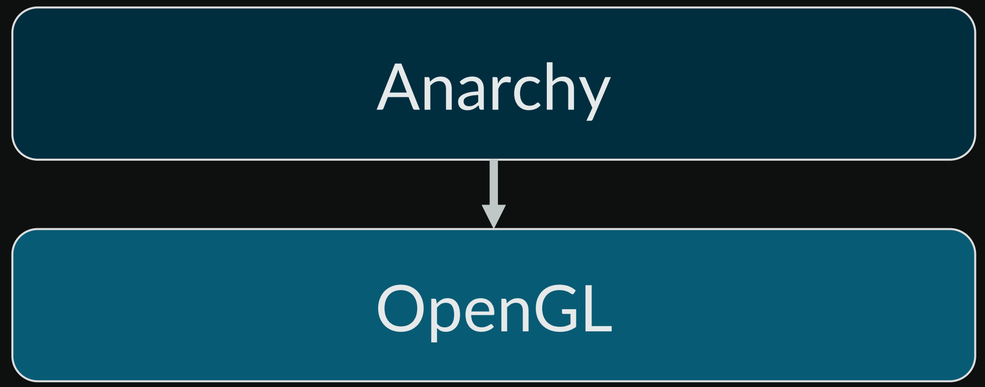
The tech stack

- Custom engine called X-System



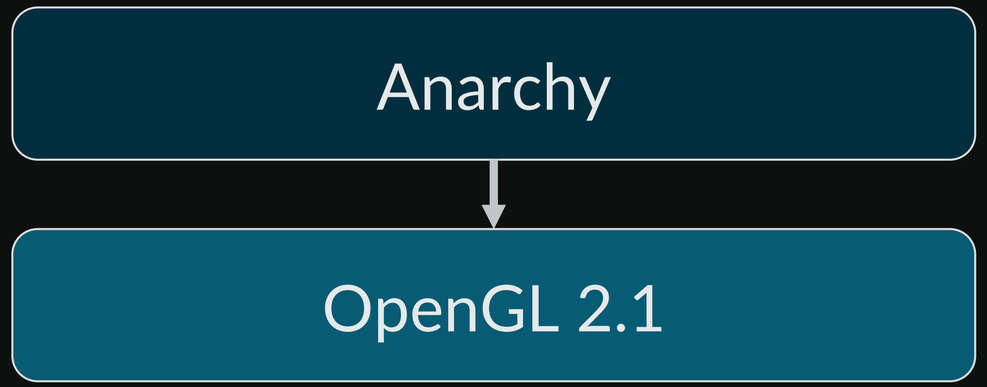
The tech stack

- ~~Custom engine called X-System~~
- Absolute anarchy called X-System
- One large monolithic codebase
- Some high level “engine” objects



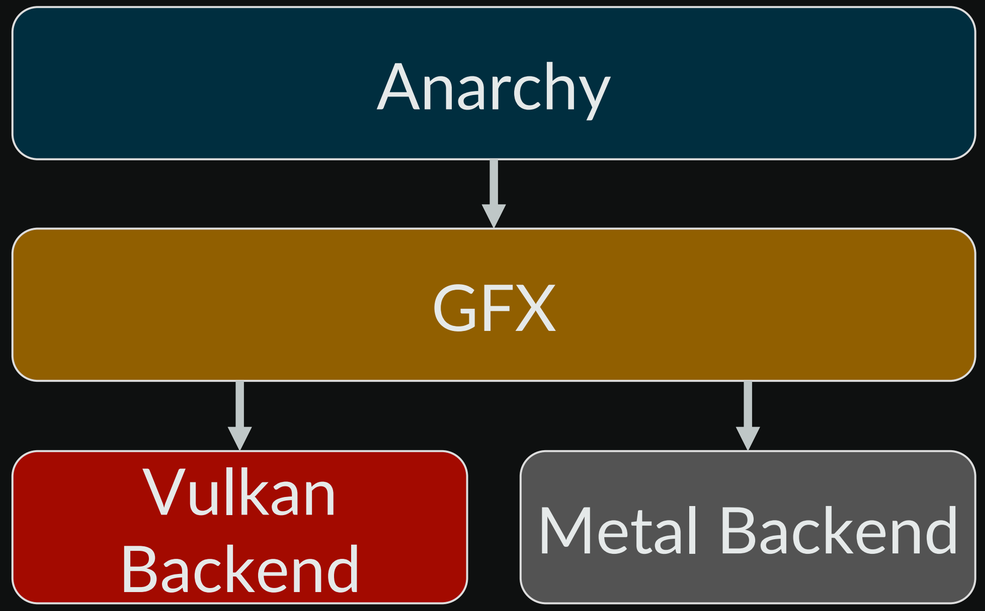
The tech stack

- ~~Custom engine called X-System~~
- Absolute anarchy called X-System
- One large monolithic codebase
- Some high level “engine” objects
- OpenGL 2.1
- No AZDO
- Just-in-time pipeline discovery



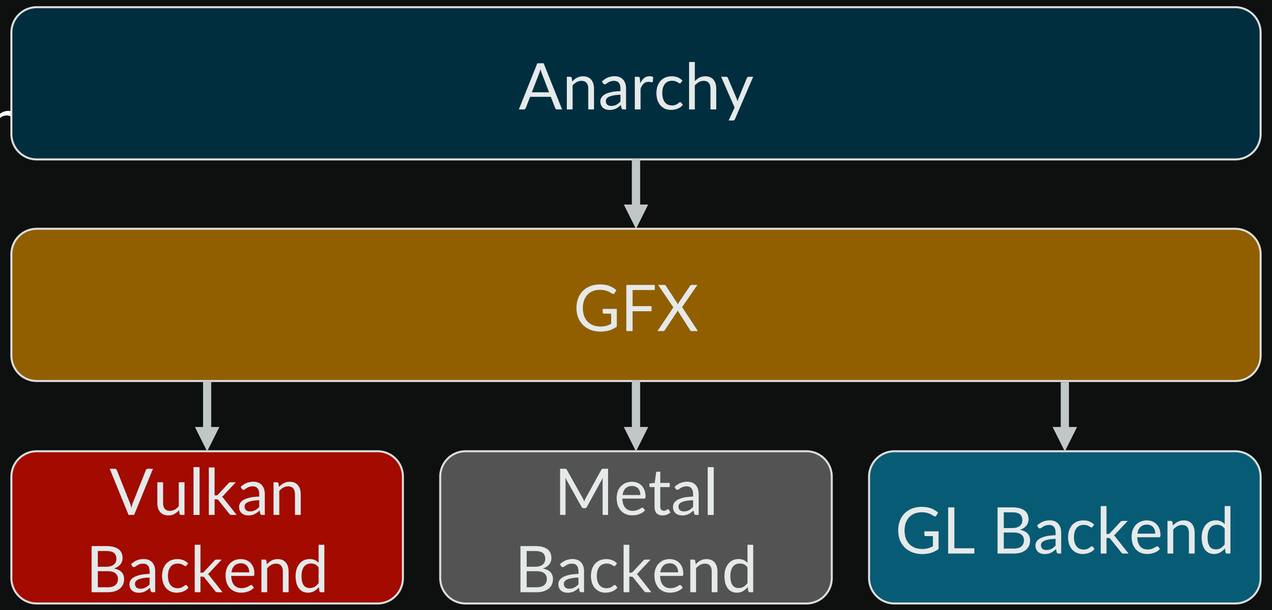
Modernization plan

- Adopt Vulkan and Metal
- No more just in time PSO compilation
- No VRAM oversubscription
- Ship spec conform code
- Don’t break plugins



Modernization plan

- Adopt Vulkan and Metal
- No more just in time PSO compilation
- No VRAM oversubscription
- Ship spec conform code
- Don’t break plugins
- Ship as feature update
- No new graphics



X-Plane as a platform

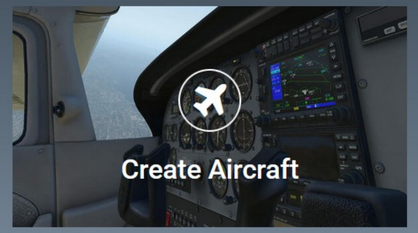
- Open platform
- SDK available to anyone
- Vast add-on ecosystem
  - Aircrafts
  - Scenery
  - Utilities

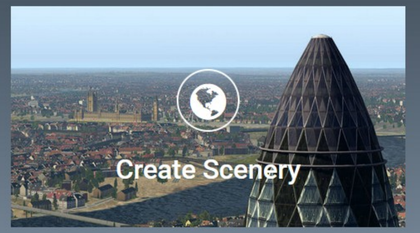
```
##############################################################################
#
#                    S T O P ! ! !
#
#   This is X-Plane 12's settings file.  This is NOT your preferences.  You
```

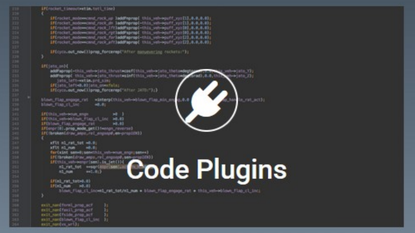
XPLANE DEVELOPER BLOG SCENERY TOOLS VIDEOS DOCUMENTATION MORE

Create aircraft, scenery, & more for X-Plane.

X-Plane is incredibly extensible. You can simulate new aircraft, make 3-D models, or even change the functionality of the program in drastic ways.








```
#
#   * Bad things include: X-Plane crashing, X-Plane hanging, X-Plane failing
#     to start, rendering artifacts, your whole computer hanging, your
#     computer catching fire, and highly trained monkeys stealing all of
#     your underwear.
#
#   You have been warned.
#
#
##############################################################################
```

The SDK

- SDK released with X-Plane 6
- Plugins are shared libraries
- Segmentation fault (core dumped)
- Callbacks for everything
- L2 cache misses for everyone

Configuring X-Plane to Use Less Virtual Memory

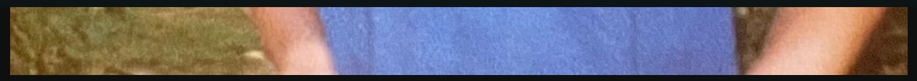
X-Plane 10 can run out of memory when the rendering settings are set too high or too many add-ons are running at once. This article explains this problem and makes some recommendations about what to do about it.

X-Plane Runs Out of Virtual Memory

Because X-Plane is a 32-bit application, it can only use a certain amount of memory. That limit varies by operating system:

- Windows 32-bit: 3 GB.
- Windows 64-bit: 4 GB.
- Mac OS X: 3.5 GB.
- Linux: 3 GB.

Because this is a virtual memory limit on each application, these limits apply no matter how much physical memory you have in your computer; even if you have 16 GB of RAM, these limits apply. There are no operating system settings that raise these limits beyond what is listed.



The SDK - Drawing

- No SDK drawing API
- Plugins use OpenGL directly
- Third-parties love the ffp
- macOS only provides two OpenGL versions
  - OpenGL 2.1
  - 

ile

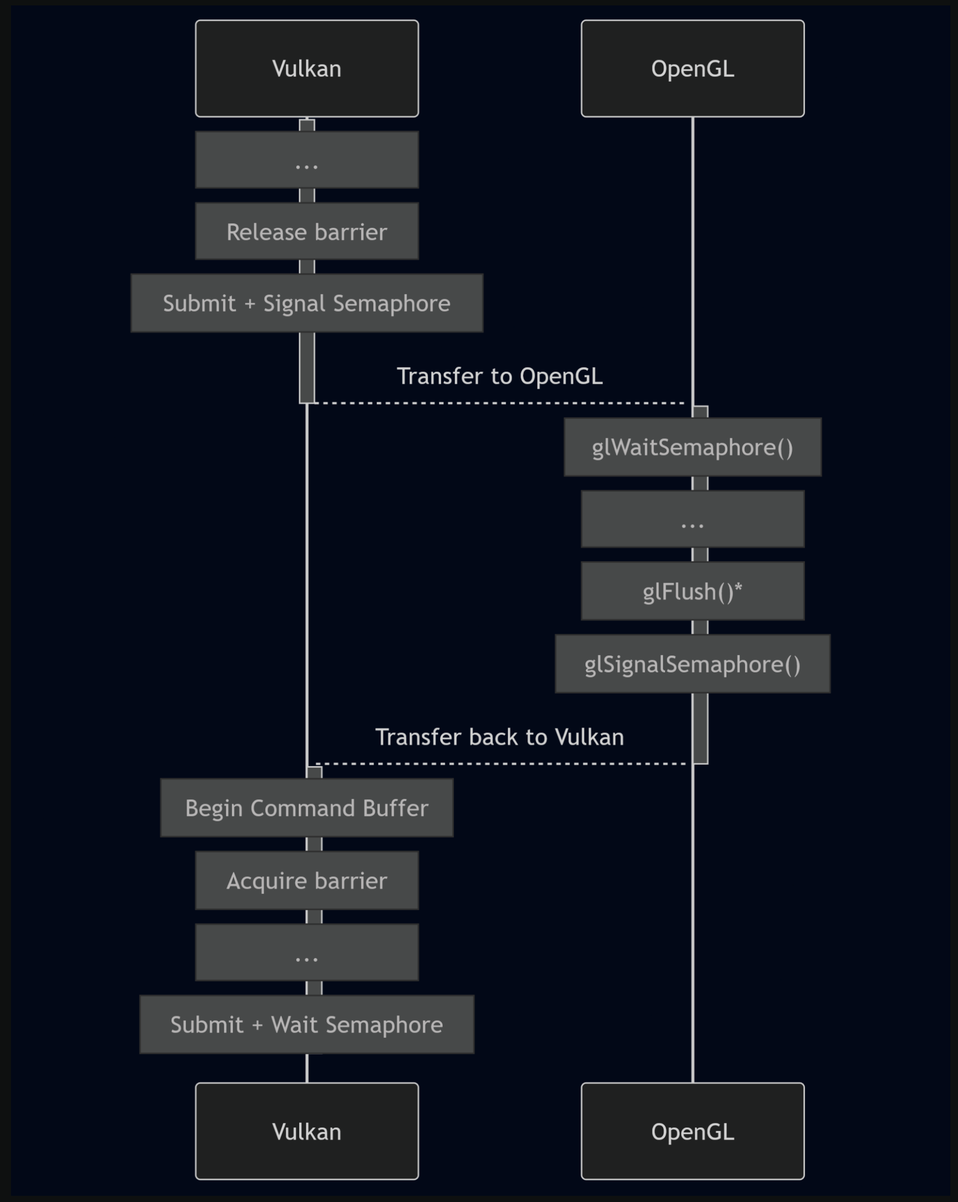
XPLANE Laminar Research OpenGL

Plugin compatibility

- Plugin bridge for compatibility
- Let them eat ~~cake~~ OpenGL 2.1
- Uses native OpenGL interop
- No plugin changes required

Vulkan bridge

- Interop uses a zoo of extensions
- Built around shared memory regions
- Bind OpenGL texture to memory
- Bind Vulkan image to memory
- Shared semaphores for synchronization



Metal bridge

- IOSurface as shared memory
- Imported into Metal and OpenGL from Mach object
- Restrictions
  - Linear tiling only
  - No mip maps
  - GL only: Texture rectangle
- Shadow textures as workaround

Bridges in general

- Lots of individual command buffers
- Easy application API
  - 

```
texture = gfx_texture_builder()
    .with_texture2d(gfx_render_surface_format::bridge_rgba, width_screen, height_screen)
    .with_usage( gfx_texture_usage::shared | gfx_texture_usage::render_target | gfx_texture_usage::sampler)
    .with_min_filter(gfx_sampler_mip_mode::trilinear)
    .with_mag_filter(gfx_sampler_mip_mode::bilinear)
    .with_name( "floating gauge") // gfx_texture_builder &
    .build(g_device, provider); // gfx_texture_handle
```

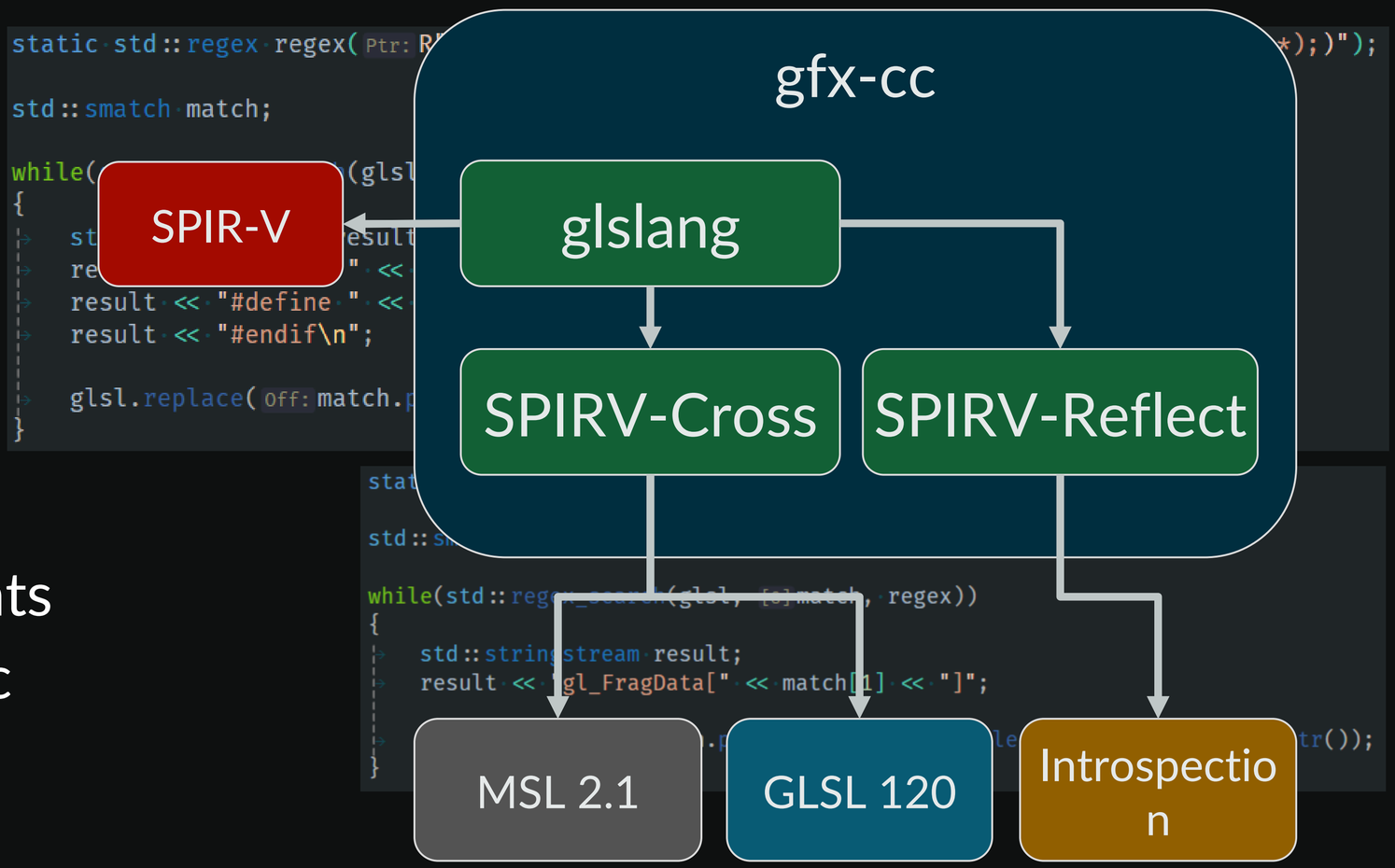
  -

```
// Transfer to the bridge only when necessary. It's expensive!
if(XPPWantsDrawingHook(inDrawingHookKind))
{
    command_buffer->transfer_to_bridge( objects: {}, consumed_objects: XPPGetConsumableFramebuffers());

    XPPDoModernDrawingHook(inDrawingHookKind); // Dispatch plugins

    command_buffer->transfer_from_bridge();
}
```

The bridge

Shaders

- Originally GLSL 120 text files
- gfx needs
  - SPIR-V
  - MSL 2.1
  - GLSL 120
- Early SPIR-V Cross experiments
- Offline shader compiler gfx-cc



Shaders

- Custom annotation for resources
- Bindings are automatically resolved
- JSON shader descriptions

```
{
    "name": "planet",
    "vertex": "planet.glsl",
    "fragment": "planet.glsl",
    "defines": [
        "RENDERING_MODE", ["GBUFFER","FORWARD"]
    ]
},
```

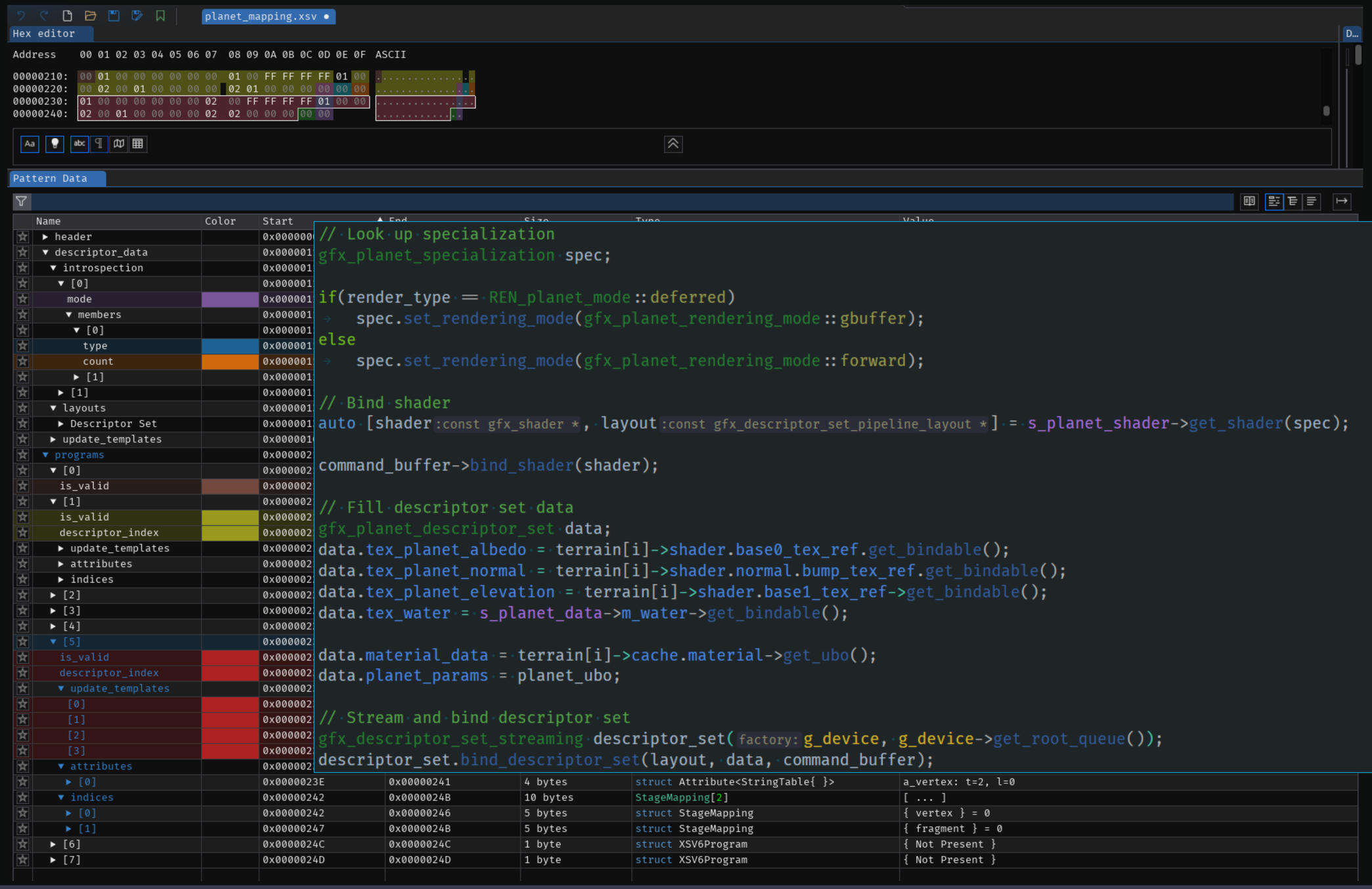
```
#sampler2D tex_planet_albedo
#sampler2D tex_planet_normal
#sampler2D tex_planet_elevation
#sampler2D tex_water

#varying vec2 v_texcoord
#varying vec3 v_position_object
#varying vec3 v_normal_eye
#varying vec3 v_position_eye

#constantBuffer u_planet_params
{
    mat4 u_eye_to_wrl;

    vec2 u_planet_clip;

    // ...
}

#if RENDERING_MODE == RENDERING_MODE_FORWARD
void output_normal(
                in vec3     final_color,
                in float    planet_rat
                )
{
    // ...
}
#elif RENDERING_MODE == RENDERING_MODE_GBUFFER
void output_gbuffer(
                in vec4     tex_color,
                in vec3     material,
                in vec3     normal_eye)
{
    // ...
}
#endif
```

Introspection

- Generated C++ introspection

```
struct alignas(16) planet_params_t
{
    planet_params_t() = default;
    planet_params_t(const planet_params_t &) = default;
    planet_params_t &operator =(const planet_params_t &other)
    {
        memcpy(this, &other, sizeof(decltype(other)));
        return *this;
    }
    volatile planet_params_t &operator =(const planet_params_t &other)
    {
        volatile_memcpy(this, &other, sizeof(decltype(other)));
        return *this;
    }
    ubo_mat4 u_eye_to_wrl;
    ubo_vec2 u_planet_clip;
    ubo_int u_planet_do_scattering;
    ubo_int u_planet_dither_mode;
    ubo_vec4 u_planet_scatter_frustum_inv;
    ubo_vec4 u_planet_scatter_ap_near_far_z;
    ubo_vec3 u_planet_scatter_resolution_inv;
};
```

```
enum class gfx_planet_rendering_mode : int32_t
{
    gbuffer,
    forward,
};

constexpr gfx_planet_rendering_mode gfx_planet_rendering_mode_range[] = {
    gfx_planet_rendering_mode::gbuffer,
    gfx_planet_rendering_mode::forward,
};

struct gfx_planet_specialization : public gfx_shader_specialization_info
{
    void set_rendering_mode(gfx_planet_rendering_mode rendering_mode)
    {
        specializations[0] = (int32_t)rendering_mode;
    }

    // ...
};
```




```
// Look up specialization
gfx_planet_specialization spec;

if(render_type == REN_planet_mode::deferred)
    spec.set_rendering_mode(gfx_planet_rendering_mode::gbuffer);
else
    spec.set_rendering_mode(gfx_planet_rendering_mode::forward);

// Bind shader
auto [shader :const gfx_shader *, layout :const gfx_descriptor_set_pipeline_layout *] = s_planet_shader->get_shader(spec);

command_buffer->bind_shader(shader);

// Fill descriptor set data
gfx_planet_descriptor_set data;
data.tex_planet_albedo = terrain[i]->shader.base0_tex_ref.get_bindable();
data.tex_planet_normal = terrain[i]->shader.normal.bump_tex_ref.get_bindable();
data.tex_planet_elevation = terrain[i]->shader.base1_tex_ref->get_bindable();
data.tex_water = s_planet_data->m_water->get_bindable();

data.material_data = terrain[i]->cache.material->get_ubo();
data.planet_params = planet_ubo;

// Stream and bind descriptor set
gfx_descriptor_set_streaming descriptor_set( factory: g_device, g_device->get_root_queue());
descriptor_set.bind_descriptor_set(layout, data, command_buffer);
```

Pipelines

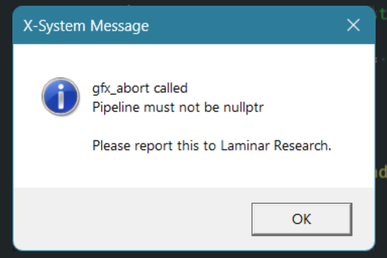
- No just-in-time PSO compilation
  - Without a PSO cache
- Shader creation requires capability tagging

```
enum class REN_shader_caps : uint32_t {

    none = 0,
    exterior = 1,       // Shader can be used for acf exterior and world
    interior = 2,       // Shader can be used for interior aircraft rendering
    instancing = 4,     // Shader can be used for an instanced object
    post_gbuffer = 8,   // Shader can be used after a Z buffer pass
    blend_glass = 16,   // Shader can be used in a blended-glass context
    prefill = 32,       // Shader can be used to prefill
    acf_attached = 64,  // OBJ can legally be ACF-attached
    snorm_vertex = 128, // OBJ has squished vertices
    all = 255,

};
```

```
OGL_terrain_shader shader;

shader.tex_mode = mode_texture;
shader.base0_tex_ref = s_planet_data->m_planet_day[i];
shader.base1_tex_ref = s_planet_data->m_planet_ele_synthetic;
shader.night_mode = mode_night_none;

shader.two_sided = xfals;
shader.geom_mode = mode_geom_normal;
shader.border_mode = mode_border_none;
shader.blend_mode = mode_blend_none;
shader.alpha.alpha_mode = mode_alpha_none;
shader.normal.normal_mode = mode_normal_displace;

s_planet_data->m_lo_res_terrain[i] = REN_shader_cache_build(shader, REN_shader_caps::exterior, debug_name: "planet", load_group);
```

```
static void REN_shader_cache_build_pipelines_for_mode_and_specific_caps(const void * why, const OGL_terrain_shader &shader, OGL_rendering_mode mode, REN_shader_caps caps, const gfx_vector<gfx_render_pass *>& rps, UTL_continuation_group *group)
{
    auto specializations :vector<gfx_shader_specialization_info> = OGL_terrain_shader_specializations(why, shader, mode,
                post_gbuf_ok: (caps & REN_shader_caps::post_gbuffer)==REN_shader_caps::post_gbuffer,
                (caps & REN_shader_caps::acf_attached)==REN_shader_caps::acf_attached);


    //printf("%p for mode %d got %zd specializations.\n", why, (xint) mode, specializations.size());

    if(specializations.empty())
        return;

#if DESKTOP
    gfx_graphics_pipeline_allowed_state allowed;
    allowed.add_color_mask(gfx_color_mask::rgba);
    allowed.add_color_mask(gfx_color_mask::none);
    allowed.add_alpha_mask(gfx_alpha_mask::none);

#if SIM
    if(mode == mode_rendering_gbuffer && REN_want_alpha_to_one())
        allowed.add_alpha_mask(gfx_alpha_mask::alpha_to_coverage);
#endif  #if SIM

    allowed.add_polygon_mode(gfx_polygon_mode::fill);

#if PLN
    allowed.add_polygon_mode(gfx_polygon_mode::line);
#endif  #if PLN

    allowed.add_stencil_op(gfx_stencil_op::none);
    allowed.add_stencil_op(gfx_stencil_op::write_zpass);
    allowed.add_stencil_op(gfx_stencil_op::test_ref_is_eq);

    if(mode == mode_rendering_depth || shader.two_sided)
        allowed.add_cull_face(gfx_cull_face::none);
    else
    {
        allowed.add_cull_face(gfx_cull_face::back);
    }
    if(g_icon_hack)
        allowed.add_cull_face(gfx_cull_face::front);

    if(shader.tex_mode == mode_line || shader.base_pt == pt_powr)
        allowed.add_primitive_type(gfx_primitive_type::lines);
    else
    {
        allowed.add_primitive_type(gfx_primitive_type::triangles);
#if PLN
        // Needed because Plane Maker wants to render with triangle strips for the default "None" object
        // There is no other case where this happens
        allowed.add_primitive_type(gfx_primitive_type::triangle_strip);
#endif  #if PLN
    }

// if(shader.blend_mode == mode_blend_glow)
//     allowed.add_depth_mode(gfx_depth_mode::none);
// else
    if(shader.poly_offset > 0)
        allowed.add_depth_mode(gfx_depth_mode::test);
    else
        allowed.add_depth_mode(gfx_depth_mode::test_and_write);

    if(mode == mode_rendering_depth)
    {
        // Depth mode - ALWAYS off - keep it simple.
        allowed.add_blend_mode(gfx_blend_mode::none);
    }
    // All others get blending - fade-with-LOD means blender is pretty
    // much *always* on.
```

```
const bool is_gbuffer = (pass.color_formats[1] != gfx_render_surface_format::none);

if(!gfx_format_is_stencil(dformat) && desc.stencil_op != gfx_stencil_op::none)
    return false;

if(mode == mode_rendering_depth && desc.stencil_op == gfx_stencil_op::write_zpass && !is_prefill)
    return false;

if(mode == mode_rendering_depth && desc.depth_mode != gfx_depth_mode::test_and_write)
    return false;

// WTF we need htis?
if(mode == mode_rendering_exterior && is_not_obj && desc.stencil_op == gfx_stencil_op::test_ref_is_eq)

if(mode != mode_rendering_depth || format != gfx_render_surface_format::none)
{
                                                    tencil_op::none)

                                                    mode_blend_glass)

                                                    dering_interior && mode != mode_rendering_exterior)
```



Memory

- Vulkan uses VMA
- VK_EXT_memory_budget to monitor memory usage
- Large number of runtime allocations
- VMA pools based on resource lifetimes
- Active VRAM defragmentation

Memory – I can haz sparse?

- Tried to use sparse memory to reduce fragmentation
- Way too slow for production

The Sad State of Hardware Virtual Textures July 1, 2025

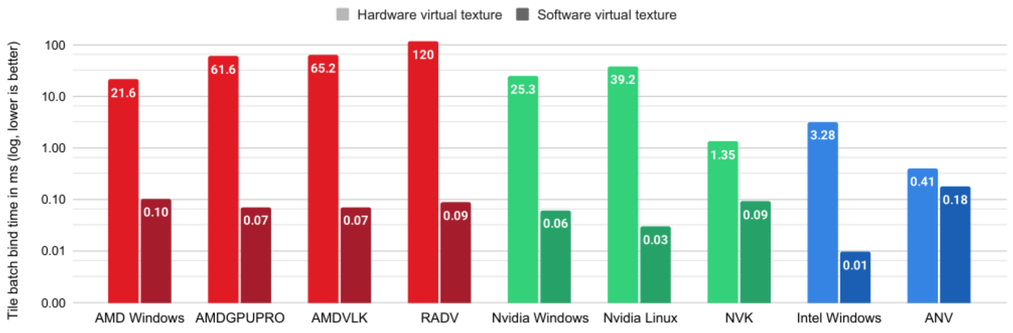
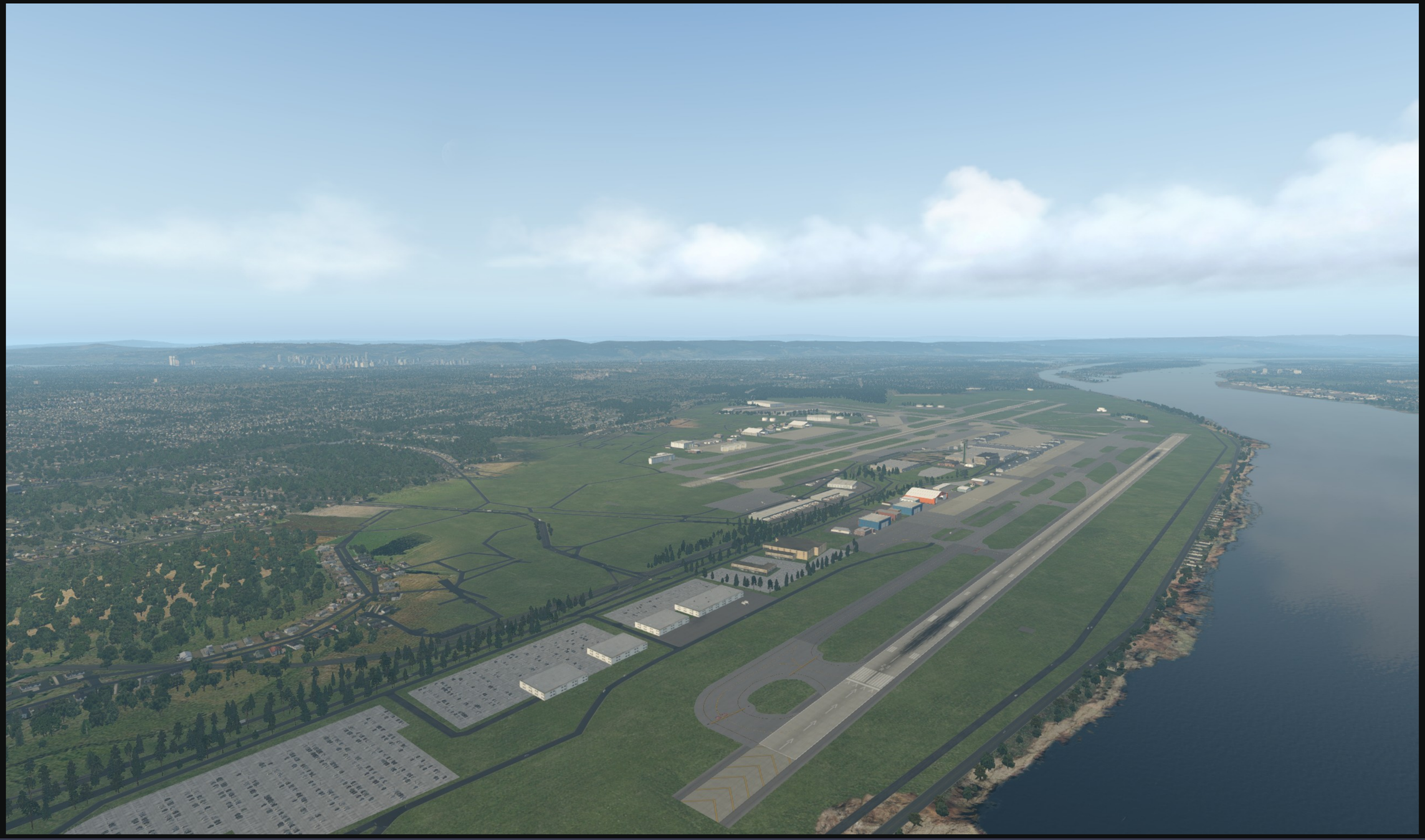


Figure 6. Tile binding performance of hardware virtual and software virtual textures under different GPU drivers. (In millisecond, lower is better, logarithmic scale)

Modernization – Post mortem

- If it’s stupid and it works, it’s not stupid
- Graphics team grew to 3 people
- 2 years of work from start to finish
- 9 months of beta period
- Complaint: Still looked the same

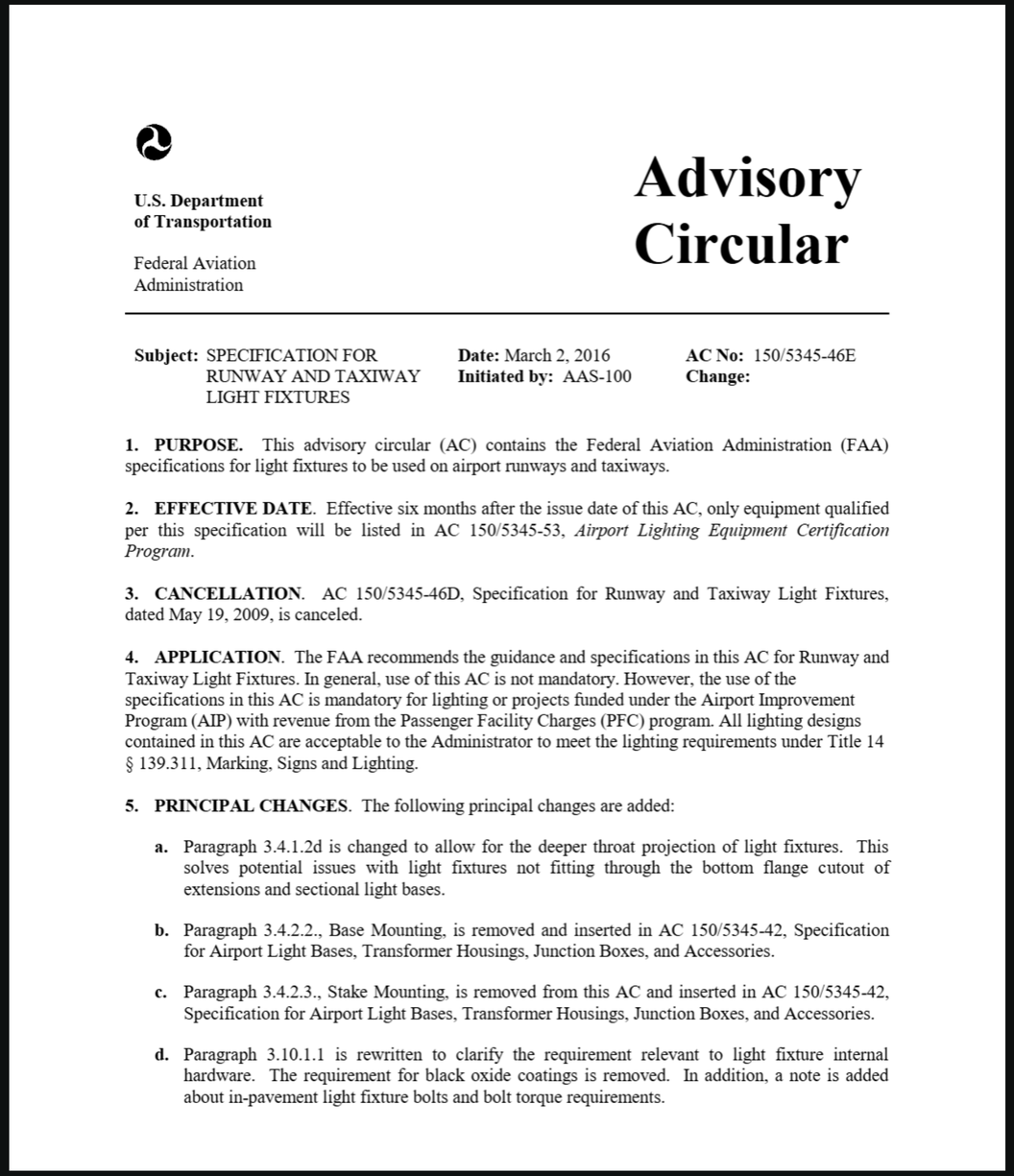


Building X-Plane 12

- No more OpenGL backend
- Focus on graphical improvements

Photometric rendering

- Radiometric values for all lighting
- Aviation industry is spec heaven



U.S. Department of Transportation

Federal Aviation Administration

Advisory Circular

Subject: SPECIFICATION FOR RUNWAY AND TAXIWAY LIGHT FIXTURES

Date: March 2, 2016 **Initiated by:** AAS-100

AC No: 150/5345-46E **Change:**

1. PURPOSE. This advisory circular (AC) contains the Federal Aviation Administration (FAA) specifications for light fixtures to be used on airport runways and taxiways.

2. EFFECTIVE DATE. Effective six months after the issue date of this AC, only equipment qualified per this specification will be listed in AC 150/5345-53, *Airport Lighting Equipment Certification Program*.

3. CANCELLATION. AC 150/5345-46D, Specification for Runway and Taxiway Light Fixtures, dated May 19, 2009, is canceled.

4. APPLICATION. The FAA recommends the guidance and specifications in this AC for Runway and Taxiway Light Fixtures. In general, use of this AC is not mandatory. However, the use of the specifications in this AC is mandatory for lighting or projects funded under the Airport Improvement Program (AIP) with revenue from the Passenger Facility Charges (PFC) program. All lighting designs contained in this AC are acceptable to the Administrator to meet the lighting requirements under Title 14 § 139.311, Marking, Signs and Lighting.

5. PRINCIPAL CHANGES. The following principal changes are added:

a. Paragraph 3.4.1.2d is changed to allow for the deeper throat projection of light fixtures. This solves potential issues with light fixtures not fitting through the bottom flange cutout of extensions and sectional light bases.

b. Paragraph 3.4.2.2., Base Mounting, is removed and inserted in AC 150/5345-42, Specification for Airport Light Bases, Transformer Housings, Junction Boxes, and Accessories.

c. Paragraph 3.4.2.3., Stake Mounting, is removed from this AC and inserted in AC 150/5345-42, Specification for Airport Light Bases, Transformer Housings, Junction Boxes, and Accessories.

d. Paragraph 3.10.1.1 is rewritten to clarify the requirement relevant to light fixture internal hardware. The requirement for black oxide coatings is removed. In addition, a note is added about in-pavement light fixture bolts and bolt torque requirements.

Photometric rendering

- Radiometric values for all lighting
- Aviation industry is spec heaven
- AgX tonemapper

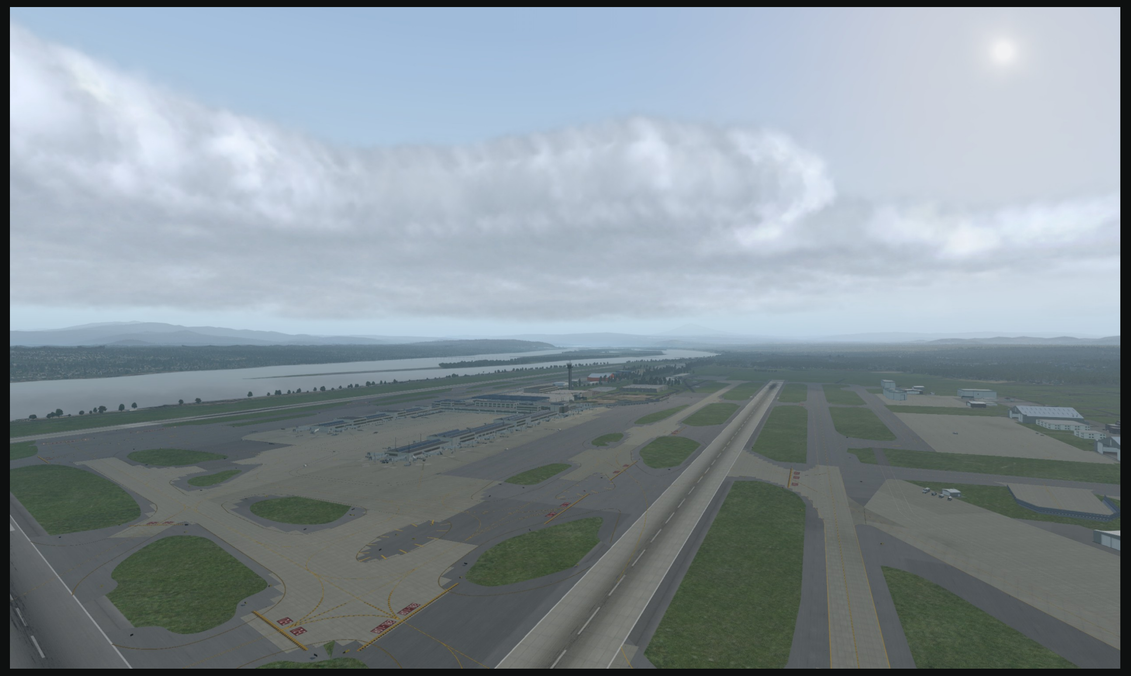
Photometric rendering

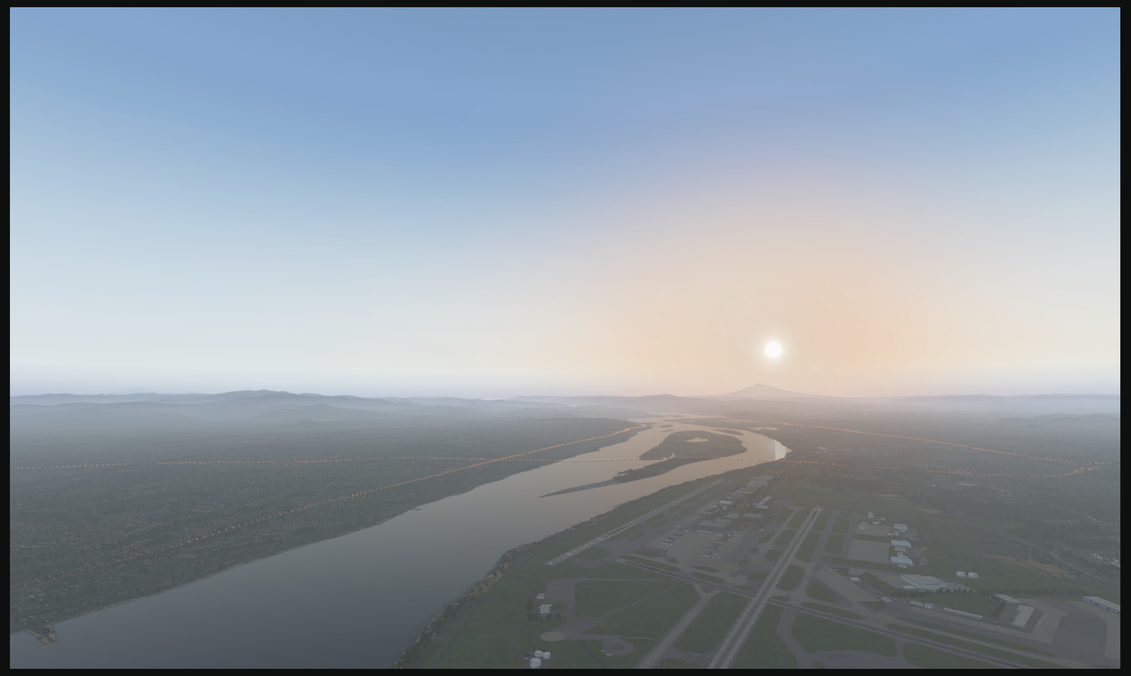
- Radiometric values for all lighting
- Aviation industry is spec heaven
- AgX tonemapper
- Exposure fusion added in an update

Home cockpits

Clouds and the atmosphere

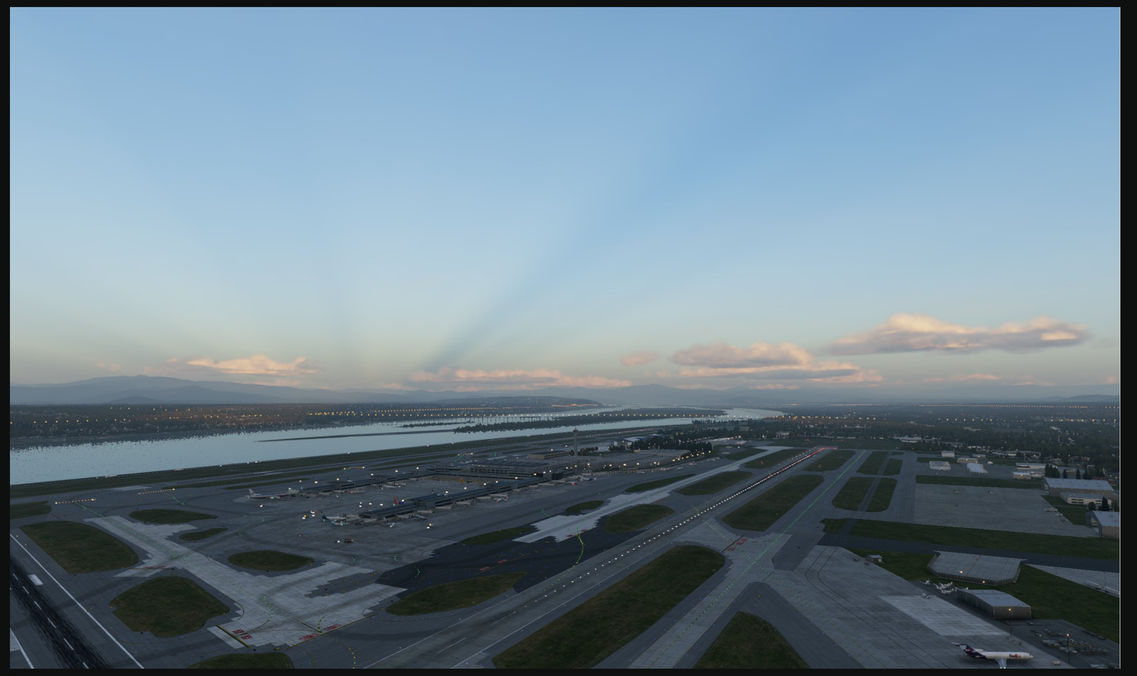
- X-Plane 11 used billboard clouds
- No smooth weather updates
- Cloud plugins tried to fix some of this
- We can do better



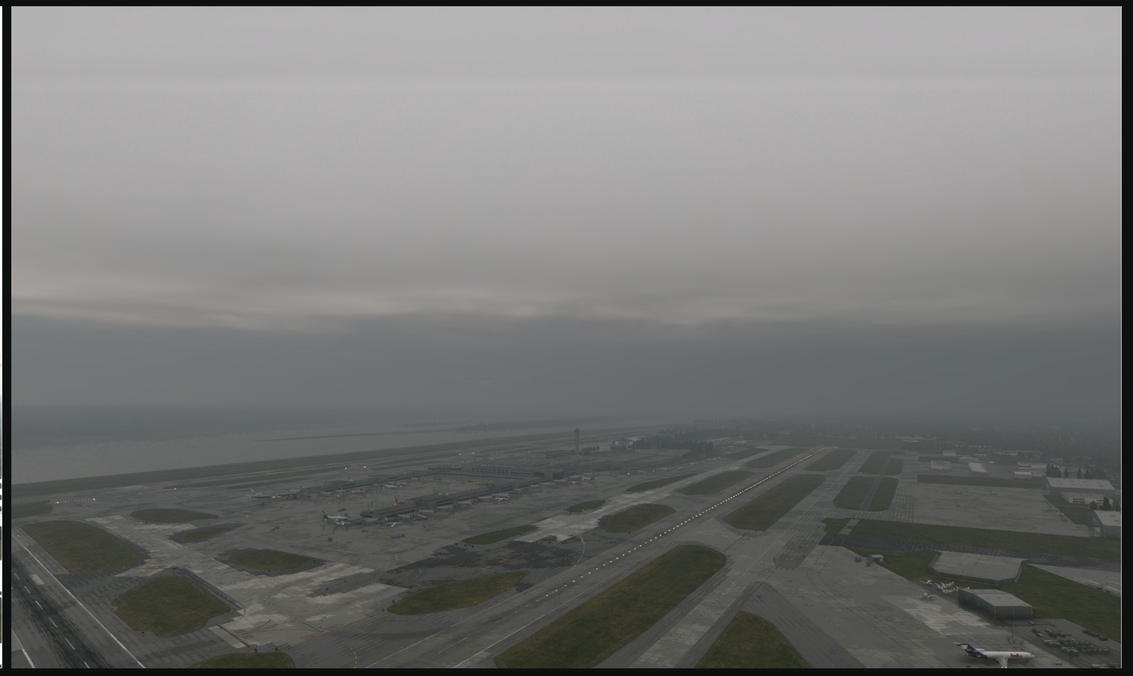


Atmospheric rendering

- Based on Hillaire’s work
- Sampled across full visible spectrum
- Improves sunset and sunrise

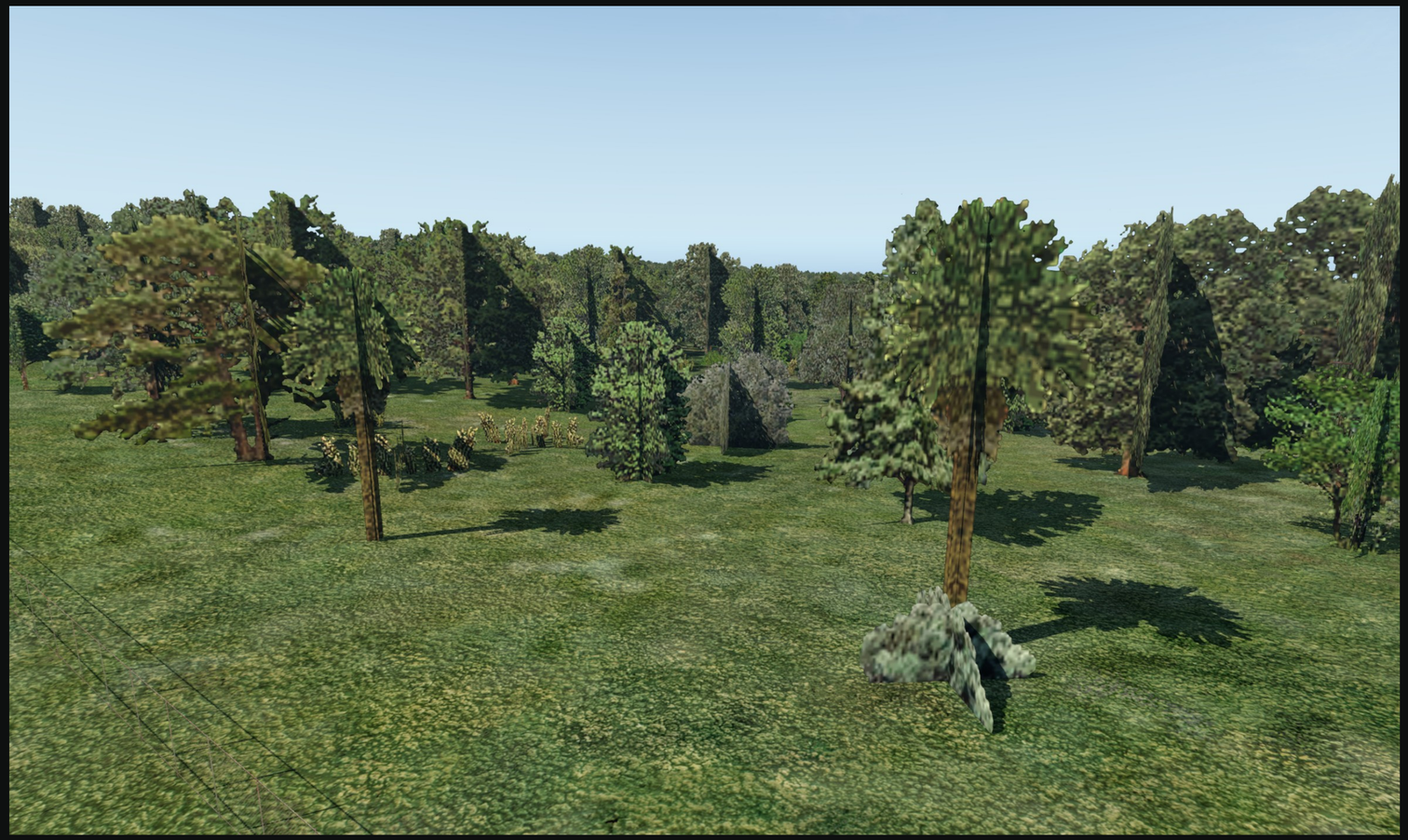


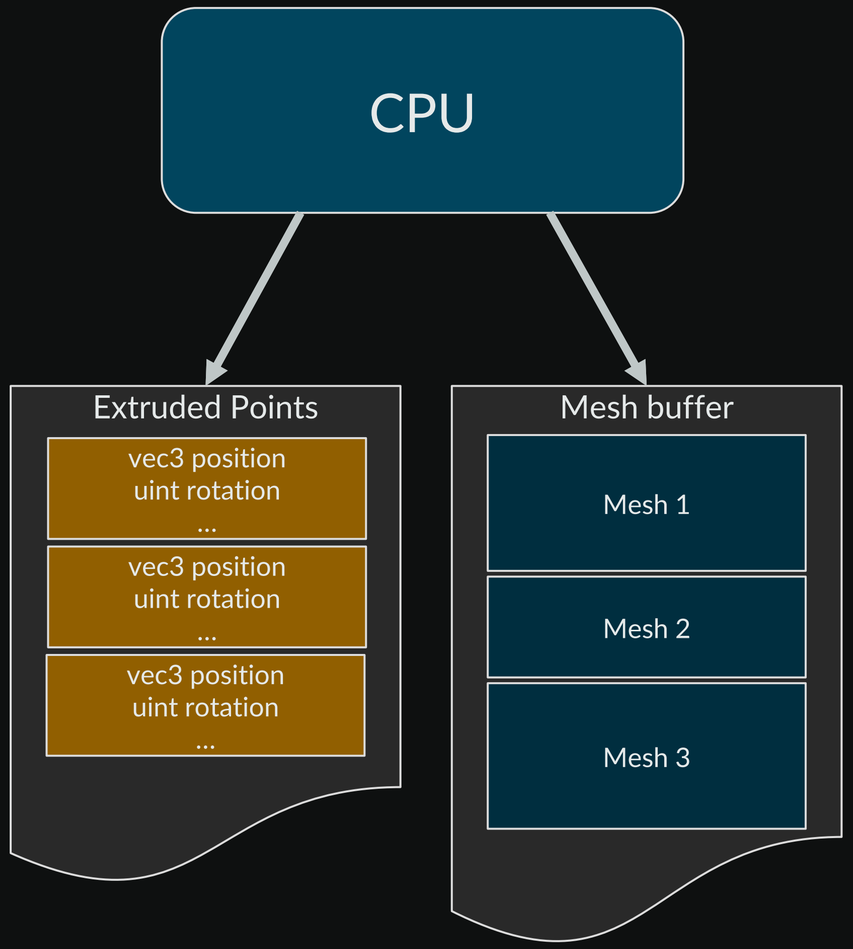
Volumetric clouds

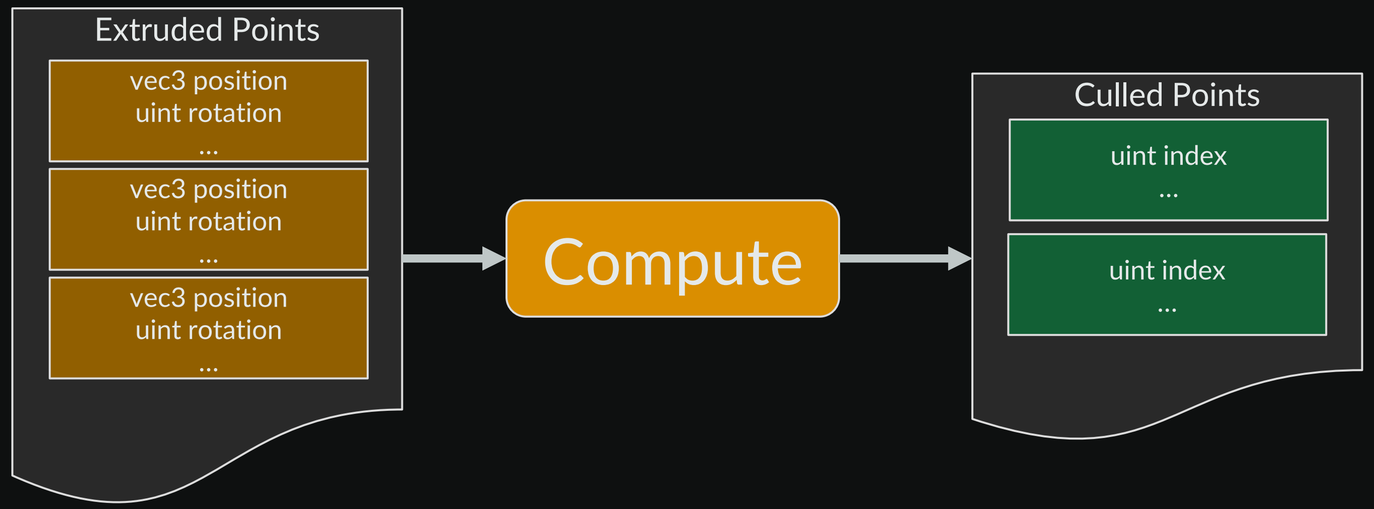


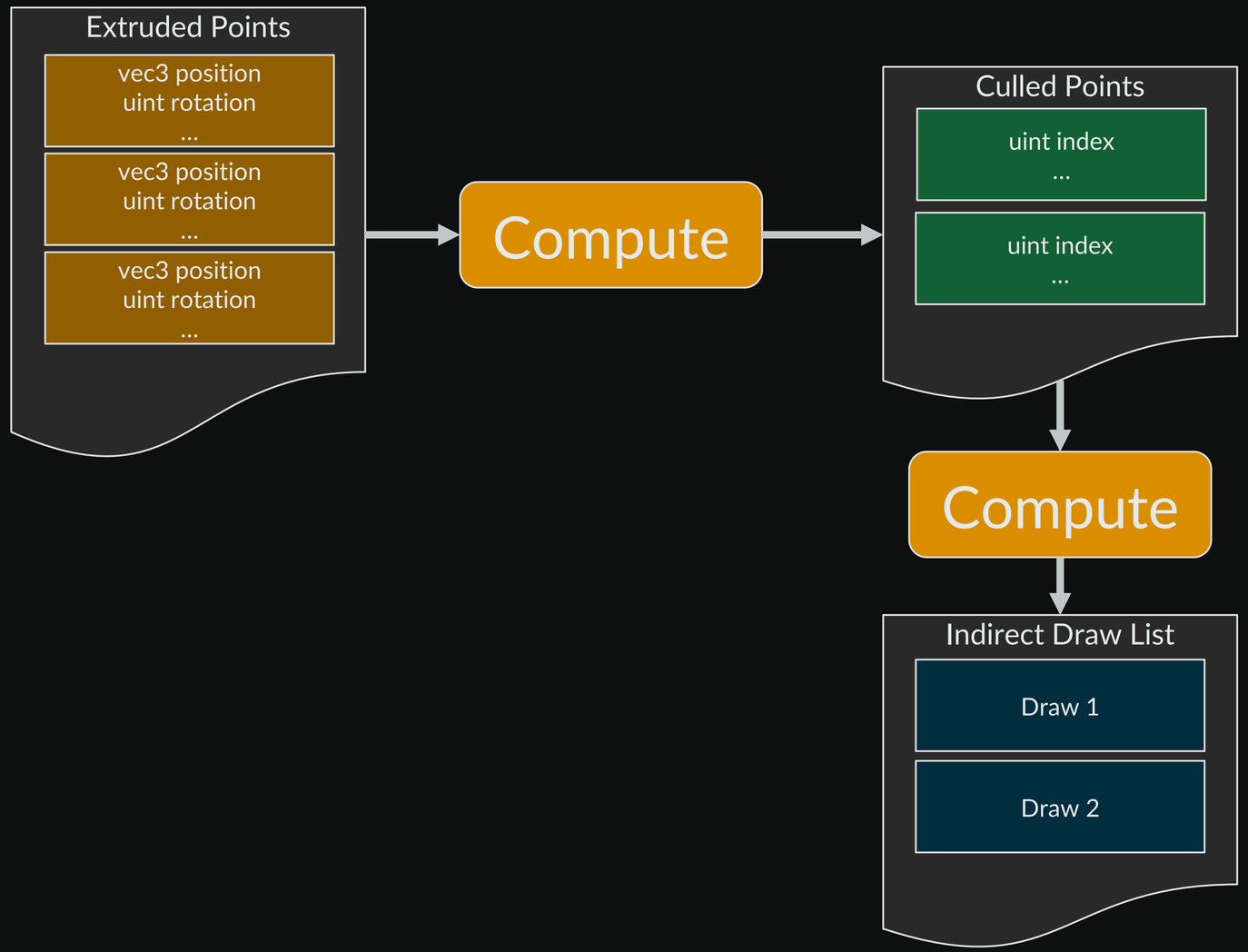
Vegetation

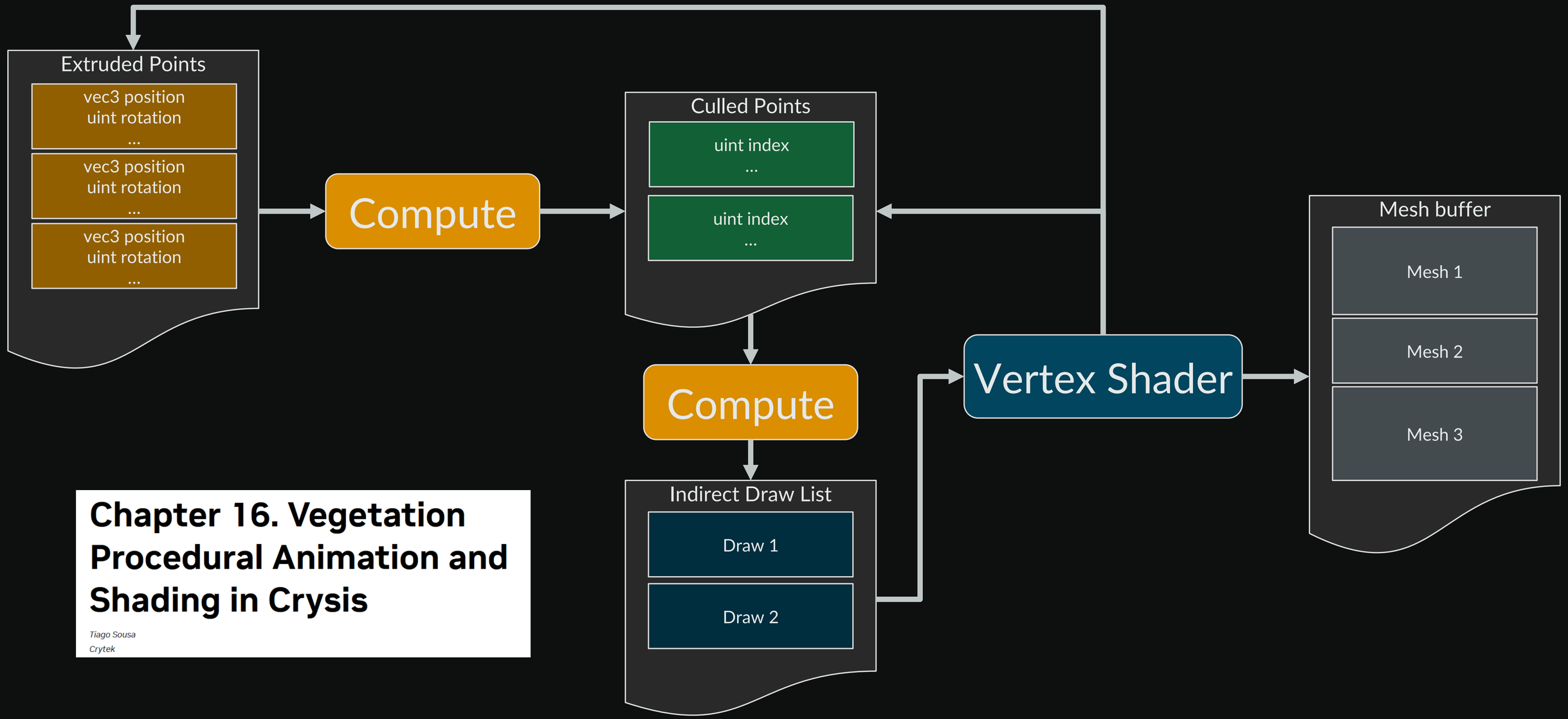
- Quad billboard meshes
- Low resolution atlas textures
- Low vegetation density
- High density 3D vegetation




CPU
Extruded Points
vec3 position
uint rotation
...
vec3 position
uint rotation
...
vec3 position
uint rotation
...
Mesh buffer
Mesh 1
Mesh 2
Mesh 3


Extruded Points
vec3 position
uint rotation
...
vec3 position
uint rotation
...
vec3 position
uint rotation
...
Compute
Culled Points
uint index
...
uint index
...


Extruded Points
vec3 position
uint rotation
...
vec3 position
uint rotation
...
vec3 position
uint rotation
...
Compute
Culled Points
uint index
...
uint index
...
Compute
Indirect Draw List
Draw 1
Draw 2

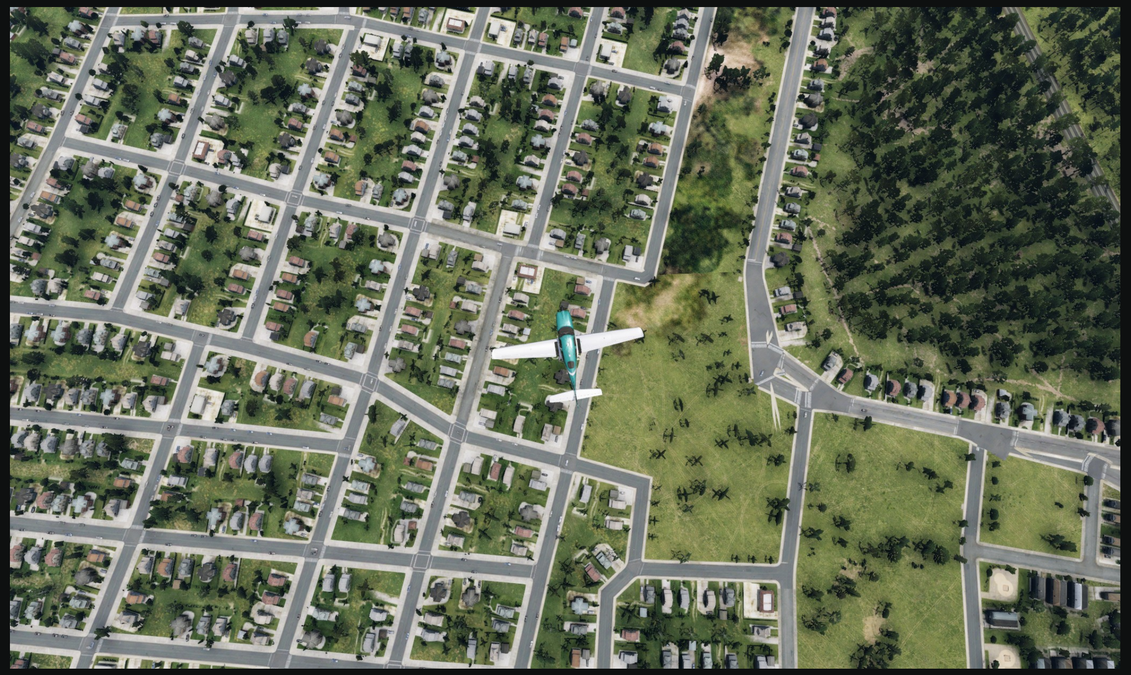


Chapter 16. Vegetation Procedural Animation and Shading in Crysis

Tiago Sousa
Crytek

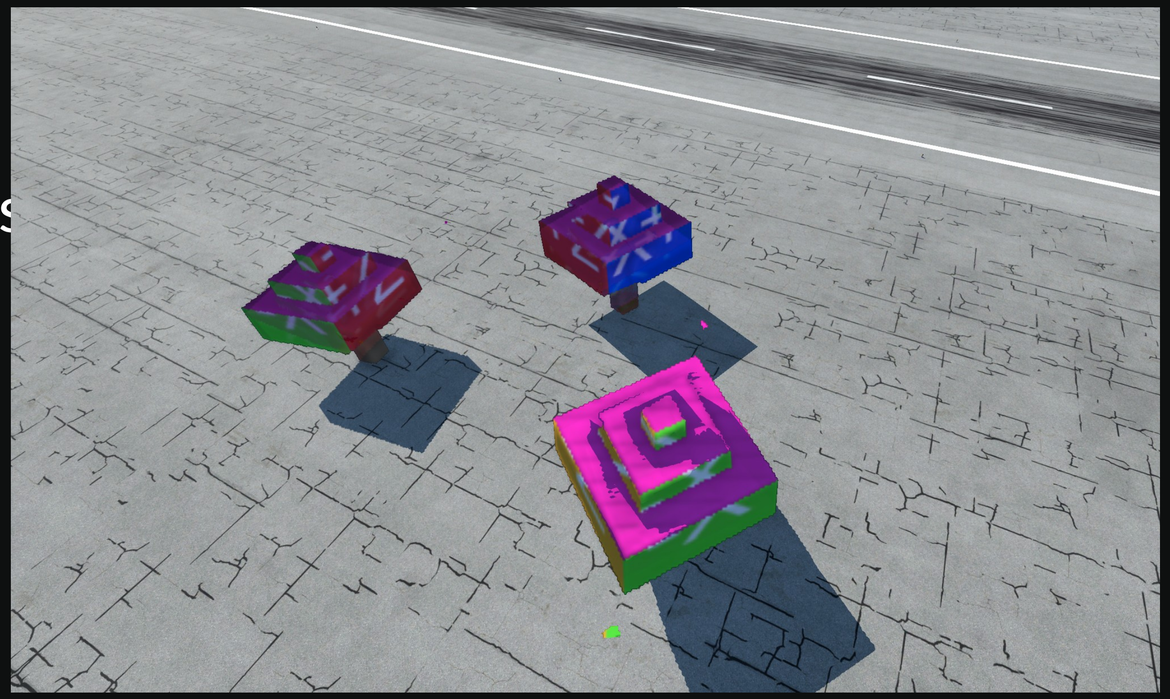
Vegetation - LODs

- Still billboard LODs in distance
- Noticeable at steep angles



Vegetation - LODs

- Still billboard LODs in distance
- Noticeable at steep angles
- Experimented with octahedral imposters
- Required too much VRAM
  - We have 2146 vegetation objects!

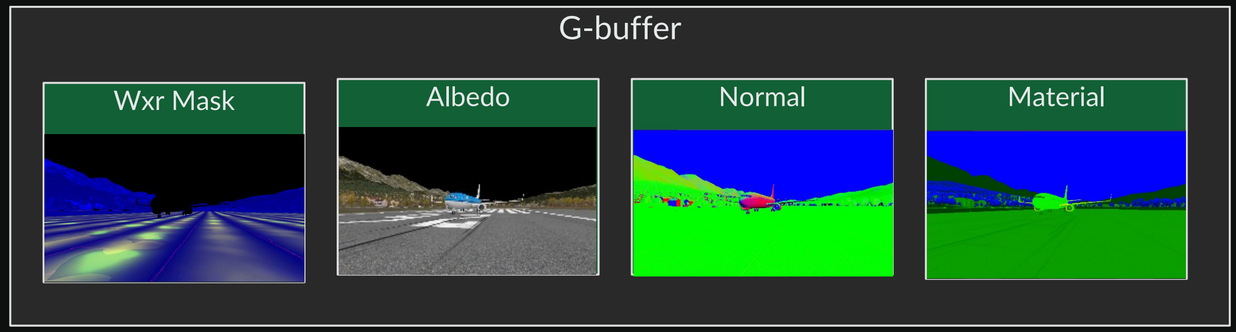


Seasons

Seasons & Weather

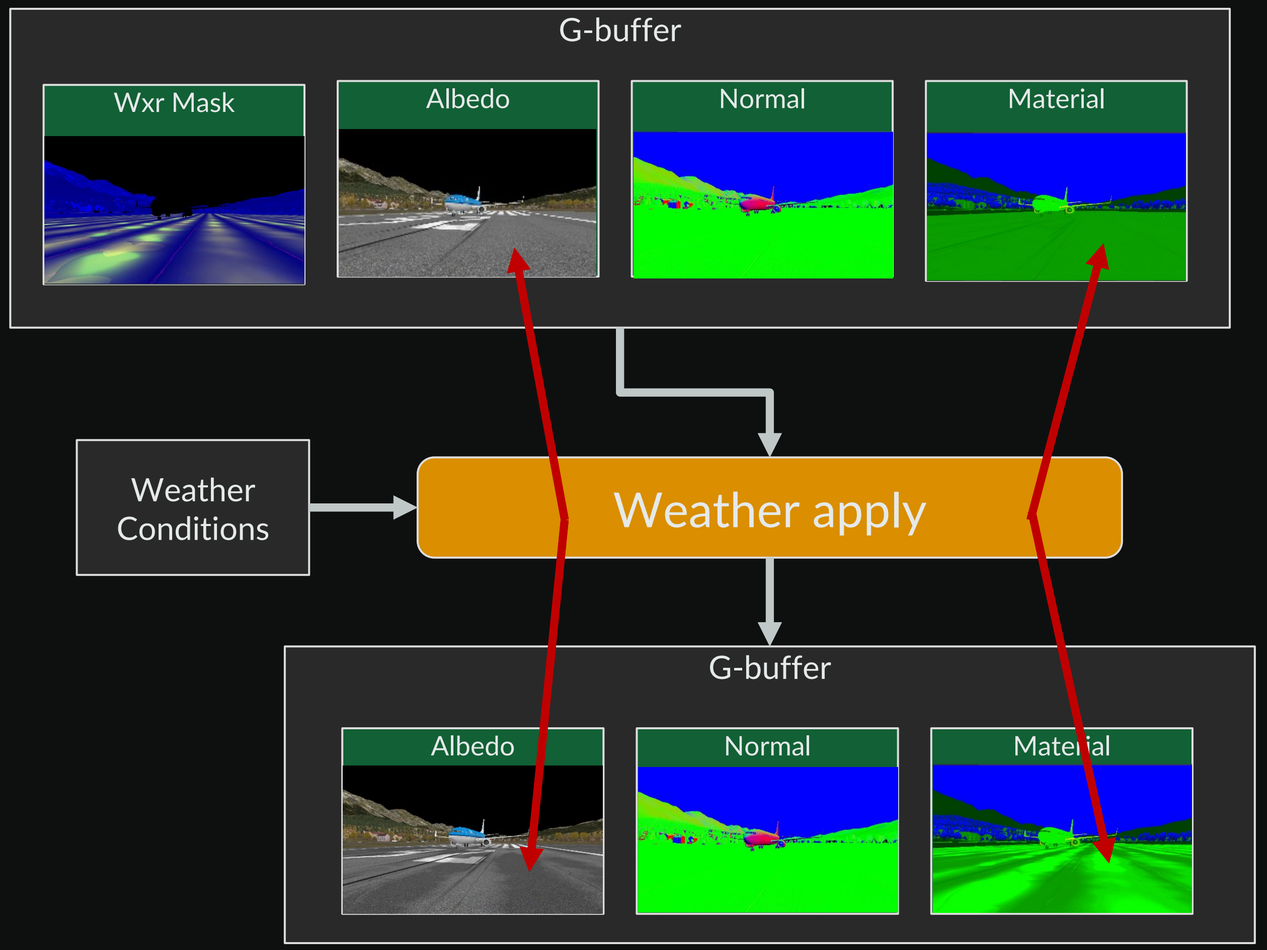
Weather effects

- G-buffer includes weather mask



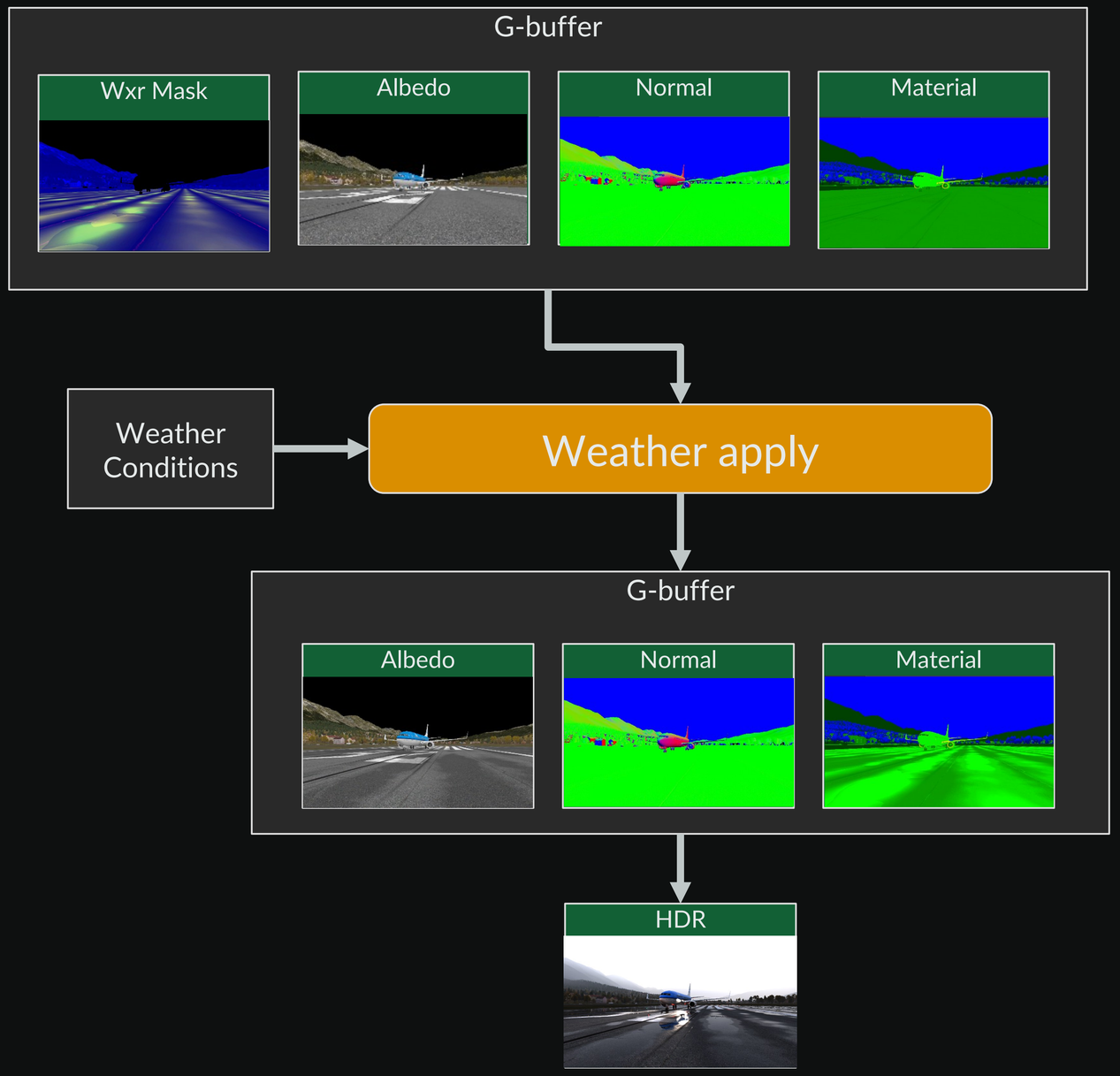
Weather effects

- G-buffer includes weather mask
- Weather apply modifies in-place
  - Albedo
  - Normal
  - Material

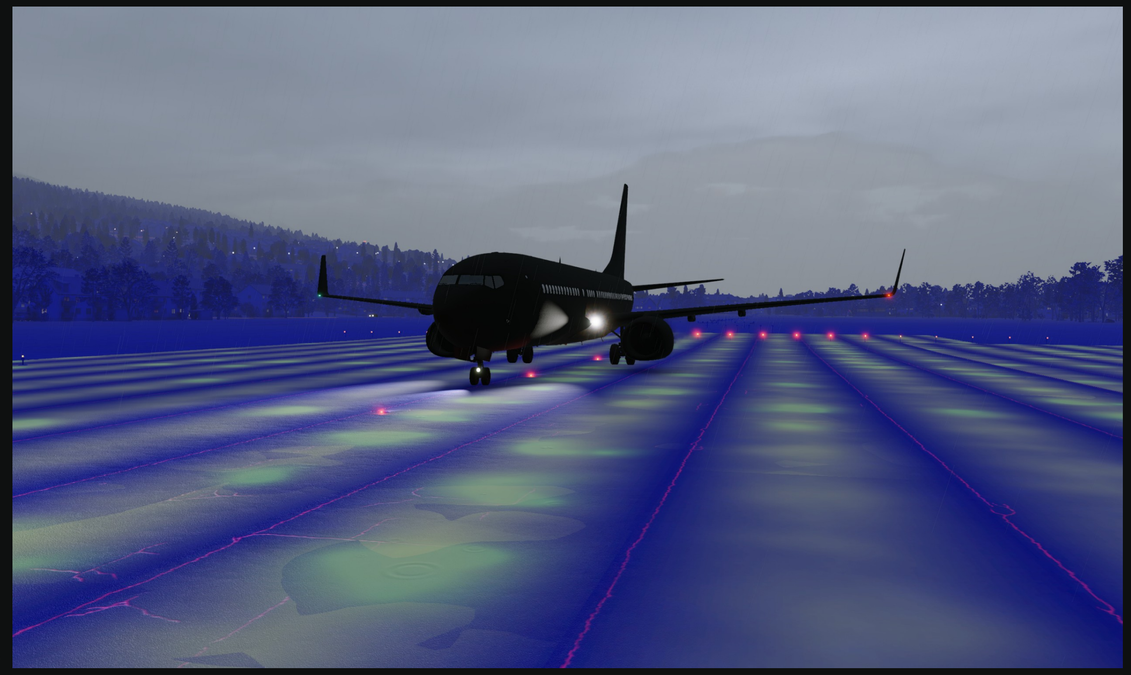


Weather effects

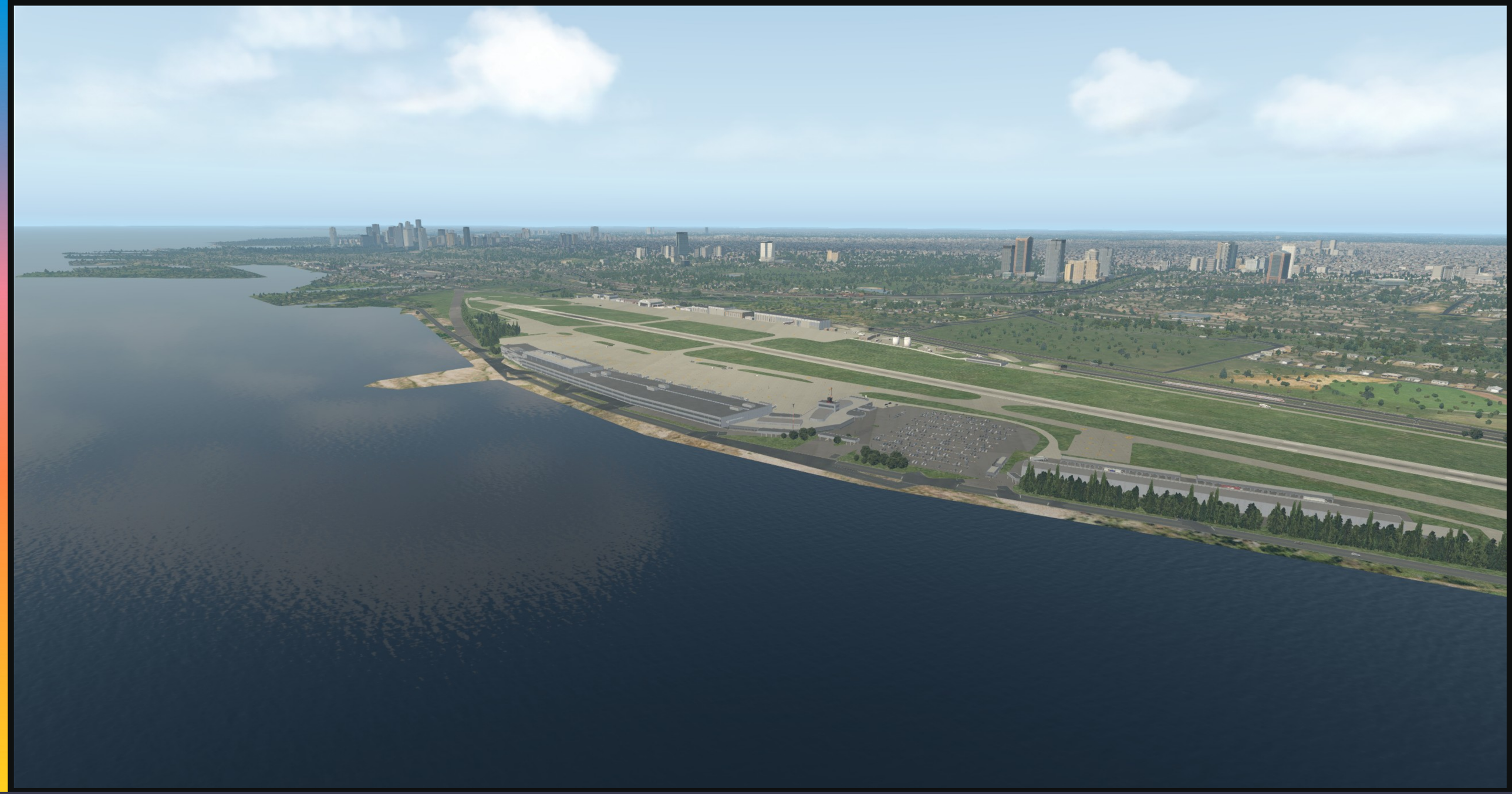
- G-buffer includes weather mask
- Weather apply modifies in-place
  - Albedo
  - Normal
  - Material
- G-buffer resolves as normal



Weather effects in practice



XPLANE Laminar Research

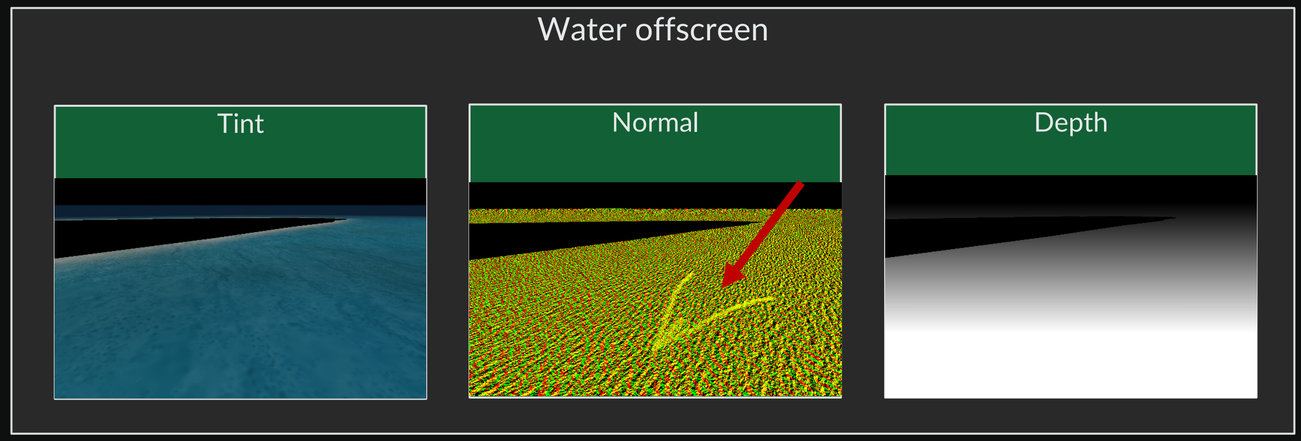


Water

- Art driven turbidity & tinting
- Not enough resolution in base mesh
- Projected grid to improve resolution
- Our world is genuinely round

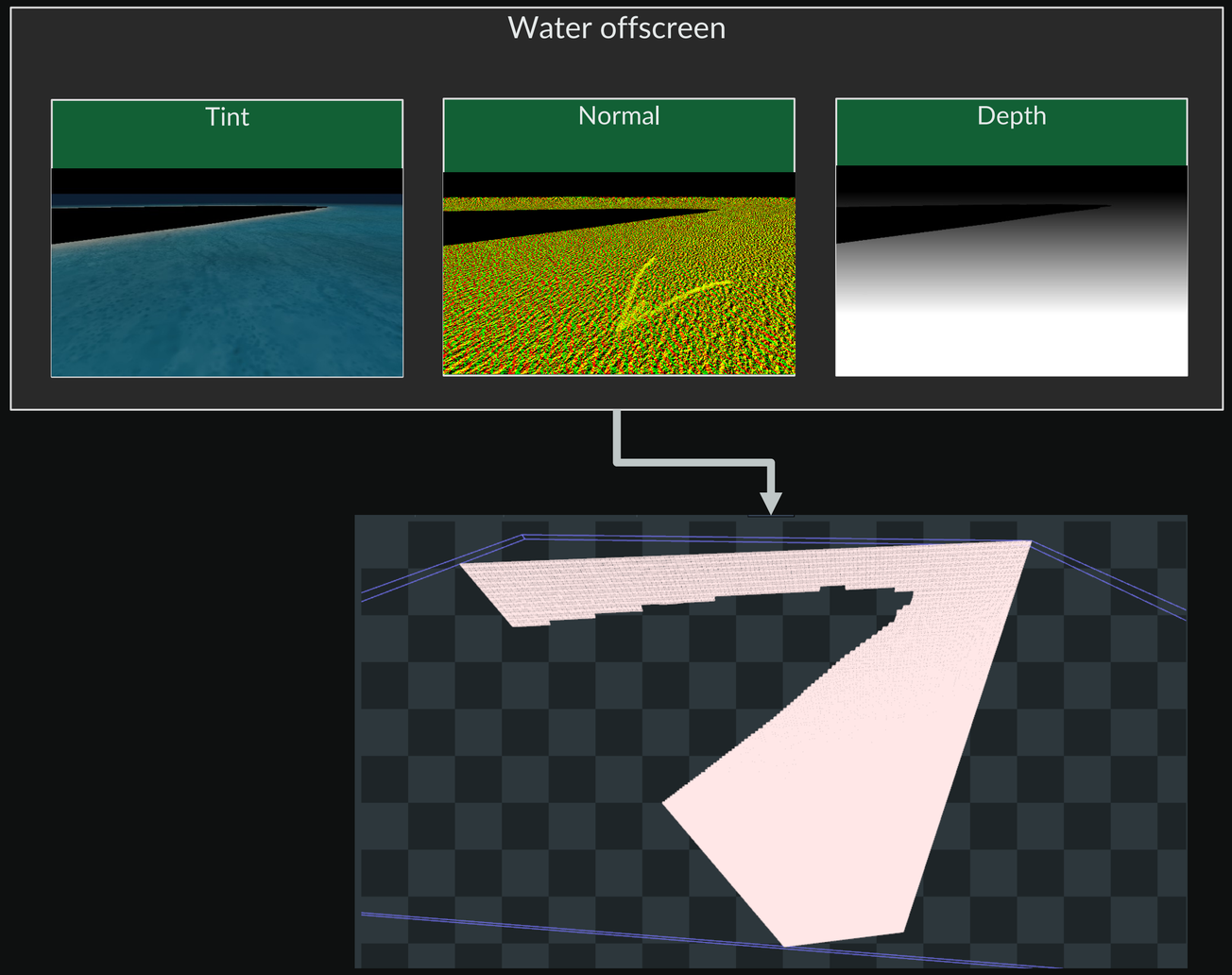
Water

- Art driven turbidity & tinting
- Not enough resolution in base mesh
- Projected grid to improve resolution
- Our world is genuinely round
- Offscreen pass for base water mesh



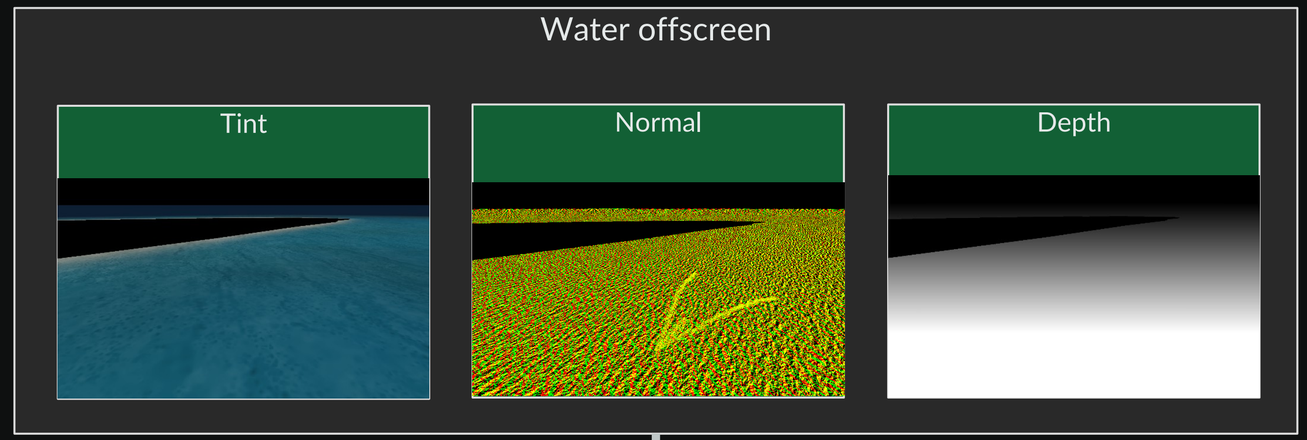
Water

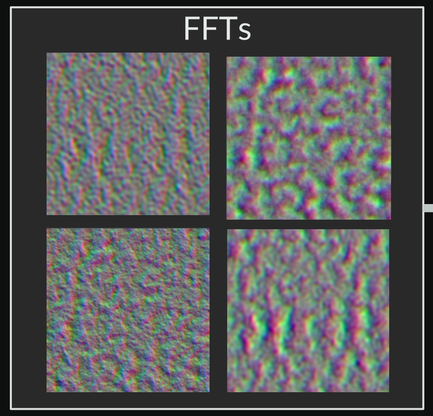
- Art driven turbidity & tinting
- Not enough resolution in base mesh
- Projected grid to improve resolution
- Our world is genuinely round
- Offscreen pass for base water mesh
- Project grid against depth

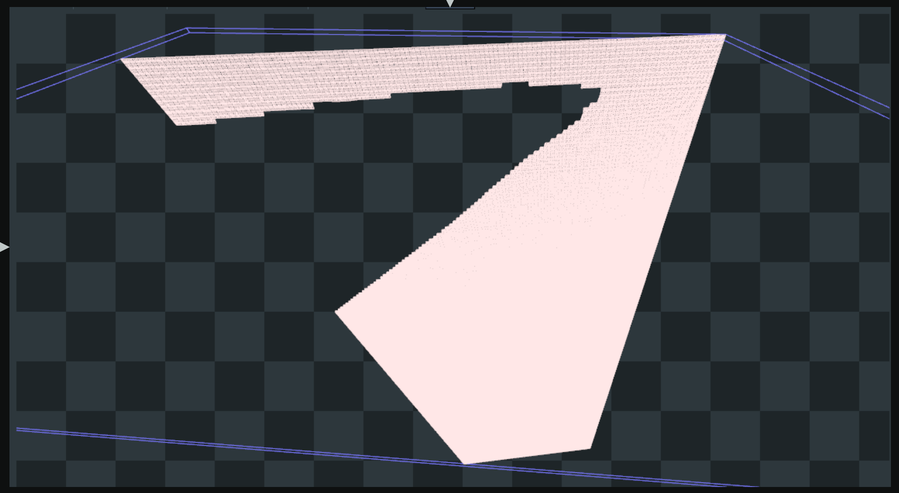


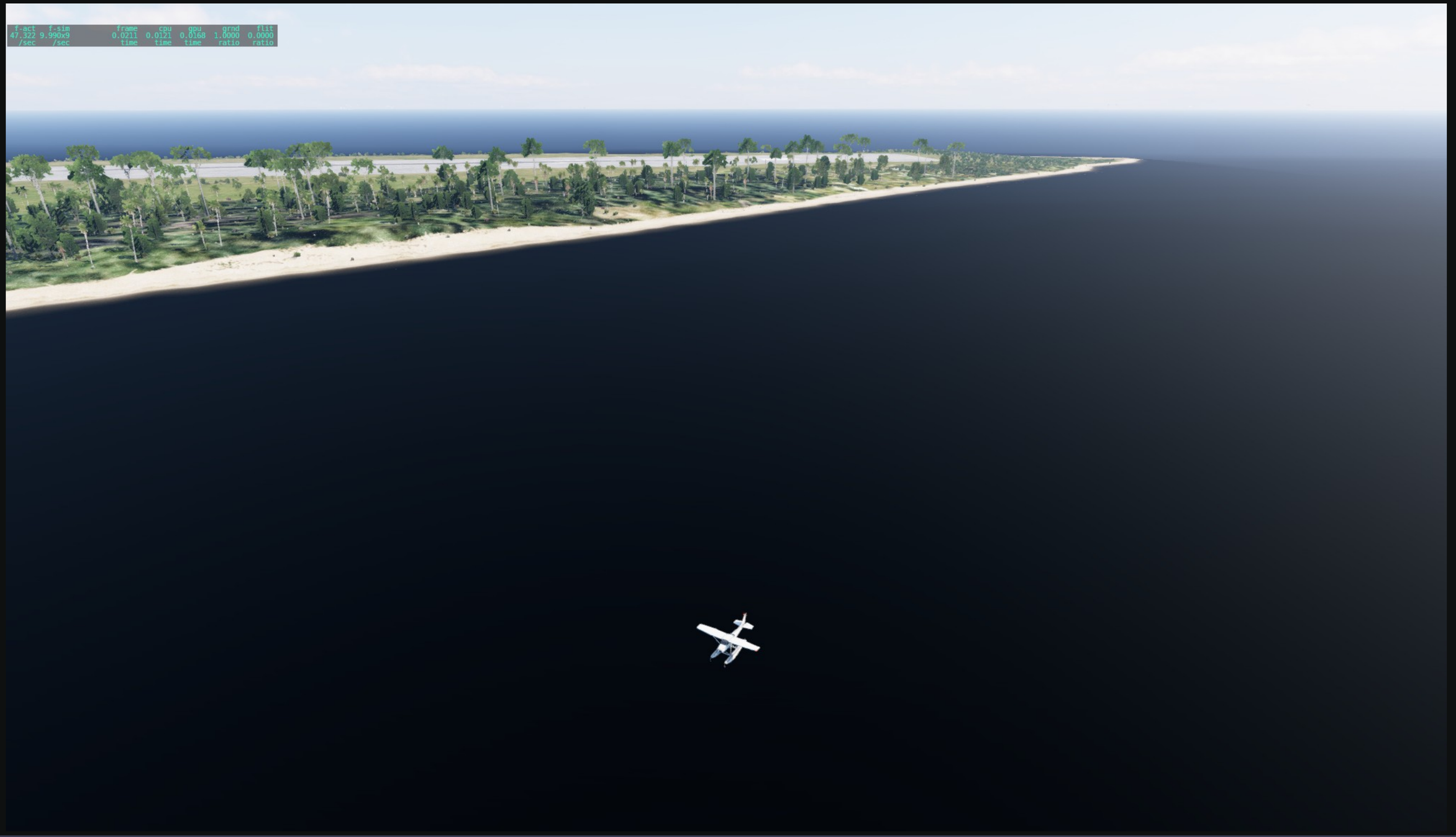
Water

- Art driven turbidity & tinting
- Not enough resolution in base mesh
- Projected grid to improve resolution
- Our world is genuinely round
- Offscreen pass for base water mesh
- Project grid against depth
- Shape water using FFTs

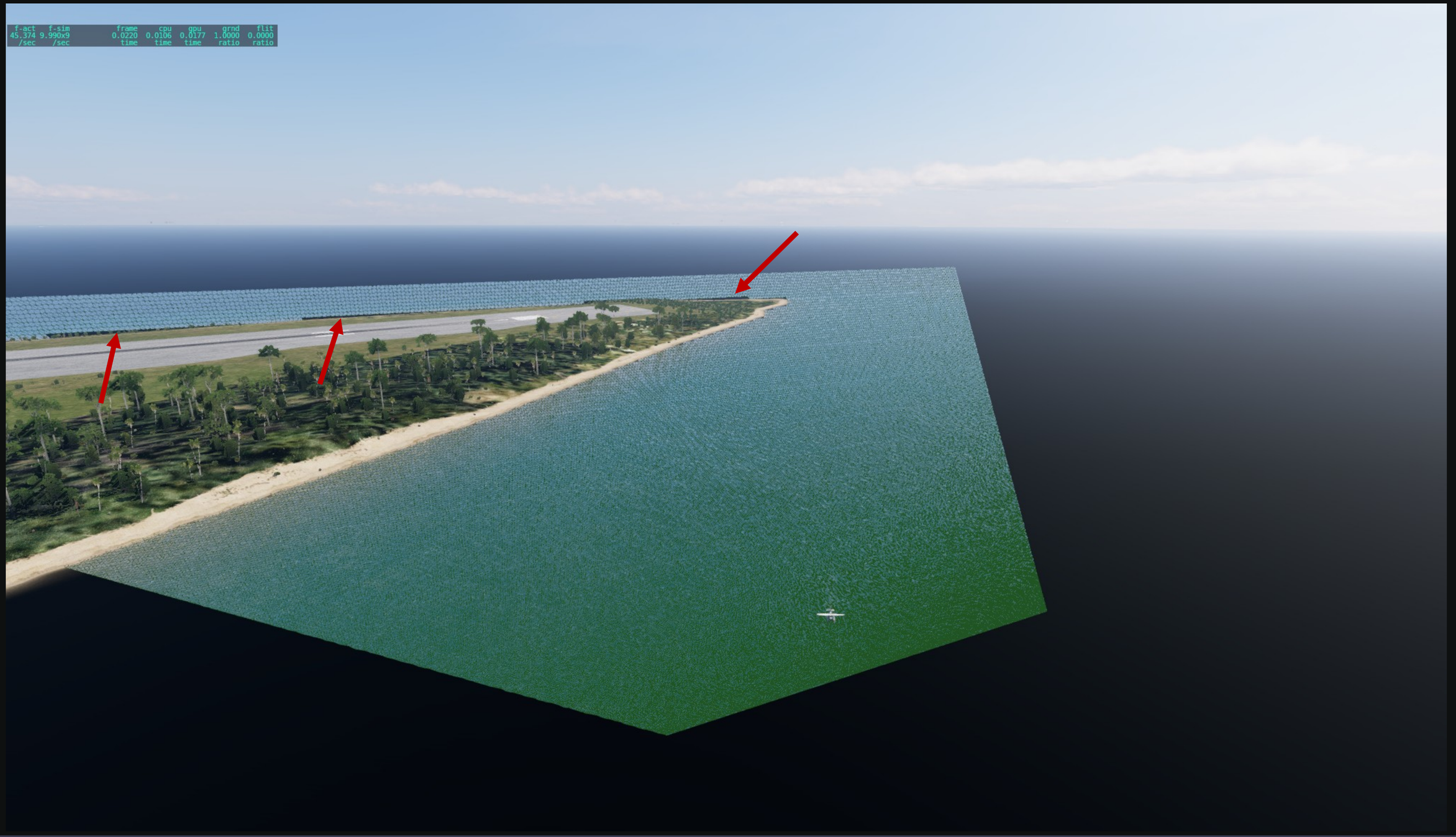






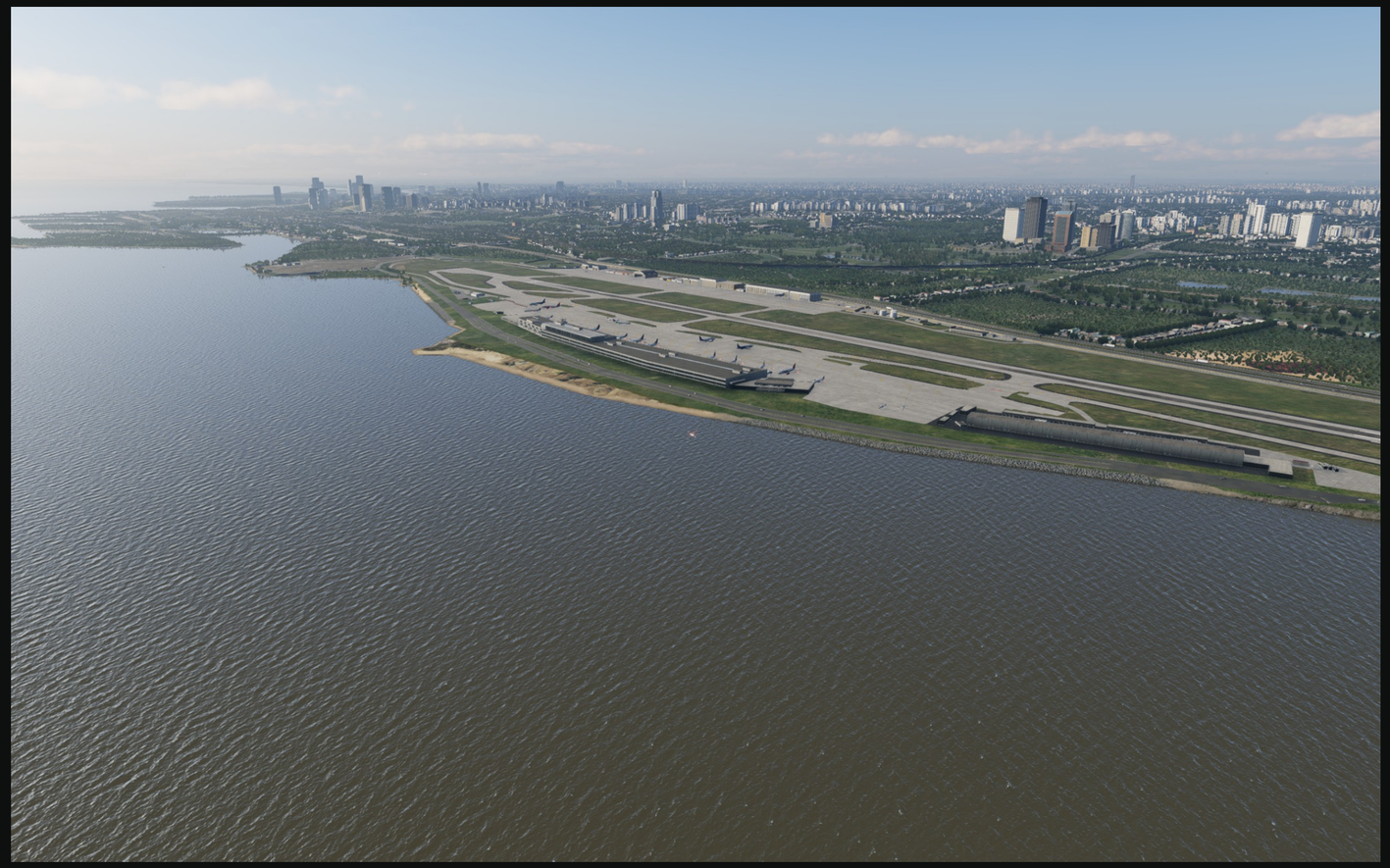

f-act f-sim frame cpu gpu grnd flit
47.322 9.990x9 0.0211 0.0121 0.0168 1.0000 0.0000
/sec /sec time time time ratio ratio

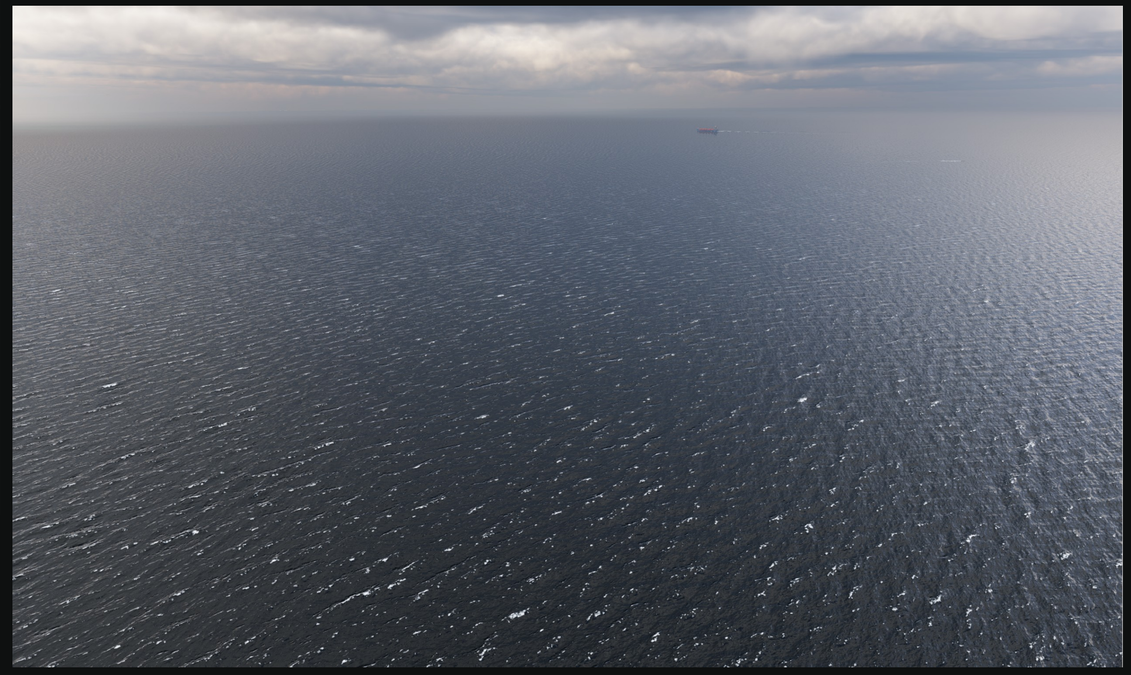

f-act f-sim frame cpu gpu grnd flit
44.248 9.990x9 0.0226 0.0122 0.0182 1.0000 0.0000
/sec /sec time time time ratio ratio


f-act f-sim frame cpu gpu grnd flit
45.374 9.990x9 0.0220 0.0106 0.0177 1.0000 0.0000
/sec /sec time time time ratio ratio


f-act f-sim frame cpu gpu grnd flit
44.156 9.990x9 0.0226 0.0112 0.0184 1.0000 0.0000
/sec /sec time time time ratio ratio

Water

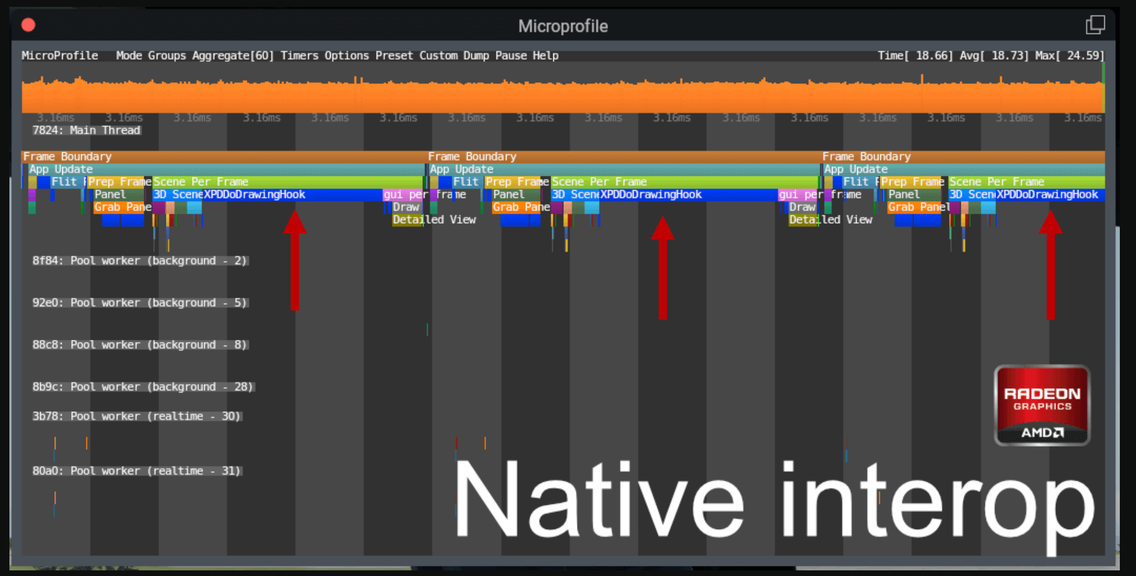




XPLANE Laminar Research

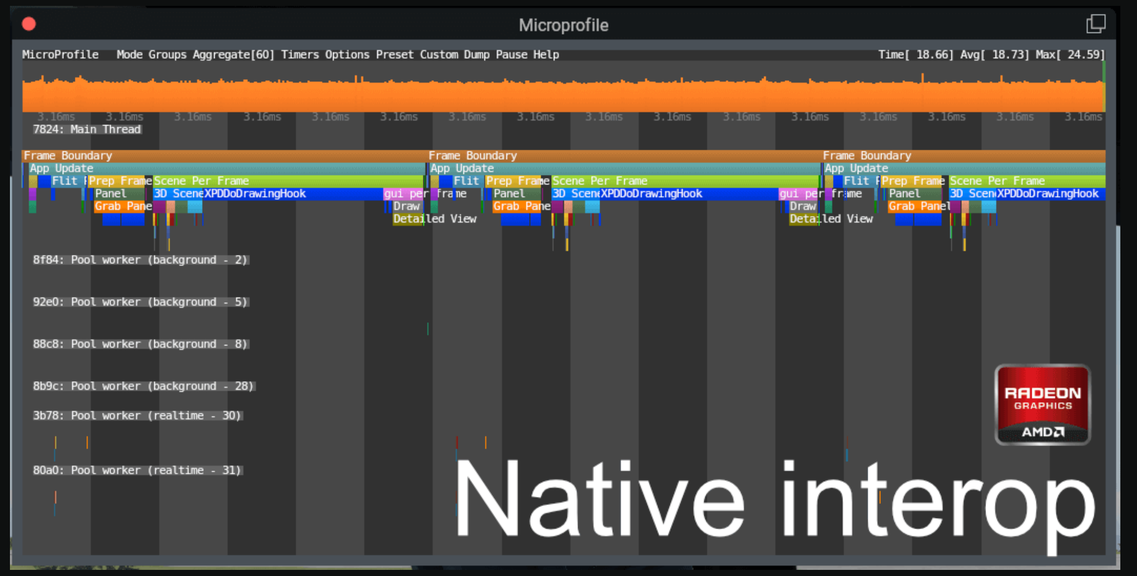
The plugin bridge again - Zink

- Vulkan/GL interop can be slow
- No more OpenGL backend fallback
- Zink implements OpenGL on top of Vulkan
- Custom opengl32.dll & libGL.so



The plugin bridge again - Zink

- Vulkan/GL interop can be slow
- No more OpenGL backend fallback
- Zink implements OpenGL on top of Vulkan
- Custom opengl32.dll & libGL.so
- Massively reduces CPU overhead
- Some issues with line emulation



Where we are going

- X-Plane is 30 years old now 🥳
- New scenery engine
- Local tangent plane coordinates (ENU)
- New object model format (GLTF)
- Multicore engine
- Motion vectors

Special Thanks

Graphics Team

Ben Supnik – Lead Coffee Drinker
Daniel Evans
Maya - just one more cloud tweak - F. Eroğlu

Art Team

Alex Unruh
Christiano Maggi
Daniela Rodríguez Careri
Jan Vogel – Resident 747 Captain
Justin Kissling
Massimo Durando
Petr Bednář – The 2461 vegetation object machine
Rodrigo Fernandez

I'm a graphics programmer at Laminar Research, AMA Random Story (self.AMA)

submitted just now by JustSid

XPLANE Laminar Research