GDC 13 EUROPE

The Art and Rendering of Remember Me

Sébastien Lagarde
Senior 3D/Engine programmer, DONTNOD
Laurent Harduin
Lighting Artist, DONTNOD

GAME DEVELOPERS CONFERENCE EUROPE
COLOGNE, GERMANY
AUGUST 19-21, 2013
EXPO DATES: AUGUST 19-20
2013

DONTNOD Entertainment

- Young French studio located in Paris
- 5 years old
- Game and studio created in parallel
- Grew from 5 up to 95 internal employees

Remember Me

- Released June 2013
- Sci-fi action-adventure third person game
- XBOX360 / PS3 / PC
- Published by Capcom
- Based on Unreal Engine 3

Conception :

Key concept

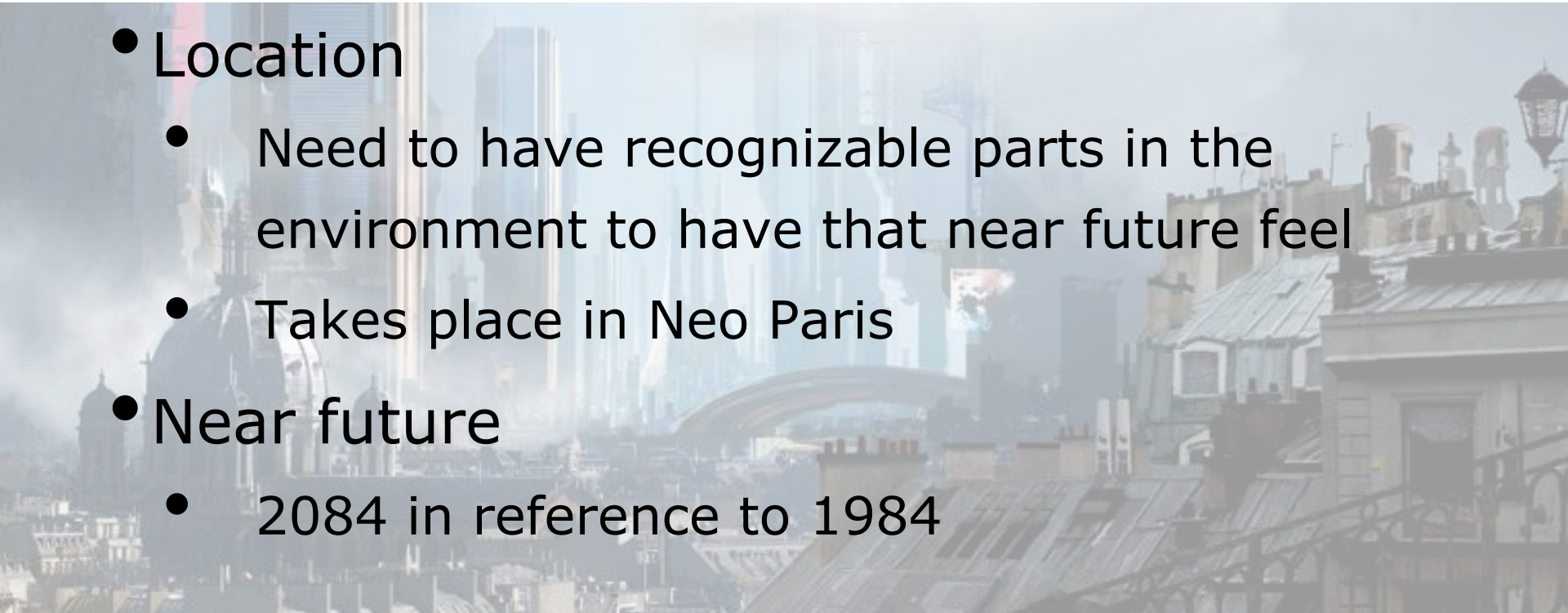
Neo-Paris

Virtual Spaces

Conception - Key concepts

- Memory manipulation
- Inspired by social networks
- Had to be perceived as an evolution of today's "share everything" way of life

Conception - Key concepts



- Location
  - Need to have recognizable parts in the environment to have that near future feel
  - Takes place in Neo Paris
- Near future
  - 2084 in reference to 1984

Conception :

Key concept

Neo-Paris

Virtual Spaces

Conception - Neo Paris

- Believable world and strong art direction
  - Stylized realism
    - Sci-fi visual codes (shiny, moody, reflections, contrast, high tech, glossy)
    - Specific color schemes

VISUAL STYLE
COLOR PALETTE

Palette color example: Episode I.

-One blue/gray desaturated overall tone.

-Hints of brighter blue, purple and red on some elements (props, tarpaulins...) and use of red artificial light.

>THE COLOR PALETTE IS CHANGING THROUGHOUT THE WHOLE GAME BUT APPLY THIS COLOR LOGIC ON ALL REMEMBER ME VISUALS: DESATURATED GLOBAL VALUE AND HINTS OF BRIGHTER COLORS.



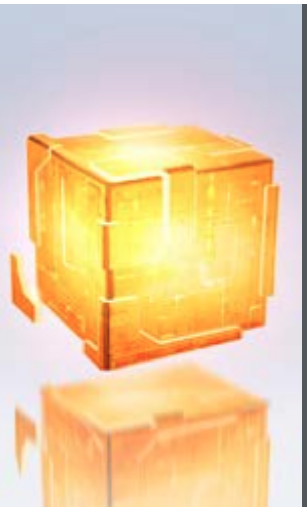







Conception - Neo Paris

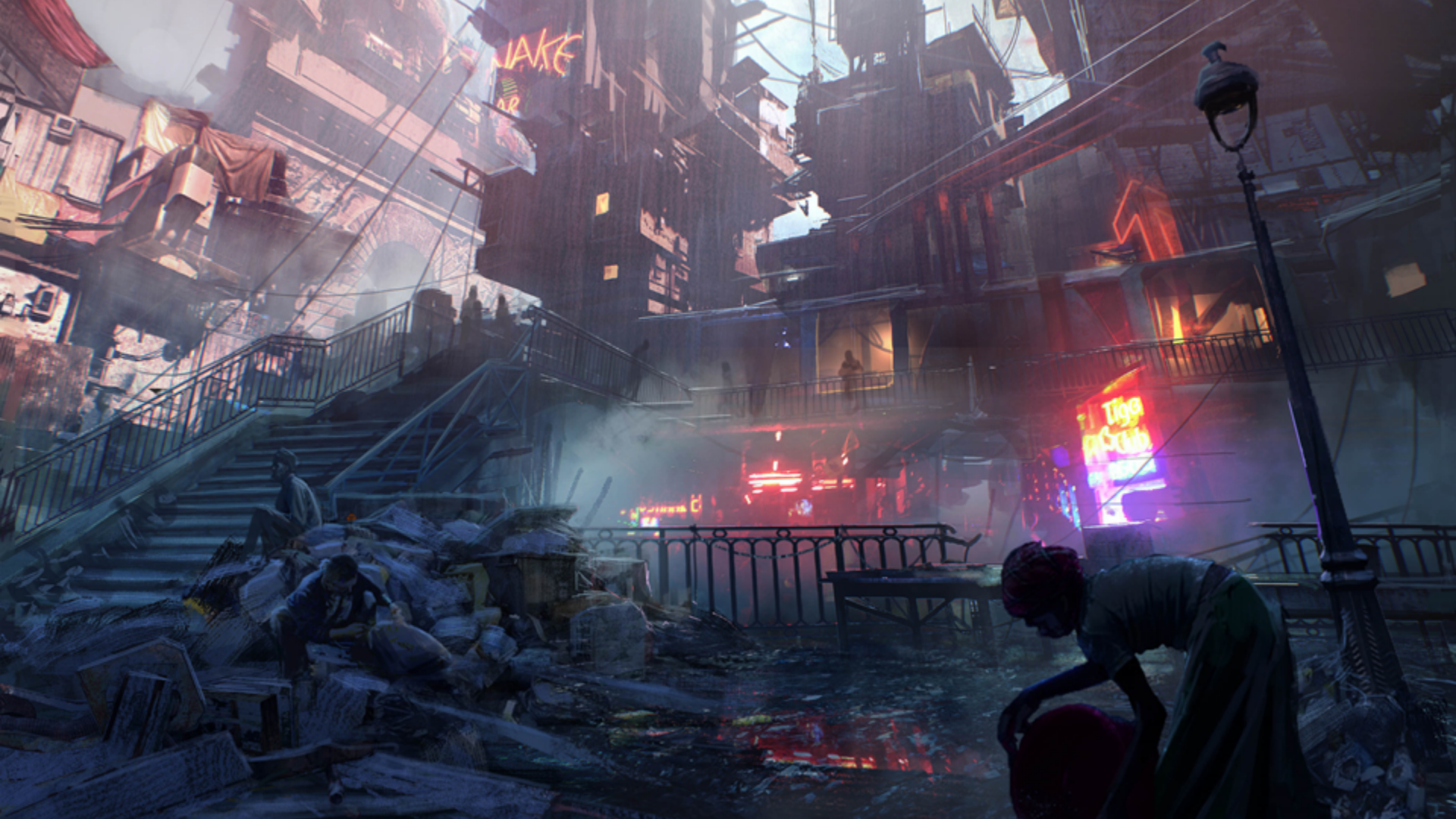
- Graphic elements



Conception - Neo Paris

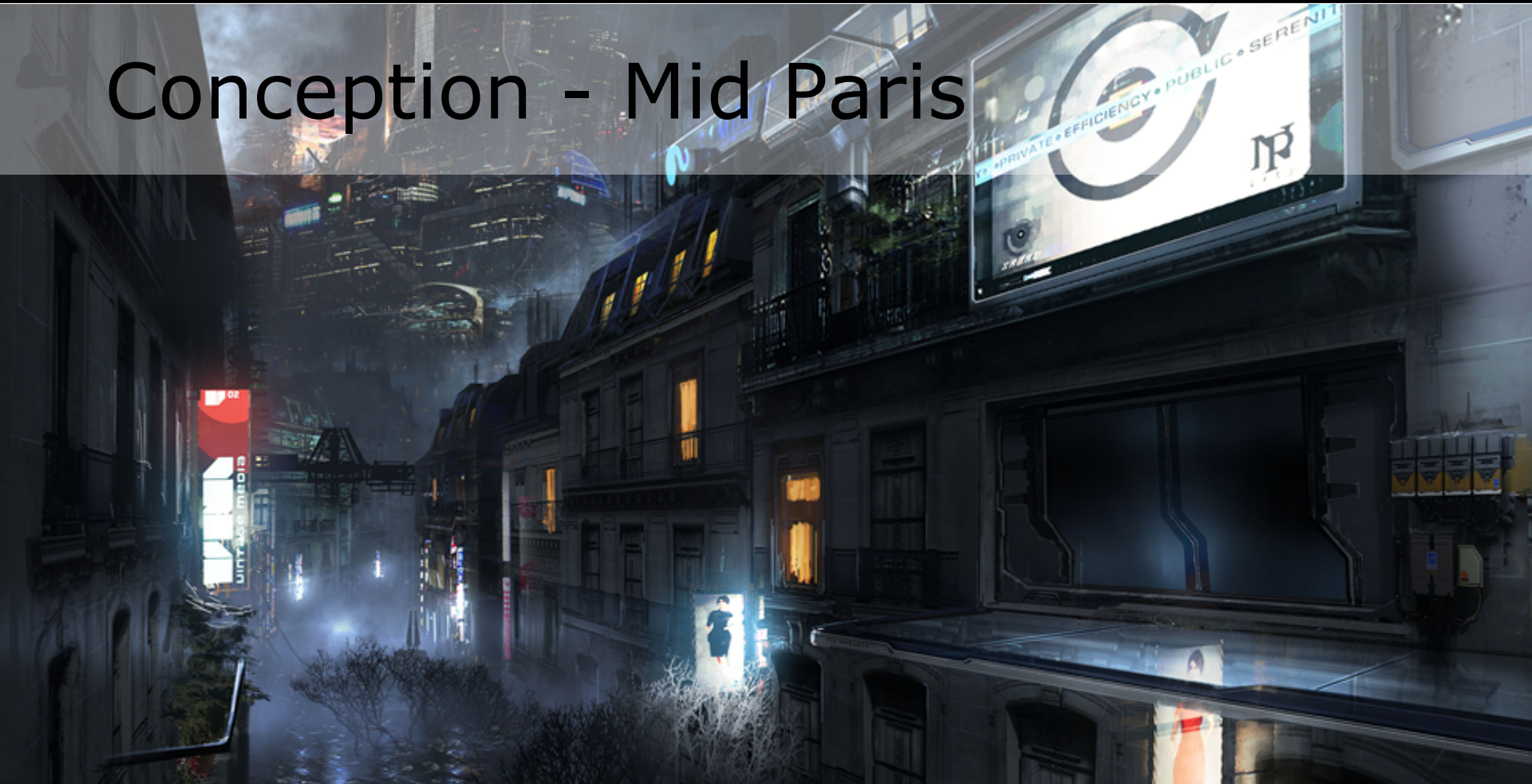
- Only one city
- Three different areas

Conception - Deep Paris




LEAKING BRAIN
QUAI
DE VALMY
SENSEN

Conception - Mid Paris

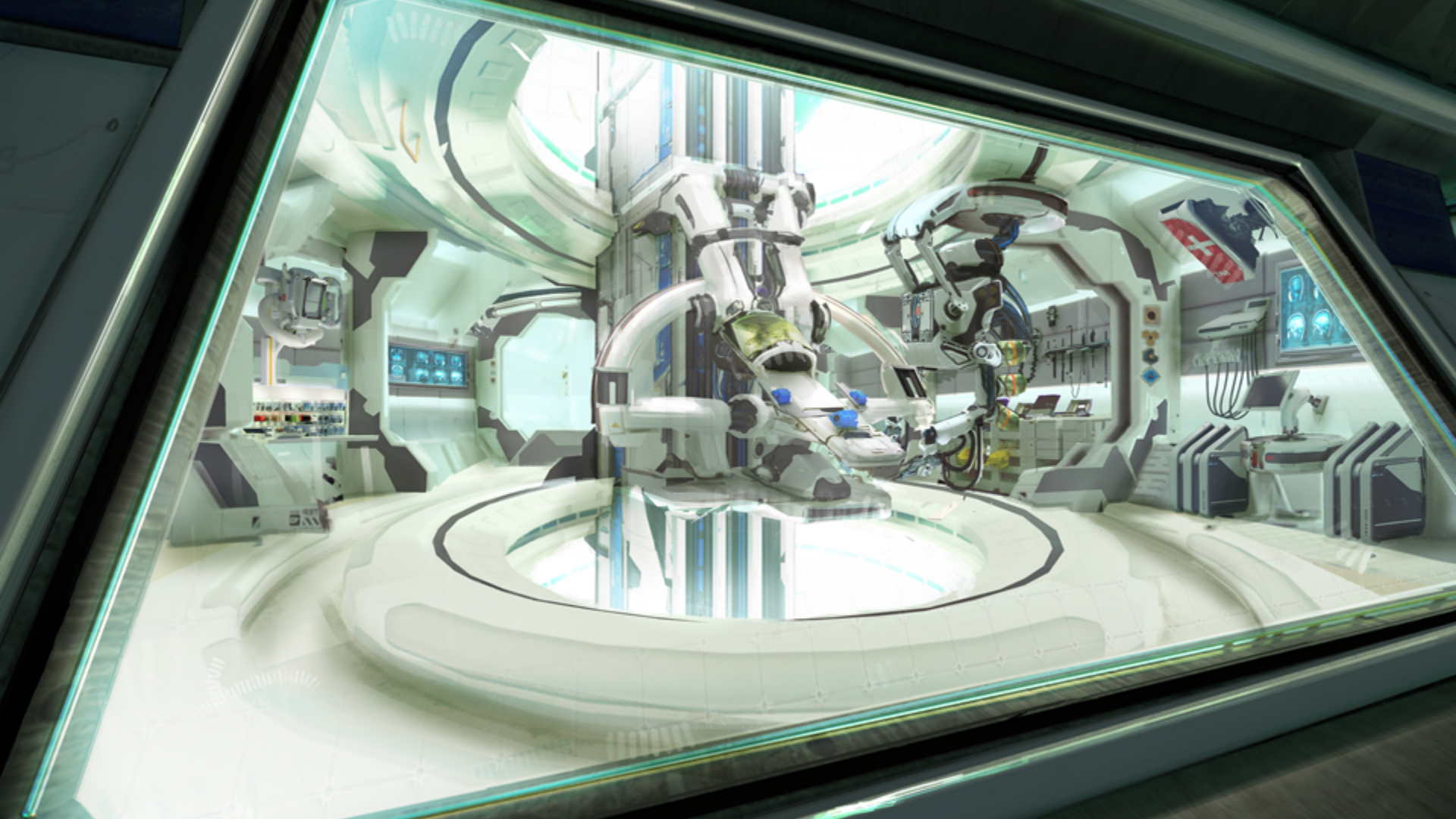



MORE HUMAN THAN HUMAN

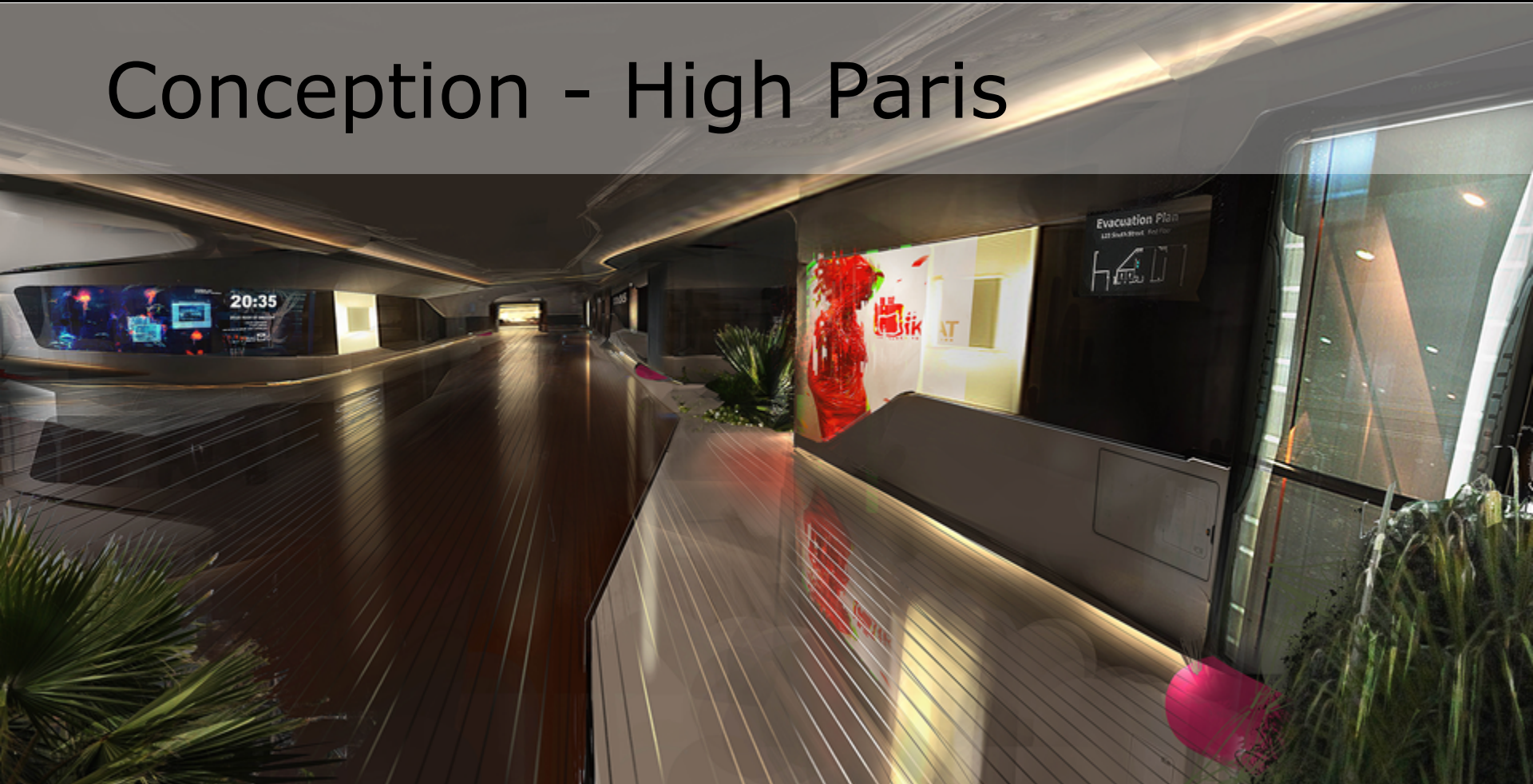

N'AYEZ PLUS PEUR
MNEMOPOLIS
CLUB FOR MEN
It's
X-MaS
eveRry
Day
KAORU SUEDAN
PARIS ARCHITECTURE

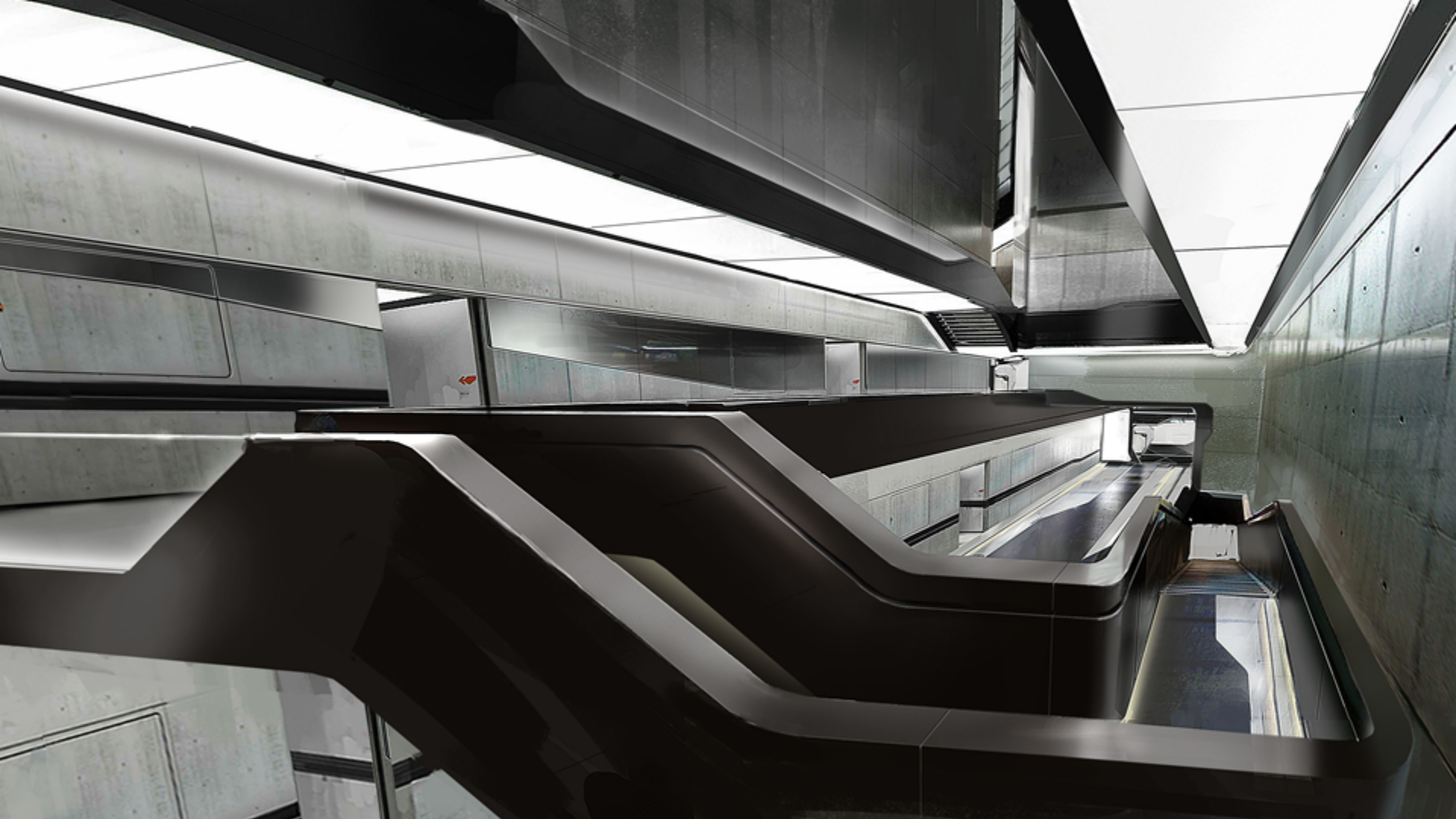

PLAN DE QUARTIER

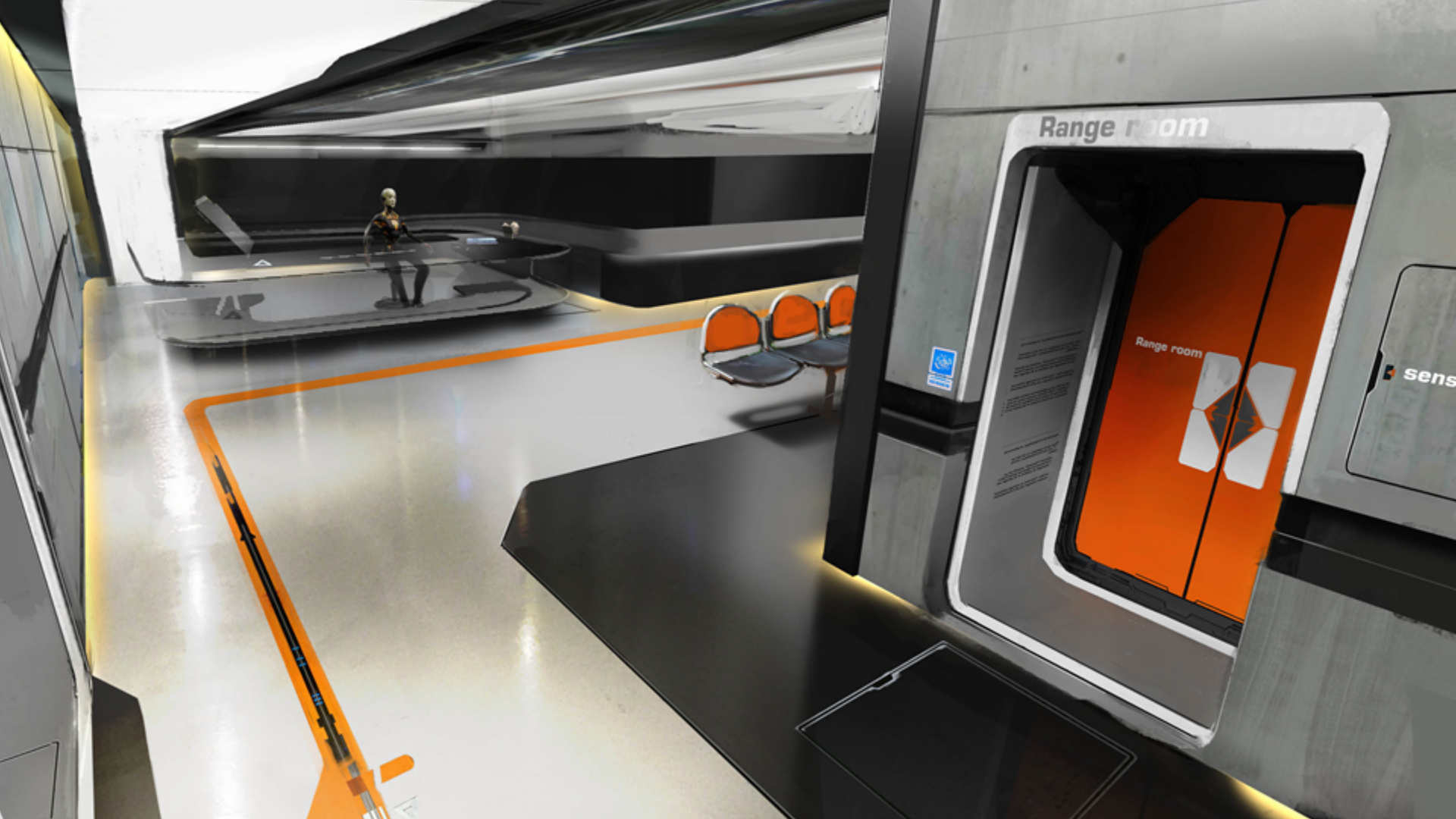
Conception – Bastille



Conception - High Paris





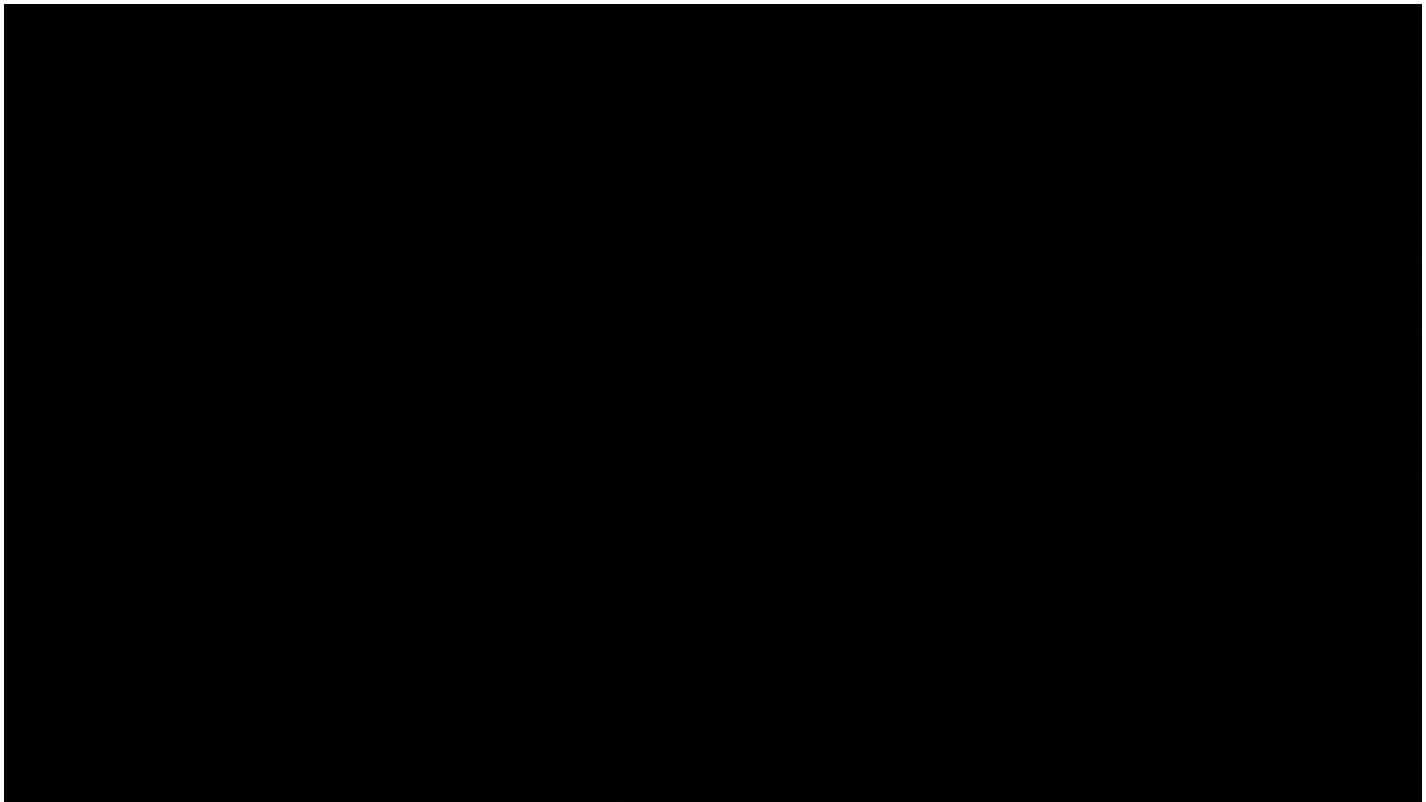

Range room
Range room


MNEMOPOLIS
Security Post
BAY 05 >

Development agenda

- Believable world
- Rainy mood
- Reflections
- Ambient occlusion
- Image enhancement

Believable world - Landmarks



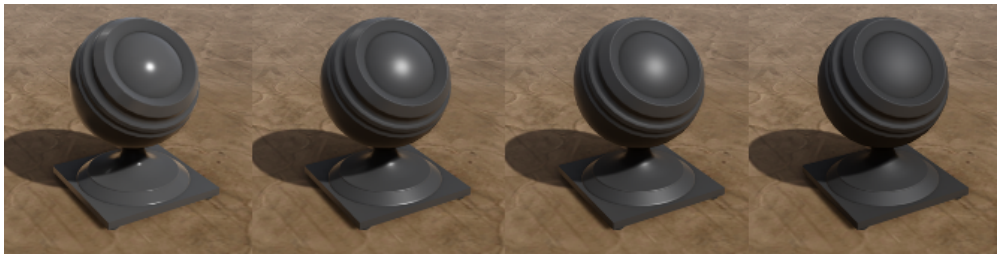
Believable world - Realistic rendering

Physically based rendering (PBR)

- Inspirational SIGGRAPH 2010 PBR course
  - Thanks to Naty Hoffman and Yoshiharu Gotanda
- Benefit:
  - Intuitive parameters
  - Fewer parameters
  - Easier to achieve photorealism
  - Consistent looks under different lighting conditions

Physically based rendering

- What is different with PBR?
  - Fresnel everywhere
  - Energy conservation



Remember Me shading model

- Rendering equation

$$L_o(v) = \int_\Omega f(l, v) L_i(l)(l \cdot n) d_{\omega_i}$$

Resulting color

For all incoming lighting

BRDF (Shading model)

Lighting

Lambert's cosine law

Remember Me shading model

- Direct lighting
  - Analytic lights
  - Naty Hoffman's Blinn microfacet model [Hoffman10]

Fresnel eyerywhere

Blinn model

$$L_o(l,v) = f(l,v)L_i(l)(l.n) = (\frac{albedo}{\pi} + F_{schlick}(specular,l,h)\frac{SpecPower+2}{8\pi}(n \cdot h)^{SpecPower})L_i(l)(l.n)$$

Energy conservation

Analytic light

Remember Me shading model

- Indirect lighting
  - Environment lights (Image based lighting)
- Separate diffuse and specular

$$L_o(v) = \int_\Omega (\frac{albedo}{\pi} + F_{schlick}(specular, l, h) \frac{SpecPower+2}{8\pi} (n \cdot h)^{SpecPower}) L_i(l)(l.n) d_{\omega_i}$$

Diffuse

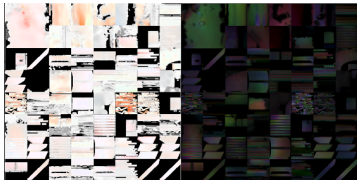
Specular

Environment light

$$L_o(v) = \int_\Omega \frac{albedo}{\pi} L_i(l)(l.n) d_{\omega_i} + \int_\Omega F_{schlick}(specular, l, h) \frac{SpecPower+2}{8\pi} (n \cdot h)^{SpecPower} L_i(l)(l.n) d_{\omega_i}$$

Remember Me shading model

- Indirect diffuse lighting
  - Lightmap
    - For background
  - Irradiance volume of spherical harmonics
    - For dynamic objects
  - Available in UE3



Remember Me shading model

- Indirect specular lighting
- Too complex to pre-integrate
  - Approximate by splitting in two parts
  - Not mathematically correct

$$L_{speco}(v) = \int_{\Omega} F_{schlick}(specular, l, h) \frac{SpecPower+2}{8\pi} (n \cdot h)^{SpecPower} L_i(l)(l.n) d_{\omega_i}$$

Glossy Fresnel

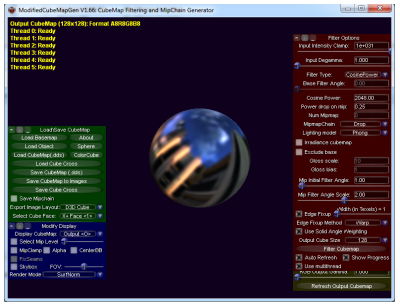
Pre-integrated cubemap

$$L_{speco}(v) = F_{glossy}(specular, n, v, glossiness) \int_{\Omega} \frac{SpecPower+2}{2\pi} (n \cdot h)^{SpecPower} L_i(l)\ (l.n) d_{\omega_i}$$

Remember Me shading model

- Pre-integrated cubemap

$$\int_{\Omega} \frac{SpecPower+2}{2\pi} \underline{(n \cdot h)^{SpecPower}} L_i(l)\,(l.n) d_{\omega_i}$$



ModifiedCubemapgen [MCube12]

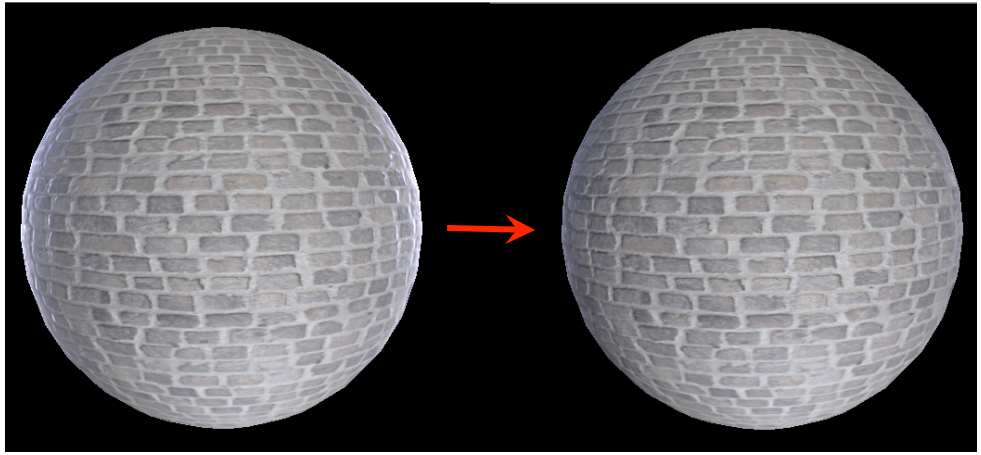
Remember Me shading model

- Glossy Fresnel

$$F_{glossy}(specular, n, v, smoothness) = specular + (\max(smoothness, specular) - specular)(1 - dot(n, v))^5$$

  - Rough surfaces reflect less light at grazing angles
  - Coarse approximation
  - Cheap and visually OK

Rough material

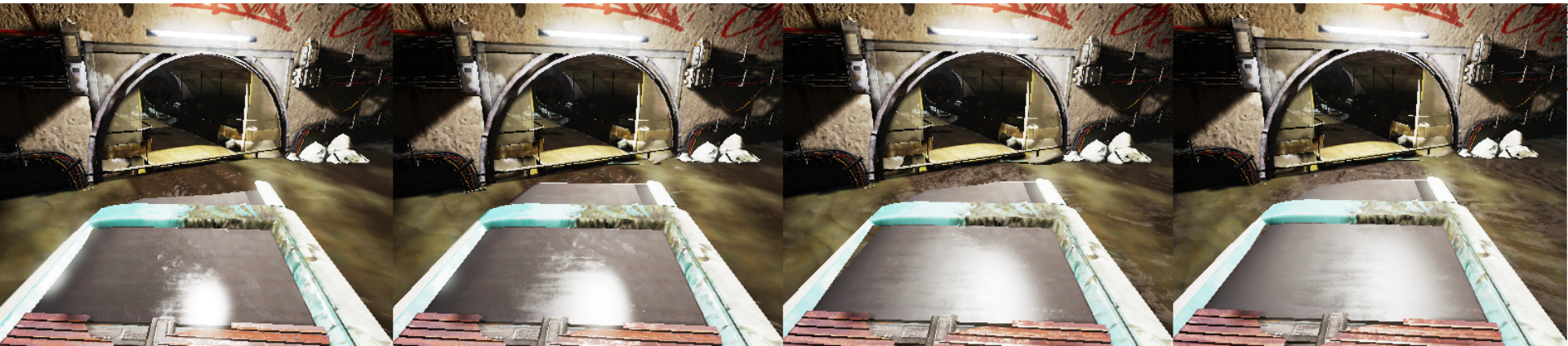


```
// Direct lighting
float3 FresnelSchlick(float3 SpecularColor,float3 E,float3 H)
{
    return SpecularColor + (1.0f - SpecularColor) * pow(1.0f - saturate(dot(E, H)), 5);
}
// For each light (Pi in energy conserving term is cancel by the Pi of punctual lights)
SpecularColor += FresnelSchlick(SpecularColor, L, H) * ((SpecularPower + 2) / 8 ) *
                    pow(saturate(dot(N, H)), SpecularPower) * dotNL * LightColor;

// Indirect lighting
float3 FresnelGlossy(SpecularColor,float3 E,float3 N,float Smoothness)
{
    return SpecularColor + (max(Smoothness, SpecularColor) - SpecularColor) *
                    pow(1 - saturate(dot(E, N)), 5);
}
// For one cubemap
float3 Envcolor = texCUBElod(EnvironmentTexture, float4(R, EnvMapMipmapScaleBias.x * Gloss +
                    EnvMapMipmapScaleBias.y)).rgb;

SpecularColor += FresnelGlossy(SpecularColor, N, E, Gloss) * Envcolor.rgb * EnvMapScaleAndModulate;
                    // EnvMapScaleAndModulate is used to decompress range
```

Physically based rendering

- Area lights
  - Important for physically based rendering
  - Spherical light hack [Gotanda11]

example of varying spherical light size



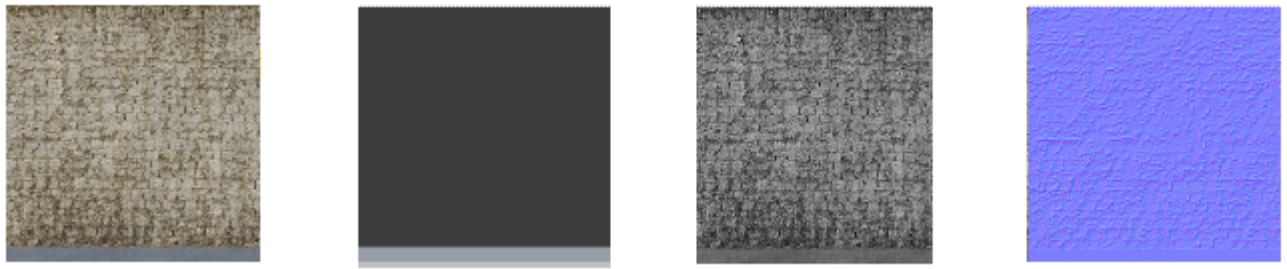
```
half SpecularPowerHack(half SpecularPower, half AttenuationFactor, half LightSizeForSpecularPower)
{
    half val = saturate(LightSizeForSpecularPower * AttenuationFactor);
    return SpecularPower * val * val;
}

half3 PointLightBlinnMicrofacet(half3 DiffuseColor,half3 SpecularColor,half SpecularPower, half3 L, half3
    N, half3 H, half AttenuationFactor)
{
    half   dotNL = saturate(dot(N, L));
    // (Pi in energy conserving terms are cancel by the Pi of punctual lights)
    half   DiffuseLighting = DiffuseColor * dotNL;

    SpecularPower = SpecularPowerHack(SpecularPower, AttenuationFactor, LightSizeForSpecularPower);
    half   SpecularLighting = FresnelSchlick(SpecularColor, L, H) * ((SpecularPower + 2) / 8 ) *
                              pow(saturate(dot(N, H)), SpecularPower) * dotNL
    return DiffuseLighting + SpecularLighting;
}
```

PBR – Artists guidelines

- Workflow - Concept art

PBR – Artists guidelines

$$\frac{albedo}{\pi} + F_{schlick}(specular, l, h)\frac{SpecPower+2}{8\pi}(n \cdot h)^{SpecPower}$$



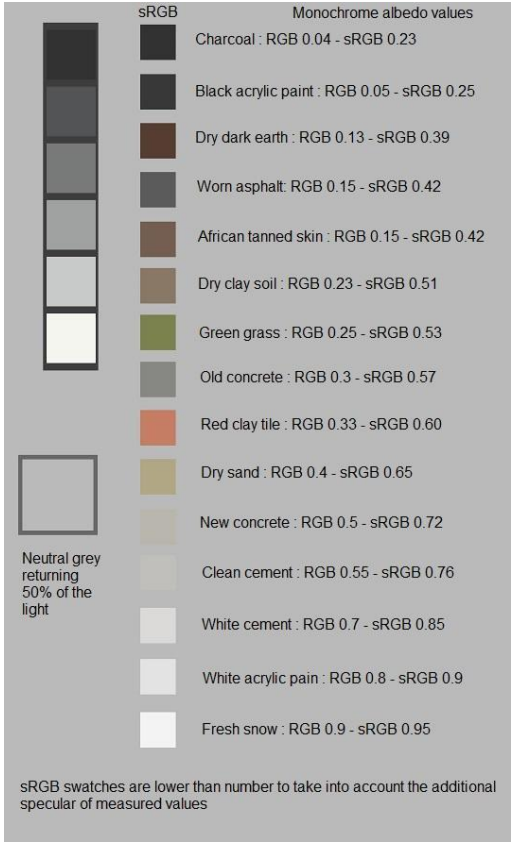
Albedo Specular Smoothness Normal

PBR – Artists guidelines

- Albedo (or diffuse color)
  - Strict name: “bi-hemispherical reflectance”
  - The characteristic color of an object
  - Has a physical meaning
- No lighting information
  - Lighting is processed by the engine
  - Exception for micro-occlusions
  - Often authored too dark

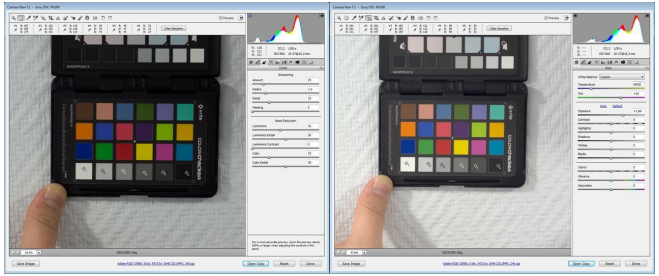
PBR – Artists guidelines

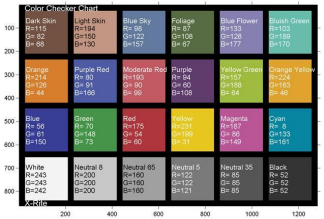
- Reference chart
  - Albedo value is between 32-243 in sRGB
  - Darkest is charcoal
  - Brightest is fresh snow
  - No albedo for pure metal
- Chart is not enough



PBR – Artists guidelines

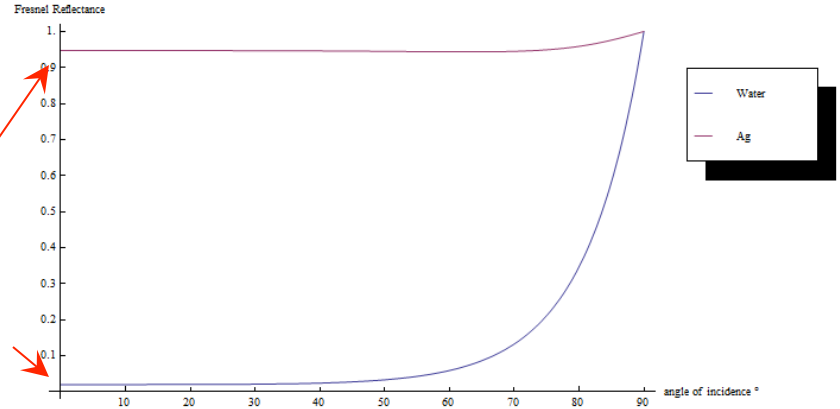
- Acquire reference from real world
  - Based on the work of Henry Labounta [Labounta11]
  - Used as references
  - Time-consuming





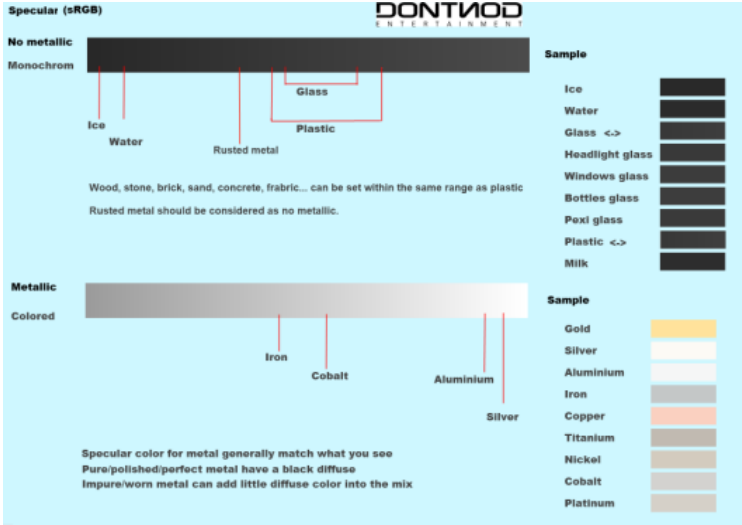
PBR – Artists guidelines

- Specular
  - Input of Fresnel equation
  - Purely physical values: index of refraction
  - Strict name: “Fresnel reflectance at normal incidence for air-surface interface”
- Two cases for game
  - Metal: high RGB values
  - Non-metal: low grey values



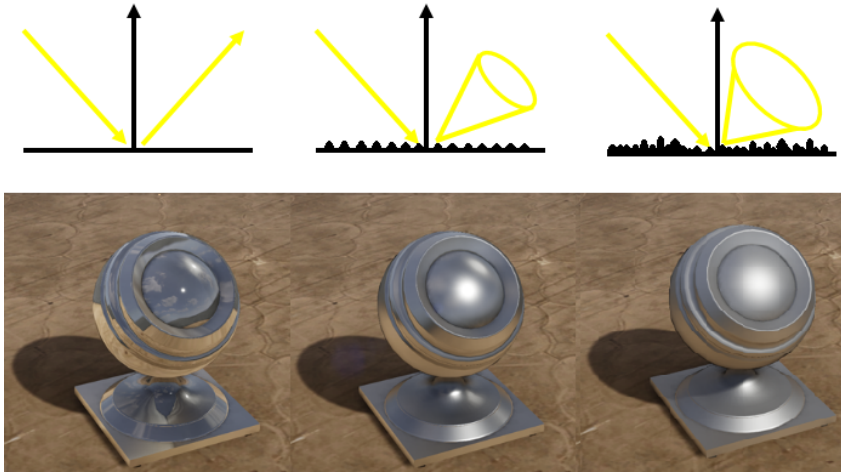
PBR – Artists guidelines

- Specular reference chart (sRGB)
  - Non-metal values are 43-65
    - Unintuitive
  - Metal values are 186-255
    - Characteristic color of metal
- In practice non-metals can be set to default 59
- Chart is available [DONTNOD12]



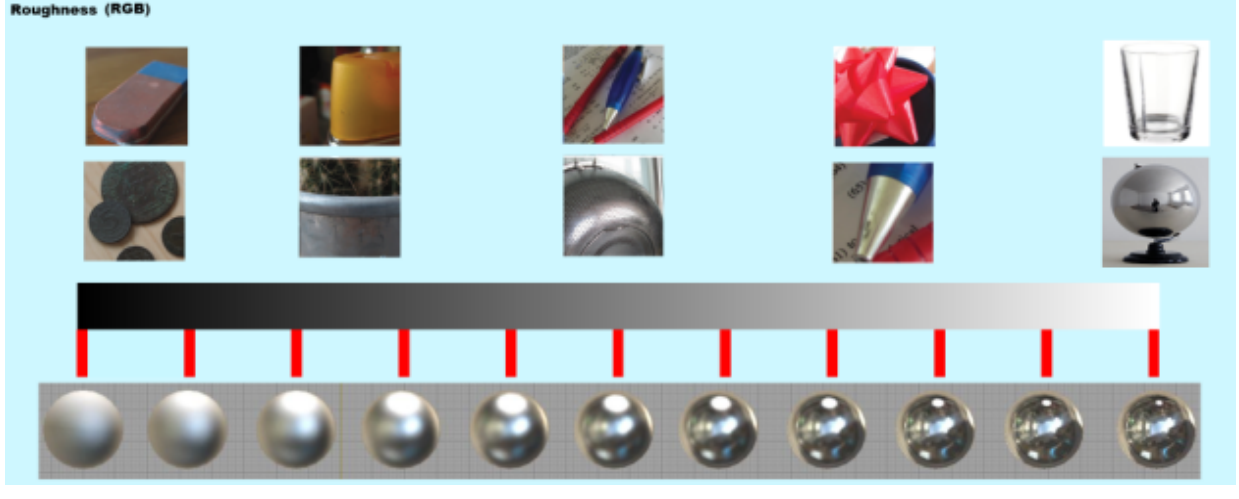
PBR – Artists guidelines

- Smoothness
  - Very important
  - Control strength of reflection blurriness
  - Values are engine -specific
    - Black: rough
    - White: smooth
  - White preferred for smooth
  - Many names: “Roughness, shininess, glossiness”



PBR – Artists guidelines

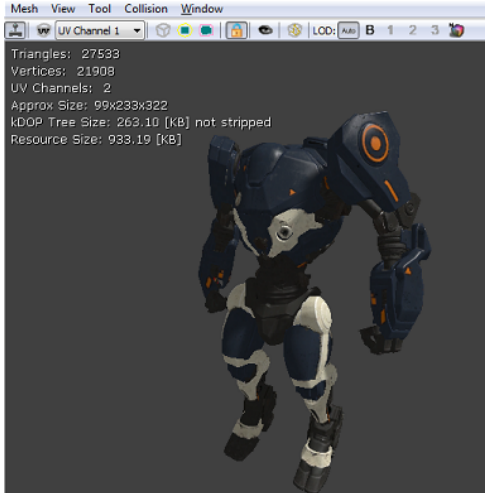
- Smoothness reference chart (grey)
- Used to select mipmap with indirect lighting
- Converted to specular power (2-2048) with direct lighting



Real world pictures courtesy of Andrea Weidlich from "Exploring the potential of layerer BRDF models" Siggraph Asia 2009).

PBR – Artists guidelines

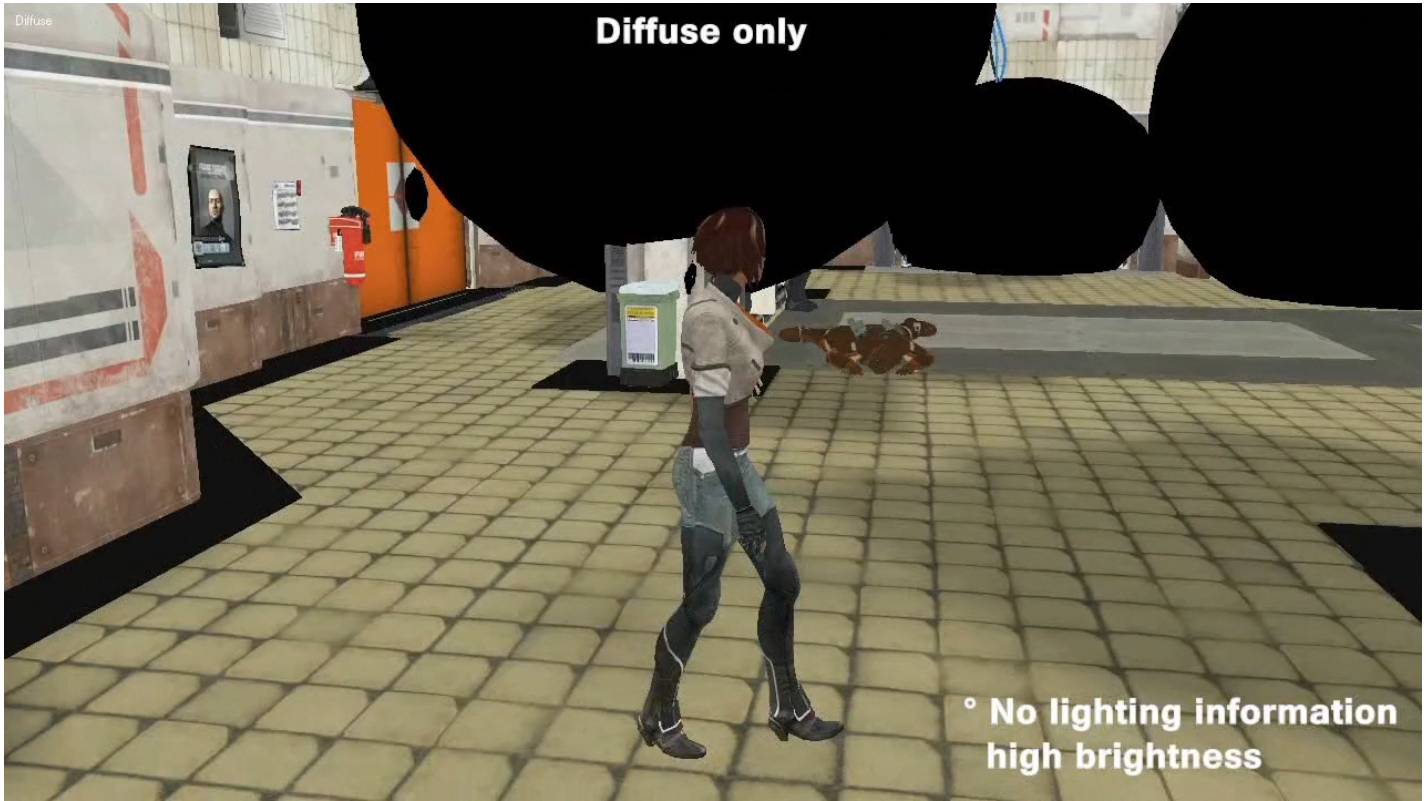
- Required fast visualization tools
  - Test scene
  - Updated UE3 mesh viewer with indirect specular lighting



PBR – Artists guidelines

- Outsourcing
  - Followed the same workflow as our in-house artists
  - Needed a simple viewer
    - Updated shaders in UDK (Nov2011)
  - Art director frequently checked and gave precise feedback

PBR – Artists guidelines



PBR – Artists guidelines

- In practice
  - No real resistance to the switch
  - Large amount of time to get textures right
  - Good results still depend on the artists’ work!
  - Great response to lighting

Development agenda

- Believable world
- Rainy mood
- Reflections
- Ambient occlusion
- Image enhancement

Rainy mood

Physically based wet surfaces

- Wet surfaces with PBR?
  - Old way: boost specular, attenuate diffuse
  - Take real life reference
    - Darker diffuse, brighter specular, hue/saturation changes
    - But not for all surfaces, only porous ones

Physically based wet surfaces

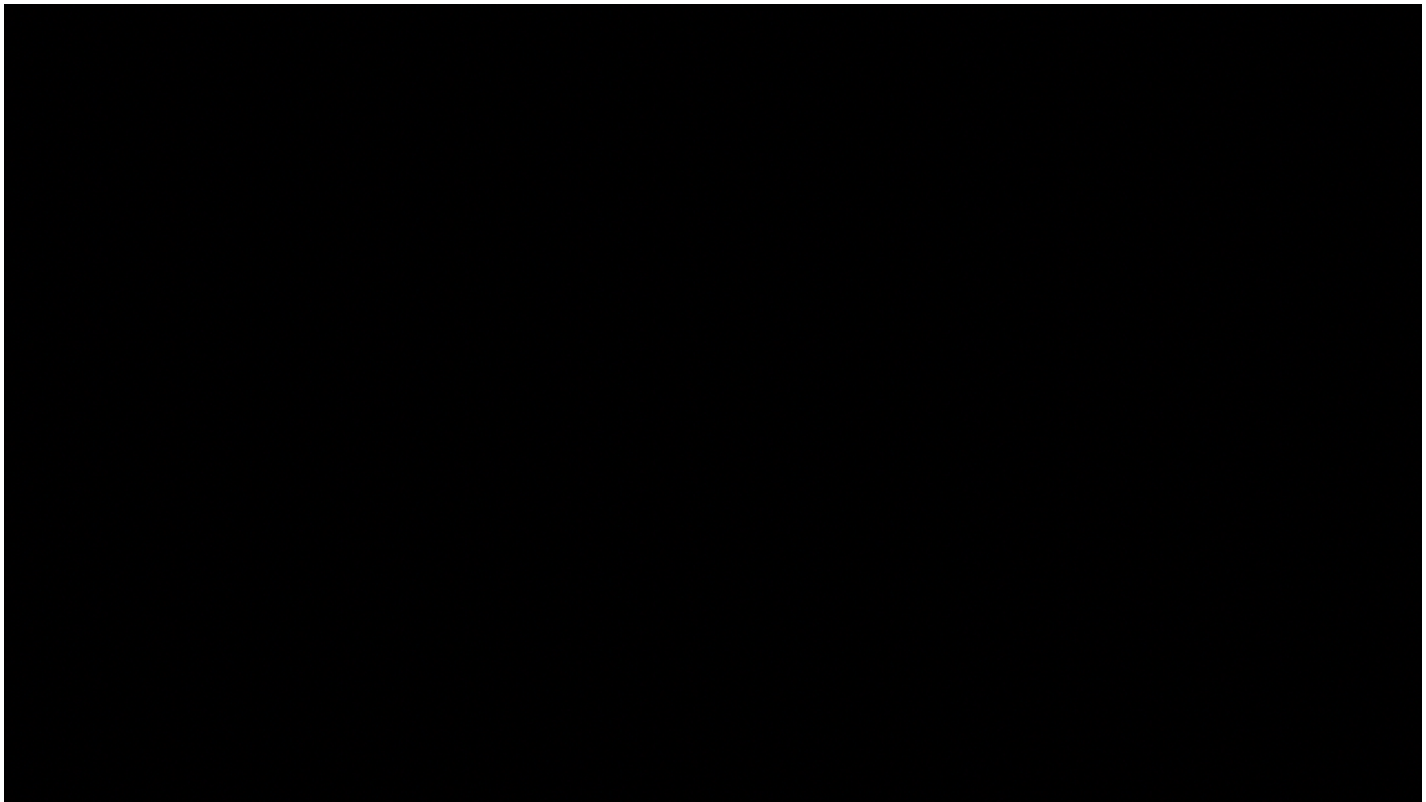
- Shading model way is too complex
- Approximate by adjusting textures
  - Assume rough, non-metallic surfaces are porous
  - Attenuate albedo
  - Increase smoothness
  - Specular doesn’t change

Physically based wet surfaces

- Optimization
  - In the end the game use static rain
  - All wet influences were baked into textures
    - Bonus: allows control from artists
  - Always fit to the context

```
float ProcessWetSurfaces(inout float3 Albedo, inout float3 Smoothness, float WetLevel)
{
    // Determine if we are a metal object (specular > 0.5), non metal (specular < 0.08)
    float Metalness = saturate((dot(specular, 0.33) * 1000 - 500) );
    // Calculate a porosity level based on Smoothness[0 rough, 1 smooth]
    float porosity = saturate(((1-Smoothness) - 0.5)) / 0.4 );
    // Calc albedo attenuation factor
    float factor = lerp(1, 0.2, (1 - Metalness) * Porosity);

    // Water influence on material parameters
    Albedo    *= lerp(1.0, factor, WetLevel); // Attenuate albedo
    // Move Smoothness toward 1 (perfect mirror)
    Smoothness = lerp(1.0, Smoothness, lerp(1, factor, 0.5 * WetLevel)); // 0.5 is an empirical factor
}
```

Rainy mood



Development agenda

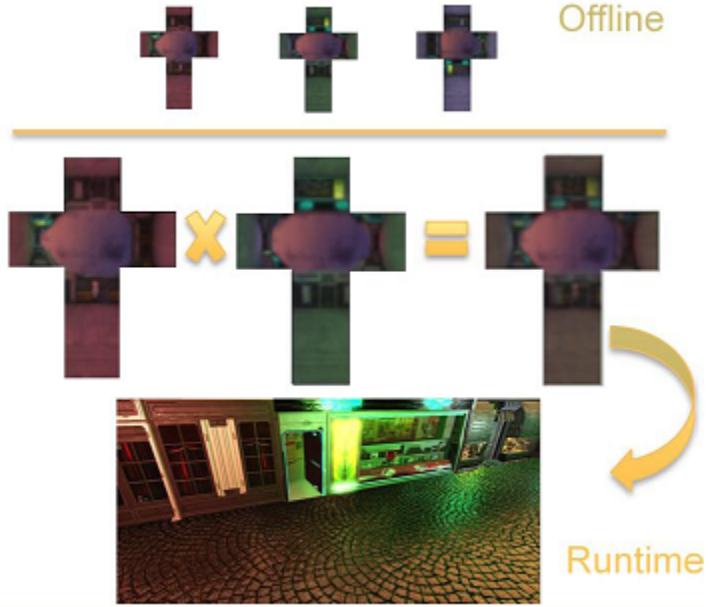
- Believable world
- Rainy mood
- Reflections
- Ambient occlusion
- Image enhancement

Reflections

- "Reflection" term
  - Alias for "indirect specular lighting"
  - Reflection improvement features here
- Reflections everywhere
  - Used artist-placed pre-integrated cubemap on every surface

Reflections

- Local image base lighting
  - Mix into one cubemap
    - Based on camera/player position
  - Apply on every surface
  - Presented at SIGGRAPH 2012 [Lagarde_Zanuttini12]

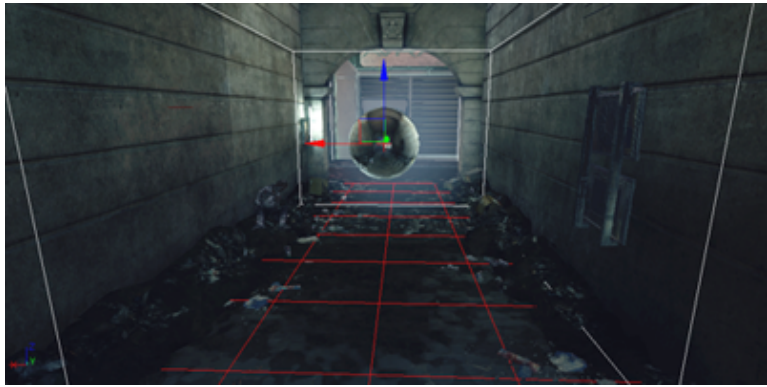


Reflections

- Parallax-corrected cubemap
  - Improves quality
  - Reflection vector adjusted
    - Based on camera location
    - ...and scene approximation
    - Supported boxes and spheres
    - [Bjorke07][Behc10]

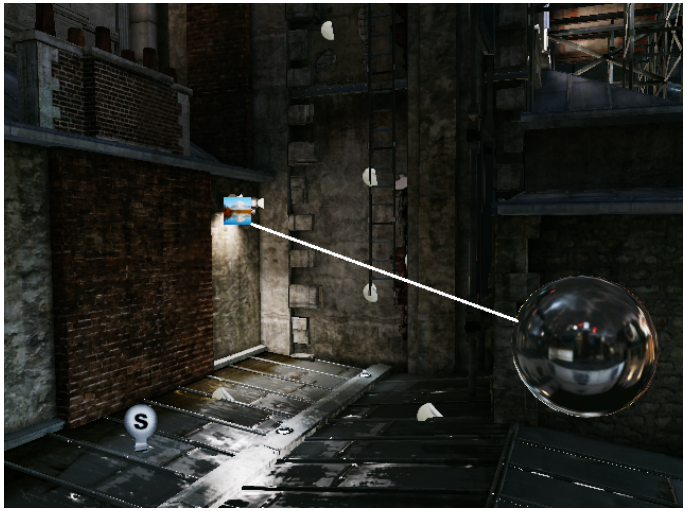
Reflections

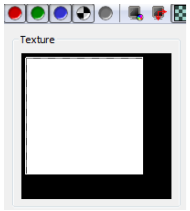
- Improved for the ground
  - Parallax-corrected cubemap
    - Support convex/concave
  - Artists placed ground plane
  - Mix cubemaps into 2D texture
  - Use result like dynamic planar reflection
  - Coarse handling of smoothness
  - See GPU Pro 4 book [Lagarde_Zanuttini13]



Reflections

- Ground image proxies
  - Billboard reflections
    - Similar to particles
  - Enhance a cubemap
  - Generated in editor
  - Authored by hand
  - [Mittring11]
  - [Wiley07]



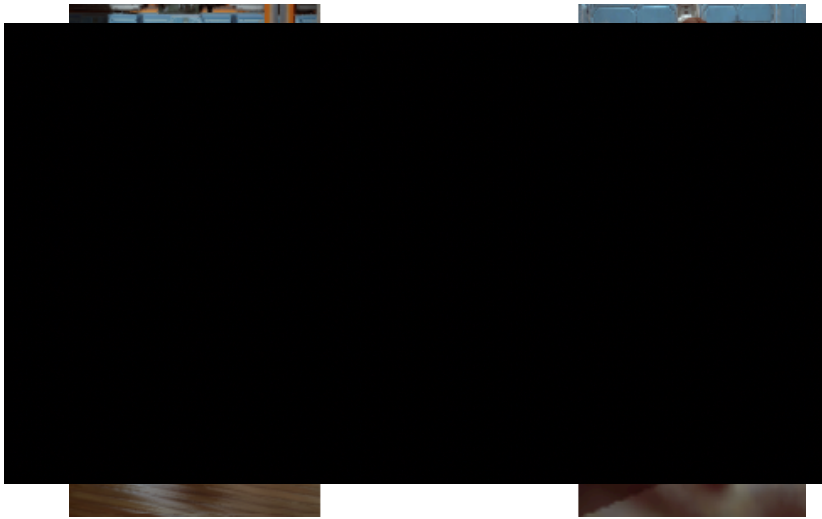


Reflections

Image proxies – best practices

Reflections

- Image proxies can be dynamically linked
- Characters
  - Stretched sphere linked to main bones
  - Don’t look at mirror surfaces!



After the rain

After the rain

- Our engine is not optimized for a lot of dynamic light sources
- Art direction influenced by the technical limitations

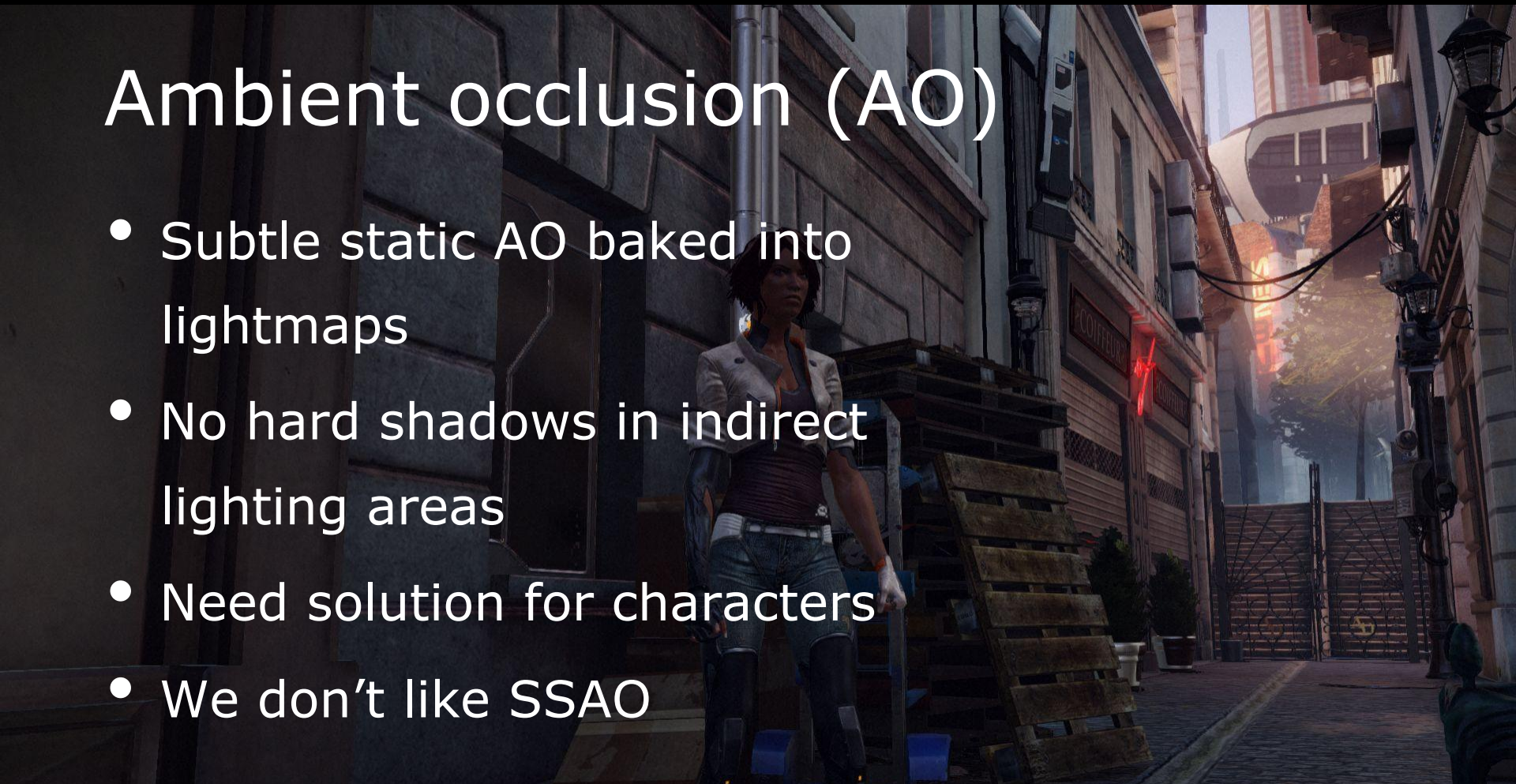

EXIT
PHILATELIE
ACHAT
VENTE
ESTIMATION
M. C. LARRE
13:42
WELCOME TO YOUR SUITE
Boulangerie traditionnelle
Croques au
pain Poisan
Restaurant 8,50
Montagne 8,50
Berger 9,50


Café

Development agenda

- Believable world
- Rainy mood
- Reflections
- Ambient occlusion
- Image enhancement

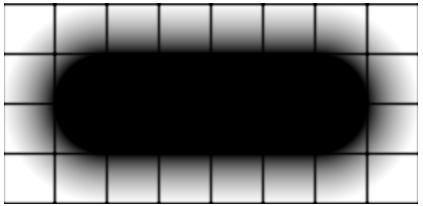
Ambient occlusion (AO)



- Subtle static AO baked into lightmaps
- No hard shadows in indirect lighting areas
- Need solution for characters
- We don’t like SSAO

Ambient occlusion volumes

- Analytic ambient occlusion
  - Volume proxy
  - Based on distance
  - Take horizon into account
- For Characters
  - Capsule linked to main bones
  - Less waste
- Similar to [Hill10]

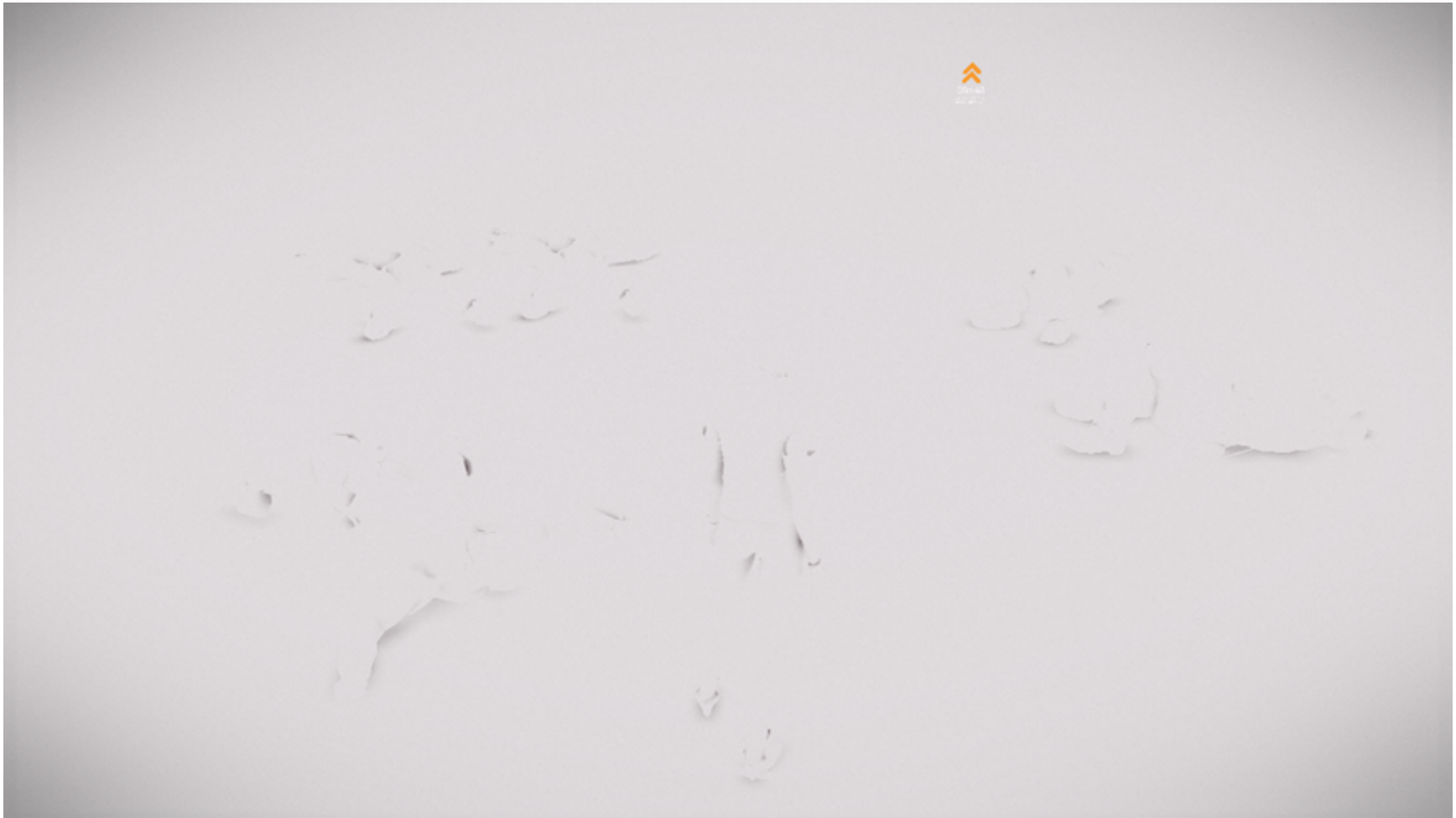


Ambient occlusion volumes

- Splat influence volumes on the screen
  - Capsules extended by influence region
  - Approximated by a box
    - Bounding box + thinnest axis size
- Requires a normal buffer
  - Forward renderer!
  - Render normals in Z prepass
    - Only for objects in contact with characters

Ambient occlusion volumes

- Steps:
  - Render normal + depth during z prepass
  - Splat extended box into AO buffer
  - Apply in main pass on indirect lighting
- Performance
  - Between 0.5 and 1.5ms on PS3
  - Faded out on close-ups to reduce fillrate cost



```
// Get vector from local position to local sphere center. In local sphere space, center is 0,0,0
half3 Vec = MulMatrix(ProxyWorldToLocal, float4(PositionWS.xyz, 1.0));
// Compute vector from point we are shading to center of the sphere
half3 VecWS = normalize(ProxyWorldPosition.xyz - PositionWS.xyz);
half3 NormalWS = tex2Dlod(NormalTexture, float4(ScreenUV,0,0)).xyz * 2.0 - 1.0;

// Cosine of the capsule's angular height above the horizon
half CosAlpha = saturate(dot(NormalWS, VecWS));
// We use a capsule shape which is computed as:
// AOVSize.w = min(size.x, min(size.y, size.z) )
// AOVSize.xyz = size - AOVSize.w
// Using length(dist3D) will result in a smooth capsule
half3 dist3D = max(abs(Vec) - AOVSize.xyz, 0.0f);
half dist = saturate(AOVSize.w / length(dist3D) * 2.0f - 1.0f);

half Occlusion = dist * CosAlpha;
Occlusion = min(Occlusion, 0.95); // Do not allow totally black to avoid removing all color
// .w contains fade factor calculated offline for transitioning to close-up shot
Occlusion *= ProxyWorldPosition.w;
Occlusion = saturate(1.0 - Occlusion); // 1 no occlusion, 0 full occlusion
```

Development agenda

- Believable world
- Rainy mood
- Reflections
- Ambient occlusion
- Image enhancement

Atmospheric tint sphere

Image enhancement

Atmospheric tint sphere

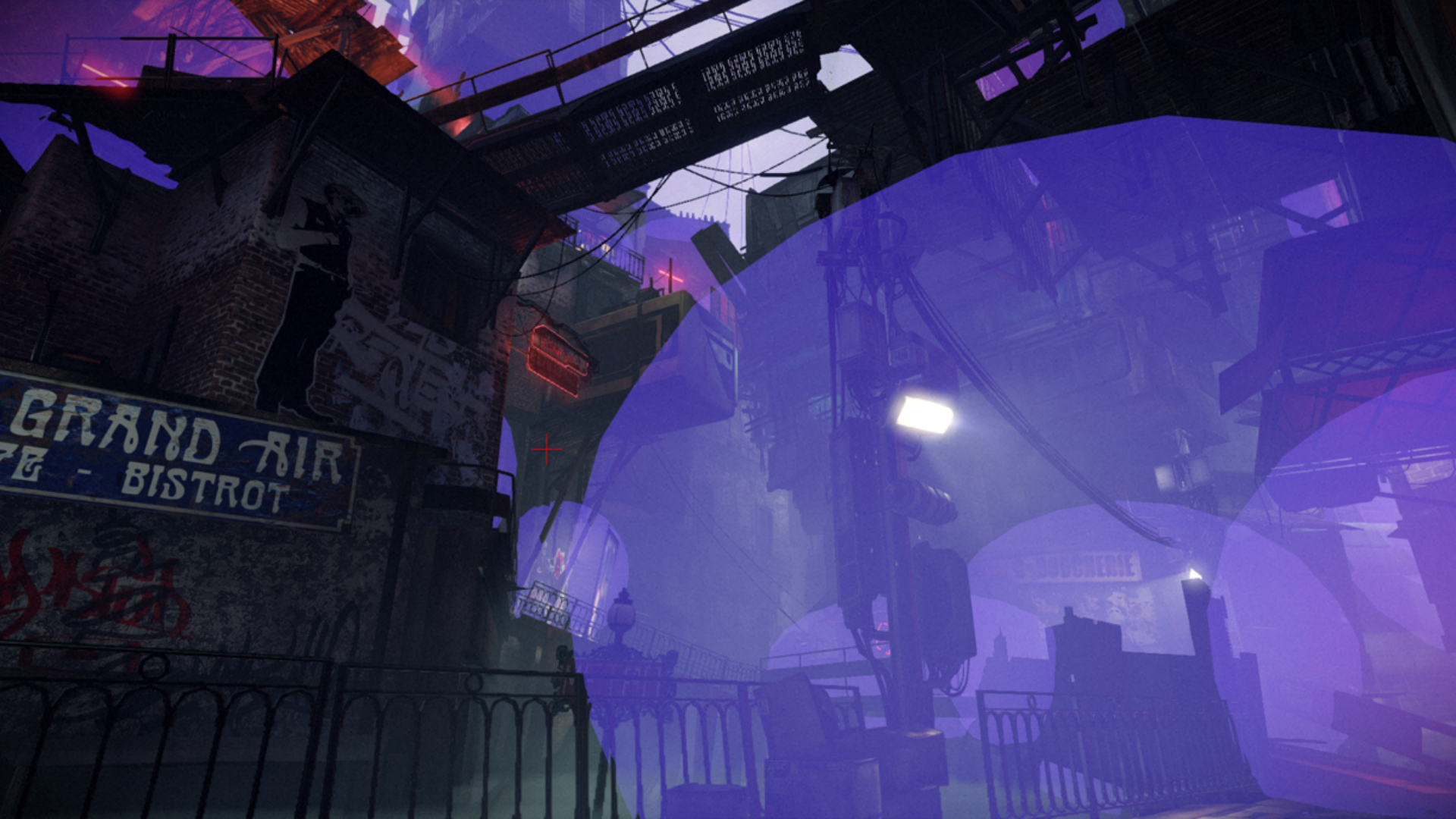
- Enhances
  - Silhouettes, image depth, light glowing, steamy atmosphere
- Cheap additive translucent sphere
  - Cubic attenuation at border

```
float3 c = Intensity * Color.rgb;
float3 result = c * pow(abs(TangentCameraVector.z), 3)
```

  - Optional features


GRAND AIR
BISTROT
BOUCHERIE


GRAND AIR
BISTROT


GRAND AIR
BISTROT


TAX


TAX

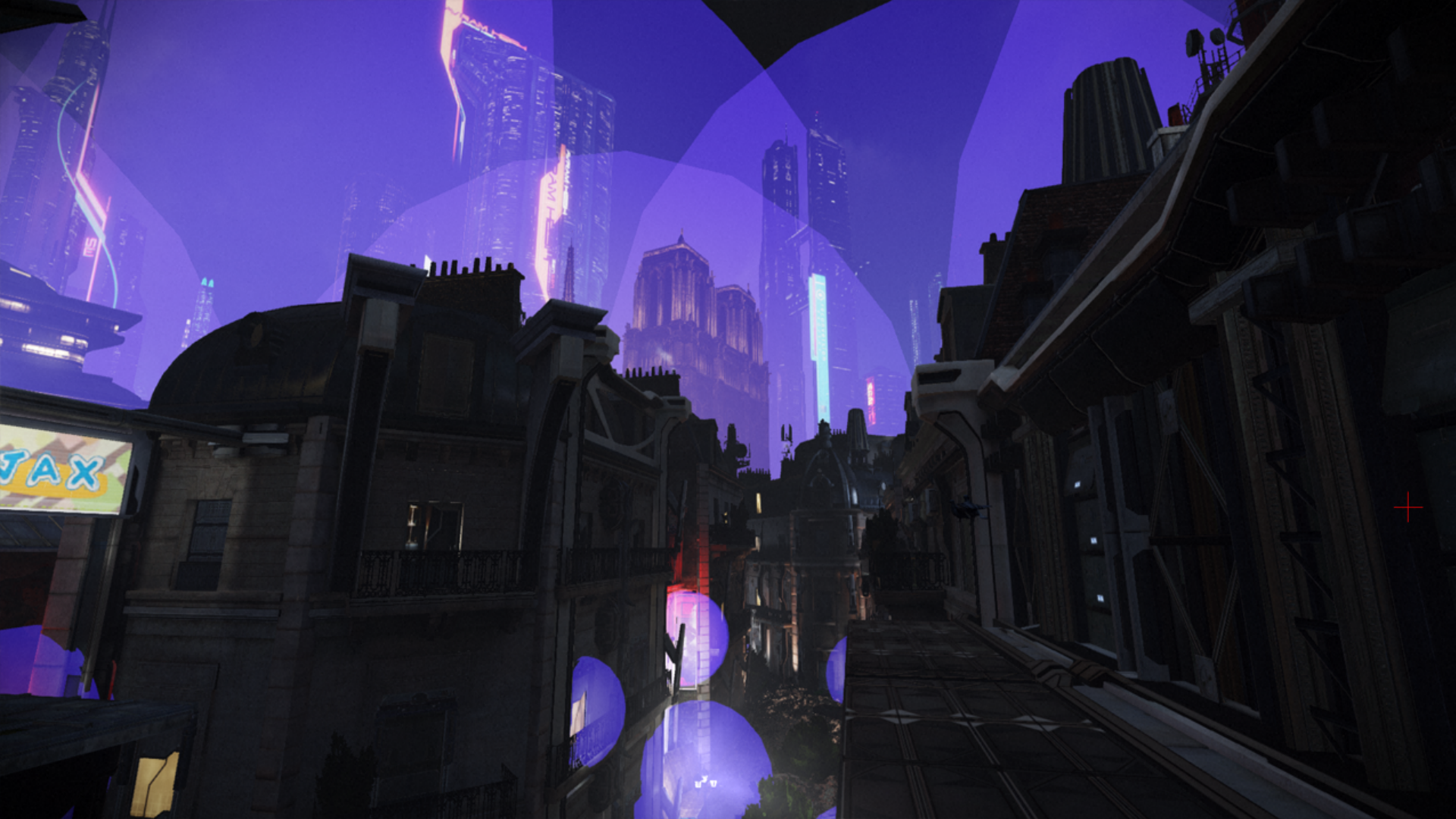

TAX

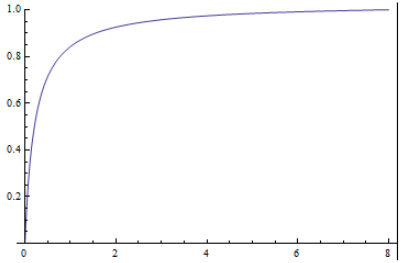
Image Enhancement

- Hand-authored sky in Photoshop
  - Default to LDR
  - Meant low reflection, no bloom
  - Tonemapper changed sky appearance
- HDR sky required with predictable results

Image Enhancement

Inverse tonemapping for sky

- Inverse tonemapping in sky material
  - Luckily the UE3 tonemapper is invertible
  - y=Bx / (x + A) => x=y * A / (B – y)
  - Values above a threshold produce bloom



Invert

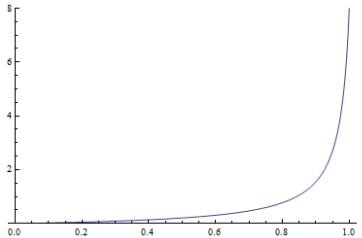


Image Enhancement

Normal sky

Inverse tonemapped sky

```
// UE3 default tonemapper range is [0;8] resulting in following code
float3 ToneMap(float3 color)
{
    return 1.0275 * color / (color + 0.22);
}

// When editing the image in Photoshop, a pure white (1) in the game is obtained with the value 214
// (0.84). Any value above 214 will bloom & the maximum in-game value (8) is reached for the value 255
float3 InverseToneMapping(float3 color)
{
    return (0.22 * color / (1.0275 - color));
}
```

Conception :

Key concept

Neo-Paris

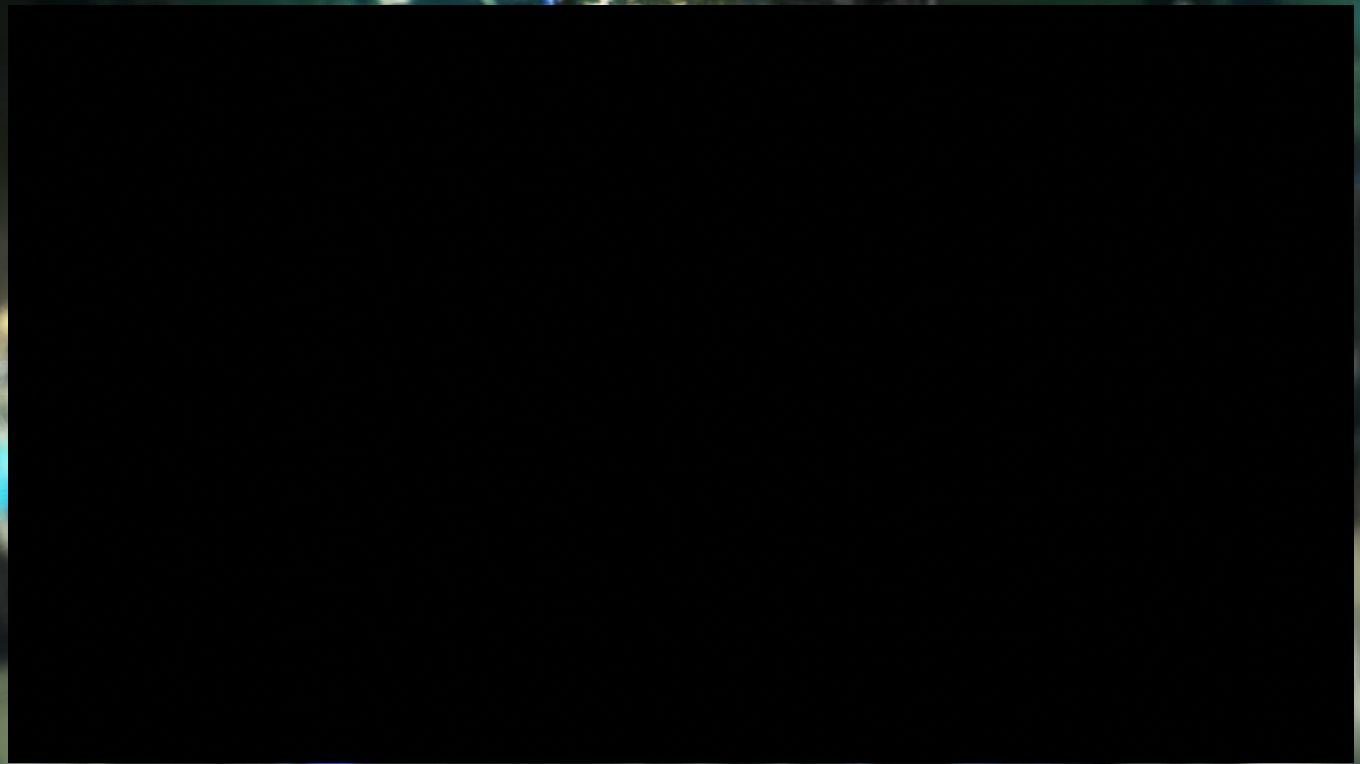
Virtual Spaces

Virtual Spaces

- Human mind seen through the Sensen technology

- Two types of levels

Virtual Spaces



Virtual Spaces in the Sensen

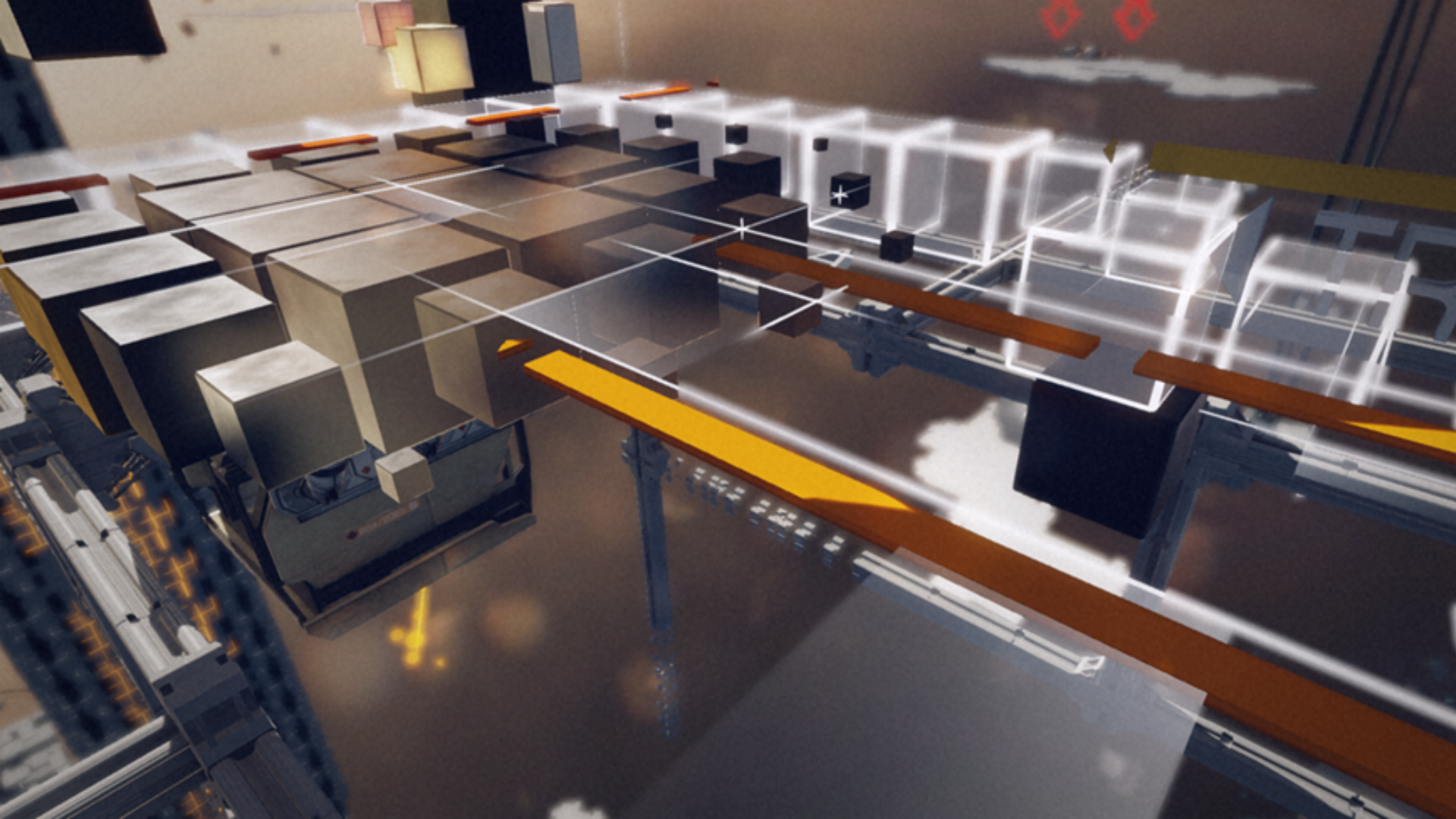
- Art style: half organic & half technological
  - Blurry space
  - Ghosting effects
  - Fade in effects
  - Glitches

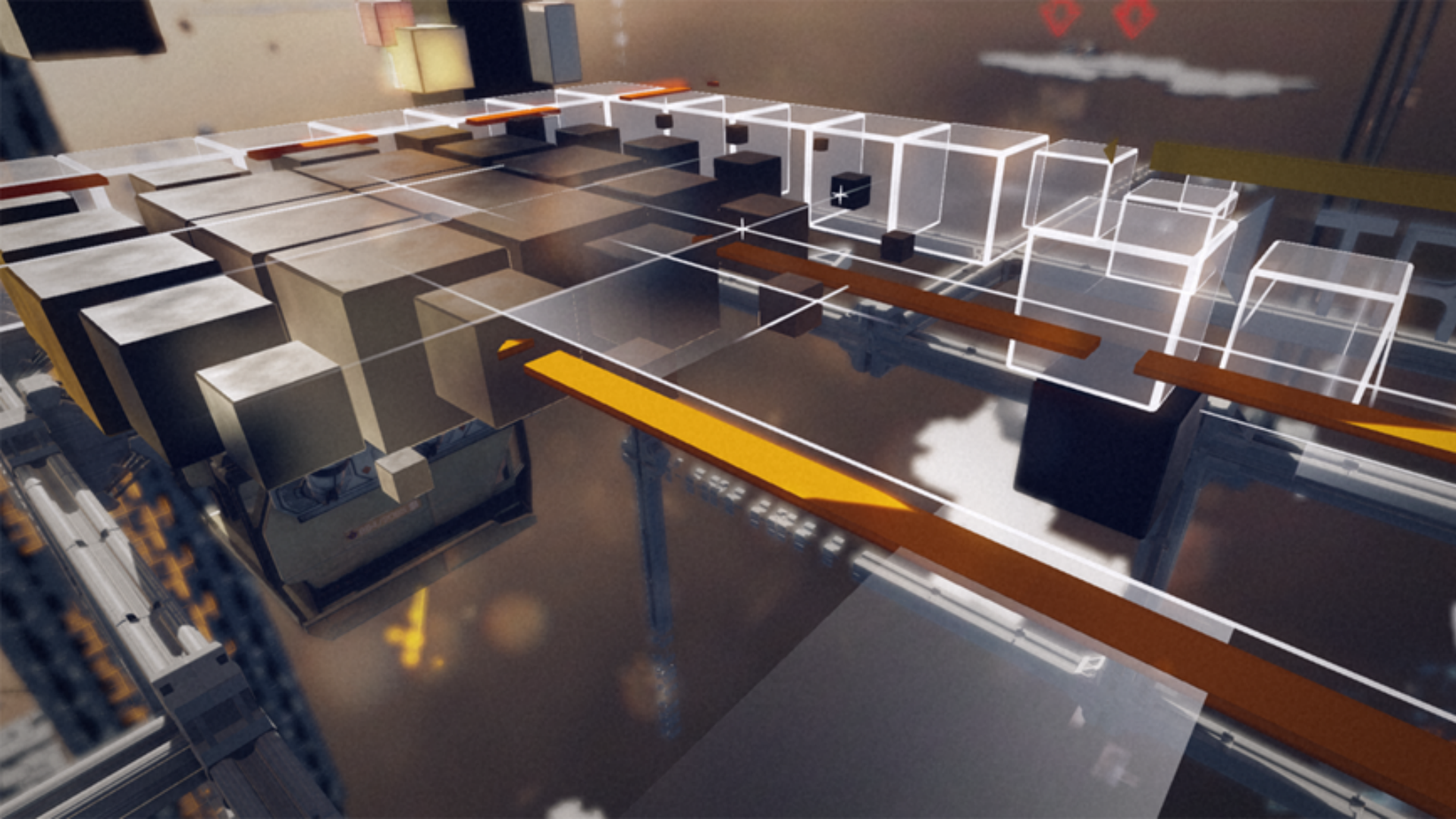
Depth of Field with translucency

- UE3 DOF is deferred
  - Wrong DOF for translucent objects
- Tried many “low cost” methods
  - Performance mattered
  - Nothing gave the quality we were looking for
- Want in-focus objects on top of out-of-focus background

Depth of Field with translucency

- Brute force solution
- Render an extra DOF pass
  - Render translucent objects behind of focal
  - Blur with stencil masked far pixel
  - Render translucent objects in front of focal
- 1.6 to 2 ms extra cost (PS3)
- Limited to memory remix and ego room






MNEMOPOLIS

Thanks!

- Frédéric Cros (Lead Lighting artist), Michel Koch (Art Director), Antoine Zanuttini, Laury Michel (Graphic programmers)
- DONTNOD and EPIC Team
- Naty Hoffman, Yoshiharu Gotanda
- Reviewers: Stéphane Hubart, Sam Hocevar, Cyril Jover, Jérôme Banal, Timothée Letourneux, Thomas Iché, Daniel Wright
- Special thanks : Stephen Hill, Brian Karis

Q&A

- Sébastien Lagarde
  - Email: Lagardese@hotmail.fr
  - Twitter : @SebLagarde
- Laurent Harduin
  - Email: harduin.laurent@gmail.com
  - Twitter : @lharduin

- Slides availables at

  http://seblagarde.wordpress.com

Q&A

- Sébastien Lagarde
  - Email: Lagardese@hotmail.fr
  - Twitter : @SebLagarde
- Laurent Harduin
  - Email: harduin.laurent@gmail.com
  - Twitter : @lharduin

- Slides availables at

  http://seblagarde.wordpress.com

References

Some of the screenshots of this talk are extract from the Dead End Thrills' Remember Me gallery. (Slides 9, 16, 30, and 123).

http://deadendthrills.com/gallery/?gid=3

@deadendthrills

References

[Behc10] "Box projected cubemap environment mapping"
http://www.gamedev.net/topic/568829-box-projected-cubemap-environment-mapping/
[Bjorke07] "Image based lighting"
http://http.developer.nvidia.com/GPUGems/gpugems_ch19.html
[DONTNOD12] DONTNOD specular and glossiness chart
http://seblagarde.wordpress.com/2012/04/30/dontnod-specular-and-glossiness-chart
[Gotanda11] "Real-time Physically Based Rendering – Implementation"
http://research.tri-ace.com/
[Hill10] "Rendering with Conviction",
http://www.selfshadow.com/talks/rwc_gdc2010_v1.pdf
[Hoffman10] "Crafting Physically Motivated Shading Models for Game Development"
http://renderwonk.com/publications/s2010-shading-course/

References

[Labounta11] GDC 2011 "Art Direction Tools for Photo Real Games"
Not available

[Lagarde_Zanuttini12] "Local Image-based Lighting With Parallax-corrected Cubemap"
http://seblagarde.wordpress.com/2012/11/28/siggraph-2012-talk/

[Lagarde_Zanuttini13] "GPU Pro 4 – Practical planar reflections using cubemaps and image proxies"
http://seblagarde.wordpress.com/2013/05/07/gpu-pro-4-practical-planar-reflections-using-cubemaps-and-image-proxies/

[MCube12] "AMD Cubemagen for physically based rendering",
http://seblagarde.wordpress.com/2012/06/10/amd-cubemapgen-for-physically-based-rendering/

References

[Mittring11] “The Technology Behind the DirectX 11 Unreal Engine "Samaritan" Demo”, http://udn.epicgames.com/Three/rsrc/Three/DirectX11Rendering/MartinM_GDC11_DX11_presentation.pdf

[Persson12] “Graphics Gems for Games – Findings from Avalanche Studios”, http://www.humus.name/index.php?page=Articles

[Wiley07] “The Art and Technology of Whiteout”, http://developer.amd.com/resources/documentation-articles/conference-presentations/gpu-technology-papers/

Bonus slides

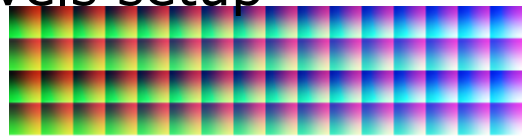
Image Enhancement

Color grading

- Intensify stylization
- Reinforce the mood
- Fake eye adaptation

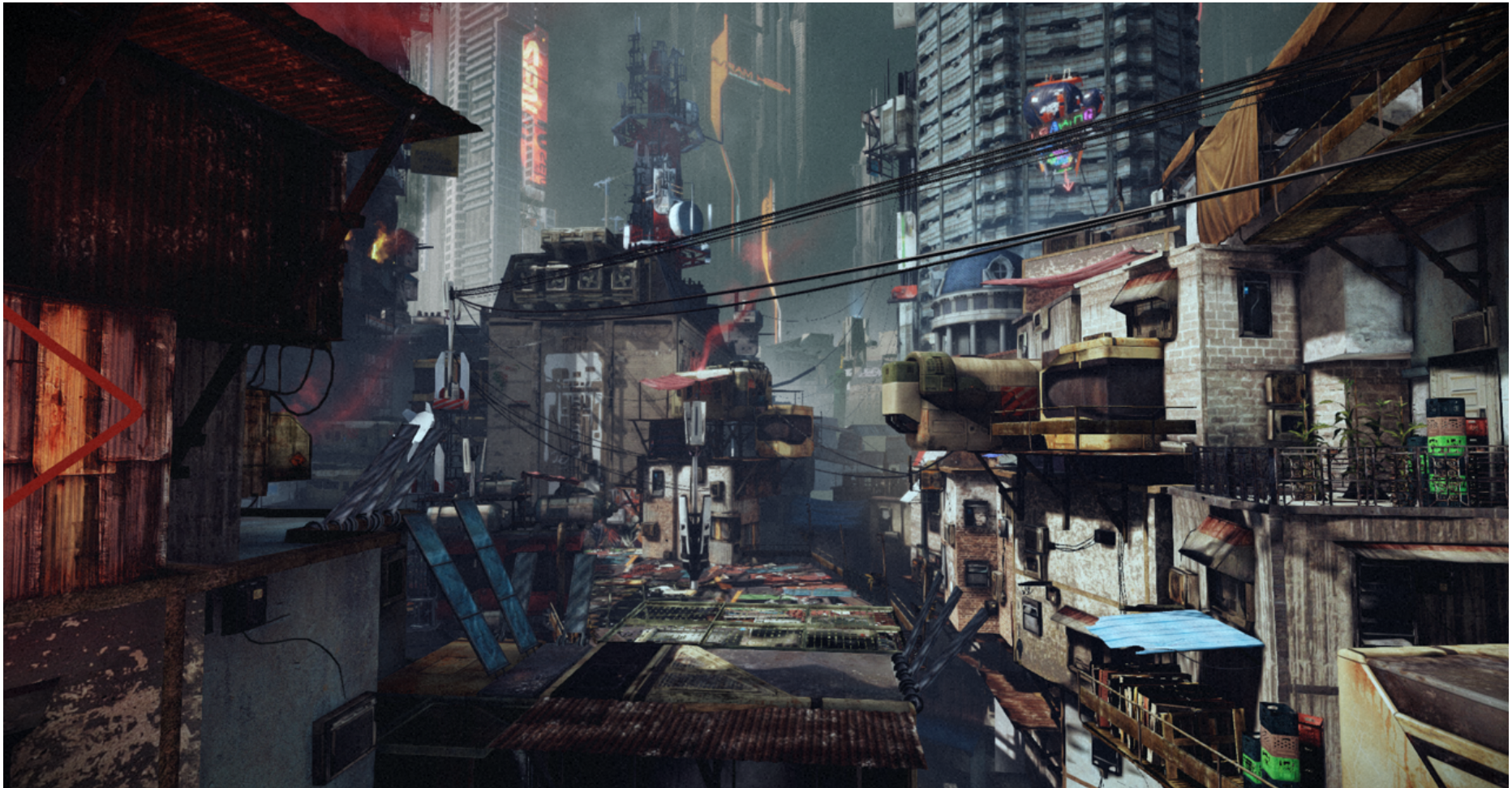
Depth color grading

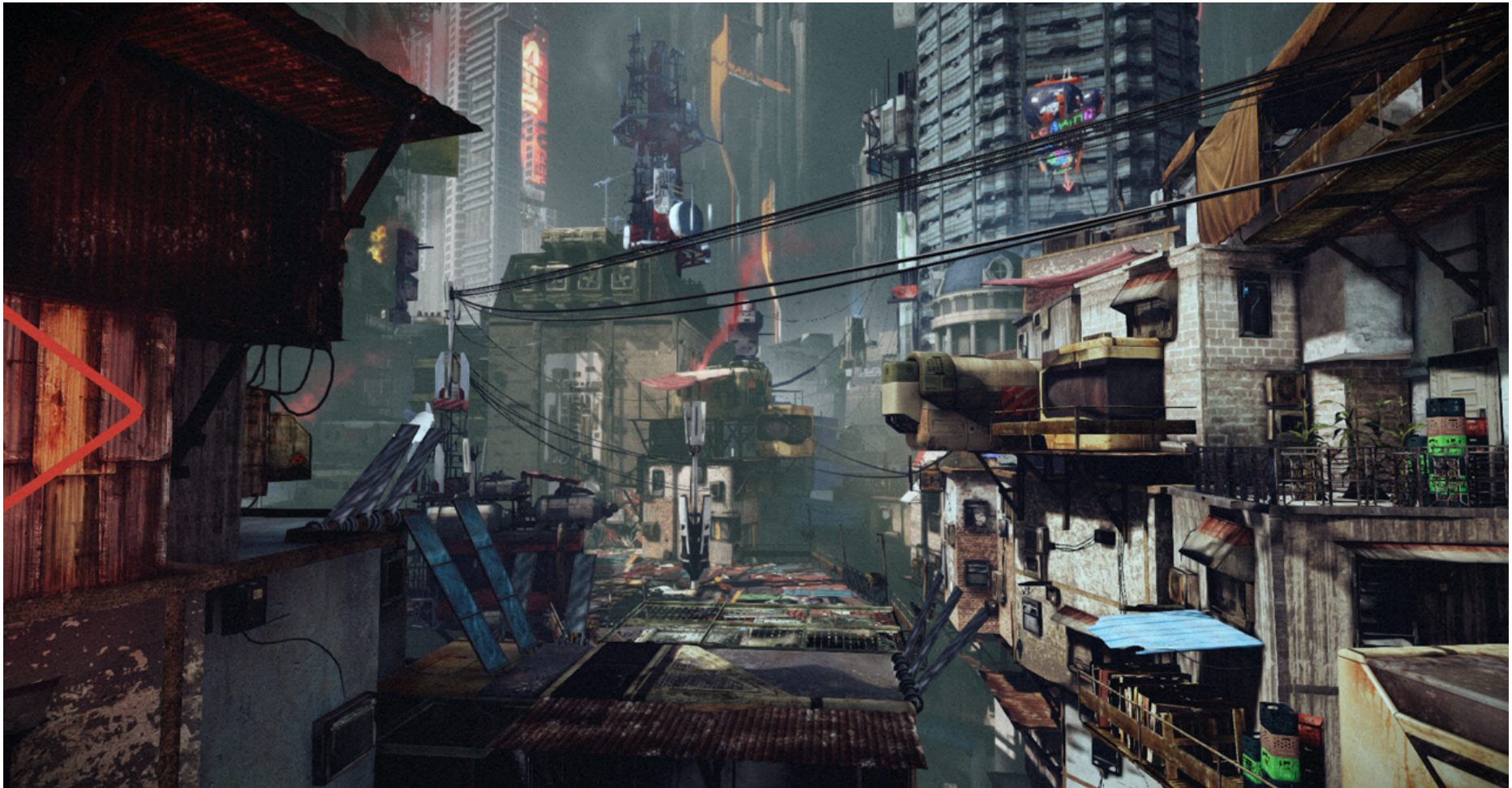
- Color grading varying with depth
- Up to 4 different levels
  - Remember Me use only one level
    - Too much time required for multiple levels setup
- 3D textures 16x16x64
- Manual interpolation in the shaders
- Add ~0.6 ms (PS3) compare to one level



```
sampler3D ColorGradingLUT;

half4 LookUpInRange(half3 InLDRColor, half IntZ)
{
    half3 UVW;
    UVW.xy = InLDRColor.xy * 15.0f / 16.0f
           + 0.5f / 16.0f;
    UVW.z = InLDRColor.z * 15.0f / 64.0f
           + 0.5f / 64.0f + IntZ * 16.0f / 64.0f;
    return tex3D(ColorGradingLUT, UVW);
}

half4 DepthTransition;
half4 DepthDistances;
```

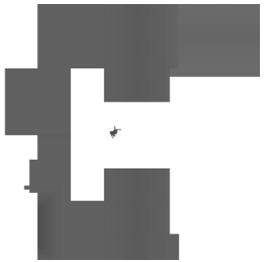
```
half4 ColorLookupTableDepth(half3 InLDRColor, half Depth)
{
    half4 DepthDistancesNear = half4(0, DepthDistances.xyz);
    half4 DepthDistancesFar = DepthDistances;
    // Get distance weights from each layer
    half4 near = half4(Depth >= DepthDistancesNear);
    half4 far  = half4(Depth < DepthDistancesFar);
    half4 weights = near * far;
    half3 Layers = half3(1, 2, 3);
    half IntZ = dot(weights.yzw, Layers);
    half3 Fraction = saturate(Depth * DepthTransition.w -
                     DepthTransition.xyz);
    half FracZ = dot(weights.xyz, Fraction);

    half4 ret = LookUpInRange(InLDRColor, IntZ);
    half4 RG1 = LookUpInRange(InLDRColor, IntZ + 1.0f);
    ret = lerp(ret, RG1, FracZ);
    return ret;
}
```





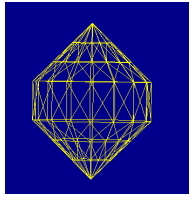
Rainy mood

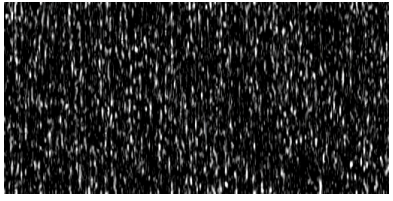
- Splashes on the background
  - Top down depth map from location above the camera
    - Read back on CPU
    - Splashes spawning location recover from depth
    - Only render tagged objects
  - ~0.5ms GPU PS3/XBOX360
    - (Splashes + depth map 256x256)

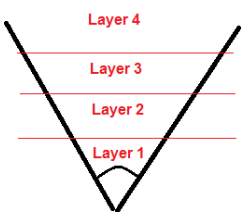


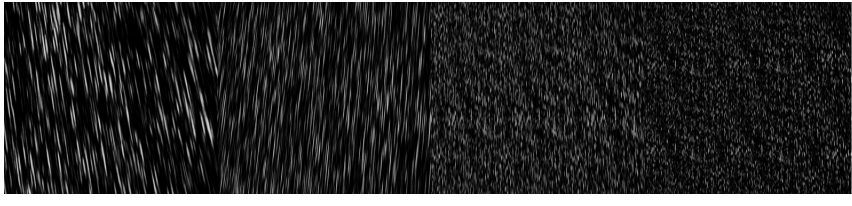
Rainy mood

- Rain
  - Cylinder with two cone caps
    - Centered around the camera
  - Only one rain texture
  - Mapped on 4 virtual layers
    - Parallax effect
    - Different scrolling/rotation







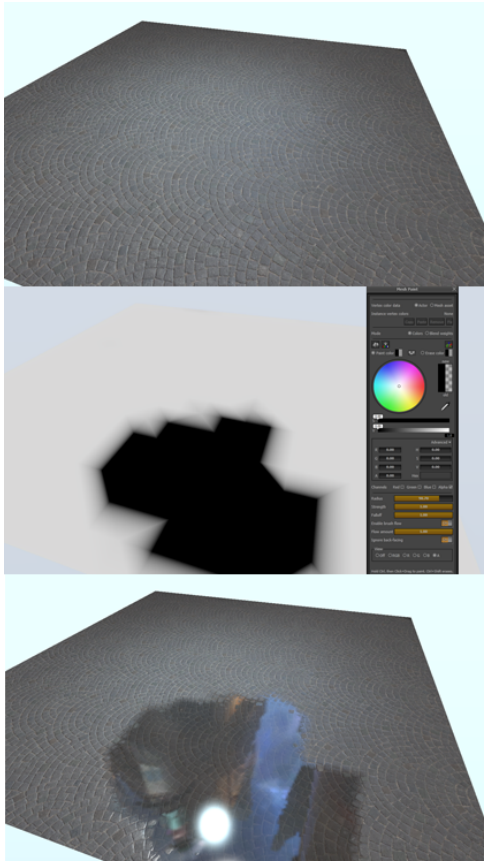


Rainy mood

- Rain
  - Drops have different intensities in rain texture
    - Hide drops based on threshold
  - Drops occluded by depth map
    - Restricted to 2 first layers for performance
    - Level designer could disable other layers
  - Heavily optimized
    - ~1.7 ms PS3/XBOX360
    - Fixed cost

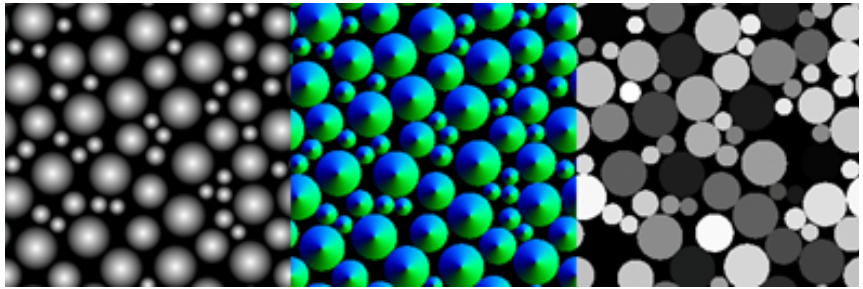
Rainy mood

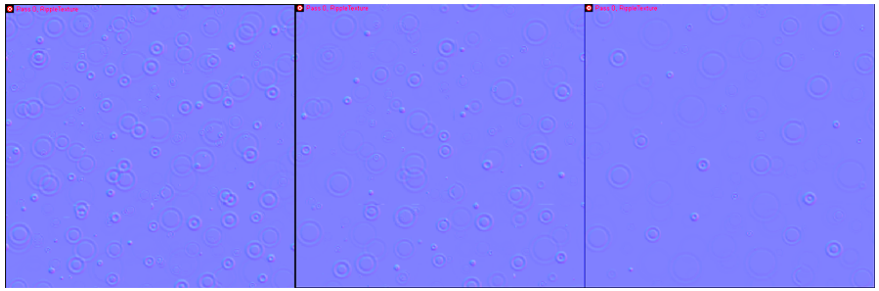
- Puddles
  - Paint vertex color to define depth
  - Based on wet surface code
    - Puddles are mirror-like
  - Vertex color fetch for footsteps
    - Sound
    - Splash FX

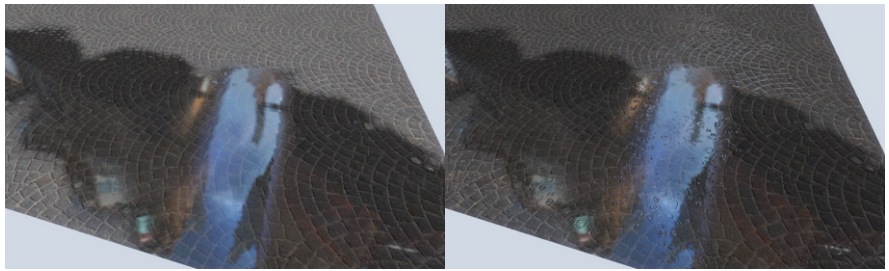


Rainy mood

- Ripples
  - Only where enough water
  - GPU semi-procedural
    - One RGBA texture
  - Mapped with world XY coordinates
  - ~0.15ms PS3/XBOX360 256x256

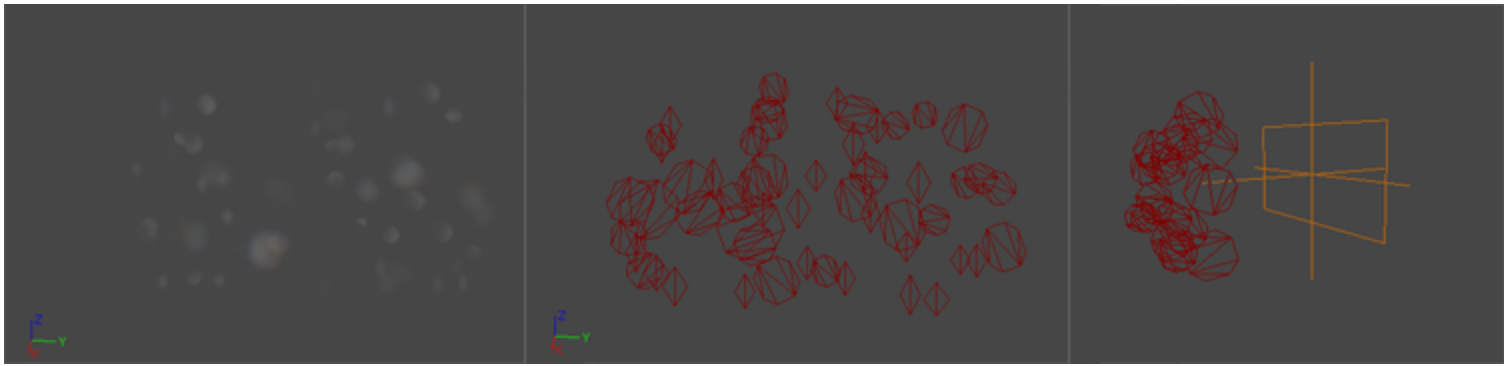






Rainy mood

- Camera droplets
  - View space particles
    - With particle trimming [Persson12]
  - ~0.4 PS3/XBOX360



Remember Me

- Published by Capcom
- Midsized team with little outsourcing
- Based on Unreal Engine 3 (UE3)
  - With in-house features ☺
- PS3 Lead platform
  - XBOX360 just works
- PC port by external company (Qloc)

Remember Me shading model

- Indirect diffuse lighting
- Pre-integrated off
  - Stored as Directional lightmaps
  - Or as Spherical harmonics
- Available in base UE3

$$L_{diffo}(v) = albedo \int_{\Omega} \frac{1}{\pi} L_i(l)(l.n) d_{\omega_i}$$