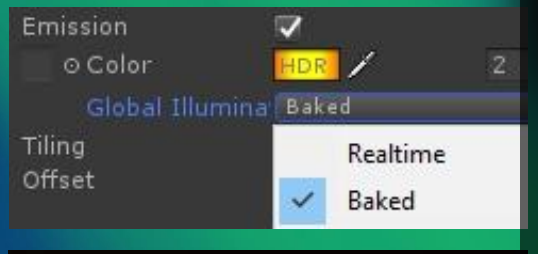
# Light types

- SpotLight / PointLight
- Directional light
  - Can be linked to procedural Skybox (default scene behavior)
- Area light
  - Only in Baked
  - Light emitted in all directions uniformly across their surface area, from one side

# Light types

- Emissive
  - Emission is a StandardShader property
  - Only in PrecomputedGI/Baked (StandardShader flag)
  - Emission will be received only by lightmap static objects
- Ambient
  - Lighting window settings

- Area & Emissive light intensity will diminish at inverse square of the distance
- IndirectMultiplier
  - 0 no indirect light
  - <1 ray light intensity decrease every bounce
  - >1 ray light intensity doesn't decrease



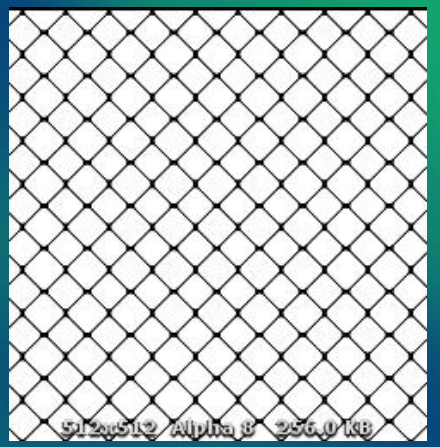

# Cookies

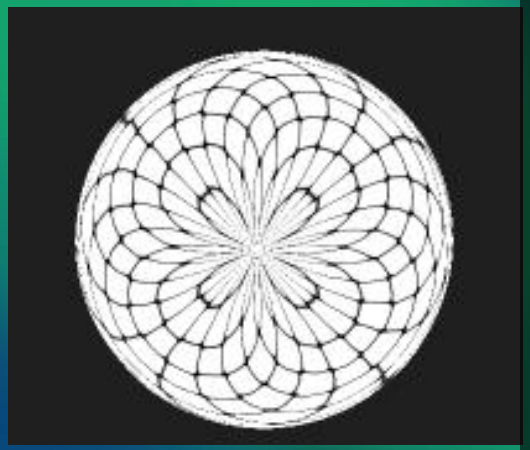
- Environment effects (Cinema, Theatre)

1. Set texture as Cookie Asset
2. Specify for what kind of light and what kind of mapping do you prefer
    1. MirroredBall / 6 Frames / LatLong / Auto
3. Set Appropriate WrapMode

Used for

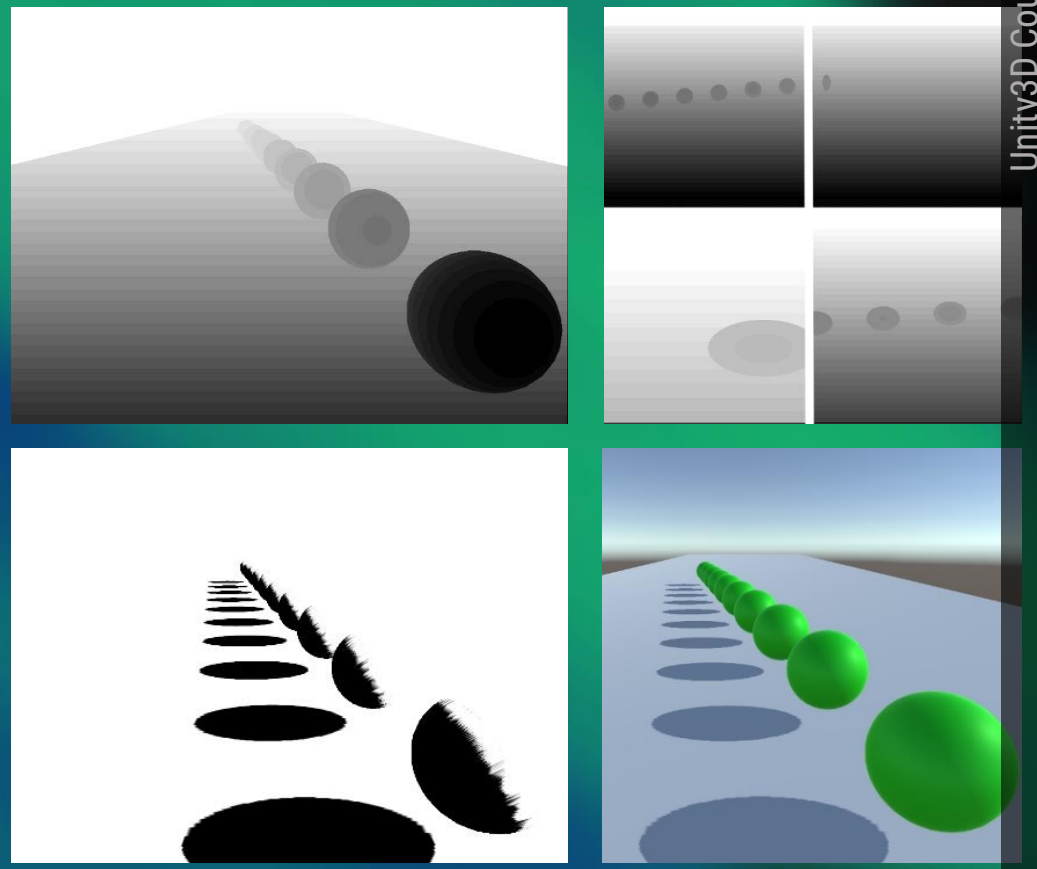
- Change the shape of a light.
    - A dark tunnel with striplights along the ceiling
    - Monitor screen glow should be restricted to a small box shape
- Can also incorporate grayscale levels. Useful for simulating dust in the path of the light
    - Torch light glass usually contains ridges that create caustic patterns





# Shadow mapping

1. Render the entire scene, but only the depth information of each fragment > screen resolution texture output
2. Values in the 0–1 range (clip space, MVP Matrix)
3. For each light, render the scene from the light source POV (again only the depth information of each fragment) - Directional light is orthographic
4. If we are using CSM, the scene is rendered N times per light
5. Make the comparison between camera depth buffer and light depth buffer
    1. set Lit texels are set 1, and shadowed texels to 0



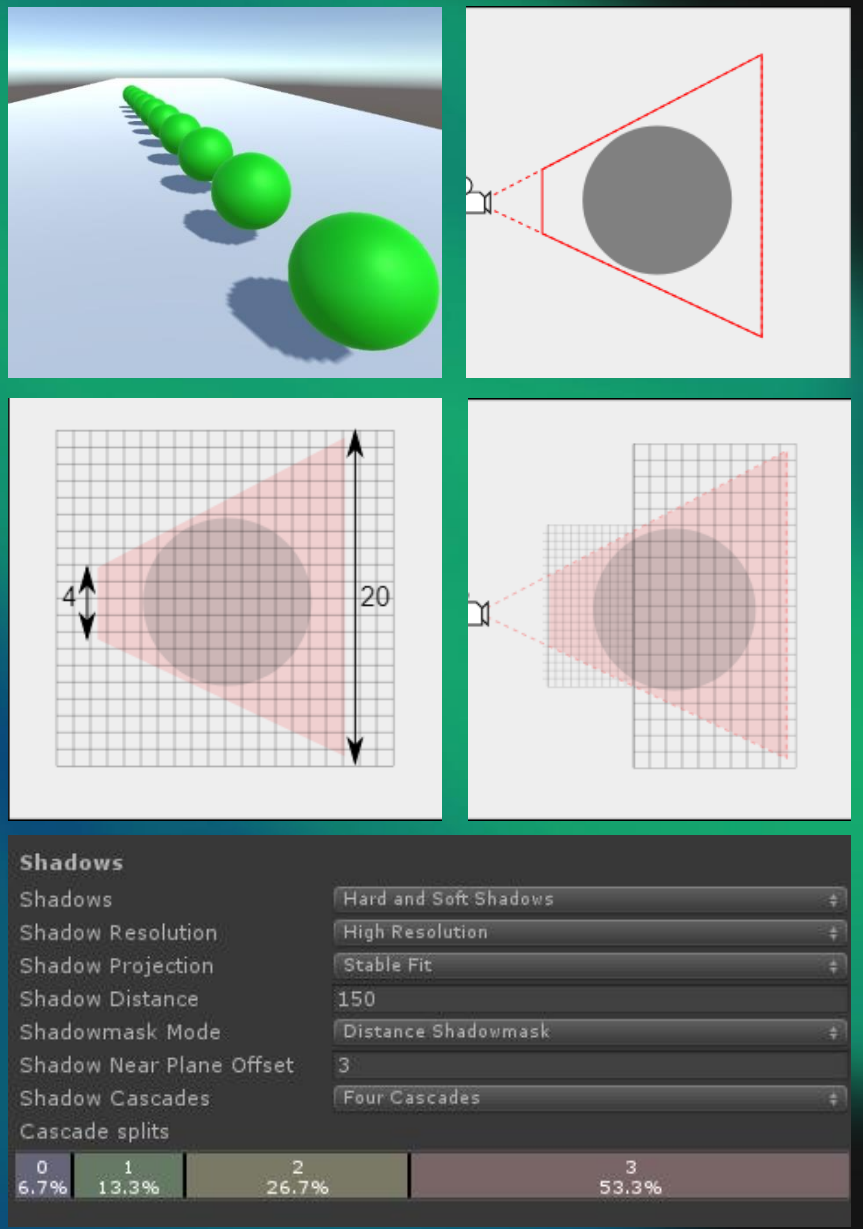
# Shadow Quality

Perspective aliasing SM pixels seen close to the camera look enlarged compared to those farther away

- Use a higher resolution for the whole map reduce the problem, but uses more memory
- The zone near the camera can use a separate shadow map with the same resolution – CSM
  - Render the scene from the light POV multiple times is better than using a very high resolution for the SM texture

Good Rule

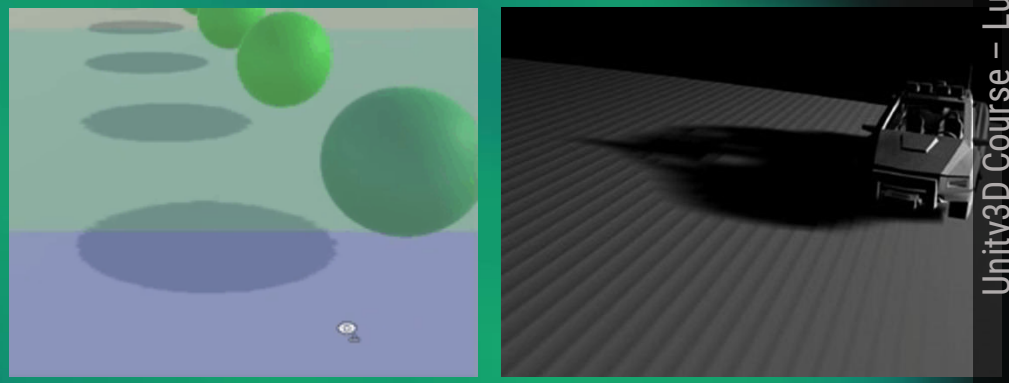
- Keep ShadowDistance as low as possible
  - Shadows that are far away often don't increase image quality
  - We can add Fog to the scene

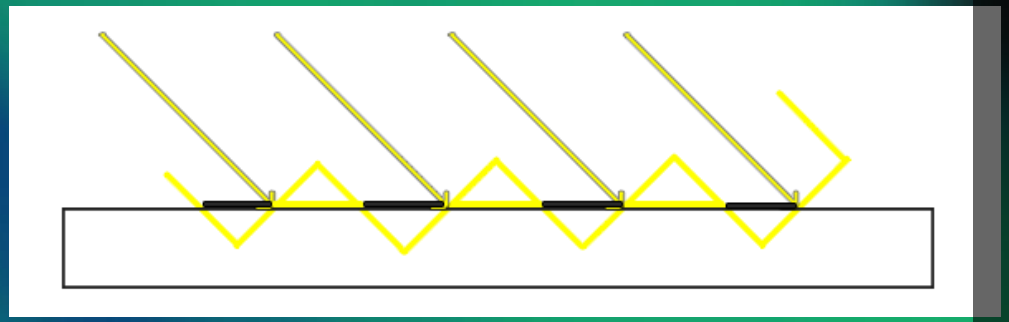


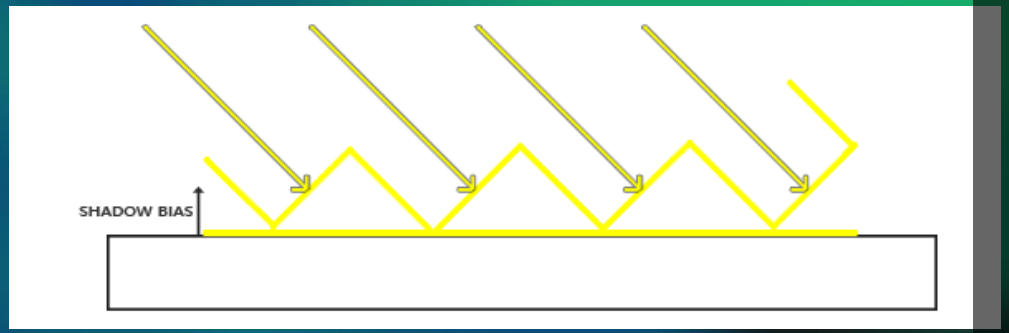
# Shadow Quality

- ShadowProjection (SceneView/ShadowCascades)
  - StableFit Coose CSM band based on Cam depth buffer
  - CloseFit Choose CSM band based on Cam pos distance

- Bias Reduces the Shadow Acne Problem
- NormalBias Reduces Self shadowing Problem



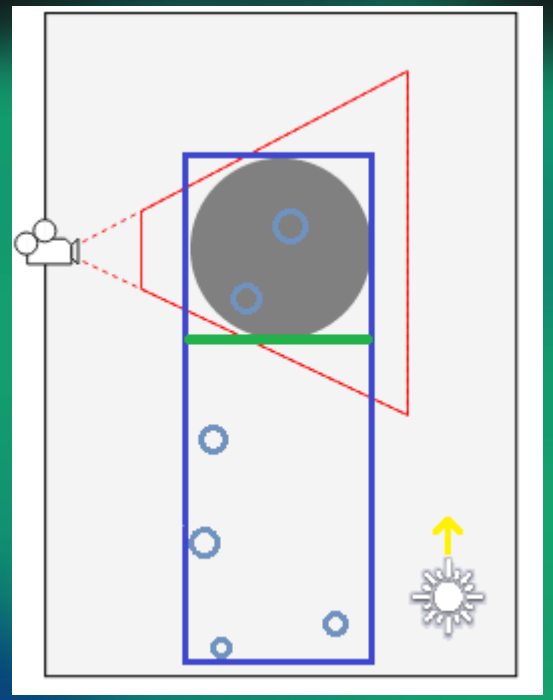


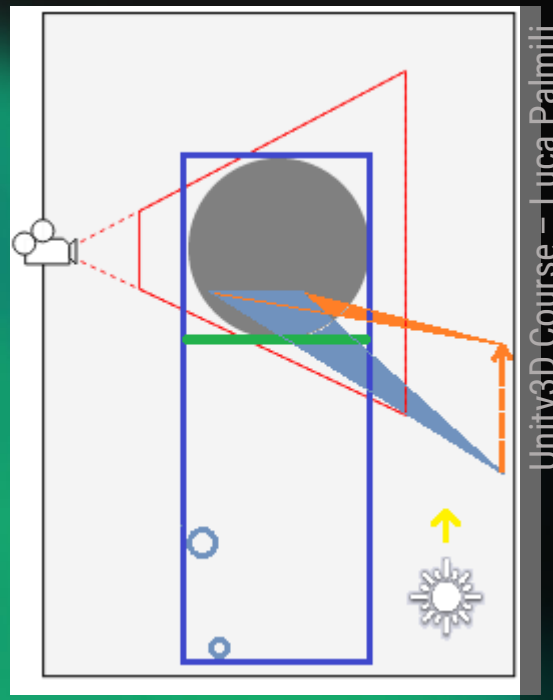


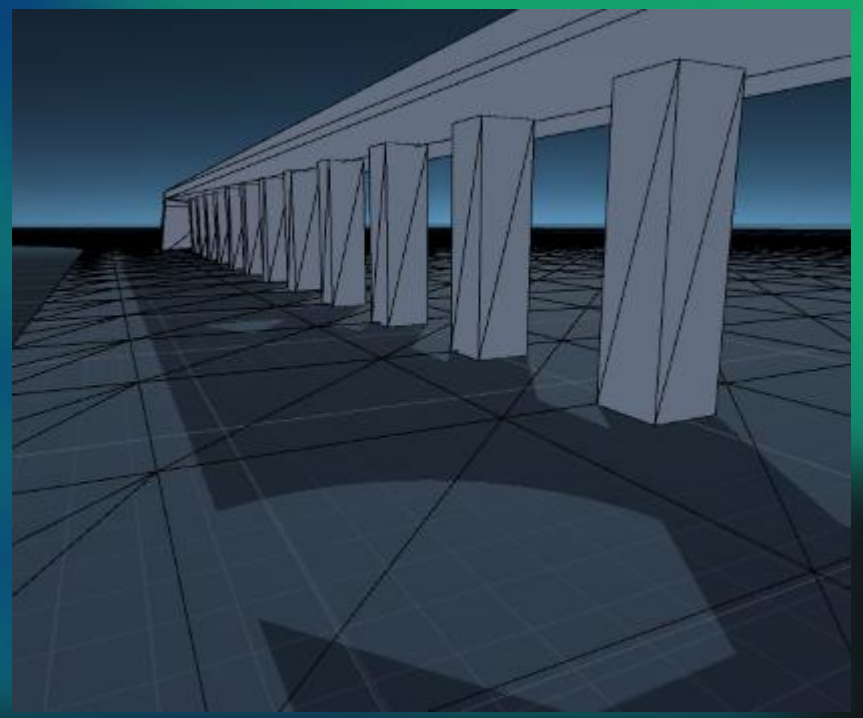
# Shadow Pancaking

Instead of render the entire scene in lightSpace

1. Calculate a near plane *Np* excluding any shadow caster not visible in the camera view frustum
2. Clamp the shadow casters in the camera VF to plane *Np* in the Vertex Shader.
    - This can create artifacts for very large triangles crossing *Np*
3. Use the QualitySettings/ShadowNearPlaneOffset to pull back Np and avoid this problem

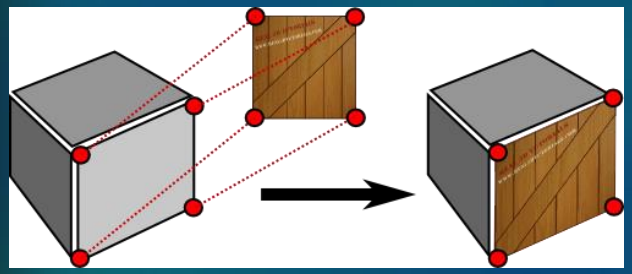
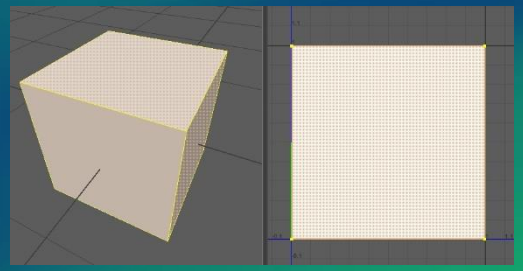
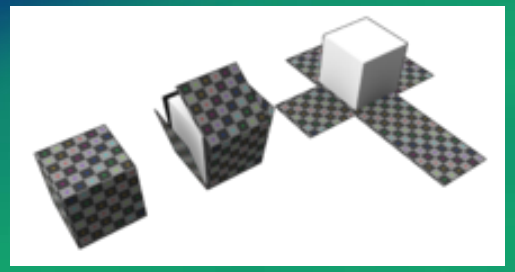
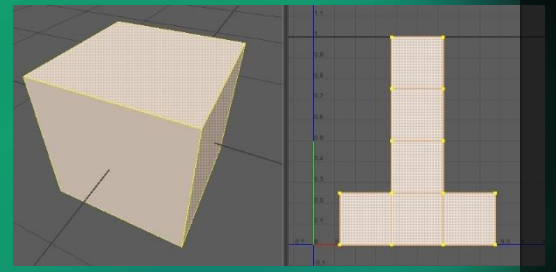
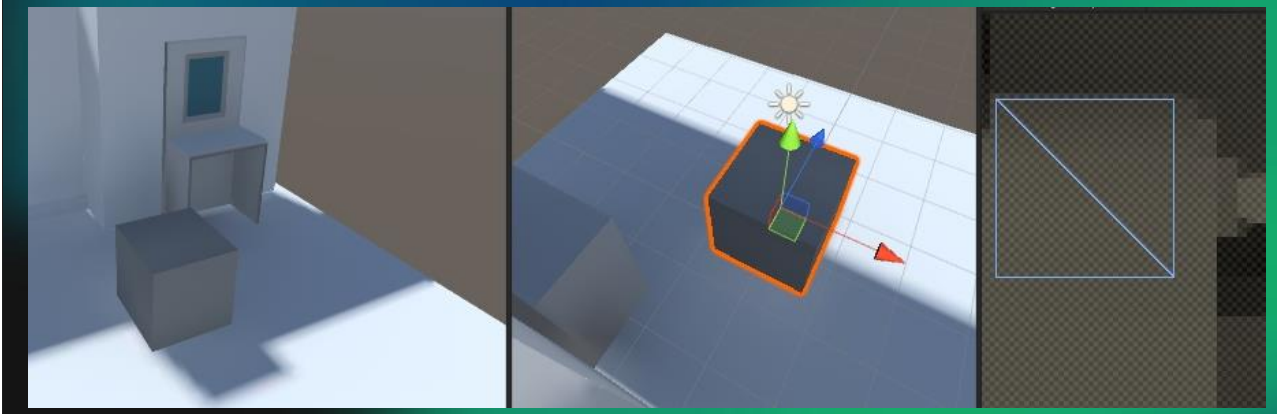
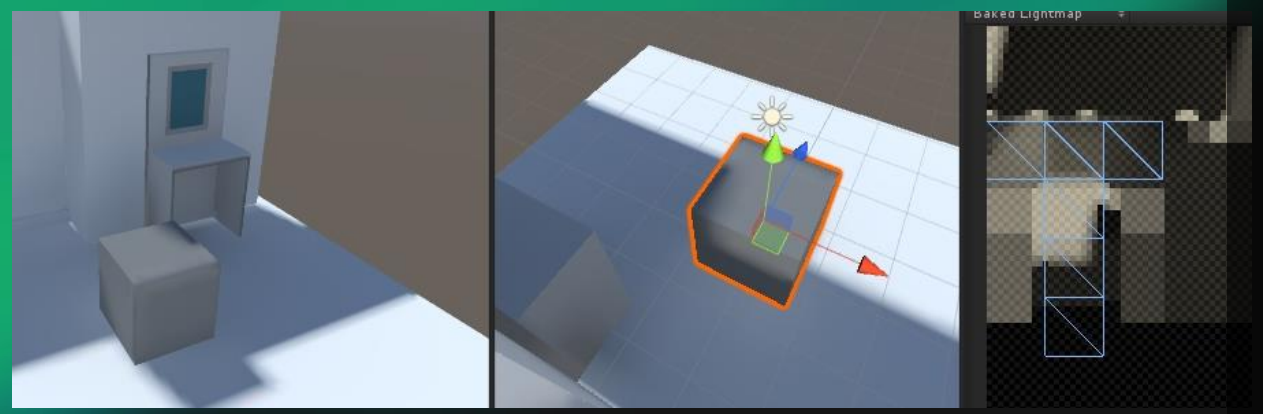






# Lightmapping UVs

- A Lightmap is a texture that contains light/shadow info and is wrapped around our obj in the same way as Albedo Textures
- BUT Albedo Texture UVs usually use optimized projection
- What happens if we use the same UVs set?
- We need 2 UVs sets: one for Texturing, one for Lightmapping

# Uvs channels

Unity supports up to 4 UVs channels

- Texturing UVSets
- Baked GI (Mixed lighting) 3DAsset Import settings/GenerateLightMapUVs
  - Direct lighting
  - Indirect lighting
  - AO
- Real-time GI Mesh Renderer Attribute/OptimizeRealtimeUvs
  - Indirect lighting only (direct light is rendered in realtime)
  - Low resolution
- Other

| Ordinal | Mesh class property | Shader Code | Used for |
|---|---|---|---|
| First | mesh.uv | UV0 | Diffuse, Metal/Spec etc. |
| Second | mesh.uv2 (& old .uv1) | UV1 | Baked lightmap |
| Third | mesh.uv3 | UV2 | Realtime GI data |
| Fourth | mesh.uv4 | UV3 | Whatever you like |

# Enlighten components

## Precompute

- Precalculates light transport in a scene
- Only depends on the static geometry and not the materials or light
  - Packing
  - Clustering
  - Compositing the light transport

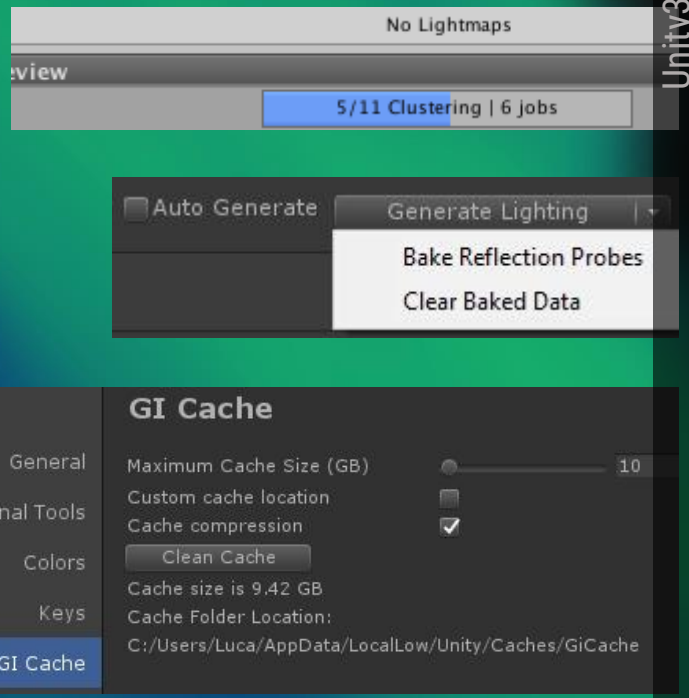
## Real-time solver

- Combines the precalculated data with the material and light information to produce light maps and probes in real time

## Light map baker

- Produces baked light maps for direct and indirect light and also AO
- Relies on the precompute data

- Autogenerate use an internal Cache. Always use Manual Generate Lighting before building for all Scenes. Unity will save lighting data as Asset files in your project folder
- To Clean GI cache: Preferences/GI Cache



# Precompute/Packing

The resolution of Realtime GI lightmap MUST be low

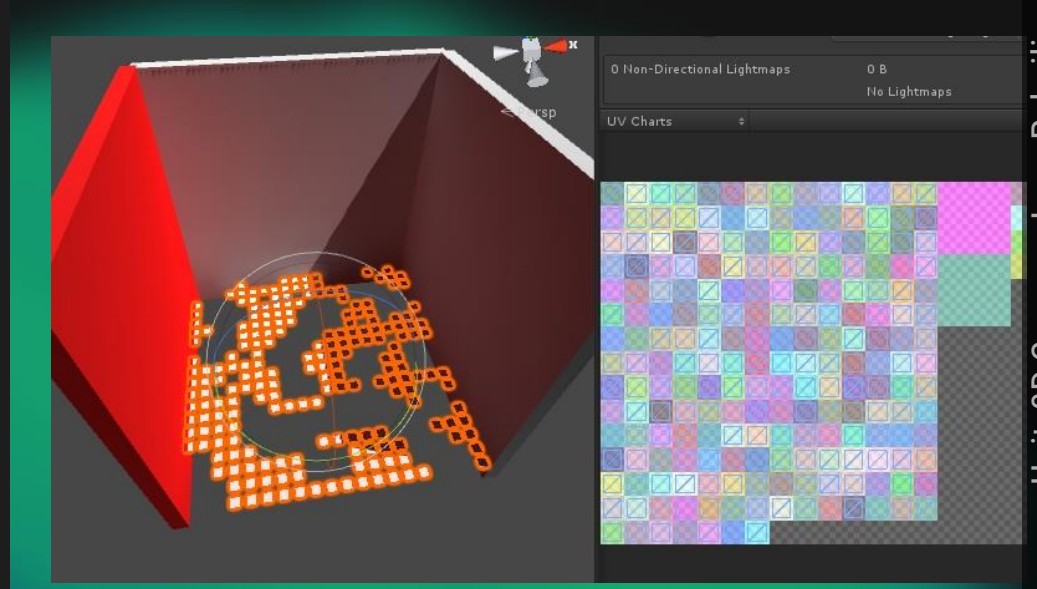
- We need another lightmap texture

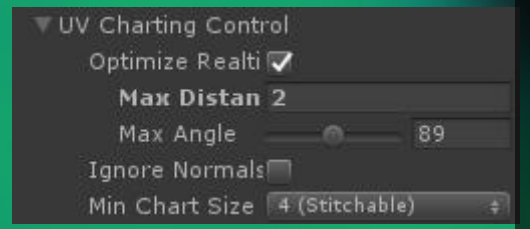
Packing main tasks

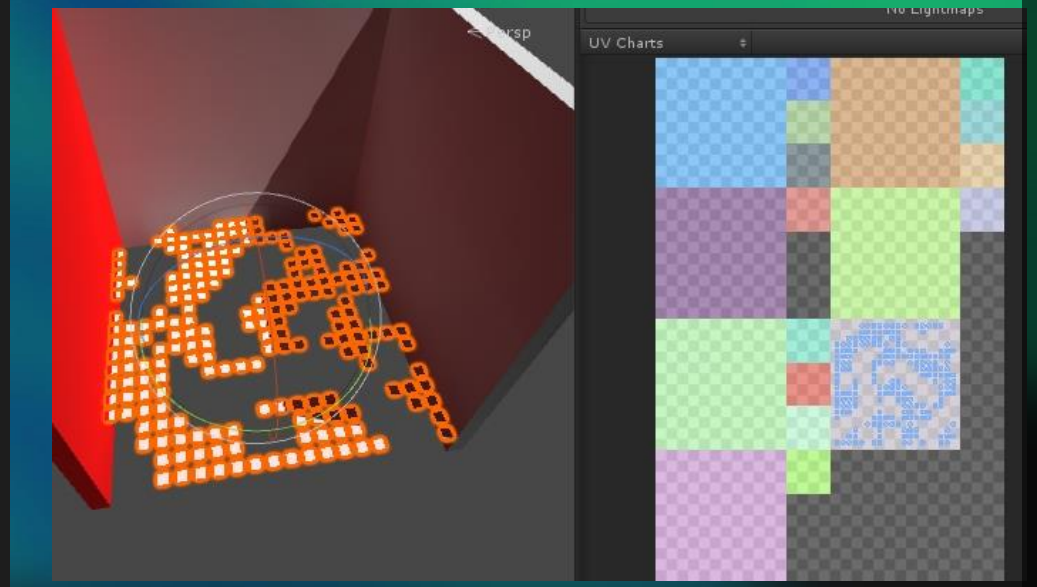
1. Identifies Charts in BGI UVs
   - Charts Groups of triangles in UVs that share vertices
2. Pack the Charts into lightmap used for real-time GI, ensuring that there is no light leaking between charts and that UVs are packed as tightly as possible

Uses

- Existing BakedGI UVs (Copy them into RealtimeGI UVs)
  - Usually produce artefacts, because of the low resolution lightmap
  - Not optimized
- Auto generated RealtimeGI UVs (from StaticGI UVs)
  - MaxDistance/MaxAngle
  - Ignore Normals
  - Min ChartSize



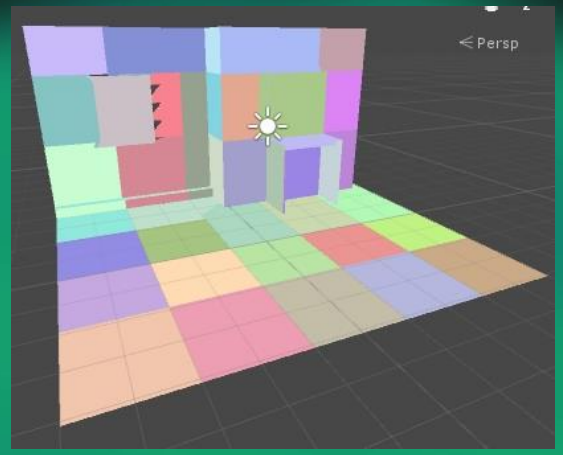



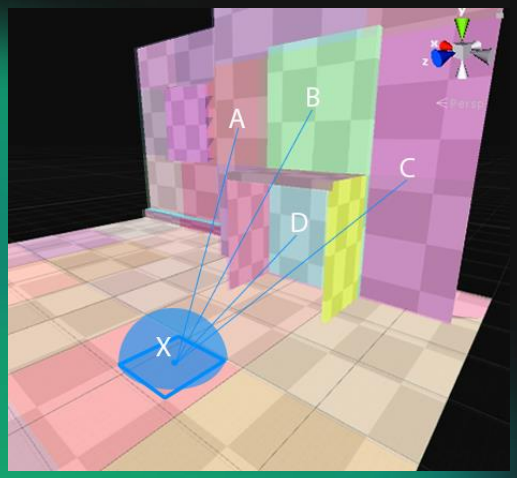
# Precompute/Clustering

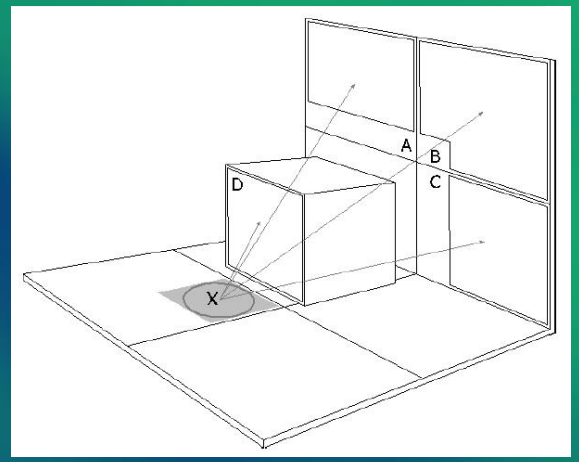
- Splits scene Geometry into clusters
- Use only triangles pos & orientation NOT UVs or Materials

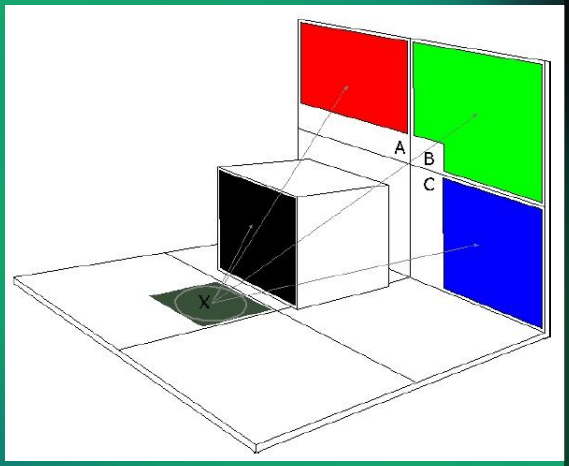
1. ∀ texel in a real-time lightmap / ∀ lightprobe
    1. Cast rays in all directions of its hemisphere / all directions
    2. Calculates the visibility of the cluster (form factor)

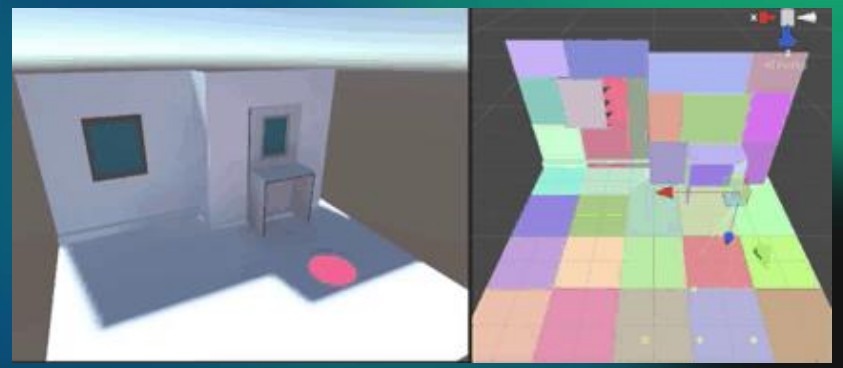
Lighting received from X = 0.05*A + 0.1*B + 0.05*C + 0.2*D



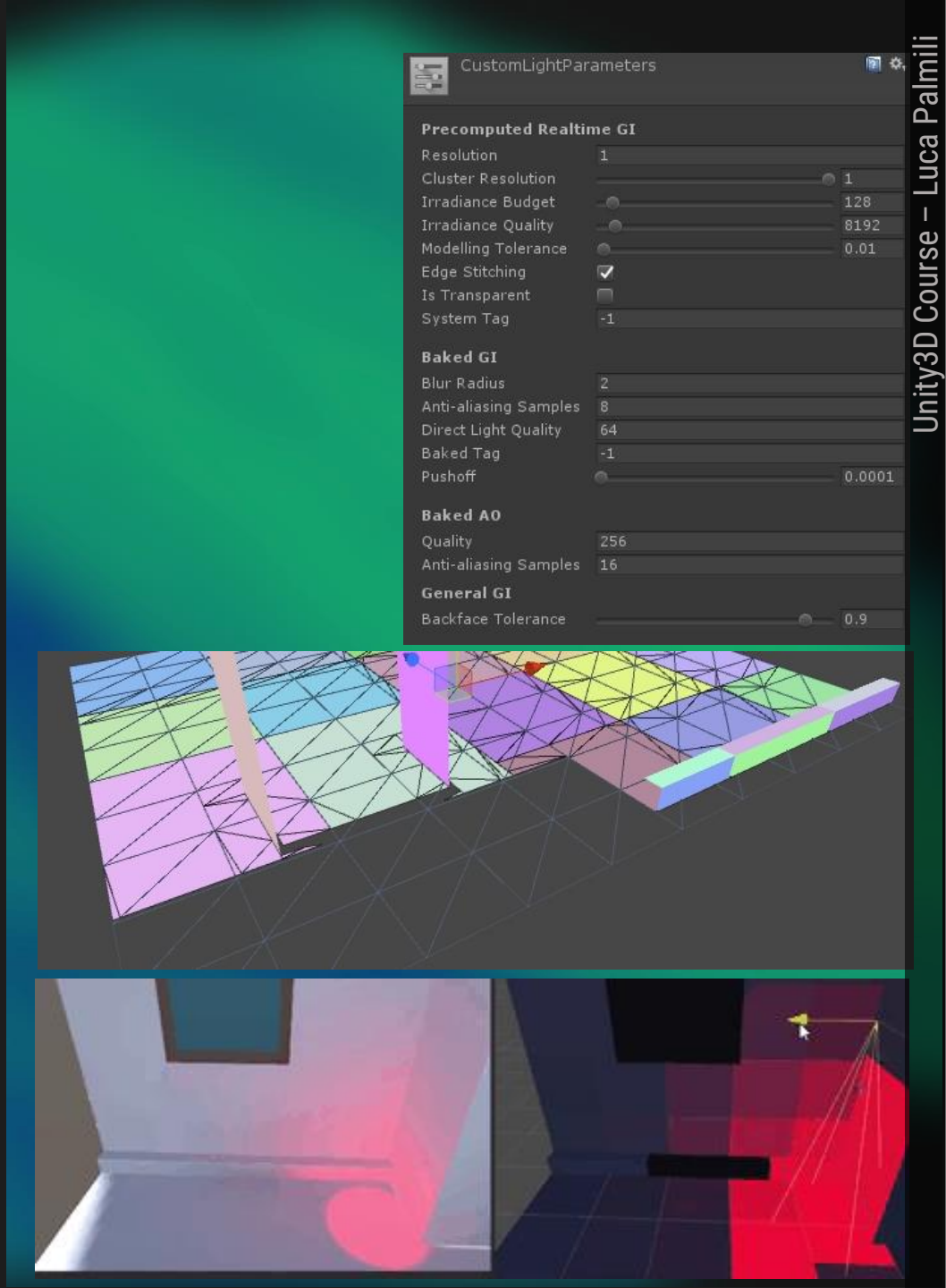



# Precompute/Clustering

Every obj can have its own lightmap parameters
- Create>LightMap Parameters

Light Transport quality based on
- Cluster resolution Default is half the real-time lightmap
- Irradiance Budget Form-factors # that Enlighten stores
- Irradiance Quality Rays # to cast per pixel
  - Multiple rays can contribute to build better form values

- BackfaceTolerance Invalidate Texel if too many of rays cast from it hit back faces



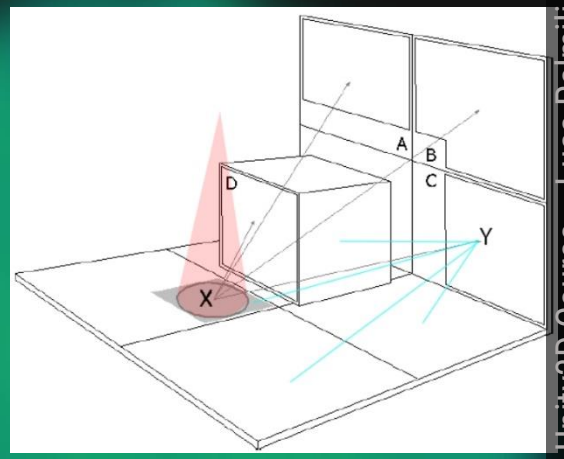
# Real-time Solver

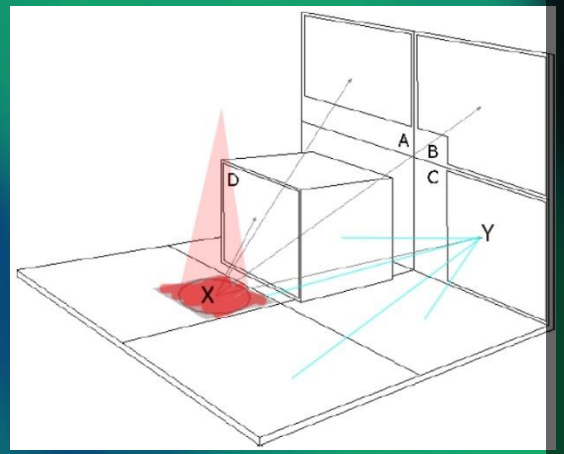
- Present in the Editor and in the Game

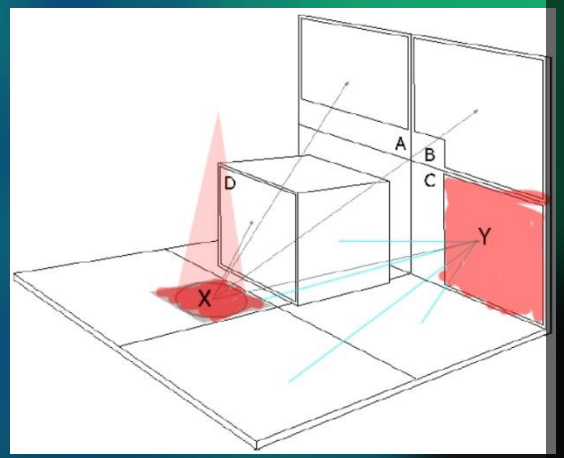
- Input lighting stage Direct lights contribute on the clusters, including dynamic lights
- Solve stage The solve stage sums up the cluster value multiplied by the stored form-factors, and stores the results in the light map

$$B_i = L_e + p_i \sum_{j=1}^{n} F_{ij} L_j$$

- Bounce stage reads back the values from the light maps and bounces light values back to the Clusters. This way Enlighten simulates multiple light bounces.
  - Light/Indirect Multiplier = Bounce Intensity



# Baking LightMaps

- Generate baked light maps that contain direct lighting, indirect lighting, and AO
- The indirect light map is generated based on the real-time results, up-sampled and filtered, to produce high-quality output
- Final gather Uses the baked light maps as input for the final light bounce
    - Switch to final gather only if you are happy with the lighting in your scene

# Setup a scene

## Turn off MixedLighting

- When happy with the lighting setup, turn on MixedLighting
  - baked light maps match the real-time rendered results (except fo soft shadows and area lights)

## Setting the indirect resolution

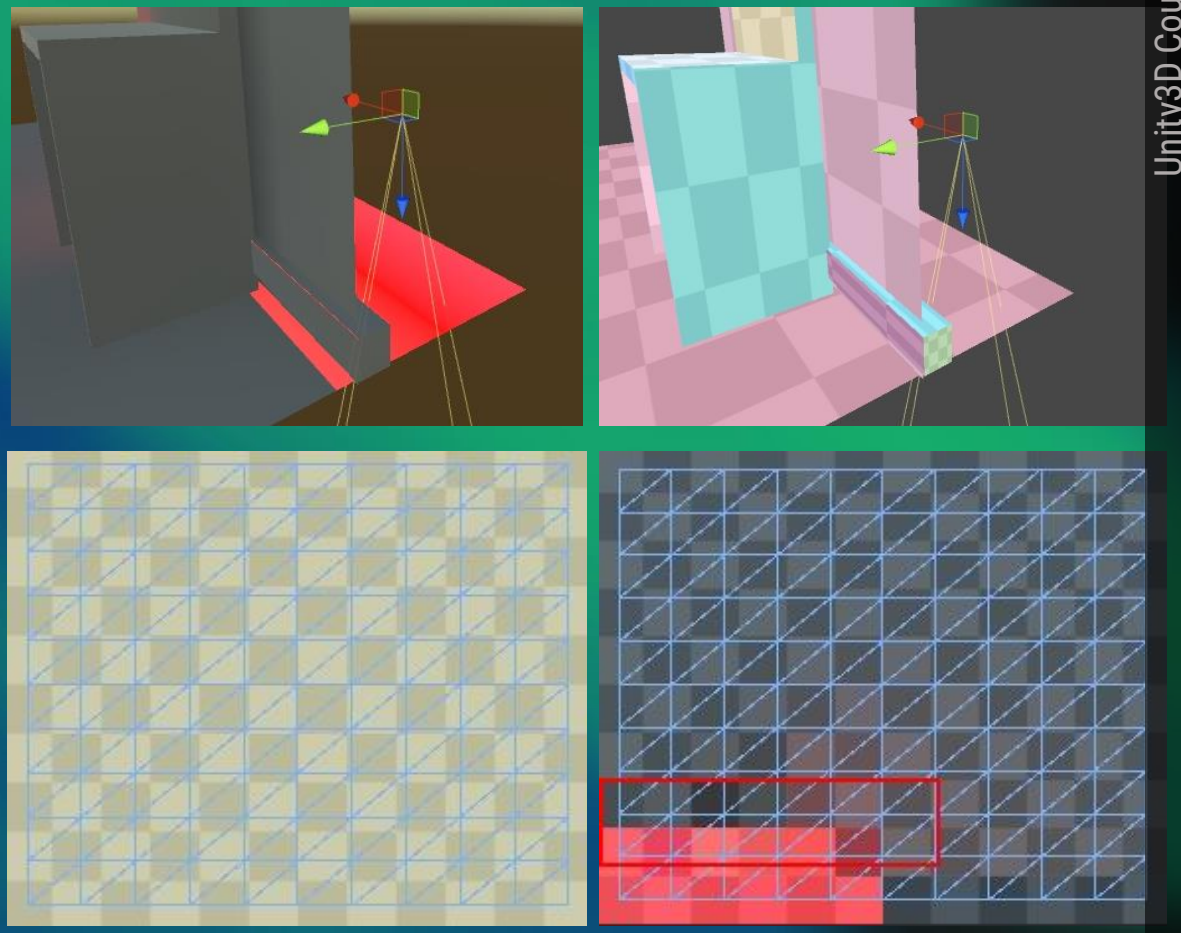
- Indoor 2-3
- Outdoor 0.5-1
- Terrains 0.1-0.5

## Pay attention to Charts disposition

- Charts that spans through a wall can produce light leaking
  - Split input UVs by hand

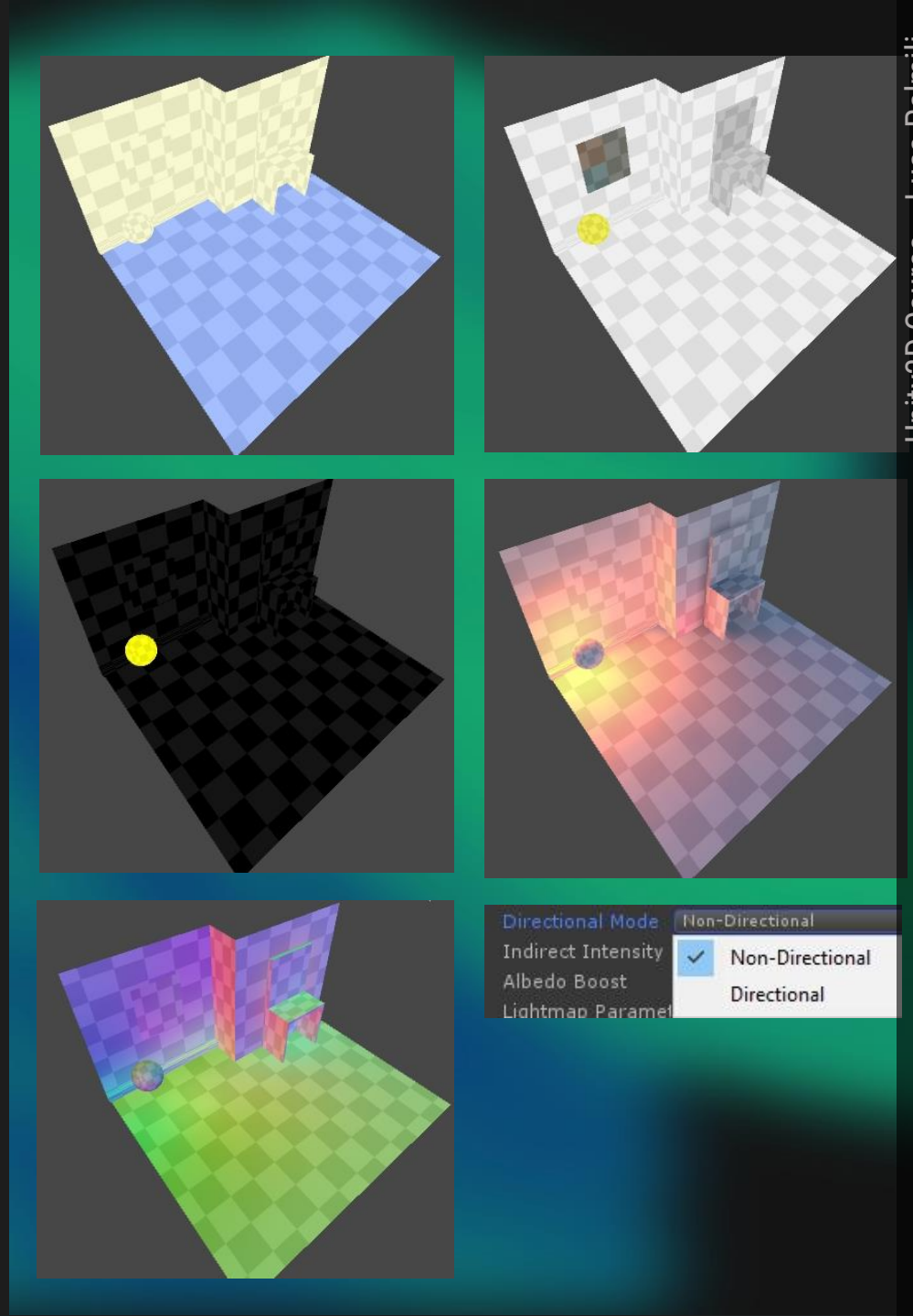
## Use emissive lighting for fake areaLights (but in realtime GI)

## Use lightprobes for small objs



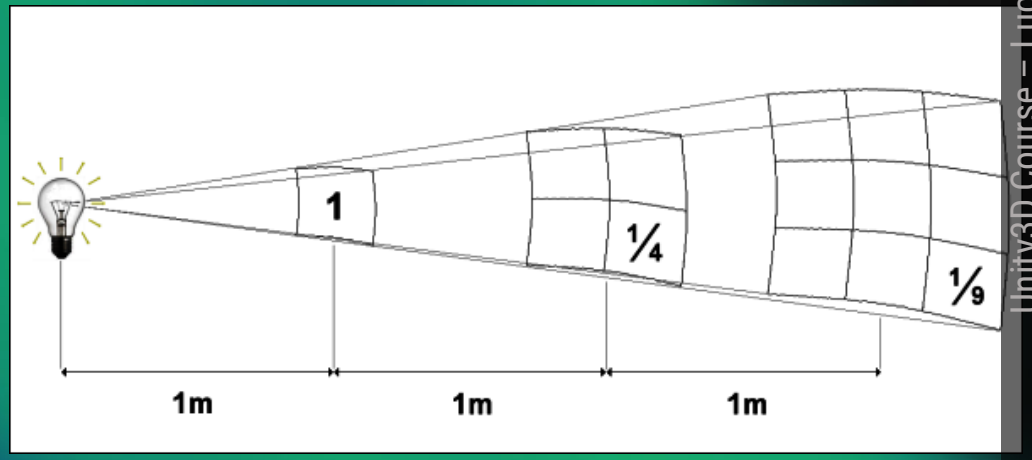
# SceneView GI Draw Modes

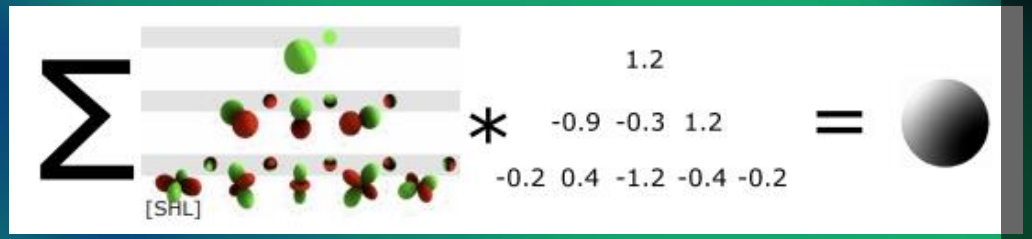
- Systems Enlighten is able to generate multiple Systems: each one is able to perform the precompute and the runtime in parallel
- Albedo Shows at what detail Enlighten uses the color information from your scene
- Emissive Color and intensity of the emitted light from emissive objects
  - Tip: we can use emissive objects with Enlighten to create area lights without any rendering cost
- Indirect Irradiance  received by surfaces using light maps
- Directionality Dominant light direction for each pixel
  - Active if LightmapSettings/DirectionMode is Directional
  - Used if the GI Shader use normal map info
  - Adds a lightmap texture to store direction info

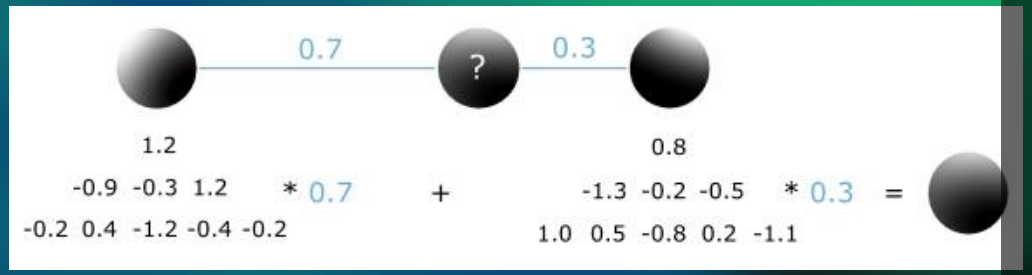


# Light Probes

- Lightmaps store info about light hitting the surfaces, Light Probes store info about light passing through empty space
  - provide high quality lighting (including indirect bounced light) on dynamic objs
  - provide the lighting information for static scenery when that scenery is using Unity's LOD system
- Irradiance
- A LightProbe use L2 SH to store directionality of RGB ray values (27 floats)
- Interpolating two probes can be done just by interpolating their coefficients.



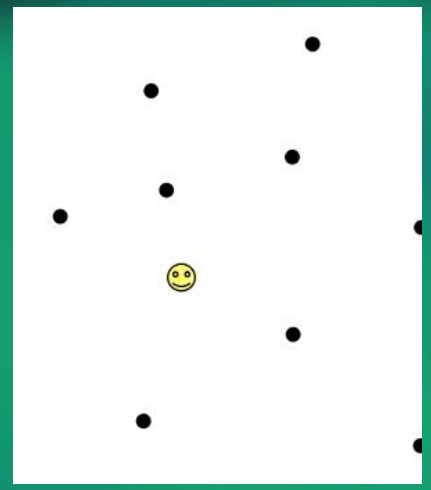
# Light Probes

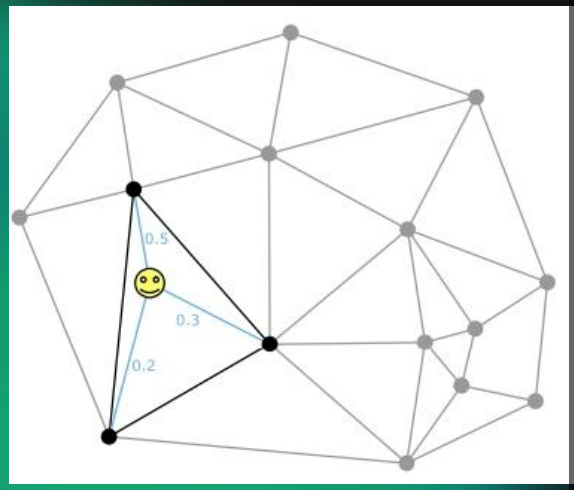
- Which probes & what wheights?
- Wow many pts do I need to define a circle?
- Delaunay Triangulation
- 2D > 3D Delaunay Tetrahedralisation

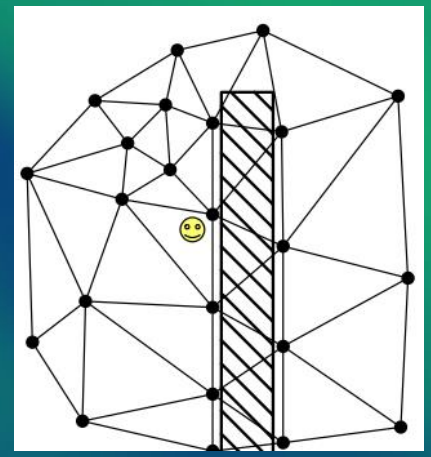
1. Cache the tetrahedron index from the previous frame
2. Calculate barycentric coordinates *Bc*
    1. *Bc > 0* => We are inside
    2. *Bc < 0* => The obj moved the most negative coordinate
3. Each tetrahedron has exactly 4 neighbours. Find the right one

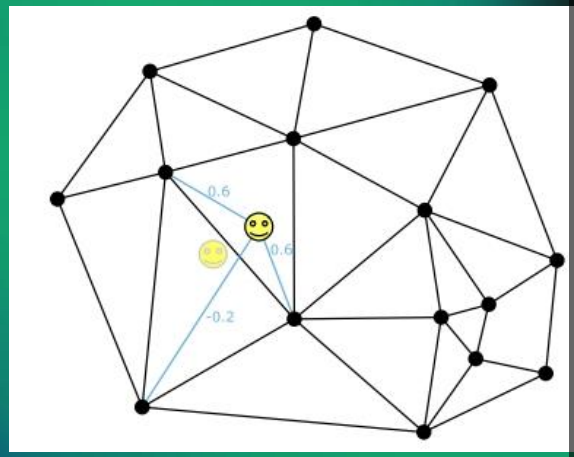
Data needed

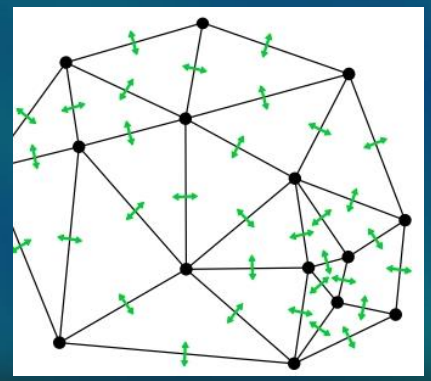
- probe positions probe_count * 3 floats
- SH coefficients probe_count * 27 floats
- hull rays hull_probe_count * 3 floats
- Tetrahedra indices 4 vertices + indices 4 neighbours

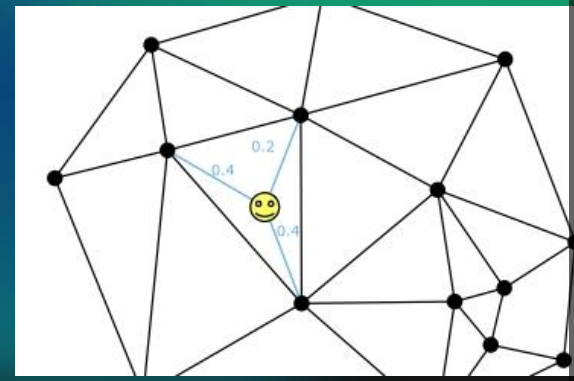






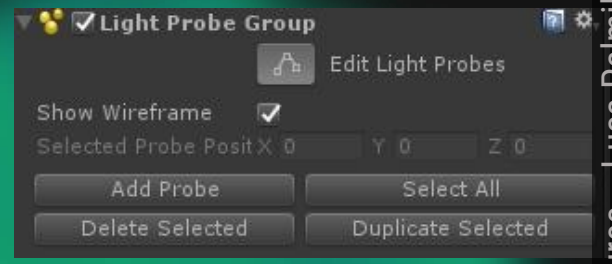



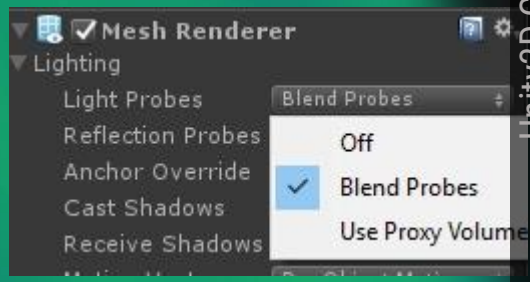


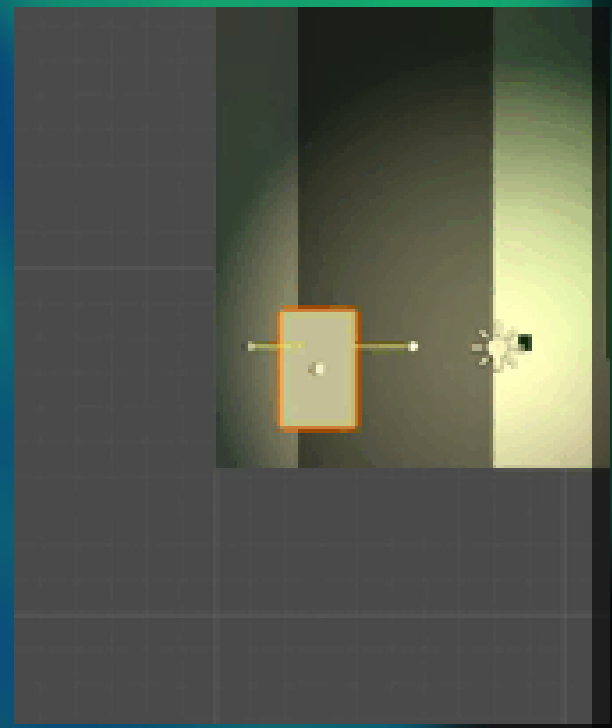
# Light Probes - Placement

- Best to add it to a new empty GameObject
- Light probe info resolution = how closely packed are the probes
- Good rule: Condensed pattern around areas that have complex or highly contrasting light
- Never use a 2D disposition (even for driving on a road games). Aat least two vertical “layers” of points in your group of probes: this will allow to calculate sensible tetrahedral volumes from the probes
- Dynamic Objs MeshRenderer LightProbes
  - BlendProbes
  - UseProxyVolumes + lightProbeProxyVolume large moving objs
  - OverridePoint
- LightProbeProxyVolume

[LightProbesTetrahedralGrid.cs]

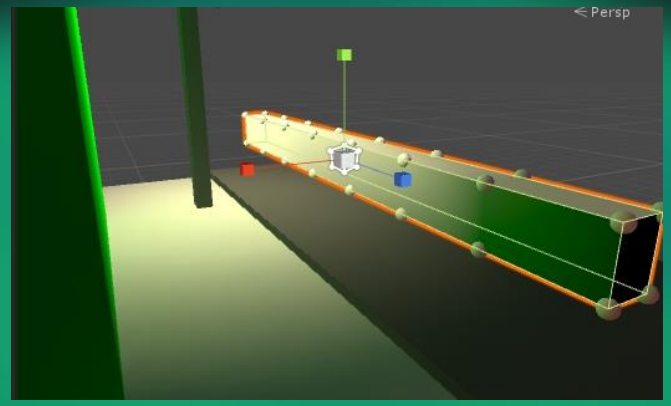


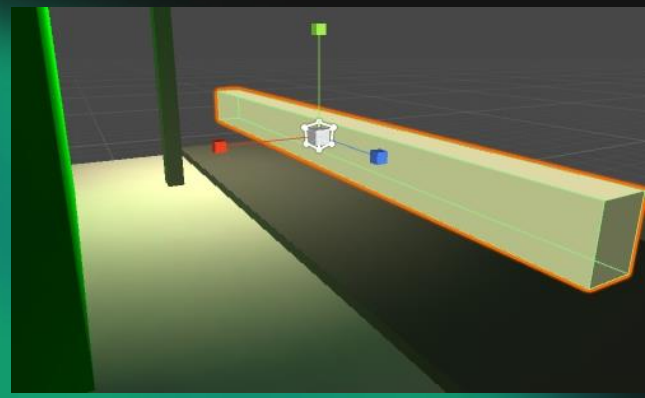



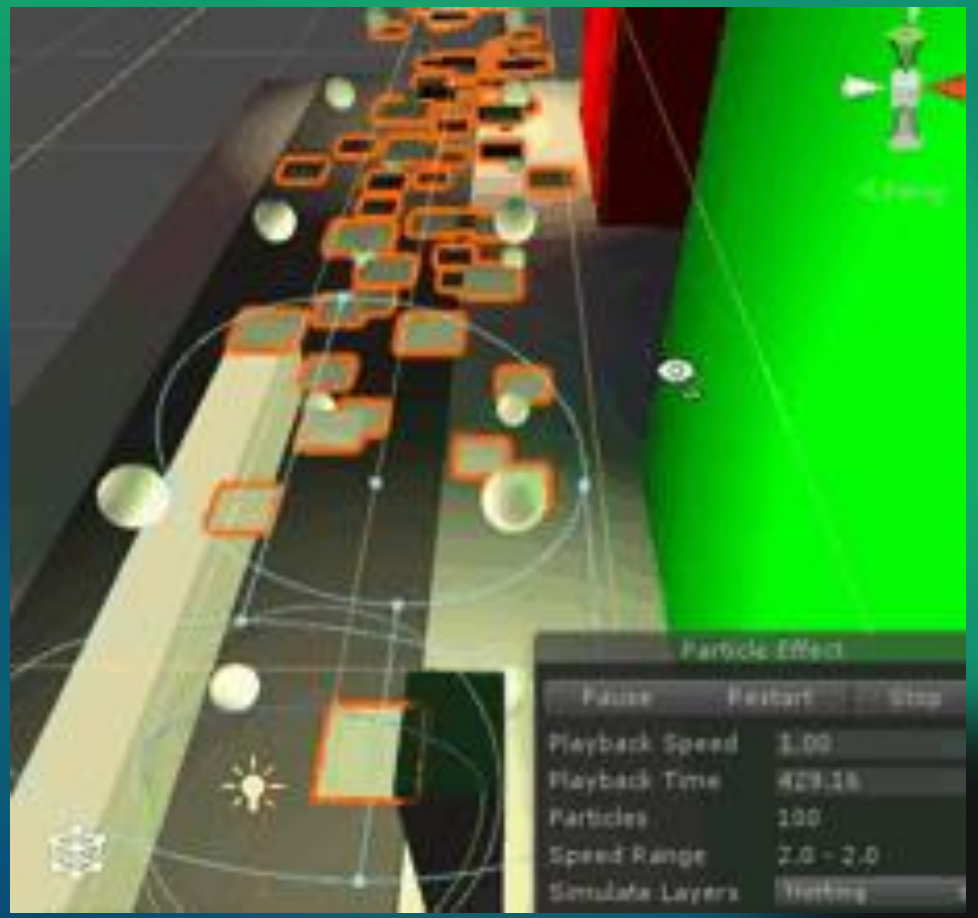



# LP Proxy Volume

- UseProxyVolumes + lightProbeProxyVolume
  - Large moving objs
  - Particles
  - ProxyVolumeResolution
  - ProbePositionMode





# Ex 39 – Italian Box

- Use Realtime GI
- Add LightProbeCreator.cs to LightProbeGroup
  - In OnValidate() it will reset LightProbeGroup LProbes, creating a cube grid of Xp x Yp x Zp Probes, which will span across a volume of Xsize x Ysize x Zsize
  - LightProbeGroup LPG = GetComponent<LightProbeGroup>();
  - List<Vector3> probePos = new List<Vector3>();
    - //Add Vector3 positions to probePos
    - LPG.probePositions = probePos.ToArray();