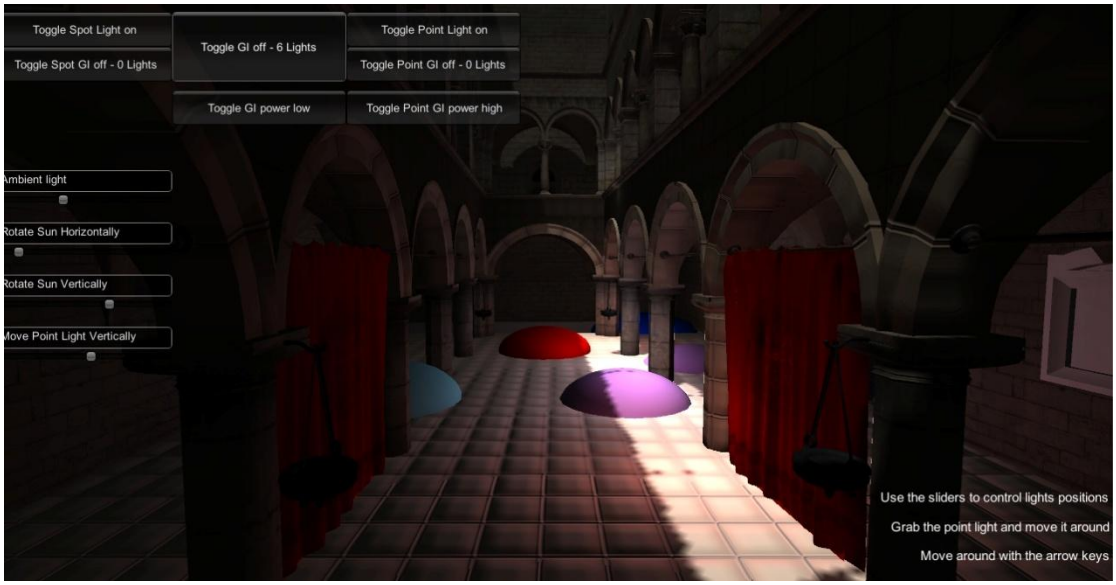
Global Illumination Proxy System



This is a pack devoted to Dynamic Global Illumination approximation through a number of unique techniques. The goal is to emulate a one bounce GI result with as little overhead as possible and with easy setup.

Instructions

In order to use the GI Proxy, the user must add the "**LightCollisionsPDM"** script to the gameobject that holds the light that will cast Indirect Illumination. All three light types are supported (point, spot and directional lights).

The point light should always be at the grid mode. The spot light supports non grid mode (it will be upgraded with grid in later versions).

Global illumination basic setup

The script will start casting point lights depending on the source light and its orientation and placement in space. The options to control the creation and destruction of the bounce lights are described below.

Point light options:

Second Bounce (v1.5): Enable 2ond light bounces. These will get their attributes from "Point Light Radius 2" parameter and will use a proportion of the main light bounce intensity. The direction of raycasting are determined by the "Hor Directions" and "Ver direction" parameters.

Not important 2ond (v1.5): Make 2ond bounces render mode "not important", for extra performance gain when needed.

Use light sampling: Look at near lights when creating a bounce light, register the target color required based on ray cast hit surface color and other lights color and assign the lerp color with the bounced light as initial value for the light color. Then lerp from the mixed color to the target color. Use when few lights are needed to smooth out color grading.

Sampling distance: Maximum distance to use when looking for near lights for the sampling.

Use grid: Use a grid to ray cast from the source. Use in directional and point lights. The grid is parallel to the ground in directional light source, expanded in X and Z dimensions.

Max positions: Max number of positions to use in the grid.

Extend X: Scale the grid along the X axis. Use for directional light grid.

Extend Y: Scale the grid along the Y (or Z) axis. Use for directional light grid.

Go random: Randomize grid positions. Use for special effects only.

Horizontal Directions: Use to define grid density for point light source.

Vertical Directions: Use to define grid density for point light source.

Point light radius: Radius around the point light to cast bounce lights.

Point Light Radius 2 (v1.5): Define 2ond bounce lights radius.

Update color distance: Distance at which the color of the already registered bounced lights (from the point light source) will be updated, by casting a ray from source to each light to grab a possible surface color change.

Follow mode:

Follow hero: Use to make ray casts follow the hero. Setup the initial position in front of the initial hero position and set the grid in an area around the hero. The ray casts will always target the area in front of the hero. **Note that the light source has to intersect a surface in order to create the rays to ray cast with, even though the actual ray cast may be moved afterwards.**

HERO: The hero gameobject.

Hero offset: Offset of grid from the hero position.

Bounce Light creation control:

Divide initial intensity: Amount by which the targeted "Bounce Intensity" for the created bounce light is divided initially. Use to balance initial light creation intensity depending on the distance between created lights defined in "Minimum Density Distance" parameter.

Bounce intensity: Target intensity for created Bounce Lights. The speed this intensity is reached depends on the "**Divide initial intensity"** and **"Appear Speed"** parameters.

Bounce Range: Bounce light range.

Degrade speed: Speed at which each a bounce light will degrade until it disappears. The degrade starts when a light passes over one of the distance checks against last bounced light, last main ray hit and distances to the light source and the main camera.

Appear Speed: The speed at which a bounced light reaches its target Bounce Intensity.

Min distance to last light: Degrade bounce light if distance to last registered light greater than this value.

Min distance to source: Degrade bounce light if its distance to the source light is greater than this value.

Min density distance: The minimum distance to all the already registered bounce lights a ray cast resulting hit position must have in order to register a new bounce light. Use to control light density.

Enable follow: Make already registered lights follow the last ray cast hit. Use when grid option is off or for special effects.

Follow distance: The minimum distance of the bounce light to the ray cast hit to start following it.

Follow speed: The speed used when following the last ray cast hit position.

Bounce color: Use when "get hit color" option is off to custom color the bounce lights. If "mix colors" is on, it will mix with the source light color.

Jitter directional: Add jitter to the light position. Use only for special effects and when small updates are needed in ray casting.

Bound up down: The jitter amount.

HDR drop speed: If the "Appear" parameter is set very high, the intensity of the light may go beyond the Bounce Intensity parameter. Use this value (must be above 0 to be enabled) to let the extra intensity drop to the target one. Use to emulate HDR effects.

Get hit color: Bounce light will be colored by the color parameter of the surface hit by the source light. Note that the material needs to have a "_Color" parameter.

Get texture color: Bounce light will be colored by the color of the texture where the ray cast hits. Texture must have Read/Write option enabled. Note that the material needs to have a "_MainTex" parameter, otherwise will revert to the surface color (if it has "_Color" parameter) and if not to the source light color.

Mix colors: Use to lerp to the ray cast hit color, from the source light (when …) or the current light color.

Dynamic update colors: For each ray cast hit, check near lights and update their color. Use to handle dynamic color changes or moving objects.

Color change speed: The speed of color transitions.

Soften color change: Multiply the above speed with delta time.

Use light color: Lerp uses source light color as initial value for bounce lights.

Debug on: Show ray casts that are used to register the bounce lights in the scene.

Debug 2ond (v1.5): Show ray casts that are used to register the 2ond bounce lights in the scene.

Add only when seen: If a ray cast to the bounce light target position from the camera returns a hit closer to the camera than the camera-hit point distance, bounce light wont be registered. Use in order to avoid registering lights that are hidden behind objects in front of the camera.

Close to camera: Use close to camera check, degrade bounce lights beyond the distance defined in "**Min distance to camera"** parameter.

Min distance to camera: Degrade bounce lights beyond this distance, if the above parameter is used.

Max hit – light distance: Use to make lights follow the source light. Degrade registered bounce lights beyond that distance to the last ray cast hit.

Give offset: Offset bounce lights from the ray cast hit position.

Placement offset: Offset in Y axis.

Origin at projector: Use the **"Projector OBJ"** as source light position.

Projector OBJ: Object to use as projector.

Light pool: The gameobject to hold the created bounce lights. Must not be parented to any object (unless lights are required to follow the rotation of an object or the projector – light source)

Update every: Seconds between ray cast updates.

Use height cutoff: Avoid light creation above a certain height from the hero. Uses "Floor if no hero" value if no hero is defined.

Cut off height: The height above which bounce lights do not get created.

Floor if no hero: Floor to check height cut off distance from, if no hero is assigned to the script.

Jove Lights (v1.5): Enable Jove lights, this will inject a script to every bounce light, that will disable it and insert a Jove point light in its place. Then this light will be updated by the injected script based on the changes to the Unity point light. The injected script is named "Control_Jove_light" and all its commented lines should be enabled in order to work, plus the Jove namespace should be added in the "using" statements.

Grab sky light (v1.5): Use IBL influence from the sky texture (must be set to "advanced" and read/write on), sky must have a collider.

Rotate grid (v1.5): Option to rotate the grid based on the Projector or light rotation.

Sky influence (v1.5): The amount of sky color to add to the Global Illumination.

Sky (v1.5): Sky object, the check is done by name, so every other object with the same name and a collider will contribute to the sky color.

Offset on normal (v1.5): Offset lights along the normal vector of the hit position.

TIPS:

There are prefabs to use for a clean setup. Place the required prefab in the scene and change it based on the scale of the scene.

For the dun light (directional), if hero follow mode is needed (this applies to the sun only), drag the hero from the scene to the script parameter.

If the hero is instantiated, he must be added through scripting. There is a commented line of code where the hero should be declared in code, make sure to leave the parameter blank in the editor, so it gets the hero transform from the code.

For the spot light, the system has to be parented to the hero prefab (or the hero in the scene) and move it to the desired location on the hero.

The point light prefab can be used in any way, it can be attach to the hero or dragged it with the Drag Transform script.

Hero mode details:

For the hero follow mode, it is desirable to have the lights hit in front of the hero and play with the "offset" parameter to define the exact placement.

To see where lights hit, enable the debug option(s) in the script inspector and the "gizmo" option in the Unity editor, hit play and then go to scene mode and see where raycasts hit and play with offset until the grid is near or around the hero. Then keep these values, exit play mode and insert them in the offset parameter in the script, so you can start with the lights hit in front of your hero, then the raycasts will follow the transform of the hero as he moves around (make sure the hero follow option is enabled too)

Some more things to consider:

- The raycasts will hit any collider, or pass through objects without colliders, so if you see lights in the middle of a shadow, it may have passed through a missing collider in the object that casts the shadow .

- There is a parameter to cut off lights registered way above the hero in Y axis, for example use this in the atrium scene to cut back the lights that may register in the roof. This threshold is in relation to the hero, so will cut light above a certain height from the hero, thus the hero can move in Y axis and the system will still work.

- The sun should also reposition itself to follow the light rotation, this will make sure the raycasts are directed correctly in relation to the gameobjects in the scene. Take care of the distance to source parameter in this case, since lights wont register if further away than this threshold from the source light.

- The hero follow mode main function is to keep the lights hitting in the initial grid in front of the hero, even if the light rotates, so a rotation will redirect the raycasts always in front of the hero.