Unityはじめるよ

～光と影①～

統合開発環境を内蔵したゲームエンジン
http://japan.unity3d.com/

※いろんな職業の方が見る資料なので説明を簡単にしてある部分があります。正確には本来の意味と違いますが上記理由のためです。ご了承ください。
この資料内の一部の画像、一部の文章はUnity公式サイトから引用しています。

光と影

光と影を意識してセットアップした世界と

なにも考えずにただ表示した世界とでは

見た目で与える印象が大きく異なる。

メリットデメリット

光に関する処理、影に関する処理は

総じて処理やメモリ、ストレージの負荷が高い。

最高なクオリティとパフォーマンスになるよう

実行環境に合わせて、

取り捨て選択をする必要がある。

やりたいこと

新興国を含む海外も対象にしたスマホゲームを作りたい。

そのためには、
低スペックのAndroid端末を考慮する必要がある。

なので、CPU性能、GPU性能、通信能力、メモリ使用量、
ストレージ使用量とも抑えめにしなくてはならない。

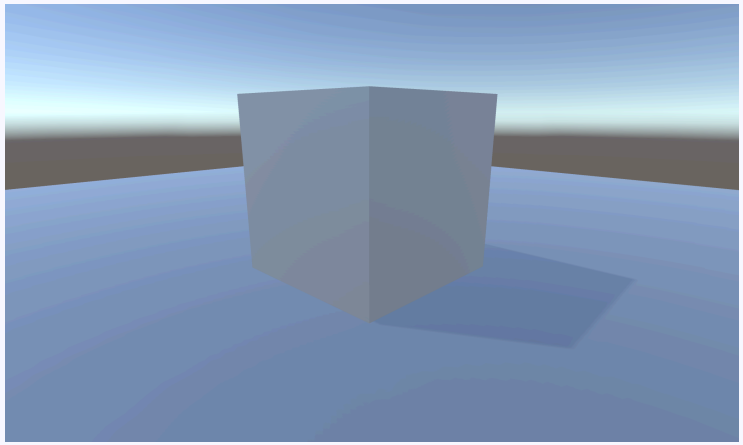
その上で、そこそこの見た目を実現したい。

光と影は見た目のクオリティを上げるのに効果的なので
どんな方法があるかを調査＆テストしている。

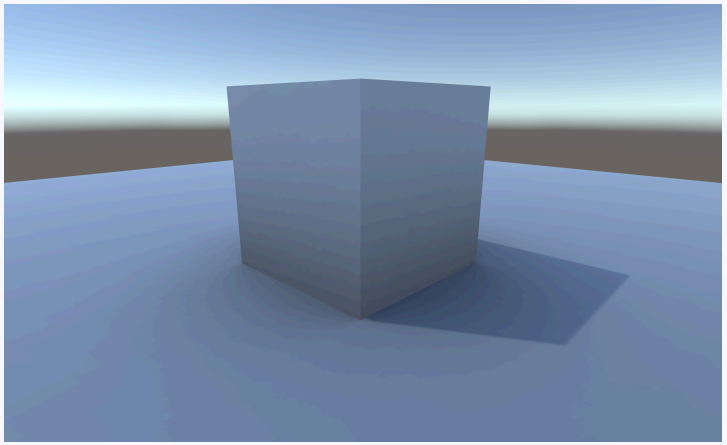
AO
(Ambient Occlusion)

AOって？

現実世界では、建物と地面が接している付近や、穴、折り目などの入り組んだ部分が互いに接近していると、環境光が遮られて暗く見える。それを表現する処理。
物を地面に置いた時の接地感が増す効果もある。



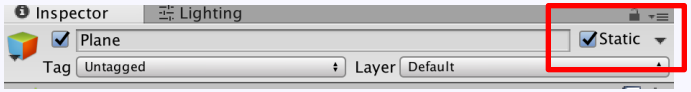
AOなし



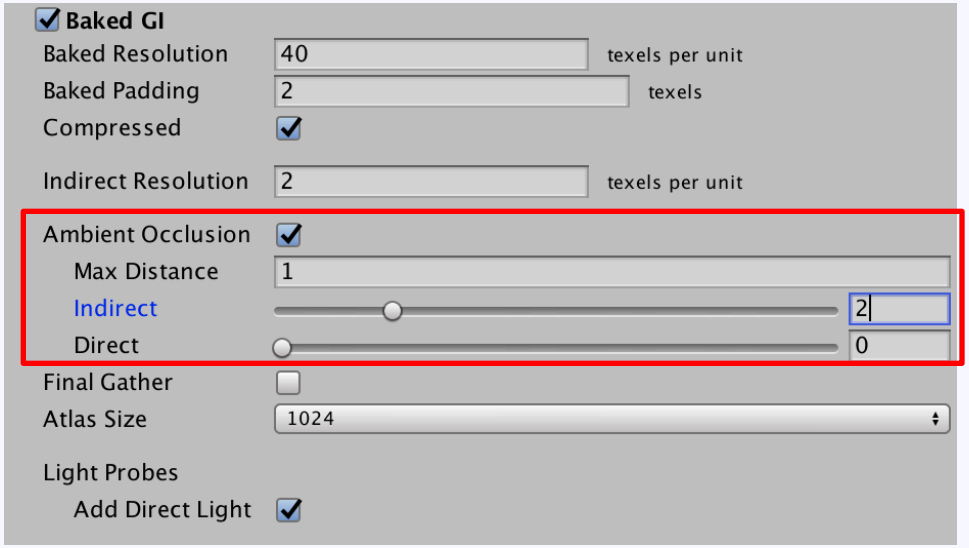
AOあり

設定方法

適用したいオブジェクトにStaticのチェックを入れる



Window > Lighting > BakedGI > AmbientOcculusion
のチェックを入れ、**Bake**する



MaxDistance
光源計算を行う為のレイを飛ばす距離。

Indirect
間接光に照らされた時のAOの強さ。
値を大きくすると入り組んだ箇所付近の面が暗くなる。
間接照明にのみAOをつける方が現実的な絵となる。

Direct
直接光に照らされた時のAOの強さ。
値を大きくすると入り組んだ箇所付近の面が暗くなる。
現実的な絵にはならないが芸術的な絵を作りたいときに使う。

メリット・デメリット

AOはライトマップに焼くことで利用できる。

メリット
・実行時の負荷は少ない

デメリット
・スタティックなオブジェクトにしか使えない
・ライトマップ用テクスチャ分のメモリとストレージを消費
※ライトマップを使う前提ならAOを使うデメリットはない
※広大なステージの場合など、
テクスチャのサイズを大きくしてクオリティを保つか、
テクスチャのサイズを抑えてクオリティを下げるかの
検討が必要。

ライトマップ使いたくないよ

って場合はどうするか。
考えてみた結果、メッシュ側に情報を持たせてはどうか。

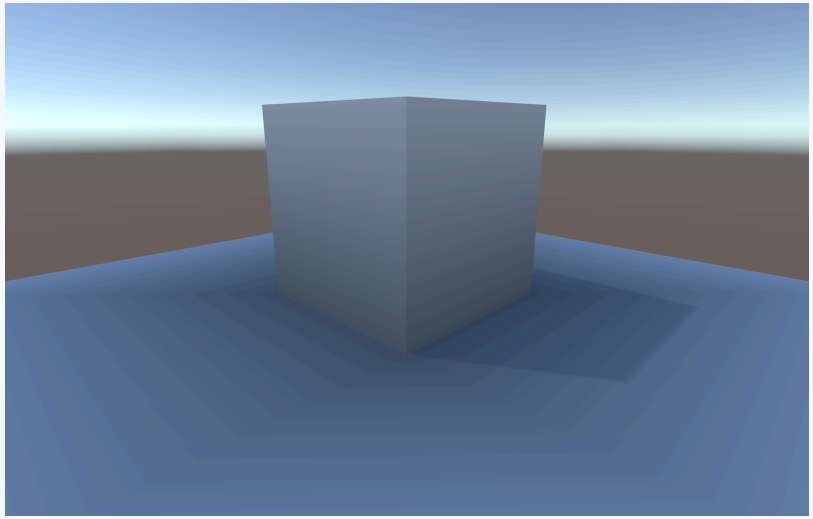
Unityのシェーダーは頂点カラーを使っていない。
なので、メッシュの頂点カラー部分に他の情報を入れておくという方法は、割と使われる手法である。

メッシュの頂点カラーにAO情報（暗さ）を持たせておけば、AOに近いことが再現できるのではないだろうか。

実はこの方法、
グラフィック性能が低かった初代PSでよく使われた。
と言ってもAOとしてではないのだが、
テクスチャ＋頂点カラー(または法線)で表現の幅を拡げていた。

画像はPS史上最高グラフィッククオリティのレースゲーム『R4』

やってみた
Blenderで地面とCubeを作って頂点カラーをセット。
頂点カラーの影響を受けるようにシェーダーをいじる。
※頂点カラーのR成分をAOに利用

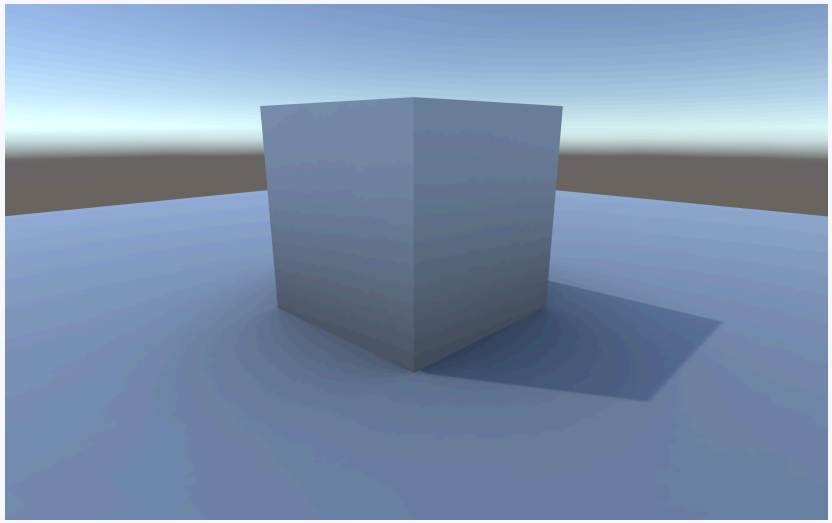


そこそこの見栄えになったんじゃない？

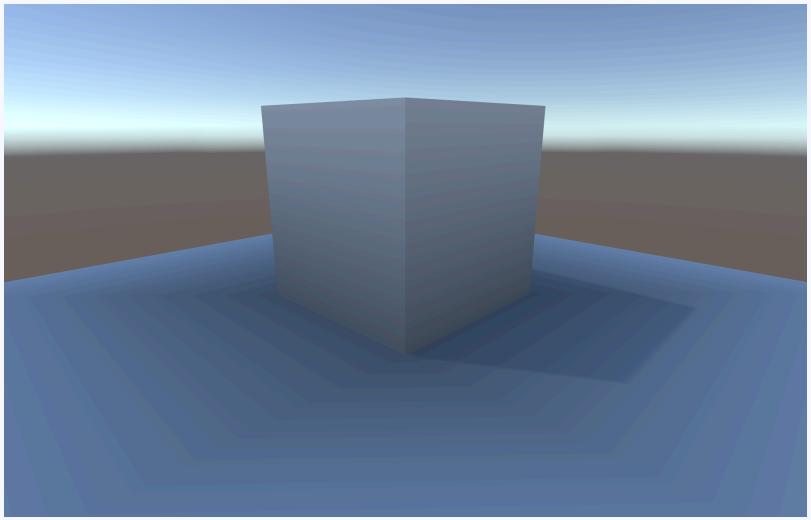
頂点カラー用シェーダー

```
Shader "Custom/VertexColorAO" {
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM
        #pragma surface surf Lambert vertex:myvert
        #pragma target 3.0

        struct Input {
            float4 vertColor;
        };

        void myvert(inout appdata_full v, out Input o){
            UNITY_INITIALIZE_OUTPUT(Input, o);
            o.vertColor = v.color;
        }

        void surf (Input IN, inout SurfaceOutput o) {
            o.Albedo = 1.0 * IN.vertColor.r;
        }
        ENDCG
    }
    FallBack "Diffuse"
}
```

比較



BakedGI AO



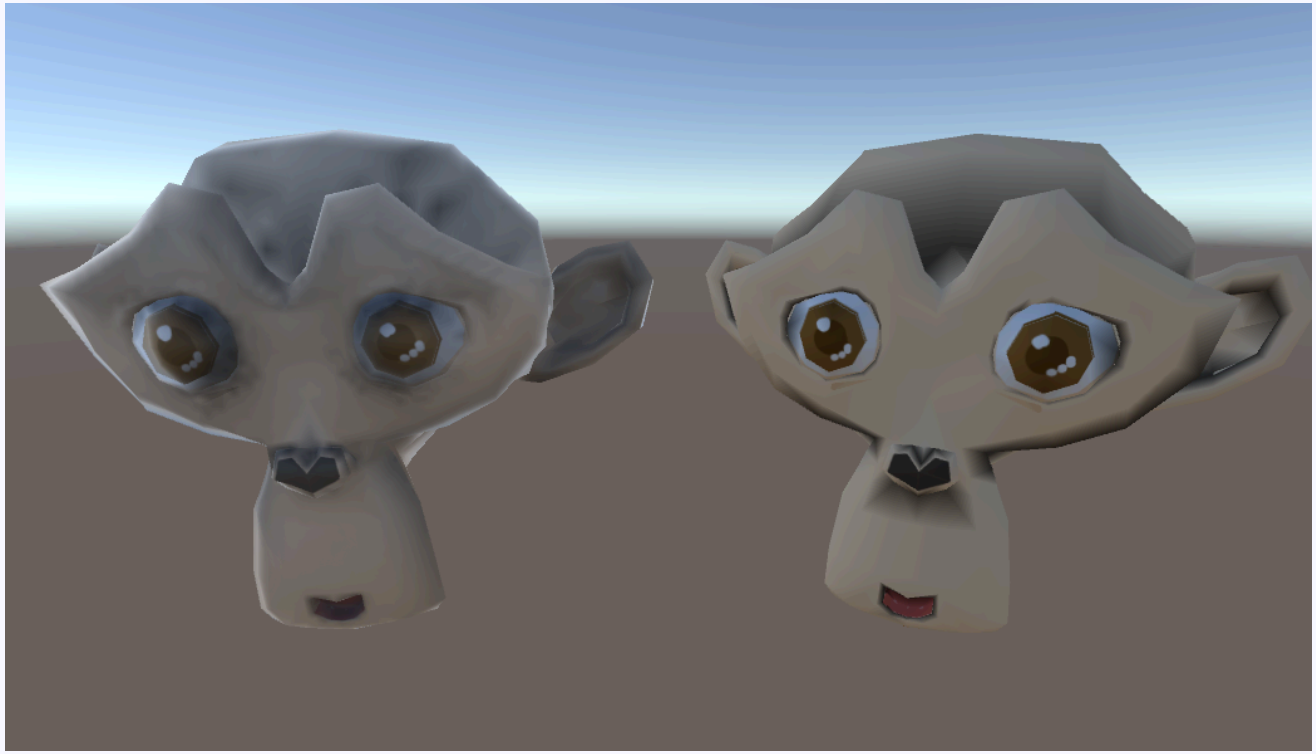
頂点カラーでAO

メリット・デメリット

メリット
・実行時の負荷はほぼなし（メモリ・CPU・GPUとも）

デメリット
・モデル作成時に頂点カラー情報を持たせなくてはならない
・地面側に影を落とすためには地面側のポリゴンの調整が必要
・シェーダーを用意する必要がある
・見た目はBakedGIには敵わない
・動かないオブジェクトにのみ使うべき

別モデルでも比較してみた



BakedGI AO

頂点カラーでAO

サル顔だと、どんなに調整を頑張ってもBakedGI AOを綺麗にすることができなかった・・・

頂点カラー+テクスチャ+カラー用シェーダー

```
Shader "Custom/VertexColorTexture" {
    Properties {
        _ColorTint ("Tint", Color) = (1,1,1,1)
        _MainTex ("Base (RGB)", 2D) = "white" {}
    }
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM
        #pragma surface surf Lambert vertex:myvert finalcolor:mycolor
        #pragma target 3.0

        struct Input {
            float4 vertColor;
            float2 uv_MainTex;
        };

        fixed4 _ColorTint;
        void mycolor (Input IN, SurfaceOutput o, inout fixed4 color)
        {
            color *= _ColorTint;
        }

        sampler2D _MainTex;
        void myvert(inout appdata_full v, out Input o){
            UNITY_INITIALIZE_OUTPUT(Input, o);
            o.vertColor = v.color;
        }

        void surf (Input IN, inout SurfaceOutput o) {
            o.Albedo = tex2D (_MainTex, IN.uv_MainTex).rgb * IN.vertColor.rgb;
        }
        ENDCG
    }
    FallBack "Diffuse"
}
```

SSAO
(Screen Space Ambient Occlusion)

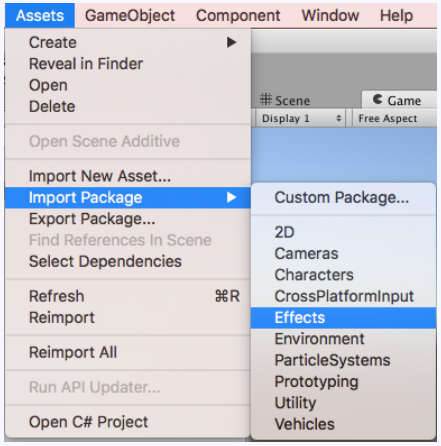
SSAOって？

AOをスクリーンスペースで計算しようというもの。
スクリーンスペースとは画面の範囲でということ。
リアルタイムでAOを行うために、画面内に映っているものに限定して計算範囲を狭めてAOを行なう手法。
なので、計算時間は解像度に依存する。
また、計算方法も簡易的となる。

設定方法

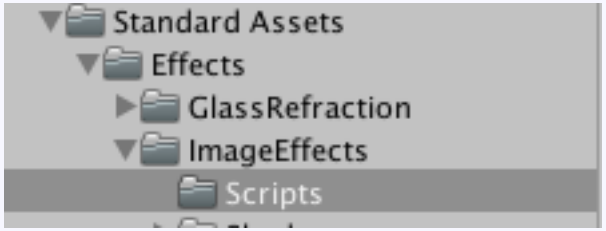
パッケージのインポート

Assets > ImportPackage > Effects



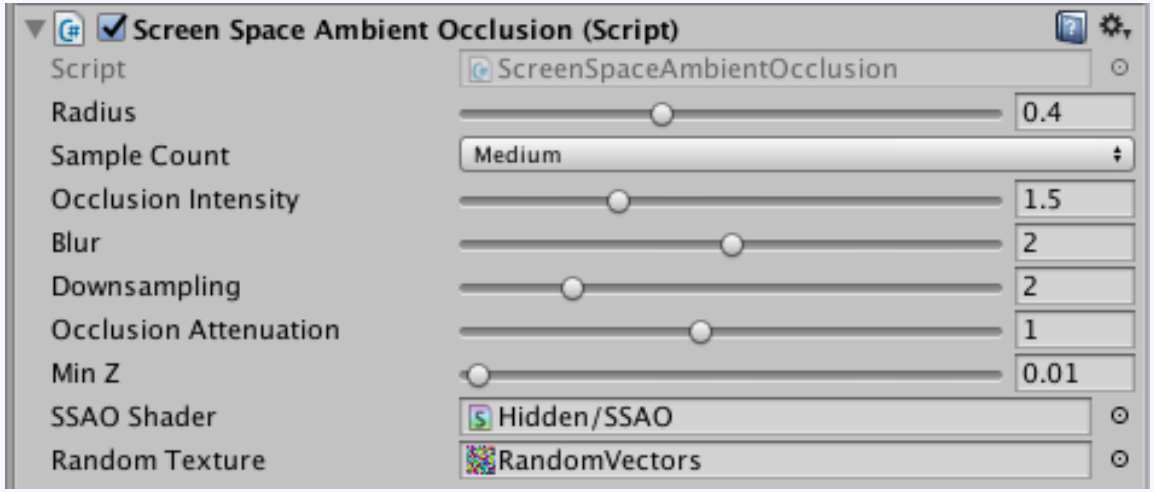
スクリプトをカメラに追加する。
プロジェクトビューの、

StandardAssets > Effects > ImageEffects > Script > ScreenSpaceAmbientOcculusion.cs



設定項目　※公式マニュアルから引用

プロパティー:	説明:
Radius	この半径より隣接距離が短い場合アンビエントオクルージョンを適用します
Sample Count	アンビエントオクルージョンのサンプリング回数 より高い値にすることで品質が上がりますが、オーバヘッドの負荷も上がります
Occlusion Intensity	アンビエントオクルージョンが適用する暗さの度合い
Blur	暗くする際に適用するにじみ にじみなし（値を0）にすることで処理速度は上がりますが、暗くした場所のノイズが増えます
Downsampling	計算を行う際の解像度（例えば値を2に設定した場合は画面解像度の半分） より大きな値にすることでレンダリング処理の時間を短縮できますが、品質は低下します
Occlusion Attenuation	オクルージョンを距離に応じて減衰させる強度
Min Z	画像の乱れが発生する場合はこの値を増加してください



実行結果

SSAO

動くとチラチラするがリアルタイムでAOが行えるのは
グラフィッククオリティを上げる意味では効果が高い。

メリット・デメリット

メリット

・キャラなどの動くモデルにAOを付けられる

デメリット

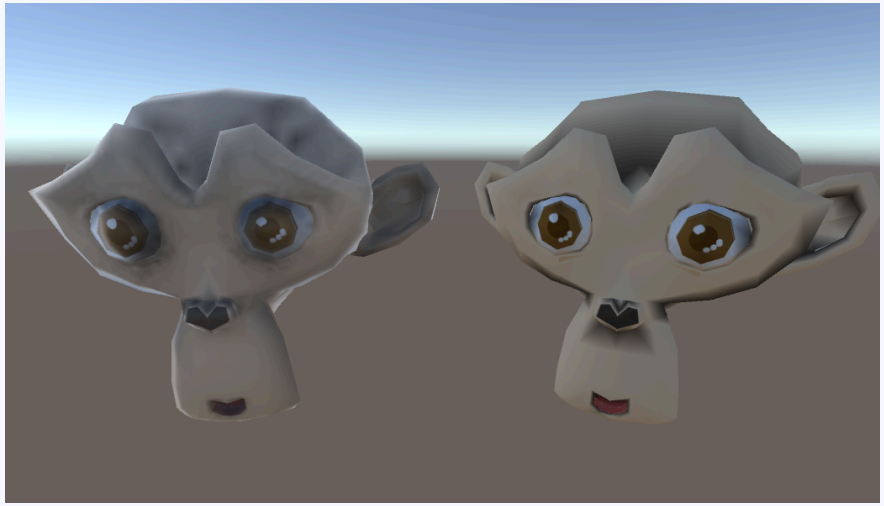
・処理が重い

・処理できるハードが限定される

シェーダーモデル3 と depth textures をサポートするグラフィックスカードが必要

比較

SSAO



BakedGI AO

頂点カラーでAO

モデルデータがよくなかったので
SSAOとBakedGI AOが良い感じしないけど、
背景オブジェクト系ならとても良く見えるようになる
素晴らしい機能ですよ。

高速に動くという以外に、
頂点カラーAOを使うメリットがないか考えてみる

※光源を無視する簡易AOであることが前提

■たくさんのキャラで装備品を共通のテクスチャにしたい場合
テクスチャ側に影を持たせるよりも、キャラモデル側に影を
持たせておいたほうが、そのキャラに合う自然な影を作れる。

まとめ

BakedGIのAOを使えば、
お手軽に見た目のクオリティを高めることができる。
ただし、ライトマップのサイズをどうするかは調整が必要。

頂点カラーAOなら低負荷でAOっぽいことができる。
ただし、モデルデータ作成時に工夫する必要があり、
開発コストは増えてしまう。

Unityのエディタ拡張でシーン上のGameObject一式に、
頂点カラーAOを適用できそうな気もする。これは要調査だな。
できればモデルデータ作成時のコストがなくなる。

■プロジェクト一式はここ

光と影①(AO/SSAO)
http://monolizm.com/sab/src/ao.zip

■参考サイト

Unity公式ドキュメント
https://docs.unity3d.com/Manual/LightingBakedAmbientOcclusion.html
https://docs.unity3d.com/jp/540/Manual/script-ScreenSpaceAmbientOcclusion.html

ご清聴ありがとうございました