光とシェーダー

VR空間の'見える'を支える物理学

かぐ（Kagu3）

自己紹介

かぐ（Kagu3）

- **バックエンドエンジニア**
- **ITエンジニアキャリア相談・雑談集会** 主催
- **法学部卒**
  - 法学部は割り算までしかいらない 😅

お断り

今回の対象

- **一般的なVRChater** を対象にしています

安心してください！

- **物理学をまったく知らなくても大丈夫！**
- **数式は一切出しません！**
- **厳密さより分かりやすさ重視！**

VR世界で使われる物理学

光の流れ

VR世界（CGの世界）には**Light**があり、それが物体に当たって**見える色**が決まります。

この見える色を決める**シェーダー**の数式が**物理学の理論**を基にしています。

使われる物理学

光学（Optics）が主に使われる

- BSDF（双方向散乱分布関数）も光学の考え方
- **エネルギー保存則**のような代表的な物理学の考え方も使われている

BSDFとは

Blender使ってる人ならプリンシプルBSDFを聞いたことがあるかも

BSDF（双方向散乱分布関数）＝

BRDF (双方向反射率分布関数) ＋ **BTDF**（双方向透過分布関数）

BRDF（双方向反射率分布関数）とは

表面に光が当たったとき、**同じ側に**反射される光の方向と強さを記述する関数。

つまり、反射を表す計算式

BTDF（双方向透過分布関数）とは

表面に光が当たったとき、**物体を通り抜けて反対側に**透過する光の方向と強さを記述する関数。

つまり、透過を表す計算式

今回の範囲

単純化のため
反射（BRDF）のみ扱います

透過も扱うと複雑なので、今回は省略！

物理学の反射の理論

大きく分けて2種類

1. **拡散反射** - 「色味」を決める
2. **鏡面反射** - 「光沢感」「ツヤ」「鏡のような映り込み」を決める

拡散反射の例：りんご

りんごの見え方

- りんごの表面は微細な凹凸がある
- 光が当たると、あらゆる方向に散らばって反射
- これが**拡散反射**
- りんごの赤い色味は、この拡散反射で決まる

鏡面反射の例：黒塗りの高級車

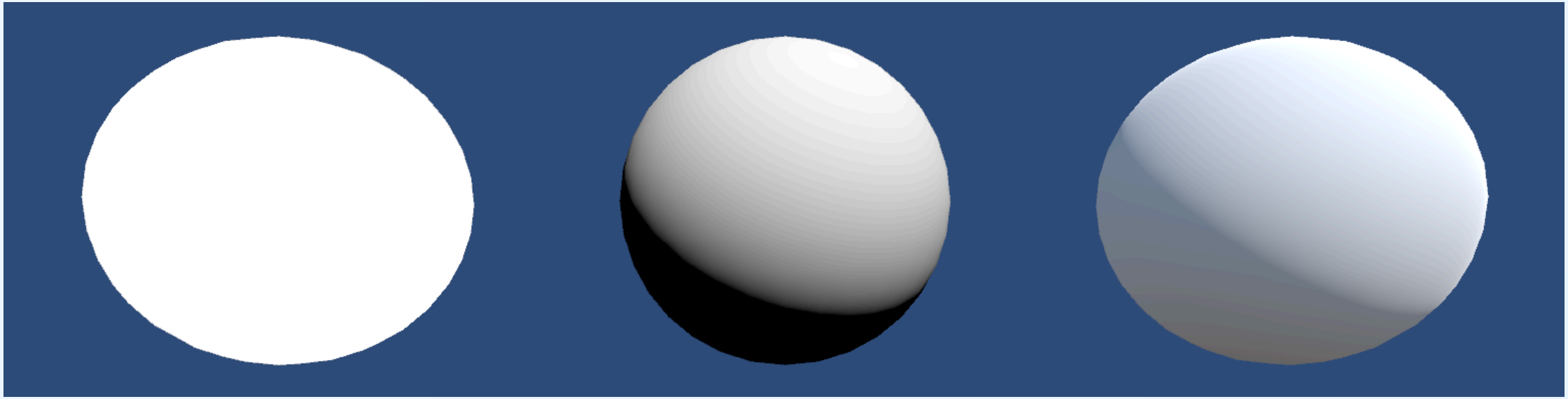
黒色での光の反射

- 黒色でも光は反射する！
- 表面が滑らかなため、光が一定の方向に反射
- これが**鏡面反射**
- 高級車のツヤツヤ感や映り込みは、この鏡面反射で決まる

実際に実装してみる

Unityでの実装例

ランバート反射



左 Unlit

中 ランバート反射

右 StandardShader

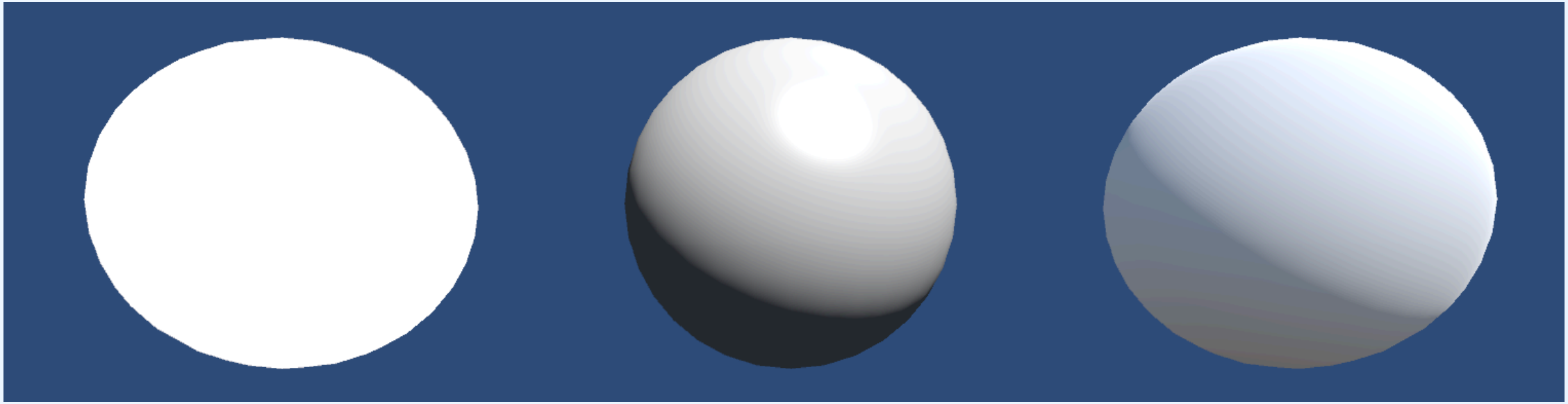
```
// 法線を正規化
float3 worldNormal = normalize(i.worldNormal);

// メインライトの方向を取得
float3 lightDir = normalize(_WorldSpaceLightPos0.xyz);

// ランバート反射の計算
float NdotL = max(0, dot(worldNormal, lightDir));

// 純粋なランバート反射モデル
fixed4 col = _Color * NdotL * fixed4(_LightColor0.rgb, 1.0);
```

Phong反射モデル



左 Unlit　　中 フォン反射　　右 StandardShader

```
// 法線を正規化
float3 worldNormal = normalize(i.worldNormal);

// メインライトの方向を取得
float3 lightDir = normalize(_WorldSpaceLightPos0.xyz);

// 【新規追加】カメラの方向を計算（鏡面反射用）
float3 viewDir = normalize(_WorldSpaceCameraPos - i.worldPos);

// 【新規追加】反射方向を計算（鏡面反射用）
float3 reflectDir = reflect(-lightDir, worldNormal);

// 【変更】ランバート反射（拡散反射）の計算（元のまま）
float NdotL = max(0, dot(worldNormal, lightDir));
fixed4 diffuse = _Color * NdotL * fixed4(_LightColor0.rgb, 1.0);

// 【新規追加】鏡面反射の計算（フォン反射モデル）
float RdotV = max(0, dot(reflectDir, viewDir));
fixed4 specular = _SpecularColor * pow(RdotV, _Shininess) * fixed4(_LightColor0.rgb, 1.0);

// 【新規追加】環境光の計算
fixed4 ambient = _Color * _AmbientIntensity * fixed4(UNITY_LIGHTMODEL_AMBIENT.rgb, 1.0);

// 【変更】最終的な色を合成（環境光 + 拡散反射 + 鏡面反射）
fixed4 col = ambient + diffuse + specular;
```

より現実的な反射の表現

物理学への理解が、**シェーダーの理解**につながる

UnityのStandardShader

PBR（物理ベースレンダリング）マテリアル

より物理現象を再現したモデルを使います

- **Specular（鏡面反射）**: Cook-Torrance BRDF（を基に改良したもの）
- **Diffuse（拡散反射）**: Disney BRDF（を基に改良したもの）

これらは**物理学の理論から導かれたもの**

余談：BSSRDF

サブサーフェススキャッタリング（SSS）

実際の物理現象による拡散反射を計算するモデル

- 拡散反射の見え方を計算しているという点では、ランバート反射等と同じ種類
- より高度な物理現象の再現

まとめ

CGの世界では様々な物理学の理論が使われている

- かなり簡単に一部だけ説明したが、実際はもっと複雑
- 透過とかも一切説明してない
- 光学以外の物理学もたくさん使われている（例：物理演算）

最後に

物理学に興味を持ったら是非！

物理学集会へ参加してみてください！