# キャラクターの真下にShaderを使って丸い影を描く（投影丸影）

## ☒ この影の作り方のイメージ

> プレイヤーの下に「黒い丸いライト」を当てて、地面を**ちょっとだけ暗く見せる**という工夫です。
> ライトなので、授業でやった点光源がわかっていれば全く同じように実装できます。
> しかも、現在のシェーダーにコンスタントバッファと、影付け部分を足すだけでできます。

こんな感じの影です☒

```
  ← プレイヤー（ジャンプ中でもOK）
  ↓
黒い光を下に照らす
  ↓
――――――――――――――――

   ● ← 黒い影（地面に丸く表示される）
――――――――――――――――
```

---

## ☒ なぜこんな影を作るの？

本物の影は、光を遮ってリアルタイムに計算するので**とても重たい（処理が大変）**です。

でも、このやり方は：

| 項目 | 内容 |
|---|---|
| ☒ ゲーム性能 | とても軽い！（超高速） |
| ☒ 理解しやすさ | 仕組みが簡単！ |
| ☒ 実装方法 | ライトと同じように扱える |

---

## ☒ どうやって作るの？

# 丸影（Blob Shadow）導入フロー

---

## 1. ☒ 目的整理

- 丸影を「地面に貼り付けるような」影として合成する
- 地面用の描画パスに統合せず、「影合成処理」は別パスとして独立させたい
- プレイヤーの現在位置（XZ）と高さを使って「影の位置・サイズ・濃さ」を計算

---

## 2. ☒ 定数バッファの設計（CBShadow）

```
struct CBShadow {
    XMFLOAT4 casterPos;    // プレイヤーのXZ座標（Yは使わない：お空の方向を表すから）
    XMFLOAT4 shadowParams; // (softness, alphaScale, unused, playerHeightY)
};
```

- `.w` に高さを埋めることで1スロットで完結
- `b2` スロットなど空いてる定数バッファにバインド

---

## 3. ☒ HLSL シェーダー統合 (3Dのhlslの後ろに追加するなど？)

### ⚙ 丸影のアルファ合成ロジックをピクセルシェーダー末尾に追加

```
float2 casterXZ = casterPos.xz;
float2 pixelXZ = inData.wpos.xz;

float2 diff = pixelXZ - casterXZ;
float distSq = dot(diff, diff);

float softness = shadowParams.x;
float alphaScale = shadowParams.y;
float heightY = shadowParams.w;

float heightRatio = saturate(heightY / 2.0f); // 最大ジャンプ2.0f想定
float radius = lerp(0.4f, 1.0f, heightRatio);
float alpha  = lerp(0.6f, 0.1f, heightRatio);

float shadowAlpha = saturate((radius * radius - distSq) * softness) * alpha;

// 丸影を黒で合成（必要に応じて色も乗せられる）
float4 shadowColor = float4(0, 0, 0, shadowAlpha);
return lerp(resultColor, shadowColor, shadowAlpha);
```

---

## 4. ☒ C++側：描画前にプレイヤーの情報を渡す

### Stage::Draw や Stage::Update にて：

```
CBShadow shadowCB;
shadowCB.casterPos = XMFLOAT4(playerPos.x, 0.0f, playerPos.z, 1.0f);
shadowCB.shadowParams = XMFLOAT4(softness, alphaScale, 0.0f, playerHeightY);

// 書き込み → バッファ更新
context->UpdateSubresource(pCBShadow, 0, nullptr, &shadowCB, 0, 0);
context->PSSetConstantBuffers(2, 1, &pCBShadow);
```

- `playerHeightY = playerPos.y - Stage::GetTerrainHeight(playerPos.x, playerPos.z)`
- プレイヤーの高さを計算して `.w` に渡すのがポイント

---

## 5. ☒ 描画順序

1. 通常の地面 + モデル描画（`Model::Draw()`）
2. その後 `Stage.hlsl` のピクセルシェーダー内で影を合成（クワッド描画不要）

---

## 6. ☒ 確認と調整

| テスト項目 | 備考 |
|---|---|
| 影が正しい位置に出るか | XZが正しく渡っているか |
| 高さでサイズ変化するか | `.w` の補間式が効いているか |
| 透明度が変化しているか | `alpha` の計算式の係数調整 |
| カメラを回しても違和感ないか | 視差が出ないか、影が地面に貼りついて見えるか |

---

## ☒ 最終的な関数と構造の一覧

| 要素 | 内容 | 追加すべき場所 / クラス |
| --- | --- | --- |
| `CBShadow` | 丸影用の定数バッファ構造体 | `Engine/ShaderStruct.h` など定数バッファ定義系ヘッダ |
| `UpdateShadowCB()` | プレイヤー情報から `CBShadow` を更新・送信する関数 | `Stage` クラス、または `ShadowManager` を作ってもOK |
| `Stage.hlsl` | 丸影の合成コードを含む HLSL | `Assets/Shader/Stage.hlsl`（または新規に丸影対応シェーダ） |
| `GetTerrainHeight(x, z)` | 指定座標の地面高さを返す | `Stage` クラスに追加（地形データを持っているなら） |

---

# ☒ より具体的に解説

## ☒ 1. `CBShadow`

```
// ShaderStruct.h または ShadowStruct.h など
struct CBShadow {
    DirectX::XMFLOAT4 casterPos;    // プレイヤーのXZ座標
    DirectX::XMFLOAT4 shadowParams;  // (softness, alphaScale, -, playerHeightY)
};
```

- すでに `CBGlobal` や `CBLight` がある場所に追加すると整理しやすい

---

## ☒ 2. `UpdateShadowCB()` のような関数

**追加先:** `Stage.cpp` 内のメンバ関数 or `ShadowManager` クラスとして独立化もOK

```
void Stage::UpdateShadowCB(const Player& player)
{
    CBShadow cb;
    auto pos = player.GetTransform().position_;
    float terrainY = GetTerrainHeight(pos.x, pos.z);

    cb.casterPos = XMFLOAT4(pos.x, 0.0f, pos.z, 1.0f);
    cb.shadowParams = XMFLOAT4(softness, alphaScale, 0.0f, pos.y - terrainY);

    D3D11_MAPPED_SUBRESOURCE mapped;
    context->Map(pCBShadow_, 0, D3D11_MAP_WRITE_DISCARD, 0, &mapped);
    memcpy(mapped.pData, &cb, sizeof(cb));
    context->Unmap(pCBShadow_, 0);

    context->PSSetConstantBuffers(2, 1, &pCBShadow_);
}
```

---

## ☒ 3. `Stage.hlsl` 内のピクセルシェーダー後半に合成コード

**追加先:** `Assets/Shader/Stage.hlsl`

```
cbuffer ShadowParam : register(b2)
{
    float4 casterPos;
    float4 shadowParams; // (softness, alphaScale, -, playerHeightY)
}

...

// ピクセルシェーダー末尾
```

```
float2 delta = inData.wpos.xz - casterPos.xz;
float distSq = dot(delta, delta);

float softness = shadowParams.x;
float alphaScale = shadowParams.y;
float playerHeightY = shadowParams.w;
float heightRatio = saturate(playerHeightY / 2.0f);
float radius = lerp(0.4f, 1.0f, heightRatio);
float alpha  = lerp(0.6f, 0.1f, heightRatio);

float shadowAlpha = saturate((radius * radius - distSq) * softness) * alpha;
float4 shadowColor = float4(0, 0, 0, shadowAlpha);

return lerp(resultColor, shadowColor, shadowAlpha);
```

## ☒ 4. `Stage::GetTerrainHeight(x, z)` 関数

**追加先:** `Stage.h` / `Stage.cpp`

```
float Stage::GetTerrainHeight(float x, float z) const
{
    // 例えば地形がグリッドなら、高さマップや地面モデルから高さを補間して返す
    return heightMap.SampleAt(x, z);
}
```

# ☒ 補足：必要な DirectX11 の初期化項目

| 名前 | 概要 |
| --- | --- |
| `ID3D11Buffer* pCBShadow_` | 丸影用定数バッファ |
| `CreateBuffer()` で生成 | 初期化時に `sizeof(CBShadow)` を渡す |

Revision #12
★Created 20 June 2025 15:58:00 by youe2
Updated 13 June 2026 04:01:09 by youe2