`sample()` の説明

入力：`dir`：レイ（光線）の方向

1：球面座標（経緯度）を計算

```
vec3 d = dir.normalized();
float phi = atan2(d.z, d.x);   // [-π, π]：水平方向の角度（経度）
float theta = acos(d.y);       // [0, π]：垂直方向の角度（緯度）
```

- `phi` は地球を一周する方向（Y軸まわりの回転角）
- `theta` は北極から赤道へ見下ろす角度（Y=1 から下方向）

2：UV座標に変換（[0,1] に正規化）

```
// [-π, π] を [0, 1] にマッピング
float u = (phi + M_PI) / (2 * M_PI);
// [0, π] を [0, 1] にマッピング
float v = theta / M_PI;
```

この処理により、球面座標を画像上の 2D 座標に変換します。

3：ピクセル座標にマッピング

```
int x = std::min(int(u * width), width - 1);
int y = std::min(int(v * height), height - 1)
```

- `u`, `v` を画像の整数ピクセル位置に変換
- `min()` を使って画像の端を超えないように制限

4：RGBピクセル値の読み取り

```
int idx = (y * width + x) * 3;
return vec3{
    image_data[idx]     / 255.f,
```

```
    image_data[idx + 1] / 255.f,
    image_data[idx + 2] / 255.f
};
```

- 各ピクセルは3バイト（RGB）で構成されている
- `255.f` で割ることで、`uint8_t` の値を `[0.0, 1.0]` に正規化

指定方向に対応する背景色を返す

この関数を使えば、任意のレイ方向に基づいて背景画像から色を取得し、リアルな背景表現が可能になります。