ランダムな三角形で地形を作っていく

☒ 地形の高さをランダムに決めて、頂点を作ろう！

☒目的：

地面の形を作るには、まず「高さのある点（＝頂点）」をたくさん並べます。
そしてこの点の高さをランダムに決めることで、でこぼこした地形になります。

☒ 使うデータ

前回までに作った `TerrainParams` を使います。

```
struct TerrainParams {
    int width = 128;      // 横に何個頂点を並べるか
    int height = 128;     // 奥に何個頂点を並べるか
    float scale = 1.0f;   // 1マスの広さ（距離）
    float heightScale = 10.0f; // 高さの最大値
};
```

☒ 乱数で高さを作る

C++では乱数を使って、毎回ちがう地形にすることができます。

```
std::mt19937 rng(std::random_device{}()); // ランダムの元
std::uniform_real_distribution<float> heightDist(0.0f, 1.0f); // 0〜1の高さ
```

☒ 頂点を作るコード

ここで、地面の「点（Vertex）」を作ります。

```
for (int z = 0; z < params_.height; ++z) {
    for (int x = 0; x < params_.width; ++x) {
        float y = heightDist(rng) * params_.heightScale; // ランダムな高さ！

        Vertex v;
        v.position = {
            x * params_.scale - halfWidth,  // X座標（左から右）
            y,                              // Y座標（高さ）
            z * params_.scale - halfHeight  // Z座標（手前から奥）
        };
        v.normal = { 0, 1, 0 }; // 法線（とりあえず上）
        v.uv = {
            static_cast<float>(x) / (params_.width - 1),
            static_cast<float>(z) / (params_.height - 1)
        };

        vertices_.push_back(v); // 頂点リストに追加！
    }
}
```

☒ なにが起こってるの？

1. `for` でグリッド状に頂点を並べる
2. 1つ1つに、ランダムな高さ `y` をつける
3. 地面の中心が (0, 0, 0) に来るように位置を調整
4. 頂点データを `vertices_` に入れる

☒ 実行するとどうなる？

こんな感じの地形ができます☒（イメージ）

```
高い ▦
   ▦   ▦
▦  ▦  ▦   ▦
  ▦▦▦▦▦▦▦ ← ランダムな高さででこぼこ
```

頂点作ったら、インデックスを考える

☒ 1. まずは四角を考えよう

たとえば、次のような 2×2 の四角形（セル）があります：

```
点の番号（vertices_ のインデックス）

↑ z方向

0———1
| ／ |
| ／ |
2———3    → x方向
```

この 4つの頂点から、2枚の三角形を作ります。

☒ 2. 三角形の作り方（インデックス配列）

1. 左下の三角形 → 点 0, 2, 1
2. 右上の三角形 → 点 2, 3, 1

※この順番（左回り／反時計回り）が**「表面」になるためのルール**です！

```
// C++ではこう書く
indices_.push_back(i0); // = 左上の頂点
indices_.push_back(i2); // = 左下の頂点
indices_.push_back(i1); // = 右上の頂点

indices_.push_back(i2); // = 左下
indices_.push_back(i3); // = 右下
indices_.push_back(i1); // = 右上
```

☒ 3. 地形全体のループでこれを繰り返す！

```
for (int z = 0; z < height - 1; ++z) {
    for (int x = 0; x < width - 1; ++x) {
        int i0 = z * width + x;       // 左上
        int i1 = i0 + 1;              // 右上
        int i2 = i0 + width;          // 左下
        int i3 = i2 + 1;              // 右下

        // 三角形①: 左上 → 左下 → 右上
        indices_.push_back(i0);
        indices_.push_back(i2);
        indices_.push_back(i1);
```

```
        // 三角形②: 左下 → 右下 → 右上
        indices_.push_back(i2);
        indices_.push_back(i3);
        indices_.push_back(i1);
    }
}
```

☒ 補足：なぜこうするの？

GPUに渡すときには、**「三角形の頂点3つ」のセット**として教える必要があるからです。
たとえば「地面を描きたい」と思ったら、こうして三角形の集合（メッシュ）として組み立てます。
順番逆にしちゃったりして、ポリゴンが表示されたりされなかったりするときは、ラスタライズステート（Direct3D.cpp）の設定をワイヤーフレーム＋カリングなし、にしてみよう。

Revision #1
★Created 25 June 2025 06:46:21 by youe2
Updated 2 June 2026 18:32:40 by youe2