アピールポイント

①フックの法則を用いた3Dソフトボディの実装
(SoftBodyBox.cpp)

②頂点バッファとインデックスバッファを自前で計算
(RenderSoftBody.cpp)

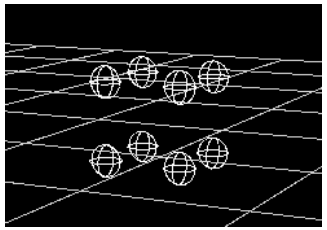
③シャドウマッピングでのキャラクターの影描画

④シングルトンでリソースの管理

⑤jsonファイルでキャラクターパラメータなどを管理

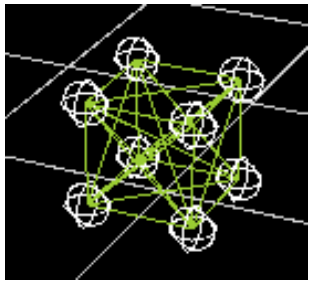
アピールポイント

①フックの法則を用いた3Dソフトボディの実装



質点を作成

斜め上から見た図



縦,横,斜め,手前,奥のバネを生成

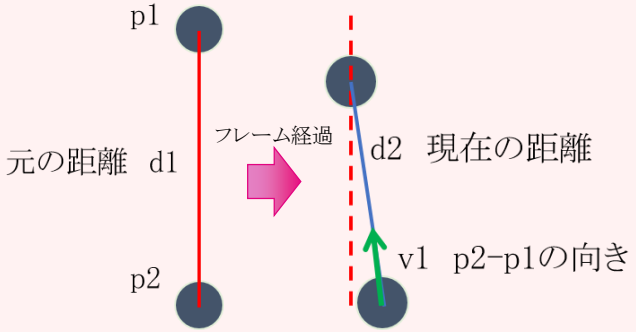
毎フレームバネの力を計算

バネの力「F=－kx」

大きさと向きを含むベクトルで弾性力を表すため「-」が付く。
xはばねの自然長からの伸び（変位）を求める。

x ＝ 頂点間の元の距離 － 頂点間の現在の距離
k ＝ 繋げる頂点から見たもう片方の頂点の向き

公式に代入し、
その後弾性率を掛けて力の強さを調整。



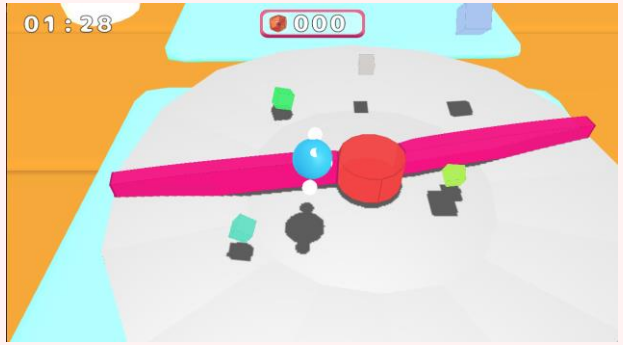
抵抗力は前フレームのバネの力の
逆ベクトルに抵抗率を掛けて計算。

アピールポイント

②頂点バッファとインデックスバッファを自作

複雑な形状に対して
１枚のテクスチャを描画。
ゼリーの表現の幅が広がる。

③シャドウマッピングで影を描画



まず描画するピクセルの座標をライト空間→
正規化デバイス座標に変換します。
その後Z座標を深度値と比較し、影になるところは
ライトの明るさをディフューズ色に掛けて描画する
ことで影を作成。

アピールポイント

④シングルトンでリソースの管理

必要なリソースを一度だけ読み込み、
unordered_mapに格納することで
keyを指定して呼び出すだけで画像やモデル、サウンドを取得できる。

```
// ジャンプ音
m_jumpSound = Resources::GetInstance()->GetSound(L"Jump");
```

⑤jsonファイルでキャラクターパラメータなどを管理

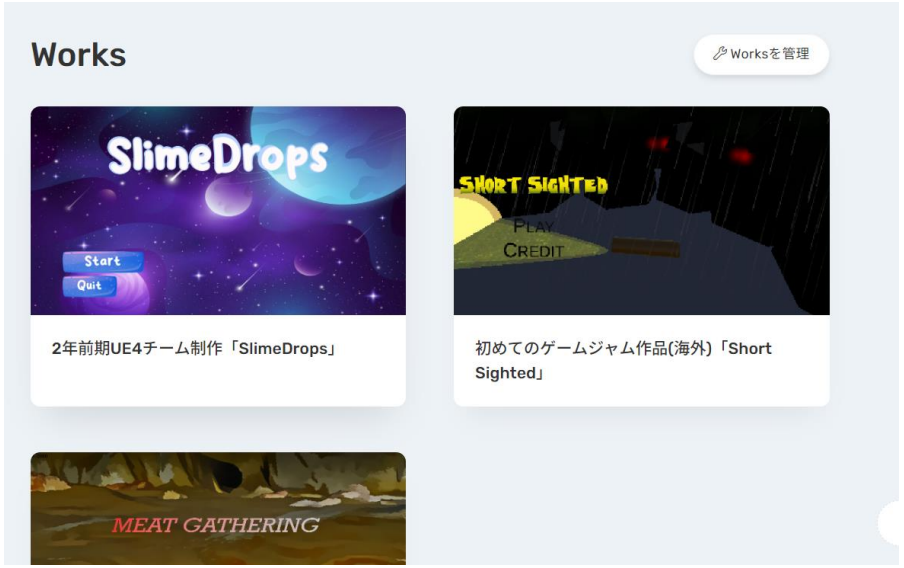
外部データで管理することで、デザイナーが設定しやすいように。

```
{
    "START_POSITION": [ [ 0.0, 0.1, 5.0 ] ],
    "MASS"          : 0.1,
    "GRAVITY"       : 0.12,
    "RADIUS"        : 0.9,
    "FRICTION"      : 0.98,
    "MOVE_SPEED"    : 0.005,
    "DECCEL_RATIO"  : 2.0,
    "JUMP_FORCE"    : 0.3
}
```

開発ブログ・ポートフォリオサイト

「C++でソフトボディ（ゼリーやスライムみたいなプルプルの動き）を作成する」

note:



ポートフォリオサイト「Resume」：

note:https://note.com/gentle_murre959/n/n3b4c53b0439a?sub_rt=share_pw
Resume:https://www.resume.id/mozu/works