# へびゲーム

## 体験入学会プログラミング体験教材

### はじめに

本日は体験入学会にお越しいただきありがとうございます。
プログラミング初心者向けの体験授業として、C++を使った「スネークゲーム」を完成させます。
コード中の `TODO` マークに沿って穴埋めし、動くゲームを作りましょう。

## ＨＥＢＩＧＡＭＥを動かしてみよう

### ✔まずはいったん遊んでみよう

- スペースキーでスタート
- キーボードのアローキー（⬆⬇⬅➡）で方向転換
- エサを食べると成長するよ！
- 自分の体を食べちゃったり、壁に当たるとゲームオーバー
- スペースキーでリスタート

---

### Step 1: 背景色を設定しよう（TODO ＃1）　62行目付近

ゲーム画面の最初に呼ばれる `Stage` クラスのコンストラクタ(初期化処理)内で、
`SetBackgroundColor` 関数を使って画面の背景色を設定します。

```
// TODO ＃1: ここに背景色を設定するRGBを入れてください
SetBackgroundColor( ____, ____, ____ );
```

- 各値は 0〜255 の範囲で指定します。
- 見やすい色（例：空色など）を選びましょう。

#### ヒント

`SetBackgroundColor(132, 255, 255);` は淡いシアン（空色）です。

---

### Step 2: ステージを描画しよう（TODO ＃2）　148行目付近

`Stage::Draw()` 内で、先ほど作ったグリッド状ステージを描画する関数を呼び出します。

```
void Stage::Draw()
{
    // TODO #2: ここにステージ描画関数を呼び出すコードを追加してください
    ____();

    if (!snake || !food)
        return;
    // 以下、蛇とフード、スコア描画...
}
```

#### ヒント

`DrawStageGrid()` を呼び出します。

---

## Step 3: スネークを作成しよう（TODO ＃3）　63行目付近

コンストラクタ内で `snake` ポインタに新しい `Snake` オブジェクトを生成します。

```
// TODO ＃3: Snakeオブジェクトを生成しよう
snake = ____;
```

### ヒント

`new Snake()` でインスタンス化します。

---

## Step 3.5: 移動タイマーを設定しよう（TODO ＃3．５）　19行目付近

スネークの移動間隔を決めるタイマーを設定します。

```
// TODO ＃3.5: スネークの移動間隔を設定しよう
SetMoveTimer( ____ );
```

### ヒント

定数 `MOVETIME` を使いましょう。

---

## Step 4: フードを作成しよう（TODO ＃4）　64行目付近

コンストラクタ内で `food` ポインタに新しい `Food` オブジェクトを生成します。

```
// TODO ＃4: Foodオブジェクトを生成しよう
food = ____;
```

### ヒント

`new Food()` を使います。

---

## Step 5: キーボード入力を処理しよう（TODO ＃5）120行目付近

`Stage::Update()` 内で矢印キーを押したときにスネークの進行方向を変更します。

```
if (Input::IsKeyDown(KEY_INPUT_UP))
    snake->SetDirection( ____ );
if (Input::IsKeyDown(KEY_INPUT_DOWN))
    snake->SetDirection( ____ );
if (Input::IsKeyDown(KEY_INPUT_LEFT))
    snake->SetDirection( ____ );
if (Input::IsKeyDown(KEY_INPUT_RIGHT))
    snake->SetDirection( ____ );
```

### ヒント

`Direction::UP` , `DOWN` , `LEFT` , `RIGHT` を使います。

---

## Step 6: ヘビを伸ばそう（TODO ＃6）　131行目付近

スネークがフードを食べたら、体を伸ばすメソッドを呼び出します。

```
if (snake->GetHeadPos() == food->GetPosition()) {
    // TODO ＃6: ヘビを伸ばす関数を呼び出そう
    snake->____();
    food->SpawnRandom();
```

```
    // スコア加算へ（TODO #8）
}
```

### ヒント

`Snake` クラスの `Grow()` メソッドを呼び出します。

---

## Step 7: スコアを描画しよう（TODO ＃7）　155行目付近

`Stage::Draw()` 内で画面上にスコアを表示します。

```
// TODO ＃7: スコアを描画しよう
DrawScore();
```

### ヒント

すでに用意した `DrawScore()` を呼び出します。

---

## Step 8: スコアを加算しよう（TODO ＃8） 136行目付近

フードを食べたときにスコアを増加させます。

```
if (snake->GetHeadPos() == food->GetPosition()) {
    // ヘビを伸ばす
    // TODO ＃8: スコアを加算しよう
    Score::AddScore( ____ );
    food->SpawnRandom();
}
```

### ヒント

1 回のフード取得につき `1` 点加算しましょう。

---

## Step 9: 壁との当たり判定をしよう（TODO ＃9）　110行目付近

`Stage::Update()` 内でスネークがステージ外に出たらゲームオーバーにします。

```
ivec3 pos = snake->GetBody()[0].GetPosition();
// TODO ＃9: 壁に当たったかチェックして、当たったら死亡処理を呼ぼう
if ( CheckHitWall( vec3(pos.x, 0, pos.z) ) ) {
    snake->SetDeath();
}
```

### ヒント

`CheckHitWall()` を使って判定します。

---

## Step 10: フードをスポーンしよう（TODO ＃4.5）　92行目付近

ステージ読み込み後にフードをランダムな位置に配置します。

```
if (food) {
    // TODO ＃4.5: フードをランダムにスポーンさせよう
    food->____();
}
```

### ヒント

`SpawnRandom()` メソッドを呼び出します。

---

## Step 11: フードを更新しよう（TODO ＃11）　142行目付近

`Stage::Update()` 内でフードのアニメーション更新を呼び出します。

```
// TODO ＃11: フードの更新処理を呼び出そう
food->____();
```

### ヒント

`Update()` メソッドを呼び出します。

---

以上でスネークゲームの基本機能が完成します！
コードをビルドして、動作を確認してみましょう。1 つずつクリアしながら進めてください。

## Step 12: ゲームオーバー時のシーン切り替え（TODO ＃12）　117行目付近

スネークが死亡したときに、別のシーンへ切り替えます。

```
if (!snake->IsAlive())
{
    // TODO ＃12: 死亡時にシーンを切り替えるコードを追加しよう
    SceneManager::ChangeScene();
}
```

### ヒント

- `SceneManager` クラスの `ChangeScene()` メソッドを呼び出します。
- 遷移先のシーンはあらかじめ設定されていることを確認してください。

以上で、全 12 ステップの穴埋め教材が完成です！ コードをビルドし、ゲームオーバー時にも正しくシーンが切り替わるか確認してみましょう。 コードをビルドして、動作を確認してみましょう。1 つずつクリアしながら進めてください。

## ✔URL

https://github.com/youetsux/HEBIGAME_TAIKEN.git

---

Revision #5
★Created 7 June 2025 23:52:25 by youe2
Updated 2 June 2026 18:31:27 by youe2