最終作品

作品名　　: **Project:NeaR**

ジャンル　: 3Dアクション

開発環境　: DxLib/Visual Studio 2022

言語　　: C++

制作期間　: 3ヶ月

所要時間数の概算　　:約510時間

開発メンバー数と構成　: 1人（プログラマー）

担当箇所　: プログラム全般

動作環境　: Windows

制作意図　:3D開発の基礎から高度なシステム実装までの習得を目的として制作をしました。

Project:NeaR

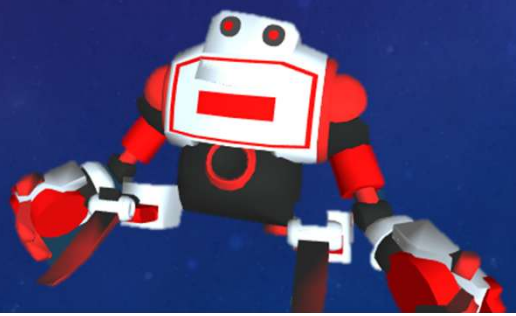
操作説明

ゲーム概要

■目標はステージの最後にいる**ボスを倒す**こと

■**剣**と**大剣**、**射撃**を駆使して敵を倒そう！！

■敵の攻撃を**ジャスト回避**して、そこから**ジャスト回避攻撃**を繰り出し大ダメージを狙おう！！



▼敵を倒しボス部屋を目指す！！

▼ボスと激闘！！

▼ステージクリア！！

▼いざボス戦！！

▼ボス撃破！！

ここが面白い①

誰でも簡単にかっこいいアクションができる！！

- ボタン連打でかっこいい連続攻撃ができる
- ジャスト回避が簡単に出せる

▼ボタン連打でコンボ

◀ジャスト回避から・・・

ジャスト回避攻撃！！▶

ここが面白い②

個性あふれるボスとの戦闘！！

- ボス毎に攻撃方法や行動パターンが違う
- ボス毎の攻略法を考えるのが楽しい

弾幕を出してくるボス

巨大ボス

第二形態に変身するボス

Project:NeaR

かっこいいアクションで
ボスを撃破せよ！！


技術紹介

技術的に1番頑張ったこと

Project:NeaR

プレイヤーの手触りを良くすること

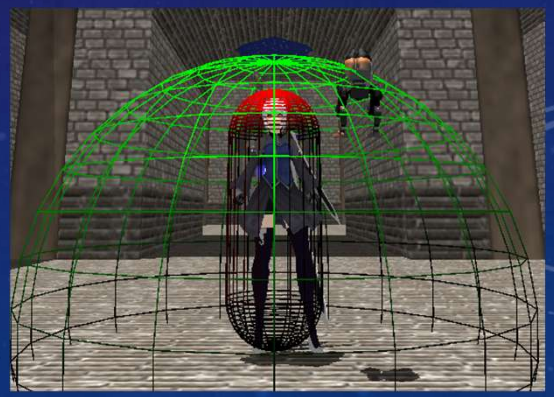
そのためにやった技術的挑戦

①先行入力

▼ボタン連打！！　▼入力を保持

②入力キャンセル

③回避用判定



技術的に1番頑張ったこと:①先行入力

Project:NeaR

入力を一定フレーム数保持する先行入力の仕組みを実装しました。

行動不可中に行われた入力を保存し、次の行動に遷移可能になったタイミングで処理することで、入力の取りこぼしを防いでいます。

その結果、コンボのつながりが良くなり、
快適な操作感を実現しました。

▼新しく入力が入った、または入力が残っているか

```
void Input::UpdateBuffer()
{
    //押された入力のフレームを減らしていく
    for (auto& [key, frame] : m_inputBufferFrame)
    {
        //フレームを減らしていく
        if (frame > 0.0f)
        {
            --frame;
        }
    }

    //新しく押された入力をバッファに登録
    for (const auto& keyInfo : m_currentInput)
    {
        if (IsTrigger(keyInfo.first))
        {
            m_inputBufferFrame[keyInfo.first] = kBufferFrame;
        }
    }
}
```

▼攻撃可能になったら即座に攻撃

▼ボタン連打！！　▼入力を保持

▼攻撃キーの入力が残っていたら攻撃する

```
//攻撃
if (input.IsBuffered("X"))
{
    ChangeState(std::make_shared<PlayerStateLightAttack>(m_pO
    return;
}
if (input.IsBuffered("Y"))
{
    ChangeState(std::make_shared<PlayerStateHeavyAttack>(m_pO
    return;
}
```

技術的に1番頑張ったこと：②入力キャンセル

攻撃・回避・待機などの行動をStateとして管理し、
攻撃後の後隙をキャンセルして次の行動へ
遷移できる仕組みを実装しました。

各行動を状態として分離したことで、入力に応じた
遷移制御が明確になり、テンポの良いアクション
を実現できました。

▼キャンセルフレーム中に入力に応じて遷移

```
//キャンセルフレーム
if ((model->GetTotalAnimFrame() - m_attackData->GetCancelFrame()) < m_frame)
{
    //攻撃の条件
    bool isChargeAttack = m_chargeCountFrame >= kChargeFrame;
    bool isCombAttack = input.IsBuffered("X");

    //武器を持つ
    owner->HaveLightSword();

    //攻撃をするか
    if (isChargeAttack || isCombAttack)
    {
        NextAttack(isChargeAttack, owner, model);
        return;
    }
    //ジャンプ
    if (owner->IsJumpable() && input.IsBuffered("A"))
    {
        ChangeState(std::make_shared<PlayerStateJump>(m_pOwner, m_isWait));
        return;
    }
    //大剣攻撃
    if (input.IsBuffered("Y"))
    {
        ChangeState(std::make_shared<PlayerStateHeavyAttack>(m_pOwner, m_isWait));
        return;
    }
}
```

▼攻撃後

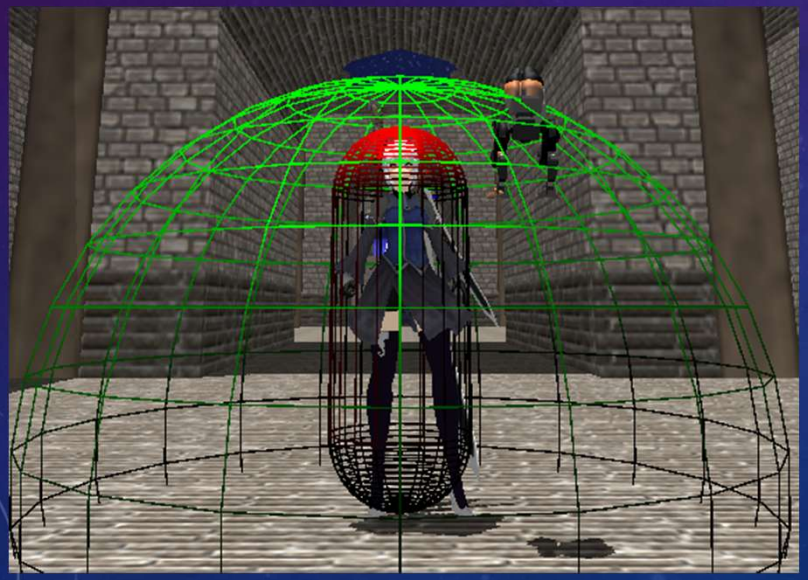
▼連続攻撃

▼回避

技術的に1番頑張ったこと：③回避用判定

喰らい判定とは別に回避専用の判定を実装しました。
これにより、入力タイミングに対する許容範囲が広がり、ジャスト回避を狙いやすくなっています。
その結果、プレイヤーがテンポよく行動でき、よりスタイリッシュなアクション体験を実現しました。

▼緑の円が回避用判定、赤のカプセルは喰らい判定



▼敵の弾を回避したとしてジャスト回避成功

技術的に2番目に頑張ったこと

カメラ制御

そのためにやった技術的挑戦

①スタック構造

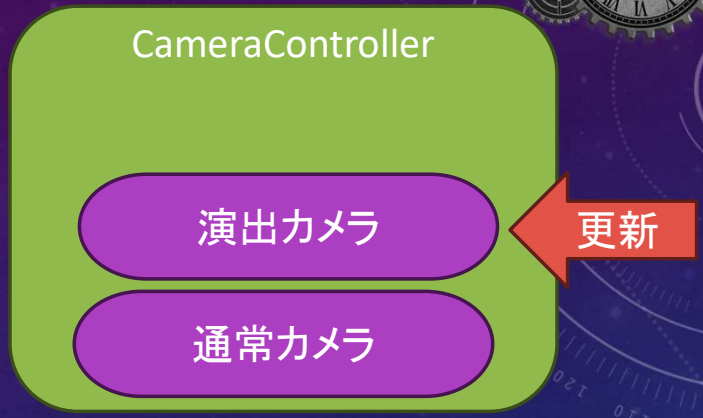
②ロックオンの挙動

③壁と床との当たり判定

技術的に2番目に頑張ったこと:①スタック構造

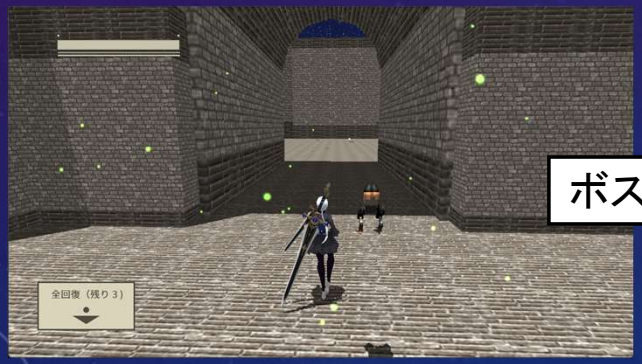
CameraControllerにカメラをスタック構造で管理し、常に最後にプッシュされたカメラのみを更新する仕組みを実装しました。

カメラ切り替え時には、前のカメラの座標や向きを引き継ぐことで、違和感のないスムーズな遷移を実現しています。



（例）通常のカメラから演出用カメラへ切り替え、演出終了後に元のカメラへ戻る

▼通常カメラ



ボス戦

▼演出カメラ

演出終了

▼通常カメラ

技術的に2番目に頑張ったこと:②ロックオンの挙動(1/2)

ロックオン時には敵の大きさに応じてカメラの高さと距離を動的に調整することで、敵のサイズに依存せず視認性の高い快適な戦闘を実現しました。

▼高さと距離を調整している部分

```
//高いほうに合わせる
basePos.y = MathSub::Max(MathSub::Max(playerPos.y,lockPos.y) , playerPos.y + kUpOffset);
//高さの差で距離を離す(でかい奴ほど引いた視点で見たいので)
basePos -= (toEnemyXZ * (kBackOffset + abs(lockPos.y - playerPos.y) * kBackRate));
//少し横から見たような視点にしたい
basePos += (playerRight * m_lockOnSide);
```

▼通常サイズの敵

▼大きいサイズの敵

技術的に2番目に頑張ったこと:②ロックオンの挙動(2/2)

ロックオン切り替え時には、プレイヤーからターゲットへの向きを基準に右スティックを倒した方向の敵を探索し、その中からカメラの向きに最も近い敵を選択することで、直感的なロックオン切り替えを可能にしています。

(例)右スティックを左に倒したとき

技術的に2番目に頑張ったこと:③壁や床との当たり判定

カメラの座標から注視点に向けてレイキャストを行い、
その間に壁や床のポリゴンが存在した場合は、ヒットした座標までカメラを移動させることで、カメラが壁にめり込まないようにしました。

技術的に3番目に頑張ったこと

Project:NeaR

見た目

そのためにやった技術的挑戦

①シェーダー

②アニメーションブレンド

技術的に3番目に頑張ったこと:①シェーダー

HLSL を用いてポストエフェクトを実装しました。
グリッチやモノクロといった機械的な映像表現を加えることで、
無機質でアンドロイドらしい世界観を演出しています。

▼グリッジ

▼ビネット

▼モノクロ

技術的に3番目に頑張ったこと：②アニメーションブレンド

キャラクターの状態に応じたアニメーションブレンドを実装しました。
移動・攻撃・回避などの遷移時に補間処理を行うことで、
アニメーションの急な切り替わりを防ぎ、操作感と見た目の両立を図っています。

▼前のアニメーション　　だんだん変化させる…　　▼次のアニメーション

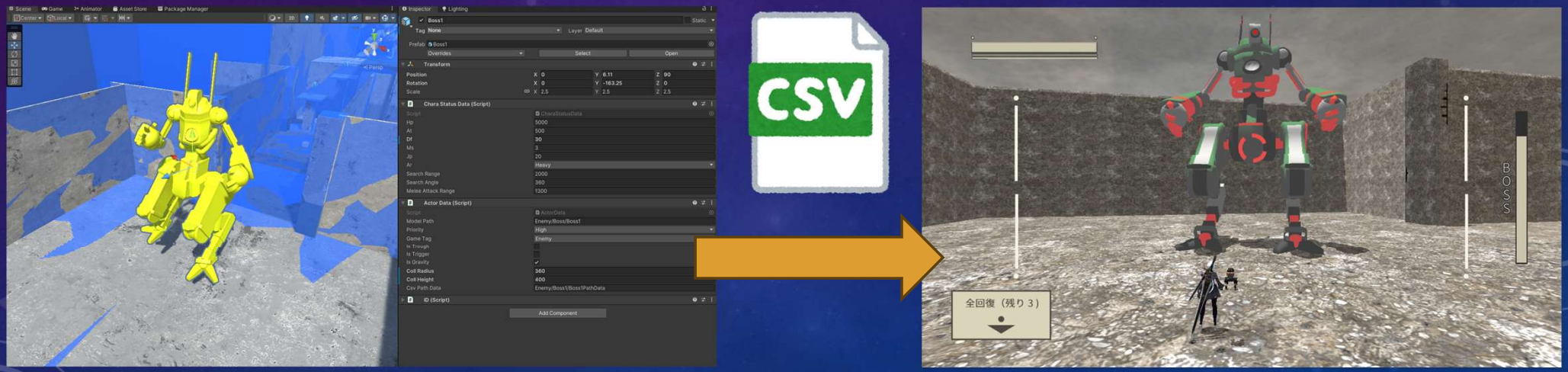
その他の技術的なこと

その他：① Unityをエディターとして使った

Project:NeaR

Unityで配置したオブジェクトの座標データなどをCSVに書き出して
それを読み込むことでステージやキャラの配置を簡単にできるようになりました。

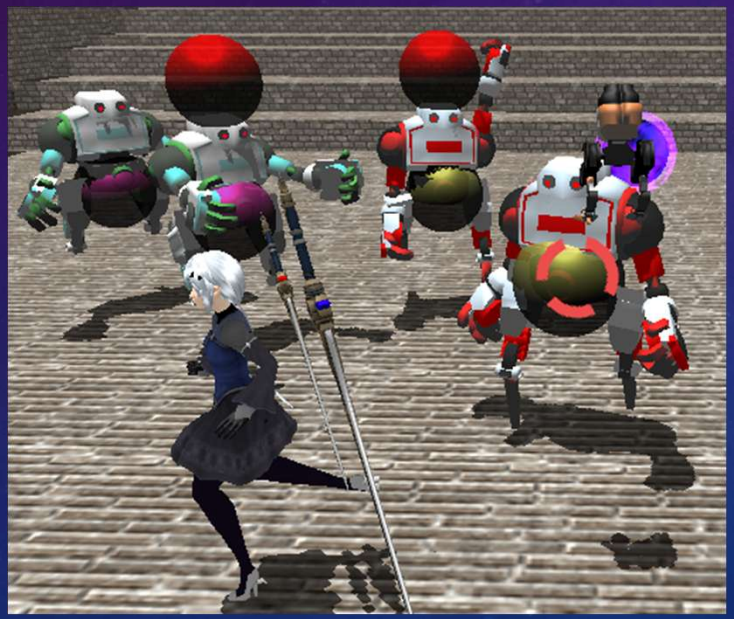
▼CSV書き出し/読み込み



その他：②グループ化

グループ内で攻撃権を順番に回すことで
プレイヤーを複数の敵が同時に攻撃しないようにしました。

▼画像では各グループをおなかの色で、攻撃権を頭の赤色の球で表しています。



▼グループ内で順番に攻撃権を回す

その他：③外部ファイル化

キャラクターの座標・大きさ・向きや攻撃、アニメーション情報などを
外部ファイルで管理しました。
これによってパラメータ調整や敵キャラの量産が効率よく行えるようになりました。

▼アニメーション

状態	武器	アニメーション
Idle	None	RobotArmature|Idle
Walk	None	RobotArmature|Walk
Run	None	RobotArmature|Run
Hit	None	RobotArmature|HitRecieve_1
HitAir	None	RobotArmature|HitAir
Fall	None	RobotArmature|Fall
FallHitAir	None	RobotArmature|HitAirFall
PunchAtta	None	RobotArmature|Punch
JumpAttac	None	RobotArmature|JumpAttack
JumpAttac	None	RobotArmature|JumpAttackComb
KickRight/	None	RobotArmature|KickRight
KickLeftA	None	RobotArmature|KickLeft
ShootAtta	None	RobotArmature|Shoot
Down	None	RobotArmature|Down
Death	None	RobotArmature|Death

▼攻撃

攻撃の名前	多段ヒット	攻撃タイプ	威力		攻撃の重さ	アーマー	ノックバッ	上方向の力	発生フレー	持続フレー	半径		長さ	前進速度
Punch	FALSE	1	10		3	3	10	0	40	20	200		200	0
KickRight	FALSE	5	10		3	3	10	0	40	10	500		0	0
KickLeft	FALSE	5	10		3	3	10	0	40	10	500		0	0
Shoot	TRUE	2	10		3	3	10	0	50	100	100		0	0
Shoot_2	TRUE	2	10		3	3	10	0	60	100	100		0	0
Shoot_3	TRUE	2	10		3	3	10	0	70	100	100		0	0
Shoot_4	FALSE	2	10		3	3	10	0	80	100	100		0	0
JumpAttac	FALSE	5	10		3	3	6	0	60	10	800		0	2
JumpAttac	TRUE	5	10		3	3	6	0	80	10	800		0	0
JumpAttac	TRUE	5	10		3	3	6	0	120	10	800		0	0
JumpAttac	FALSE	5	10		3	3	6	0	150	10	800		0	0

▼各ステージ情報

名前	ID	アクター	座標X	座標Y	座標Z	回転X	回転Y	回転Z	大きさX	大きさY	大きさZ
Player	147	Character	0	0	0	0	0	0	1	1	1
Boss3	290	Character	0	10	130	0	180	0	4	4	4
Robot1	273	Character	0	2.73	31.81	0	180	0	0.3	0.3	0.3
Robot1	36	Character	0	2.22	55	0	180	0	0.3	0.3	0.3
Robot1 (1)	338	Character	5.15	5.97	62.48	0	180	0	0.3	0.3	0.3
Robot1 (2)	138	Character	-5.35	5.83	60.74	0	180	0	0.3	0.3	0.3
Robot2	288	Character	2	4	66.02	0	196.75	0	0.3	0.3	0.3
Robot2 (1)	161	Character	-2	4	65.96	0	196.75	0	0.3	0.3	0.3