クレシタ crescita

crescita

制作期間 2023/10/1～2023/12/8
制作時間 約300時間
開発環境 Unity ver.2021.3.15f1 Photon
開発言語 Unity/C# PHPMySQL
対応機種 PC

紹介動画

作品説明
ハスクラ風MMOアクションRPG

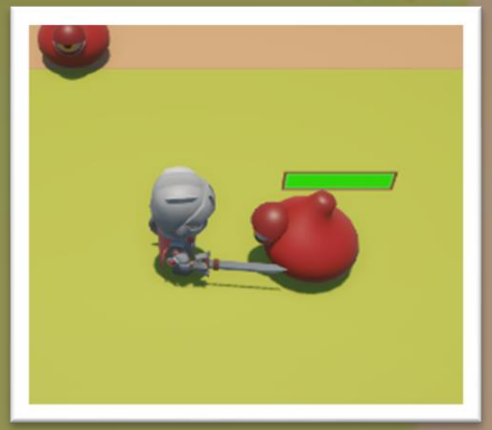
イチ押しポイント
Photonを使ったリアルタイム通信。
PHP,MySQLを使った通信。
最大4人でのマルチプレイ可能。

使用モデル
UnityAssetStore, Itch.io
使用音源
DOVA-SYNDROME, 効果音ラボ, 甘茶の音楽工房

GitURL
https://github.com/west-field/crescita.git

ゲーム内容①：ゲーム概要

- 敵を倒す



- ステージで素材を入手する

- 町で武器を強化する

- 強敵に挑戦だ！！

強くなったら

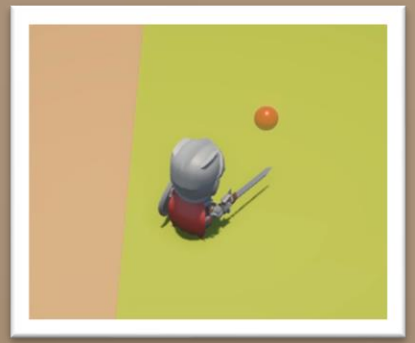
ゲーム内容②：一人プレイ

➢ ステージを選択

➢ 敵を倒す

敵は一定時間たつと復活する

➢ ステージで素材を入手する



➢ 町で武器を強化する

ゲーム内容③：マルチプレイ

- ステージを選択

- 同時に4人までプレイ可能

- 時間制限のあるステージで
制限時間内にボスを倒す

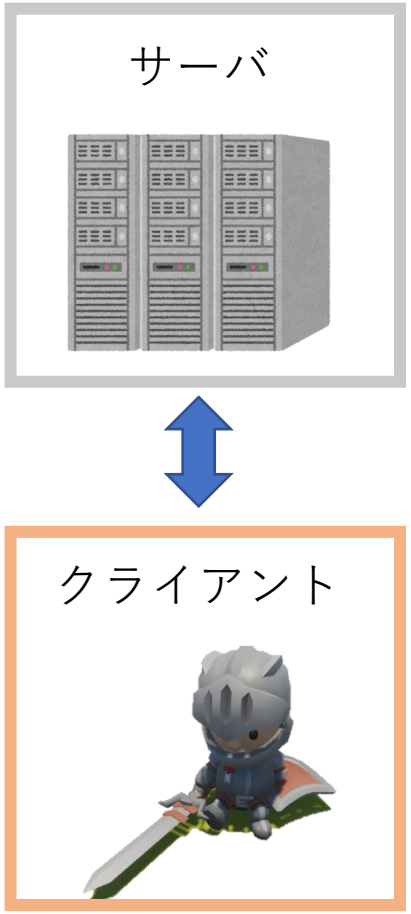
- 4人で集まって
強敵に挑戦だ!!

技術紹介：ネットワーク通信

以下の処理はネットワーク通信を用いて、
Webサーバー上のDBと連携して処理を行っています。

- ログイン
- アイテムデータのロードorセーブ
- ステータスのロードorセーブ
- ガチャ

サーバ環境はLAMPです。
(※LAMP：Linux,Apache,MySQL,PHP)



技術紹介①：ログイン

・UnityからPHP
入力された名前とパスワードを渡す

```
//Post通信　ログイン
1 個の参照
IEnumerator LoginRequest(string host,string postUrl,string name,string pass)
{
    //PostはURLのみの通信ではなく、URLに対してFormを渡すことで通信する
    WWWForm form = new WWWForm();
    form.AddField("name", name);
    form.AddField("pass", pass);

    string url = host + postUrl;

    using UnityWebRequest postRequest = UnityWebRequest.Post(url, form);

    yield return postRequest.SendWebRequest();
```

・PHPからデータベースにアクセス

```
try{
    $pdo = new PDO($dsn,$user,$password);
    //プリペアドステートメントのエミュレーションを無効にする
    $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
    //例外がスローされる設定にする
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    //echo "データベース{$dbName}に接続<br>";

    //SQL文を作る
    $sql = "SELECT * FROM {$tableName}";
    //プリペアドステートメントを作る
    $stm = $pdo->prepare($sql);
    //SQL文を実行する
    $stm->execute();
    //結果の取得(連想配列で受け取る)
    $result = $stm->fetchAll(PDO::FETCH_ASSOC)
```

id	name	password
1	test	0419
2	orias	0419
4	aaa	123456

・ PHPからUnity
名前とパスワードが同じかどうかを確認し結果を渡す

```
//名前とパスワードがあっているかを確認する
$isName = false;
$isPass = false;
$id = 0;
foreach($result as $temp)
{
    //名前が一致しているか
    if($name === $temp[$Colum2])
    {
        //名前が一致
        $isName = true;
        //パスワードが一致しているか
        if($pass === $temp[$Colum3])
        {
            //パスワードが一致
            $isPass = true;
            $id = $temp[$Colum1];
        }
    }
}

//接続を解除する
$pdo=NULL;
//echo "データベース{$dbName}の接続を解除<br>";


catch(Exception $e)
{
    echo "<span class='error'>エラーがありました。</span><br>"
    echo $e->getMessag
    exit();
```

```
if($isName && $isPass)
{
    echo "全て一致\n{$id}\n";
}
else
{
    echo "どちらかが間違っています\n";
}
```

・ Unity
渡された結果によって次の処理を変更する

```
//通信エラー処理
if (postRequest.result == UnityWebRequest.Result.ConnectionError)
{
    //通信失敗　エラー内容を表示
    Debug.Log(postRequest.error);
}
else
{
    //通信成功
    Debug.Log(postRequest.downloadHandler.text);
    var str = postRequest.downloadHandler.text;

    //送られてきたテキストを'¥n'で区切る
    string[] splitText = str.Split('¥n');

    nextText.text = splitText[0];
    if (splitText[0] == "全て一致")
    {
        Debug.Log("一致");
        HoldVariable.id = splitText[1];

        //アイテムデータを取得する
        itemData.GetItemData();
        //ステータスを取得する
        startas.GetStartasData();

        nextText.text = "ログイン成功";

        isLoginData = true;
    }
}
```

技術紹介②：アイテムデータをロードorセーブする

通信でデータベースからアイテムの数を取得します。

Unity

```
/// <summary>
/// アイテムのロード
/// </summary>
1 個の参照
IEnumerator LoadItemRequest(string url)
{
    WWWForm form = new WWWForm();
    form.AddField("id", HoldVariable.id);

    Debug.Log($"{HoldVariable.id}のアイテムロード");

    using UnityWebRequest postRequest = UnityWebRequest.Post(url, form);
    yield return postRequest.SendWebRequest();

    //通信エラー処理
    if (postRequest.result == UnityWebRequest.Result.ConnectionError)
    {
        //通信失敗　エラー内容を表示
        Debug.Log(postRequest.error);
        isLoad = false;
    }
    else
    {
        //通信成功　レスポンスを表示
        Debug.Log(postRequest.downloadHandler.text);//送られてきたテキストを表示

        var str = postRequest.downloadHandler.text;
        string[] splitText = str.Split('¥n');

        for (int i = 0; i < HoldVariable.itemDataList.Count; i++)
        {
            HoldVariable.itemDataList[i].count = int.Parse(splitText[i]);
            Debug.Log(HoldVariable.itemDataList[i].count);
        }

        isLoad = true;
    }
}
```

PHP

```
//SQL文を作る
$sql = "SELECT * FROM {$tableName}";
//プリペアドステートメントを作る
$stm = $pdo->prepare($sql);
//SQL文を実行する
$stm->execute();
//結果の取得(連想配列で受け取る)
$result = $stm->fetchAll(PDO::FETCH_ASSOC);

//名前とパスワードがあっているかを確認する
$isName = false;
$isPass = false;

foreach($result as $temp)
{
    if($temp["member_id"] === $id)
    {
        echo $temp["gold"]."\n";
        echo $temp["redStuffy"]."\n";
        echo $temp["redJigglyCore"]."\n";
        echo $temp["frogHorn"]."\n";
        echo $temp["frog'sPearl"]."\n";
        echo $temp["frog'sGoldenBeads"]."\n";
        echo $temp["frog'sBlackJewel"]."\n";
        echo $temp["lotusLeaf"]."\n";
        echo $temp["scarabSwallowtail"]."\n";
        break;
    }
}
```

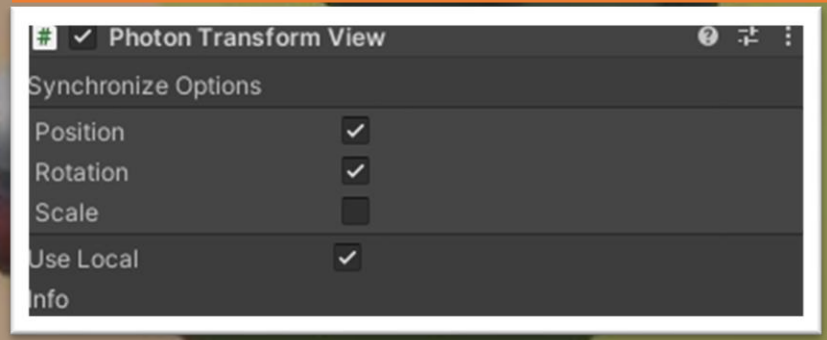
member_id	gold	redStuffy	redJigglyCore	frogHorn	frog'sPearl	frog'sGoldenBeads	frog'sBlackJewel	lotusLeaf	scarabSwallowtail
1	449	1	12	3	11	1	1	8	1
2	1930	5	5	5	1	0	0	0	5
4	100	0	0	0	0	0	0	0	0
5	100	0	0	0	0	0	0	0	0
6	100	0	0	0	0	0	0	0	0
7	100	0	0	0	0	0	0	0	0
8	100	0	0	0	0	0	0	0	0
9	100	0	0	0	0	0	0	0	0
10	100	0	0	0	0	0	0	0	0
11	100	0	0	0	0	0	0	0	0
12	2924	50	39	38	4	4	2	15	24
13	87336	123	70	78	2	15	2	10	4
14	100	0	0	0	0	0	0	0	0
15	100	0	0	0	0	0	0	0	0
16	34	2	0	0	0	0	0	0	0
17	61	3	5	4	0	2	0	0	1
18	774	3	8	0	0	0	0	0	0
19	160	3	2	3	0	0	0	5	0
20	20	0	0	1	0	0	0	0	0
21	30	0	3	0	0	0	0	1	0

ログインしたときに取得した
プレイヤーのIDを渡して、
そのプレイヤーのアイテムを
探しています。

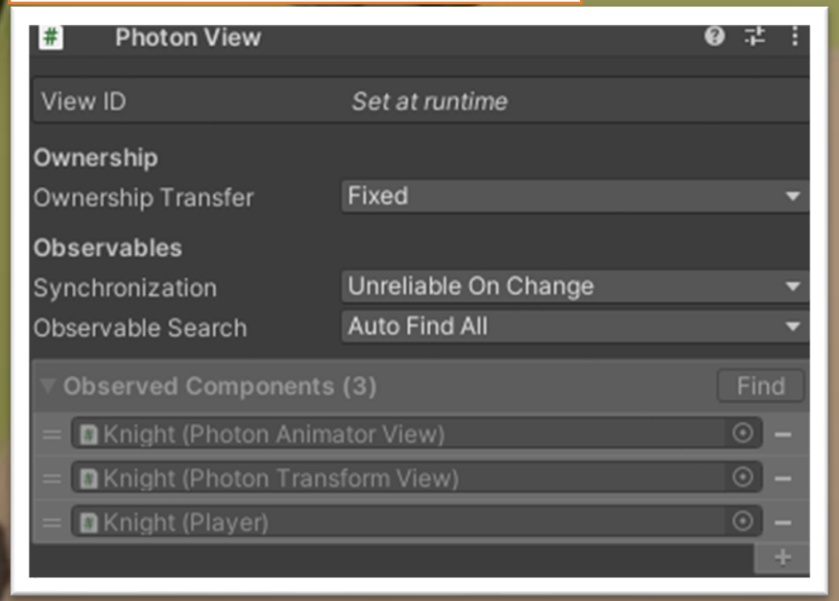
技術紹介③：マルチプレイ

PhotonUnityNetworking2を使用して制作しました。

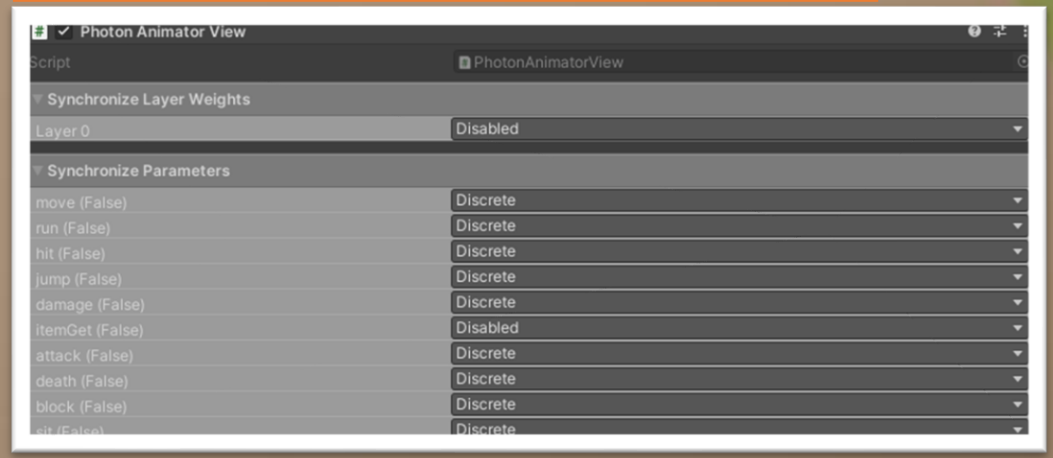
オブジェクトの座標を他プレイヤーと自動的に同期する



オブジェクト同期の基礎



アニメーションパラメータを同期する



技術紹介④：マルチプレイ　クライアント

マルチプレイ時に他プレイヤーの画面でも同じHPとなるように現在のHPを送受信する。

Player

```
//HPバー送受信
3 個の参照
void IPunObservable.OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    //このクライアントがPhotonViewの所有者である場合
    if (stream.IsWriting)
    {
        //送信する
        stream.SendNext(nowHp);//現在のHP
        stream.SendNext(isWeponEnable);//武器の当たり判定
    }
    else
    {
        //受信する
        nowHp = (int)stream.ReceiveNext();//現在のHPを変更する
        isWeponEnable = (bool)stream.ReceiveNext();//武器の当たり判定を変更する
    }
}

/// <summary>
/// 最大HPを送る
/// </summary>
/// <param name="hp">最大HP</param>
[PunRPC]
1 個の参照
private void MaxHp(int hp)
{
    maxHp = hp;
}
```

・武器の当たり判定を送受信し
　当たり判定を同期させている。

Enemy

```
//データ送受信
3 個の参照
void IPunObservable.OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    //このオブジェクトがPhotonViewの所有者である場合
    if (stream.IsWriting)
    {
        stream.SendNext(currentHp);//現在のHP
        stream.SendNext(isHpBarDraw);//HPバーの表示非表示
        stream.SendNext(isDeath);//死んだかどうか
    }
    else
    {
        currentHp = (int)stream.ReceiveNext();//現在のHPを変更する
        isHpBarDraw = (bool)stream.ReceiveNext();//HPバーの表示非表示を変更
        isDeath = (bool)stream.ReceiveNext();//死んだかどうかのフラグを変更
    }
}
```

・HPバーの表示非表示を送受信し
　他のプレイヤー画面でも同じものが
　見えるように同期させている
・死んだかどうかのフラグを送受信し
　誰かの画面で生き残らせないように同期させている。