

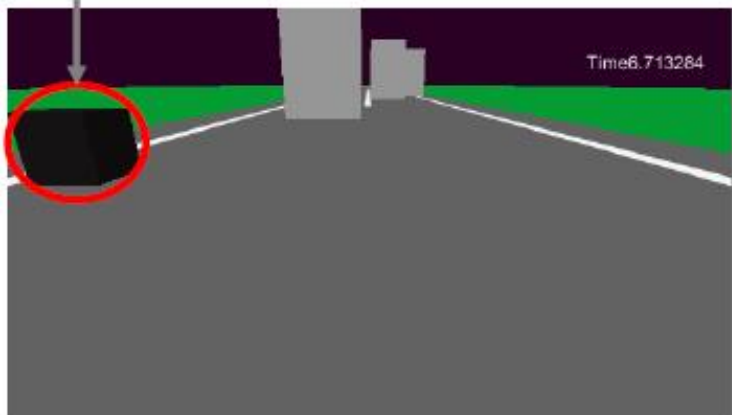
# 【体験会おまけコンテンツ】 シューティングゲームに にゴースト機能を追加しよう！

※このコンテンツを行う前に、mBaaSデータストアのScoreクラスを削除してください

- ゴースト機能とは？
- ゴースト機能追加の流れ
- ゴーストデータの作成・保存方法
  - プレイヤーの走行ログ生成
  - 走行ログのサーバーへの保存
- ゴーストデータの再生
  - 走行ログの引き出し
  - ゴーストの読み込み
  - 走行ログによるゴーストの操作



これがゴースト

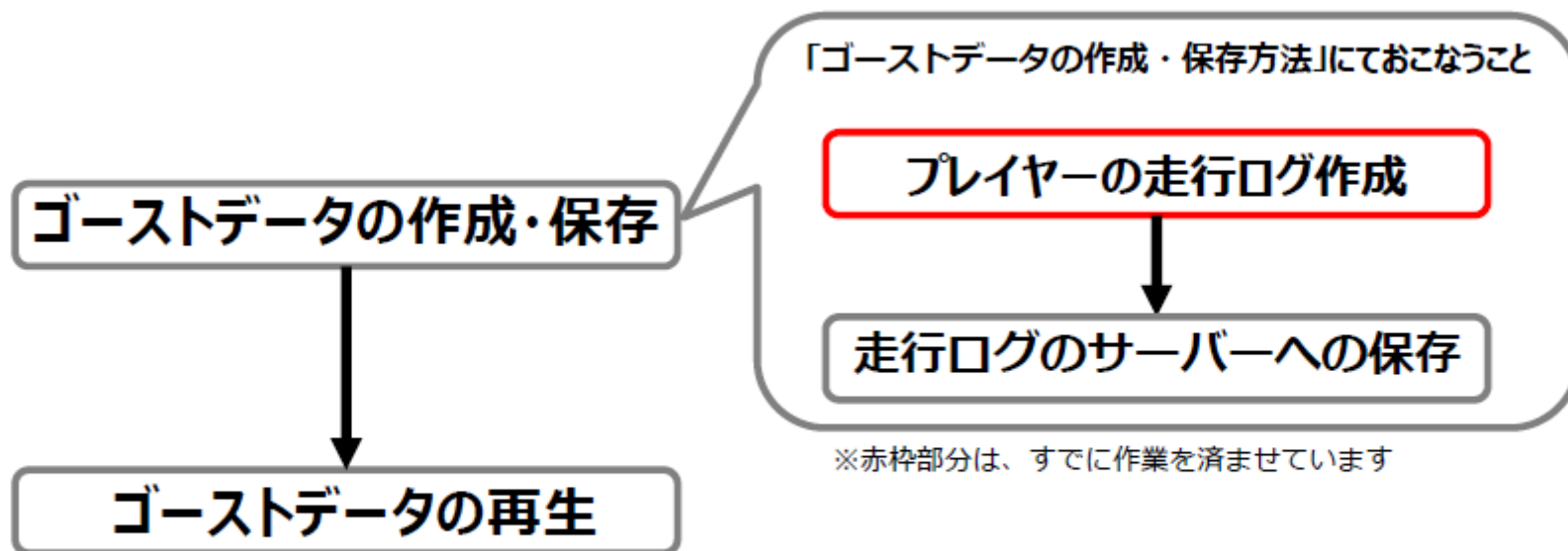


## ゴースト機能とは？

レーシングゲームや  
シューティングゲームで  
上手な人のプレイログを  
見ながら自分もプレイできる  
機能です。

ユーザーは自分の  
プレイ技術を向上させるために  
この機能を利用します。

これを体験会で作ったシュー  
ティングゲームに導入しましょ  
う！



※今回は本ページの作業を行う必要はありません。

Asset>Scriptsの「Player.cs」の下記のコードで走行ログを作成しています

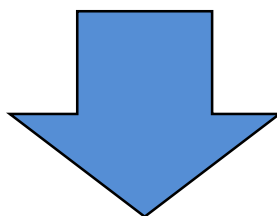
```
//---ゴーストをつくるため、ポジションをリスト化する-----  
float[] postion = new float[2];  
postion [0] = transform.position.x;  
postion [1] = transform.position.y;  
posList.Add(postion);  
//-----
```

上記のコードで、プレイヤーの現在地の座標を取得配列をリスト化しています。

スコアをサーバーに保存するコードを変更し、このposListも一緒に保存するようにします。

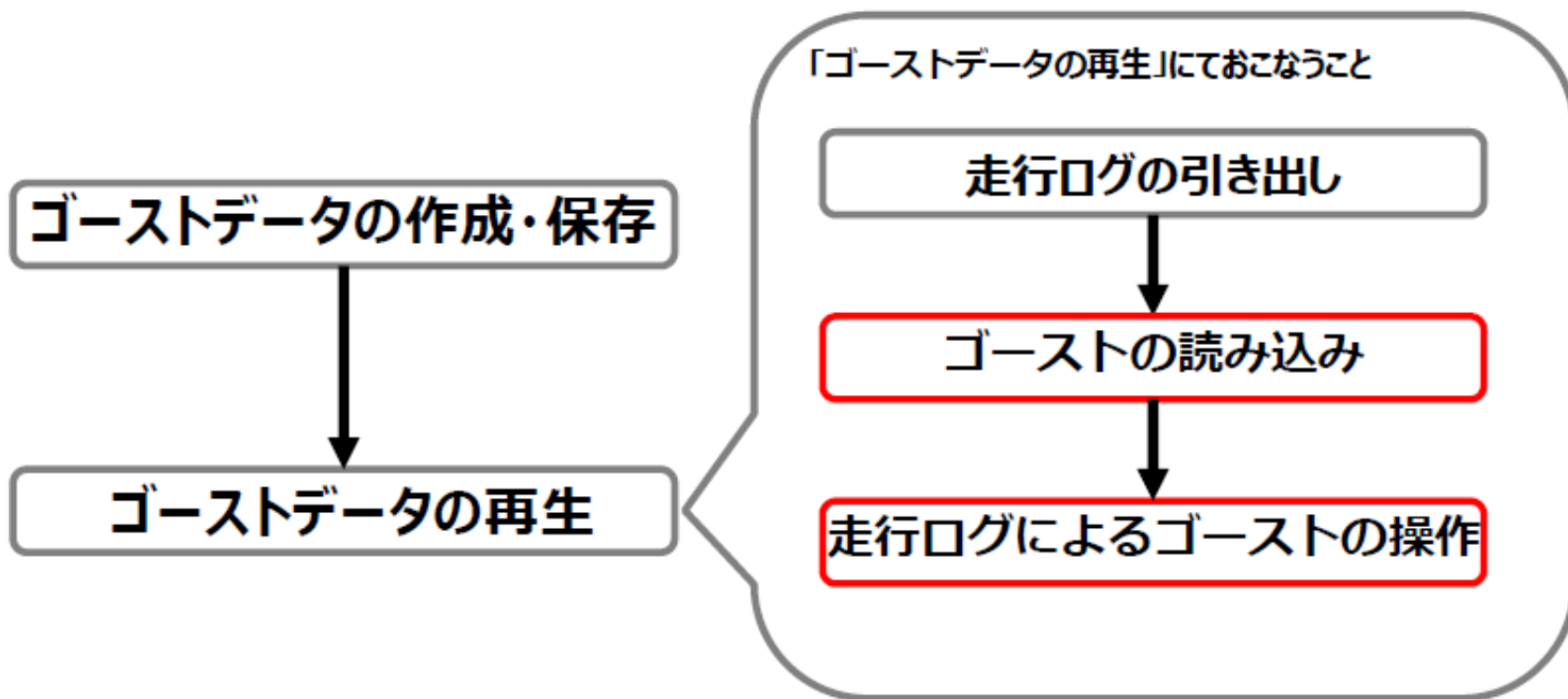
Asset>Scriptsの「SaveScore.cs」のvoid saveメソッド内の下記の部分を変更し走行ログもサーバーに保存できるようにします。

```
NCMBOBJECT obj = new NCMBOBJECT ("Score");  
obj ["name"] = name;//オブジェクトに名前とスコアを設定  
obj ["score"] = score;  
obj.SaveAsync ();//この処理でサーバーに書き込む
```



```
NCMBOBJECT obj = new NCMBOBJECT ("Score");  
obj ["name"] = name;//オブジェクトに名前とスコアを設定  
obj ["score"] = score;  
obj ["Log"] = Player.posList;  
obj.SaveAsync ();//この処理でサーバーに書き込む
```

それでは一回ゲームをプレイしてみましょう！



※赤枠部分は、すでに作業を済ませています



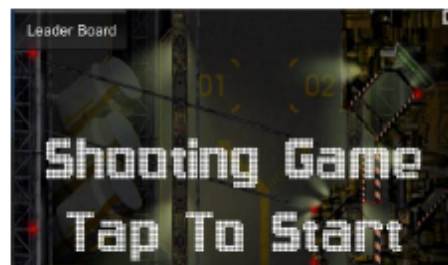
Asset>Scripts>Ghostの「Bg\_ghost.cs」のvoid Start(){}の中に  
下記のコードを実装してください

```
NCMBQuery<NCMBObject> query = new NCMBQuery<NCMBObject> ("Score");
query.OrderByDescending ("score");
query.Limit = 1;
query.FindAsync ((List<NCMBObject> objList ,NCMBException e) => {

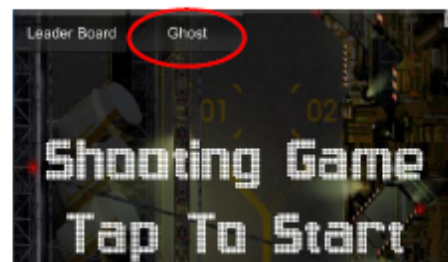
    if (e != null) {
        //検索失敗時の処理
    } else {
        //検索成功時の処理
        // 取得したレコードをscoreクラスとして保存
        foreach (NCMBObject obj in objList) {
            posObj = obj;
        }
        readyGhost = true;
    }
});
```

上記のコードは「ステージ」シーンにて駆動するように設定しています。  
走行ログの取得が完了したら、readyGhostというフラグを変更し、  
「Ghost」ボタンを表示するようにしています。  
「Ghost」ボタンを押下するとゴーストが表示されます。

左記コード実装前の画面



左記コードを  
実装すると



左記コード実装後の  
ログ取得後に「Ghost」ボタン  
が表示される



# ※参考資料※ゴーストの読み込み

※今回は本ページの作業を行う必要はありません。

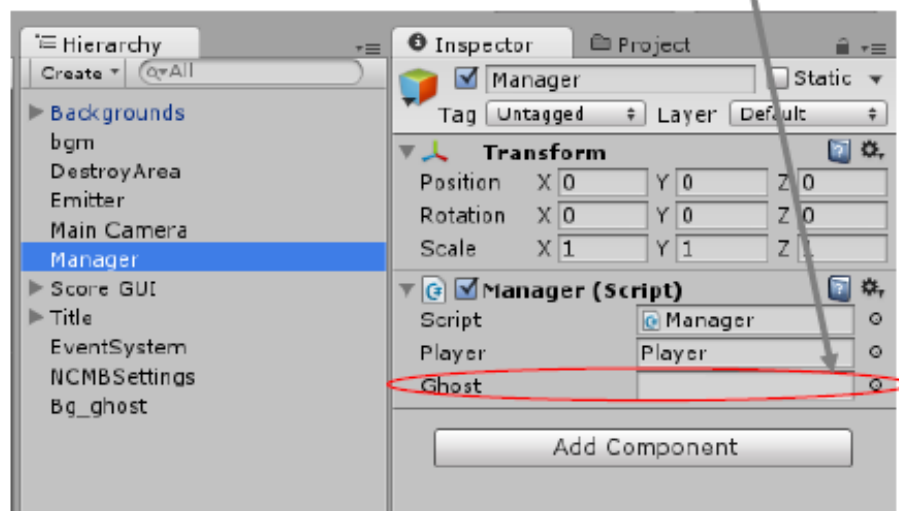
Asset>Prefabsに「Ghost」という、ゴーストとなるオブジェクトを作成しています(左図の赤丸内)。前頁の「Ghost」ボタン押下時にゴーストオブジェクトを読み込むようにします。

コードとしてはAsset>Scripts>Cameraの「Manager.cs」のvoid GameStartメソッド内の下記のコードにてPlayerオブジェクトの読み込みと同時に行っています。

```
if (withGhost == true) {  
    // ゴーストボタンを押下したらゴーストを表示する  
    Instantiate (ghost, ghost.transform.position, ghost.transform.rotation);  
    Instantiate (player, player.transform.position, player.transform.rotation);  
}
```

また、読み込みを行うためにもインスペクターにて設定が必要です。

「Manager.cs」をアタッチしているGameObject「Manager」の左図の場所にゴーストオブジェクトをドラック&ドロップしてください。



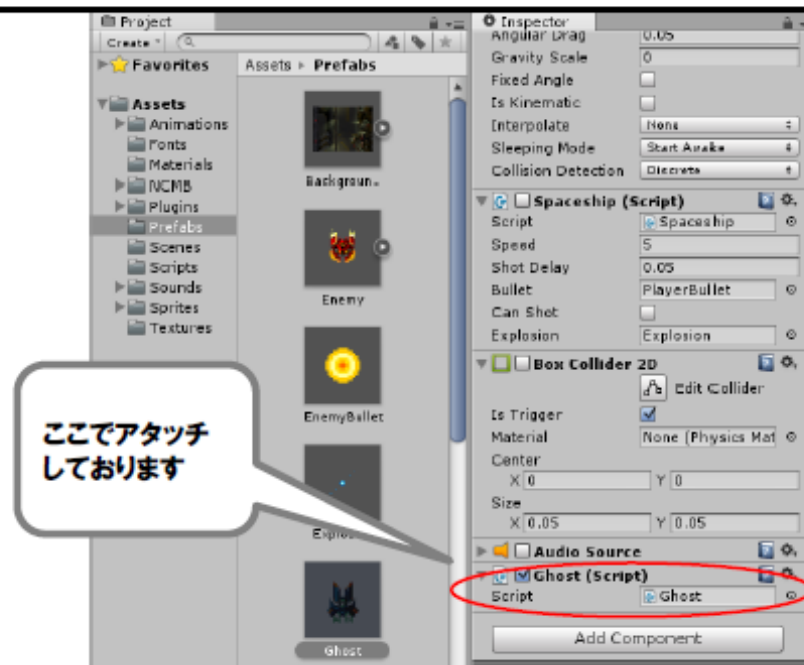
※今回は本ページの作業を行う必要はありません。

Asset>Scripts>Ghostの「Ghost.cs」のvoid Update(){}の中に実装している、下記のコードでゴーストオブジェクトを操作しています。

```
float x =  
    (float) System.Convert.ToDouble(((ArrayList)((ArrayList)Bg_ghost.posObj["Log"])[flameCount])[0]);  
float y =  
    (float) System.Convert.ToDouble(((ArrayList)((ArrayList)Bg_ghost.posObj["Log"])[flameCount])[1]);  
transform.position = new Vector2 (x,y);  
flameCount ++;
```

上記の「Ghost.cs」は前頁のゴーストオブジェクトにアタッチされています。  
左図のようにご確認ください。

「Ghost」ボタンを押して  
ゲームをスタートしてみてください！  
Ghostが表示されます。



@nifty



NIFTY  
Cloud  
ニフティクラウド

ありがとう 5周年

Going  
Next  
Stage

ニフティとなら、きっとかなう。  
With Us, You Can.

NIFTY Cloud  
ニフティクラウド