C#で逝く AtsEX 入門

T.Wattz

2024 年 5 月 8 日

こんにちは。T.Wattz です。
早速ですが、AtsEX とはなにかについて少し説明をしましょう。

What is AtsEX??

鉄道シミュレータ「BVE Trainsim」のプラグイン（Ats プラグイン）を拡張（ex tend）する C#ライブラリ。
DirectX にまで干渉できるほど拡張性は高く、知識と技術さえあれば基本的に何でも作れてしまうスゴい代物。
文化祭で公開したゲーム「MetroDrive 日比谷編」も本体にはこれを使用している。

お察しかとは思いますが、そこそこの程度の C#の知識は必要です。（Unity が満足に使えるレベルなら十分）
あとは、要所要所で Unity の類似機能を紹介しながらなので、Unity 勢にはわかりやすいかもしれません。

1. 主要機能の説明

まずは主要機能について軽く説明します。
読み飛ばして頂いても構いません。

PluginMain

PluginMain は Unity でいうところの Start 関数（厳密には違う）です。
ここで最初に一回呼び出したいものやイベントの登録などを行います。

リスト 1　Sample

```
1 public MapPluginMain(PluginBuilder
      builder) : base(builder)
```

と宣言することで使用可能になります。

Dispose

シナリオの終了時に呼び出す関数です。(Unity で言えば OnApplicationQuit() が近い.... かも)
イベントの購読解除とかに使います。

TickResult

Unity の Update 関数にあたるもので、シナリオが読み込まれた後毎フレーム呼び出されるものです。
詳細は後述しますが、BveHacker.Scenario などシナリオの実行中にしか読めない値はここでしか読めません。
基本的にはフレーム間で変わらないかもしれないもの（速度など）は毎フレーム代入するスタンスで OK です。

2. プラグインを作ろう

では、実際にプラグインを制作していきましょう。
今回の最終的な目標は「DirectX でマスコンの値、ブレーキの値を表示する」こととします。

環境設定 https://github.com/stop-pattern/AtsExCsTemplate を ¡code¿→ Download zip より Zip で落とし、任意の場所に展開しましょう。
次に、Visual Stadio をインストールします。（ここらへんはアップデートのたびにやり方が変わりそうなので、適当にググってください）
ワークロードは「.NET デスクトップ開発」をインストールすれば OK です。
インストールが済んだら MapPlugin\AtsExCsTemplate.sln を開いてください。
開いたら検索¿nuget パッケージの管理 より、「プレリリースを含める」にチェックを付け、参照¿AtsEX と検索して AtsEx.CoreExtions と AtsEx.PluginHost をインストールしてください。
AtsEX は執筆中の現在 (2024-05-07 時点) あくまで正式リリース**候補**なので、最新バージョンかどうかは https://www.okaoka-depot.com/AtsEX/download/ からご確認ください。（今時点では ver1.00-RC8 が最新バージョンです。）
では MapPlugin.cs を編集していきましょう。まずは最低限のコードでビルドできるか確認してみます。

リスト 2　MapPlugin.cs

```
using System;
using AtsEx.PluginHost.Plugins;
namespace BuildSample
{
/// プラグインの本体
[PluginType(PluginType.MapPlugin)]
internal class MapPluginMain :
      AssemblyPluginBase
{
    int sample;
    bool isFive;
    /// プラグインが読み込まれた時に呼ばれる
    /// 初期化を実装する
    public MapPluginMain(PluginBuilder
         builder) : base(builder)
    {
      sample = 1
      isFive = false;
    }
    /// プラグインが解放されたときに呼ばれる
    /// 後処理を実装する
    public override void Dispose()
    {}
    /// シナリオ読み込み中に毎フレーム呼び出
        される
    public override TickResult Tick(
         TimeSpan elapsed)
    {
      sample++;
      if(sample == 5)
      {
        isFive = true;
      }
      return new MapPluginTickResult();
    }
  }
}
```

以上のコードは特に意味がないものなのですが、一応これをベースとしてやっていきます。
これをコピペしても多分エラーは出ないと思いますが、もし出たら後述の「トラブルシューティング」をご確認ください。
では、ビルドしていきましょう。
といっても、やることは簡単でctrl+B をクリックするだけです。
コードのウインドウの左下に「問題は見つかりませんでした」と表示されていれば成功するはずです。
するとカレントディレクトリ\bin\Debug\net48 の中にプラグインの dll が生成されたと思います。
もし指定のディレクトリの中になければ、「プロジェクト」「デバッグ」と書いてある箇所の下を「debug」「AnyCPU」と変更します。
エラーが出てビルドに失敗する場合は、AtsEx.PluginHost がインストールされていない可能性があります。もう一度 nuget を確認してください。
ビルドが完了したら以下の状態に戻してください。

リスト 3　MapPlugin.cs

```
using System;
using AtsEx.PluginHost.Plugins;
namespace BuildSample
{
  [PluginType(PluginType.MapPlugin)]
  internal class MapPluginMain :
       AssemblyPluginBase
  {
    public MapPluginMain(PluginBuilder
         builder) : base(builder)
    {}
    public override void Dispose()
    {}
    public override TickResult Tick(
         TimeSpan elapsed)
    {
      return new MapPluginTickResult();
    }
  }
}
```

では、実際にコードを編集していきます。
今回のゴールはマスコン/ブレーキの値の表示なので、まずはそれらの値を取得するところから始めましょう。
結論から言うと、

リスト 4　MapPlugin.cs

```
using System;
using AtsEx.PluginHost.Plugins;

namespace DrawNotch;
{
  [PluginType(PluginType.MapPlugin)]
  internal class MapPluginMain :
       AssemblyPluginBase
  {
    int power;
    int break;
    public MapPluginMain(PluginBuilder
         builder) : base(builder)
    {}
    public override TickResult Tick(
         TimeSpan elapsed)
    {
      power = Native.Handles.Power.
           Notch;
      break = Native.Handles.Break.
           Notch;
      return new MapPluginTickResult();
    }
  }
}
```

となるのですが、おそらく実際にはこの値のみを使ってプラグインを作ることは少ないと思うので、その他の機能についても簡潔に触れておこうかと思います。
(といっても、機能が多すぎるために特に分かりづらいもののみですが...) 他の機能についてはオブジェクトブラウザーを使うのがいいのですが、やっぱり公式の Discord 鯖で質問しないとわからないことも多々あります (筆者のプログラミング能力が低いだけかも)

リスト 5　MapPlugin.cs

```
//BveTypes.ClassWrappers\
      Station に入っている値(通過時刻とか)
      を参照する場合
var station = BveHacker.Scenario.Route.
      Stations[(調べたい駅のインデックス、変
      数も可能)] as Station;
if (station == null){
    arriveMilli = station.
         ArrivalTimeMilliseconds;//通過
         時刻 (ミリ秒)
    passMilli = station.
         DepartureTimeMilliseconds;//停
         車時刻 (ミリ秒)
    pass = station.Pass;//通過/停車の判定
}
//駅インデックス
index = BveHacker.Scenario.Route.
      Stations.CurrentIndex + 1;//停車時
      は前駅のインデックスが代入され、開始時
      に
      index がマイナスになることを避けるために
      +1している
```