

C#で逝く AtsEX 入門

T.Wattz

2024 年 6 月 4 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。
BVE は Windows ソフトなので
Mac では動かない点はご注意。（プ
ラグインを使うと Wine を使っ
ても）

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

```
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 OnAp
plicationQuit() 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、Bve-
Hacker.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
力行ノッチの値、ブレーキノッチ
の値を表示する」こととします。
環境設定

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 Studio をインストー
ルします。（こちらへんはアップ
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 MapPlu
gin\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/](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

```
1 using System;
2 using AtsEx.PluginHost.Plugins;
3 namespace BuildSample
4 {
5     /// プラグインの本体
6     [PluginType(PluginType.MapPlugin)]
7     internal class MapPluginMain :
6         AssemblyPluginBase
8     {
9         int sample;
10        bool isFive;
11        /// プラグインが読み込まれた時に呼ばれる
12        /// 初期化を実装する
13        public MapPluginMain(PluginBuilder
14            builder) : base(builder)
15        {
16            sample = 1;
17            isFive = false;
18        }
19        /// プラグインが解放されたときに呼ばれる
20        /// 後処理を実装する
21        public override void Dispose()
22        {
23        }
24        /// シナリオ読み込み中に毎フレーム呼び出
25        される
26        public override TickResult Tick(
27            TimeSpan elapsed)
28        {
29        }
30        {
31            sample++;
32            if(sample == 5)
33            {
34                isFive = true;
35            }
36            return new MapPluginTickResult();
37        }
38    }
39 }
```

以上のコードは特に意味がないものののですが、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 を押すだけです。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

```
1 using System;
2 using AtsEx.PluginHost.Plugins;
3 namespace BuildSample
4 {
5     [PluginType(PluginType.MapPlugin)]
6     internal class MapPluginMain :
6         AssemblyPluginBase
7     {
8         public MapPluginMain(PluginBuilder
9             builder) : base(builder)
10        {
11        }
12        public override void Dispose()
13        {
14        }
15        public override TickResult Tick(
16            TimeSpan elapsed)
17        {
18            return new MapPluginTickResult();
19        }
20    }
21 }
```

では、実際にコードを編集していきます。今回のゴールは力行 (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

```
1 using System;
2 using AtsEx.PluginHost.Plugins;
3 namespace DrawNotch;
4 {
5     [PluginType(PluginType.MapPlugin)]
6     internal class MapPluginMain :
6         AssemblyPluginBase
7     {
8         int power;
9         int break;
10        public MapPluginMain(PluginBuilder
11            builder) : base(builder)
12        {
13        }
14        public override TickResult Tick(
15            TimeSpan elapsed)
16        {
17        }
```

```
15        power = Native.Handles.Power.
16            Notch;
17        break = Native.Handles.Break.
18            Notch;
19        return new MapPluginTickResult();
20    }
21 }
```

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) okaoka-depot.com/AtsEX/wiki/ こちらからどうぞ。質問しすぎるとシカトされます

リスト 5 機能リスト

```
1 //BveTypes.ClassWrappers\
2 Station に入っている値(通過時刻とか)
3 を参照する場合
4 var station = BveHacker.Scenario.Route.
5 Stations[<調べたい駅のインデックス、変
6 数も可能>] as Station;
7 if (station == null){
8     arriveMilli = station.
9     ArrivalTimeMilliseconds;//通過
10    時刻(ミリ秒)
11    passMilli = station.
12    DepartureTimeMilliseconds;//停
13    車時刻(ミリ秒)
14    pass = station.Pass;//通過/停車の判定
15 }
16 //駅インデックス
17 index = BveHacker.Scenario.Route.
18 Stations.CurrentIndex + 1;//停車時
19 は前駅のインデックスが代入され、開始時
20 に
21 index がマイナスになることを避けるために
22 +1している
23 //地上子通過イベントの登録 (PluginMain 内)
24 Native.BeaconPassed += new
25 BeaconPassedEventHandler((呼び出し
26 たい関数名));
27 //地上子通過イベントの購読解除 (Dispose 内)
28 Native.BeaconPassed -= new
29 BeaconPassedEventHandler((呼び出し
30 たい関数名));
31 //地上子イベント発生時に呼び出す関数
32 public void BeaconPassed(
33     BeaconPassedEventArgs e)//e.
34     Type で地上子のタイプを指定できる (読み出し)
35 {
36 }
```

```

17  switch (e.Type)//メトロ総合プラグイン
    場合
18  {
19      case 10://信号 0
20          atc = 0;
21          break;
22      case 19://ATC45
23          atc = 45;
24          break;
25  }
26  //他にも e.SignalIndex(int)で対となる閉
    塞のインデックス,e.Distance(float)
    で対となる閉塞までの距離を取得可能。
27  }

```

こんなものでしょうか（長くてすみません）

正直 AtsEX はできることが多すぎる上、Unity ほど文献が充実していないので、繰り返になります公式 Discord の利用がおすすめです（ちなみに、設立は俺が提案しました。えらい。）

余談にはなりますが、個人的に Unity が使いやすいのはそういう所だったりします。Godot や Unreal などは文献的な面で Unity 程初心者ができるような代物ではないのかな～と思います。（知らんけど）

では開発の方に戻りましょう。以上では力行ノッチとブレーキノッチの値を取得しました。今度は、それが実行できているかを確認してみましょう。

以下のコードを PluginMain 内に記述してください。

リスト 6 MapPluginMain()

```

1  if (!System.Diagnostics.Debugger.
    IsAttached)
2  {System.Diagnostics.Debugger.Launch();}

```

これはデバッガーを起動するためのもので、VisualStudio がインストールされている場合は自動的にデバッガーが起動します。

勿論 Plugin だけでは動作しないので、BveTrainism5.8(と DirectX9.0cRunTime,.NET Framework3.5),AtsEX をインストールしましょう。（ここはあまり関係ないので省略）

次に車両を用意します。（ここもググること前提で省略します。）

Map ファイルは以下のようになります。

リスト 7 Map.txt

```

1  BveTs Map 2.02
2  include '[[AtsEx::NOMPI]] このマップでは
    AtsEX マッププラグインを使用していま
    す。正常動作には AtsEX のインストール
    が必要です。詳細は https://
    automatic9045.github.io をご覧ください。';
3  include '[[AtsEx::NOMPI]] This map
    requires to install AtsEX to
    execute map plugins. For more
    information, visit https://
    automatic9045.github.io';
4  include '<AtsEx::USEATSEX>';
5  include '<AtsEx::READDEPTH1';
6  include '<AtsEx::MapPluginUsing>
    MapPluginUsing.xml';

```

（ご自身の Map に組み込む場合は上 4 行をメインの Map ファイルの最上段に組み込んで下さい）

次に MapPluginUsing.xml を作成して Map.txt と同ディレクトリに置きます。

内容は以下の通りです。

リスト 8 MapPluginUsing.xml

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <AtsExPluginUsing xmlns="http://
    automatic9045.github.io/ns/
    xmlschemas/
    AtsExPluginUsingXmlSchema.xsd">
3  <Assembly Path="(プラグインdll へのパス)
    " />
4  </AtsExPluginUsing>

```

これで動くはずですが。（動かない場合は、パスを確認して下さい。）

次に、ブレークポイントの設定をします。

MapPlugin.cs に戻って下さい。そして、コードの左側をクリックして power = Native.Handles.Power.Notch; の左にブレークポイントを設定して下さい。

では、実際に BVE を起動し、シナリオを読み込んで下さい。（上に同じく）

すると、以下画像のようにデバッ

ガーが起動すると思います。

起動したら、以下画像のようになるかと思います。

ブレークポイントで止まると思いますので、止まったら（BveTs.exe が応答しなくなったら）VisualStudio のウィンドウに移動し、左下を御覧ください。

すると、おそらく現在の power,break の値が表示されると思います。（通常のシナリオなら、power は 0,break は非常ブレーキの値に）では、早速これを反映する UI を作成していきましょう。

まずは DirectX を最低限触ります。Nuget より Lib.Harmony,ObjectiveHarmonyPatch,SlimIo をインストールし、using ディレクティブには FastMember,TypeWrapping,ObjectiveHarmonyPatch を追加して下さい。

リスト 9 MapPluginMain()

```

1  ClassMemberSet assistantDrawerMembers
    = BveHacker.BveTypes.
    GetClassInfoOf<AssistantDrawer
    >();
2  FastMethod drawMethod =
    assistantDrawerMembers.
    GetSourceMethodOf(nameof(
    AssistantDrawer.Draw));
3  HarmonyPatch drawPatch = HarmonyPatch
    .Patch(Name, drawMethod.Source,
    PatchType.Prefix);
4  drawPatch.Invoked += (sender, e) =>
5  {
6      //ここにある処理を書く
7  };

```

このコードは、BVE 本体にパッチを当てるもので、BveHacker.BveTypes.GetClassInfoOf<AssistantDrawer>では BVE 本体の補助表示のコードに対するものです。つまり、これを唱えれば補助表示の位置（運転台よりも手前）に描画することが可能という訳です。