

## Halte3 —— 泛用游戏资源抽取/回封工具

名称： Halte3

开发者： HatsuneTakumi

目前版本： GUI Ver3.0Ext + Core Ver2.0Ext （LastUpdate 2011-07-20）

目前插件数量： 43 个

目前状态： 稳定

平台： Window7 下测试通过。

接口状态：

抽取 —— 稳定

回封 —— 向下兼容更新

简介：

这是一个泛用的游戏资源抽取工具。最初的想法来自于三年前，当时就感觉到每解一个游戏都要重新写一个程序很麻烦，并且重复工作量很大。于是在当时写了第一版的 Halte。

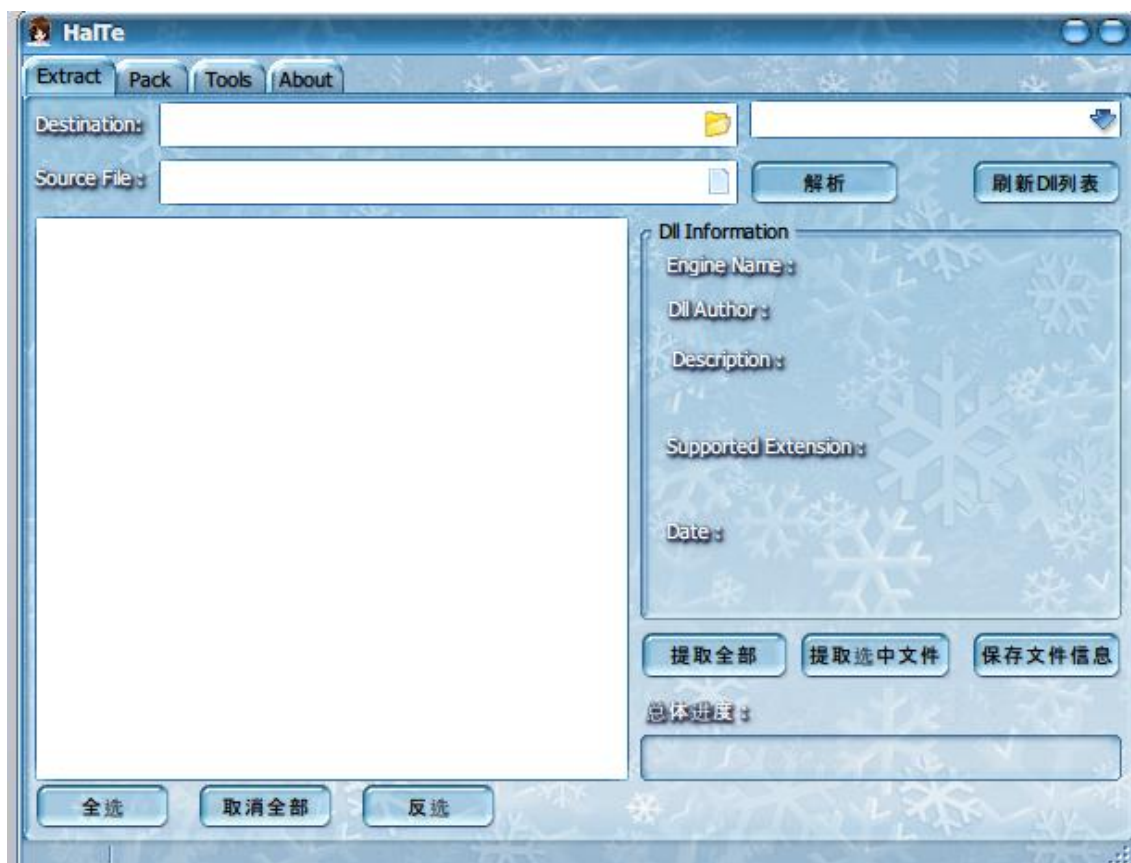


Figure 1 第一版的 Halte

当时单纯在追求缩短开发周期，所以将封包的结构和处理流程做了假定，使得只能处理 90% 的结构，当然好处是开发周期异常的短，一个没有加密的典型封包只需要填写 5 个常量外加

几行代码便可以提取。不过这种结构却导致连 Yuka 这种简单的封包都无法提取。

于是后来决定改进内核，便有了现在所使用的 Core2.0，这次则是充分考虑的扩展性和代码复用，直到近几天看 crass 源代码时我才发现其实竟然跟他的结构有一点点像……

抽取分为了 3 个阶段，分别是 Indexs 抽取，文件解密（目前没有任何作用），文件抽取。所有的读取操作都可以交给 exe 完成，而插件只需要负责填写一个 Indexs 结构和处理文件数据。当然如果自带的读取功能满足不了要求也可以自行处理一切。

同时还增加了一个辅助类 TGLcompress，可以处理大部分压缩解压，并且可以直接用来解密（因为内置了自管理的 Output）。而 Gui 则类似 Halte，组成了 Halte2。

至于 Halte3，则是因为 Halte2 的交互和外置功能上有不小的问题，为了解决这个问题，我便改进了 GUI 的设计，让它更能够方便的使用和调试。



Figure 2 界面上与 Halte 没差多少的 Halte2

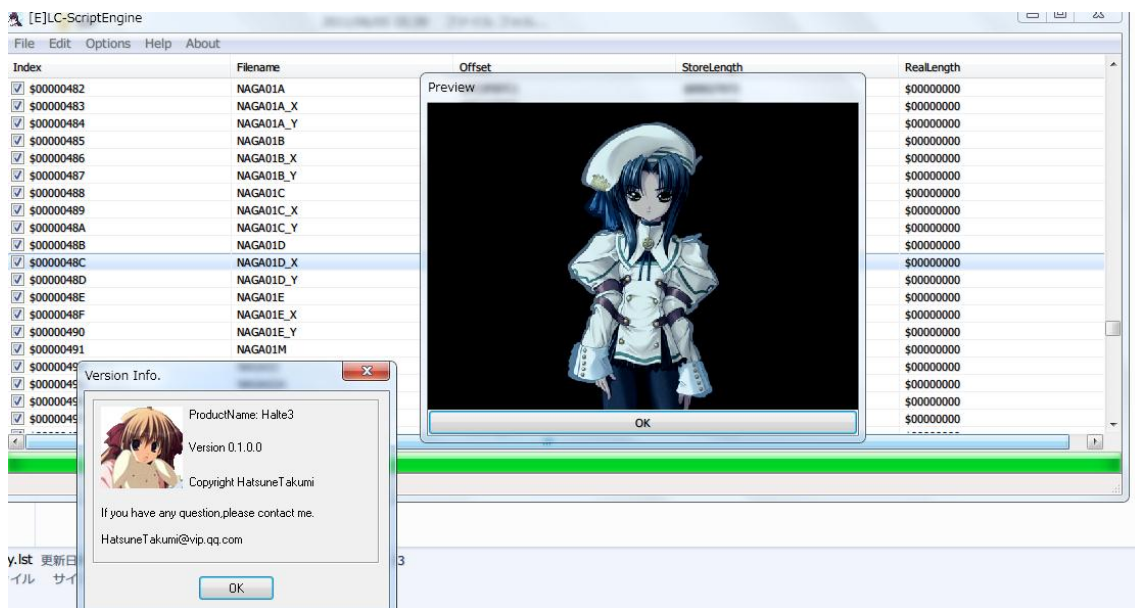


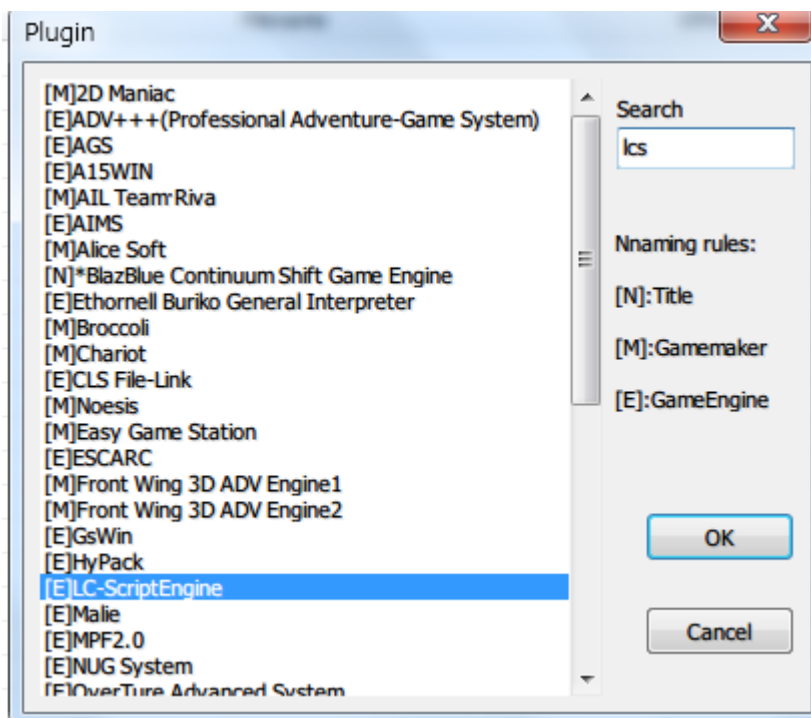
Figure 3 目前的 Halte3

使用说明:

以 LC-SE 引擎为例, 简单说下。

首先在启动之时, 新版的 GUI 增加了 OnProcess 过程, 使得程序不会卡死, 等待信息获取完毕后插件会出现在列表之中。

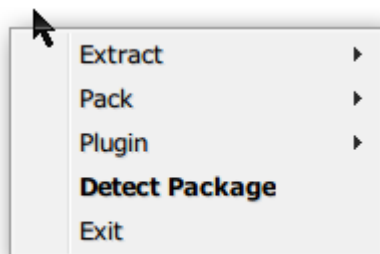
首先需要选择插件:



支持搜索，只需输入名字即可自动匹配。

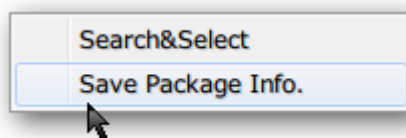
然后拖放或者选择 OpenPackage 打开文件

在新版中，修改了界面布局，因此与之前可能有所差别：

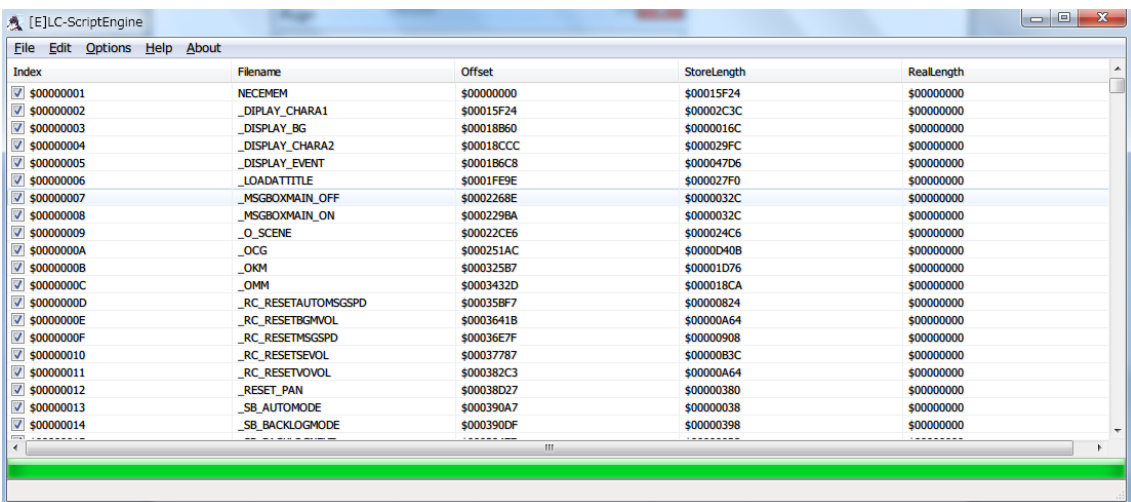
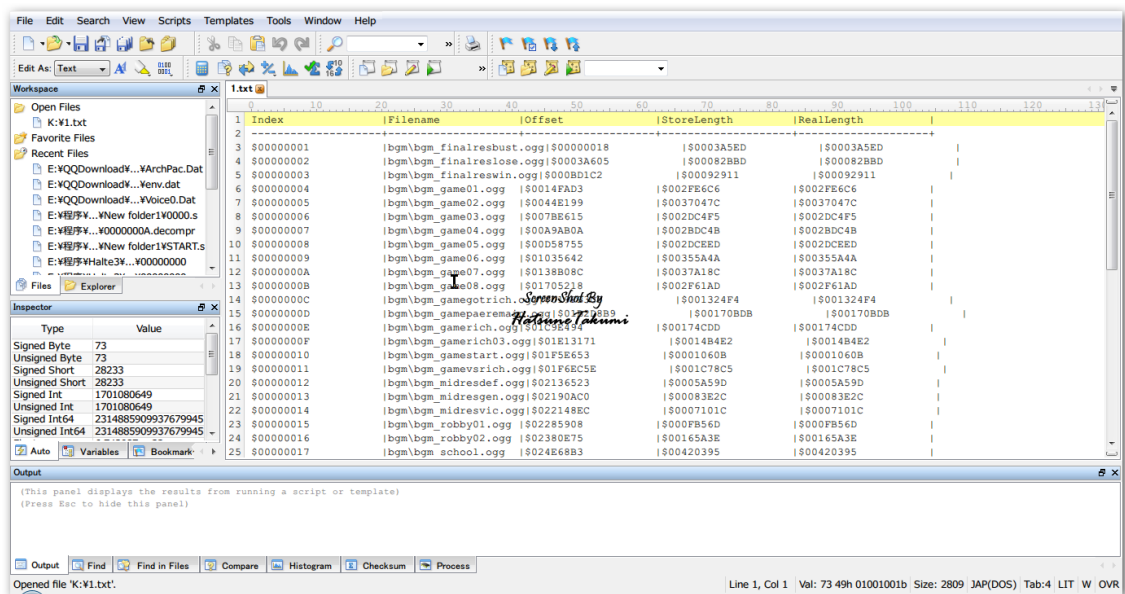


自动检测放进了 files 菜单，并且将步骤重新整理为 Extract 和 Pack，便于理解。

抽取玩 Indexs 可以选择保存文件信息：



可以得到如下结果：



可以使用 edit – search 来搜索文件：

Search&Select

Extension

KeyWord

Index

Other

Key Word

EV

Delimiter is ' , ' .(Do not contain space and quotes)

It is OR between different keyword.NOT AND.

Search Option

☒ Check
☐ Uncheck

OK

Cancel

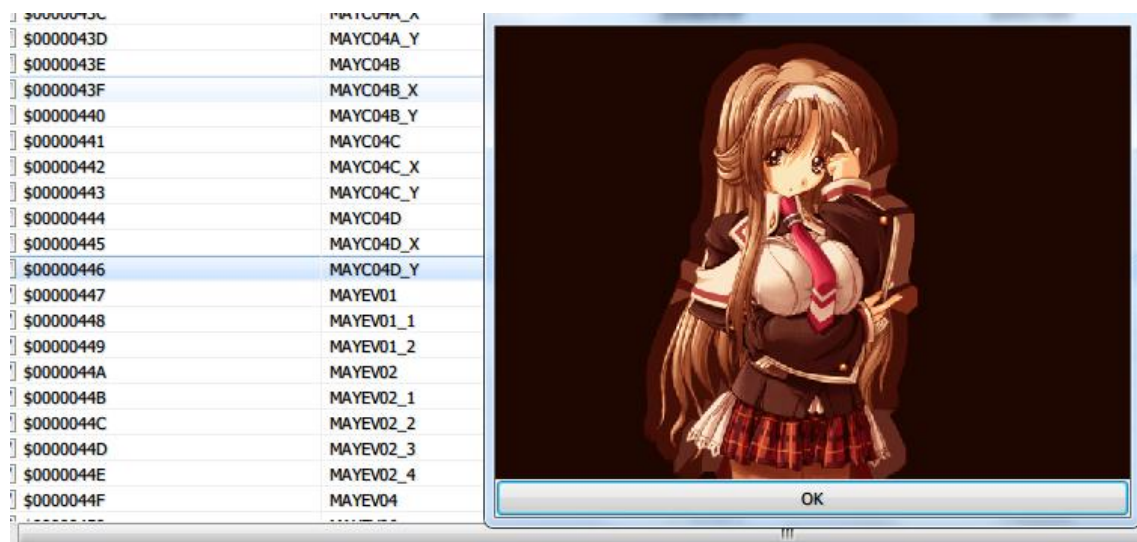
可以看到匹配结果：

<input type="checkbox"/>	\$0000043C	MAYC04A_X	\$0A864FA
<input type="checkbox"/>	\$0000043D	MAYC04A_Y	\$0A88C06
<input type="checkbox"/>	\$0000043E	MAYC04B	\$0A8B250
<input type="checkbox"/>	\$0000043F	MAYC04B_X	\$0A8DA06
<input type="checkbox"/>	\$00000440	MAYC04B_Y	\$0A90115
<input type="checkbox"/>	\$00000441	MAYC04C	\$0A92764
<input type="checkbox"/>	\$00000442	MAYC04C_X	\$0A94F33
<input type="checkbox"/>	\$00000443	MAYC04C_Y	\$0A97655
<input type="checkbox"/>	\$00000444	MAYC04D	\$0A99CB5
<input type="checkbox"/>	\$00000445	MAYC04D_X	\$0A9C47E
<input type="checkbox"/>	\$00000446	MAYC04D_Y	\$0A9EB9A
<input checked="" type="checkbox"/>	\$00000447	MAYEV01	\$0AA11F8
<input checked="" type="checkbox"/>	\$00000448	MAYEV01_1	\$0AA8CE5
<input checked="" type="checkbox"/>	\$00000449	MAYEV01_2	\$0AB075E
<input checked="" type="checkbox"/>	\$0000044A	MAYEV02	\$0AB8214
<input checked="" type="checkbox"/>	\$0000044B	MAYEV02_1	\$0AC056C
<input checked="" type="checkbox"/>	\$0000044C	MAYEV02_2	\$0AC891A
<input checked="" type="checkbox"/>	\$0000044D	MAYEV02_3	\$0AD0CFE
<input checked="" type="checkbox"/>	\$0000044E	MAYEV02_4	\$0AD9113
<input checked="" type="checkbox"/>	\$0000044F	MAYEV04	\$0AE156E
<input type="checkbox"/>	-----	-----	-----

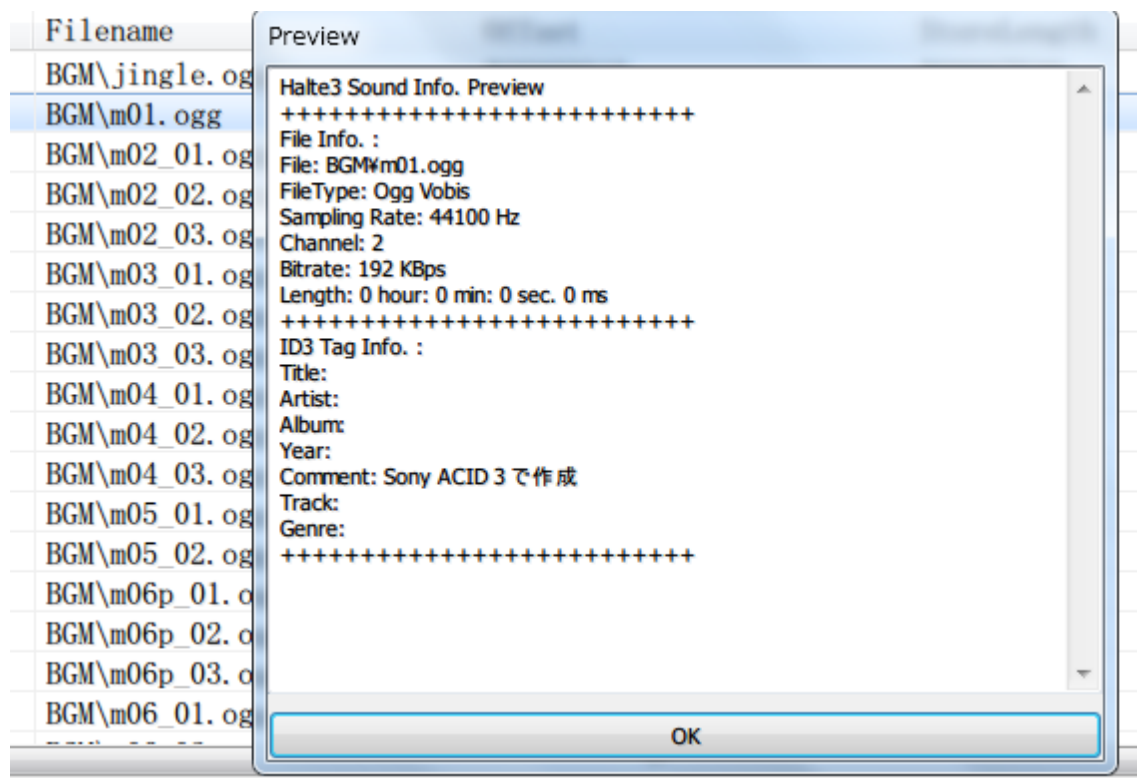
然后选择 ExtractFiles 解出文件。

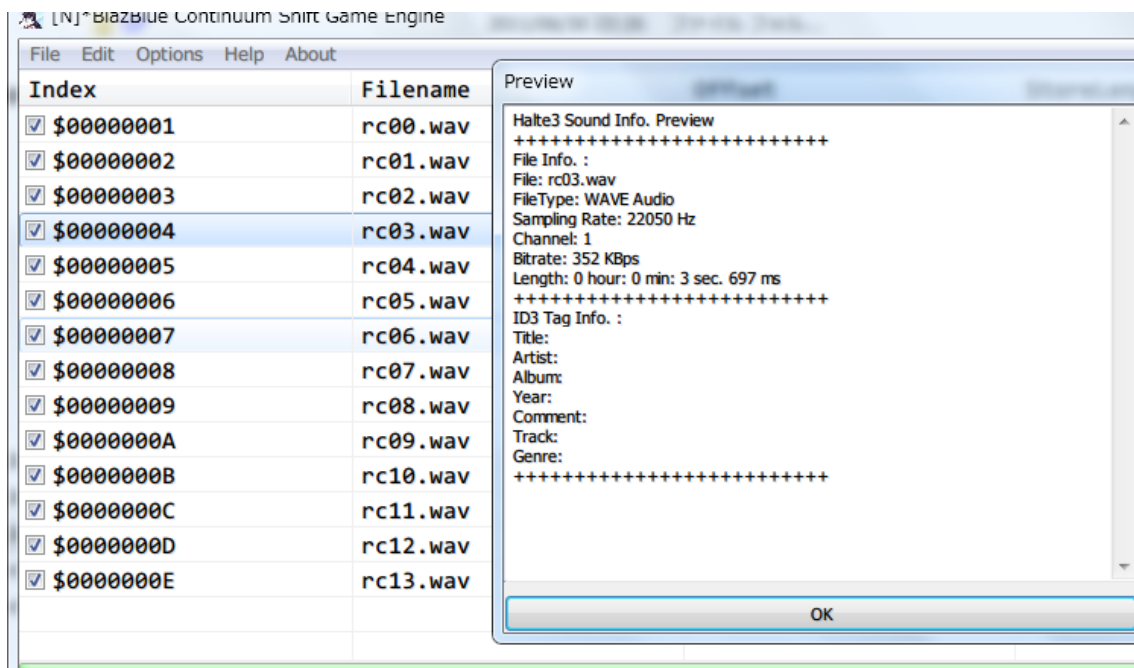
另外也可以双击文件预览（部分）。





目前支持部分文本、BMP、PNG、Jpeg、OGG、WAV、MP3 格式的预览。



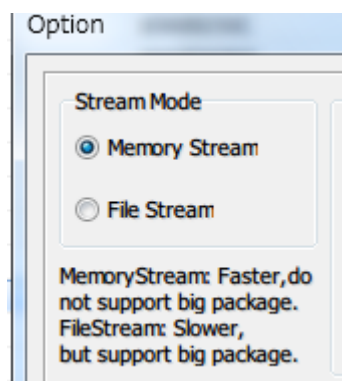


并且，预览并不会影响其他任何操作。

此时可以抽取文件，打开别的封包甚至是更换插件，这并不影响当前文件的继续预览（音乐播放也不会停止）。如果不关掉预览窗口的话，可以预览其他非音频文件而不影响音频播放。

不过有一点需要注意，就是预览使用的是 **MemoryBuffer**，如果有几百 MB 的超大音频（如果有的话，因为有部分引擎把全部语音集中到一起，然后用一张表来 **Seek** 播放，插件并不一定会自己把文件拆开）建议不要预览，否则会占用很多内存。

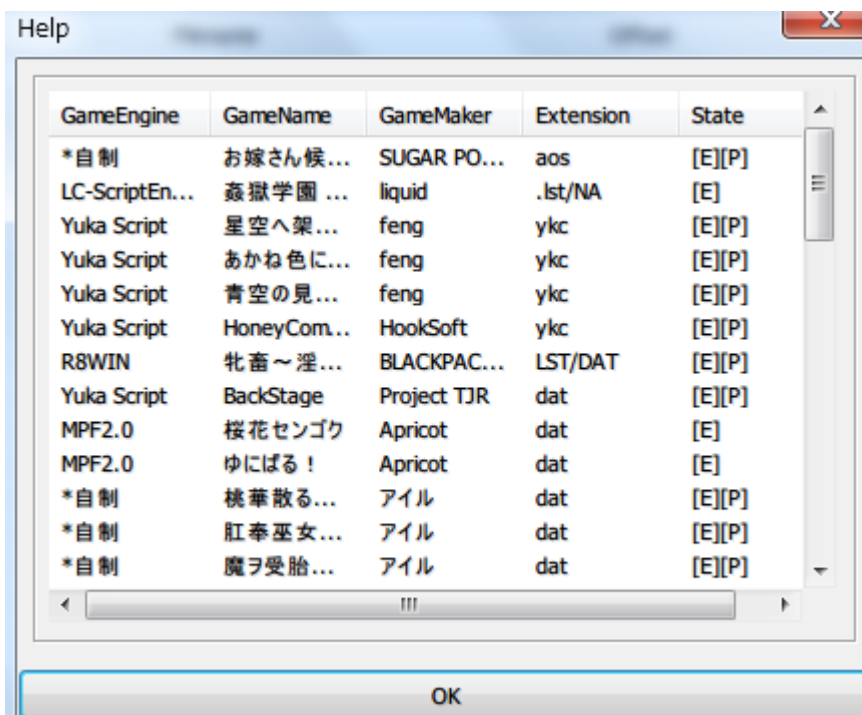
另外需要注意的是选项：



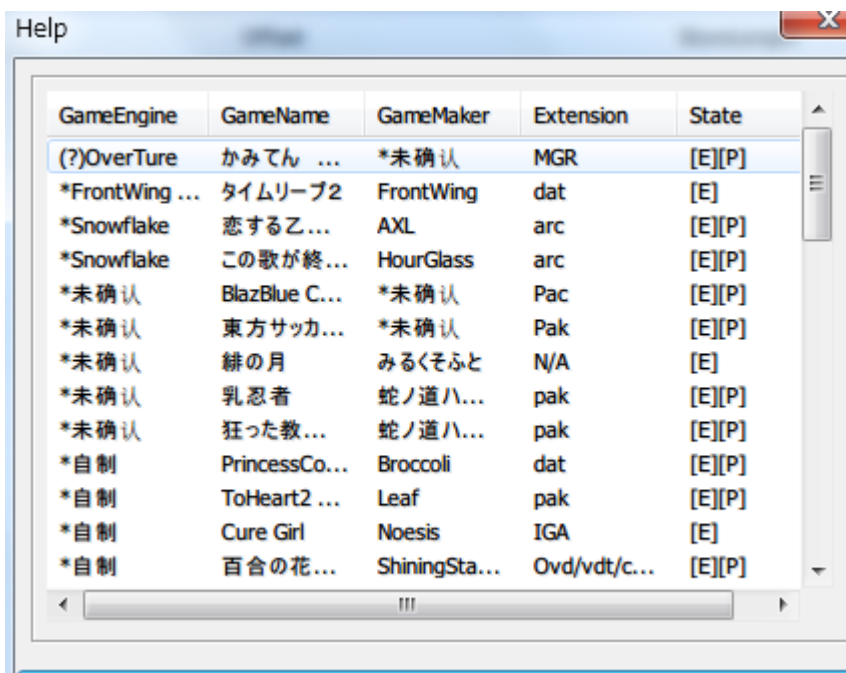
在抽取超大文件时（尤其是 **LIGHT** 的），请务必选择 **FileSteam**。其他情况使用 **Memory** 足够。

**Help** 实际上是 **supportlist**:

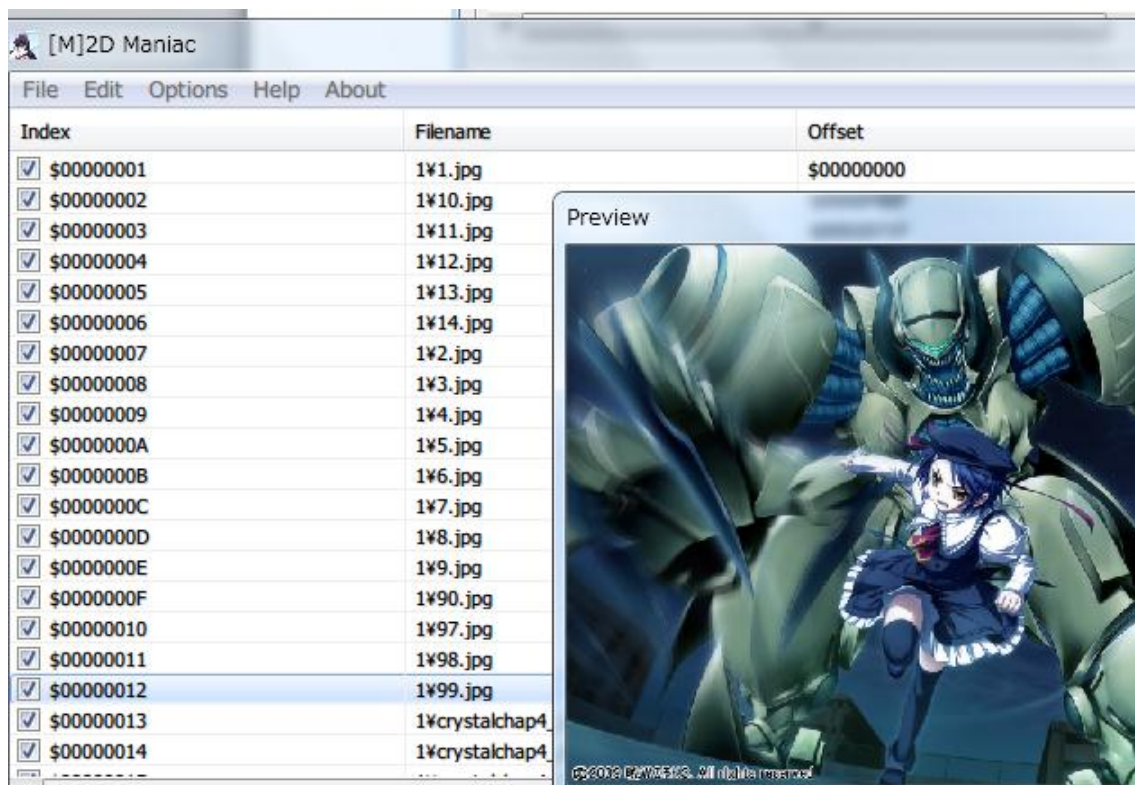




点击标签可以排序，下面就是引擎排序结果：

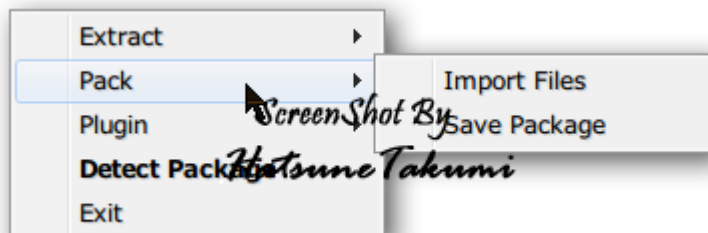


回封类似，只是首先选择的是 ImportFiles，并且依然支持预览：



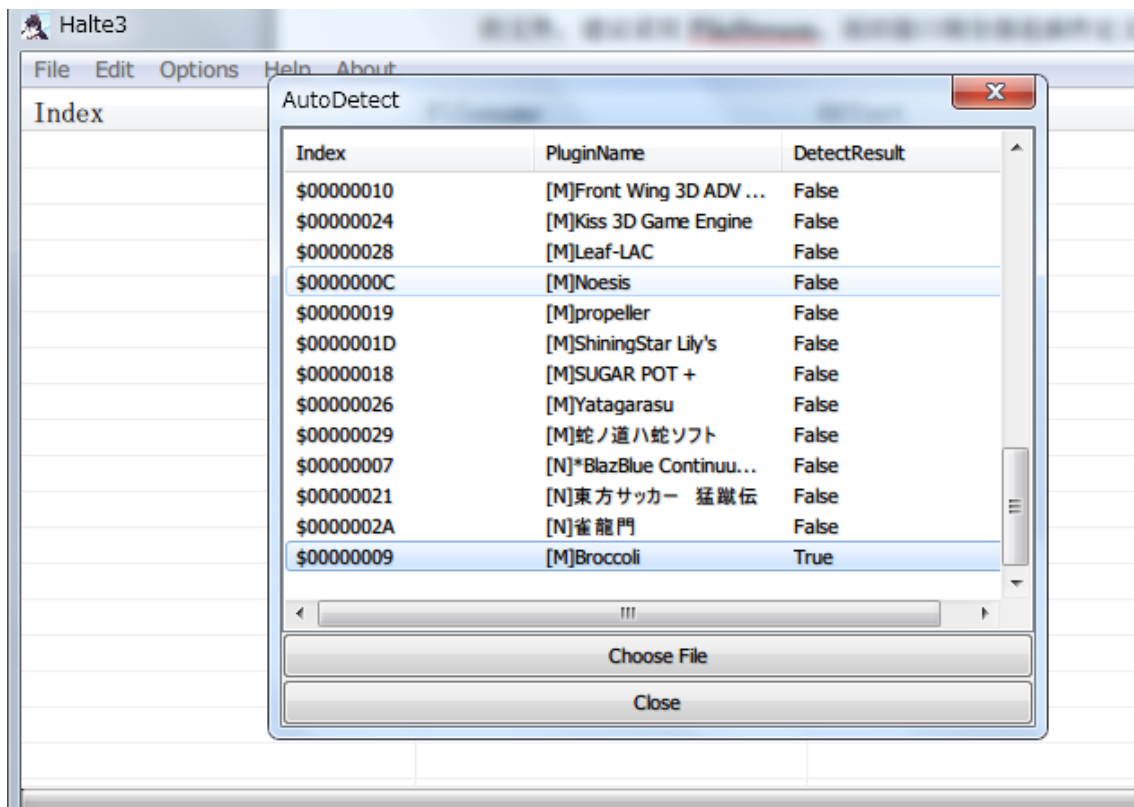
然后只需要 SavePackage 即可。

不过回封需要特别注意，对于 1 型（新接口）的辅助流类型取决于设定，雨果需要回封很大的文件，建议采用 FileStream。而旧接口则全部是插件定义好的，因此不能改变。



新版修改了之前的布局，现在回封只需要在 Pack 一栏里完成。

从 GUI 升级之后，插件开始支持一个外置的自动检测功能。



选择 edit-AutoDetect 既可以打开这个界面。同样支持排序。

特别注意，因为某些封包需要计算 Hash，因此在检测部分大封包时可能会消耗较多内存，并且有段时间的卡死。

下面对检测结果做一些说明。

**True:** 这表示封包与预想特征完全一致，当然有可能会出误判，比如 AGS 与 BGI，对少数 BGI 封包恰好可以满足 AGS 的特征，此时请换一个封包测试。

**False:** 不支持，表示插件一定不支持。此时请勿再尝试用这个插件去解。

**Unknown:** 不确定。可能是因为在引擎更新导致结构变化或者本身封包就没有很强的特征。此时请多换几个封包，比较结果。

**Error:** 不明原因的错误，请反馈给作者。

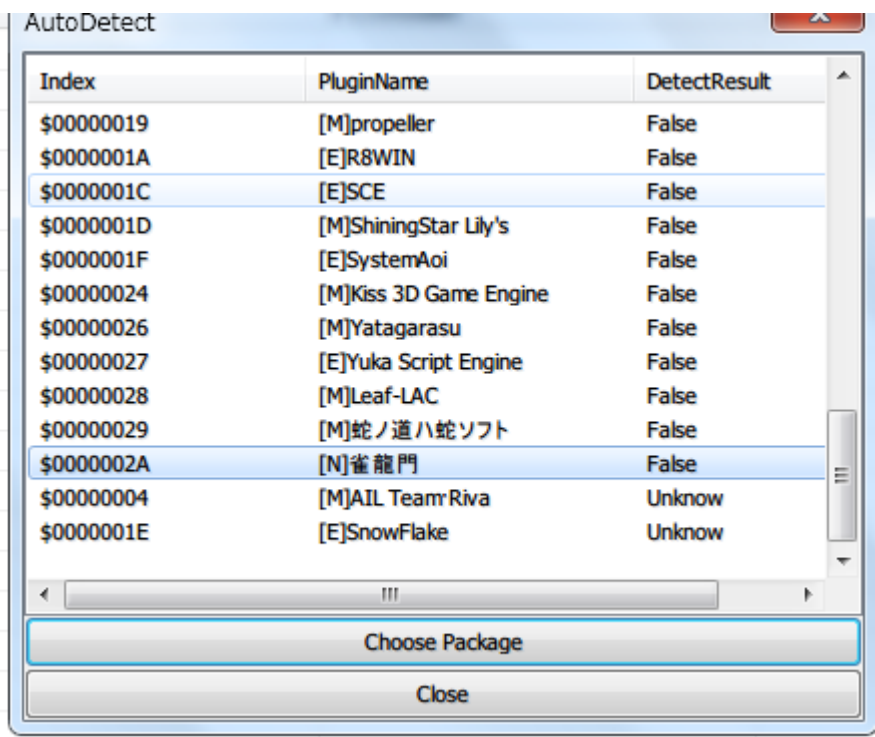
**Unsupported:** 这个插件暂时不支持检测，一般是因为这个插件还没开发完毕。

挑选插件的一般规则：请尽量使用 2 个以上封包来得出结果，优先选择报告 True 的。

当没有插件报告 True 时，可能是因为这个封包本身没有特征，因此查看有没有报告 Unknown 的。

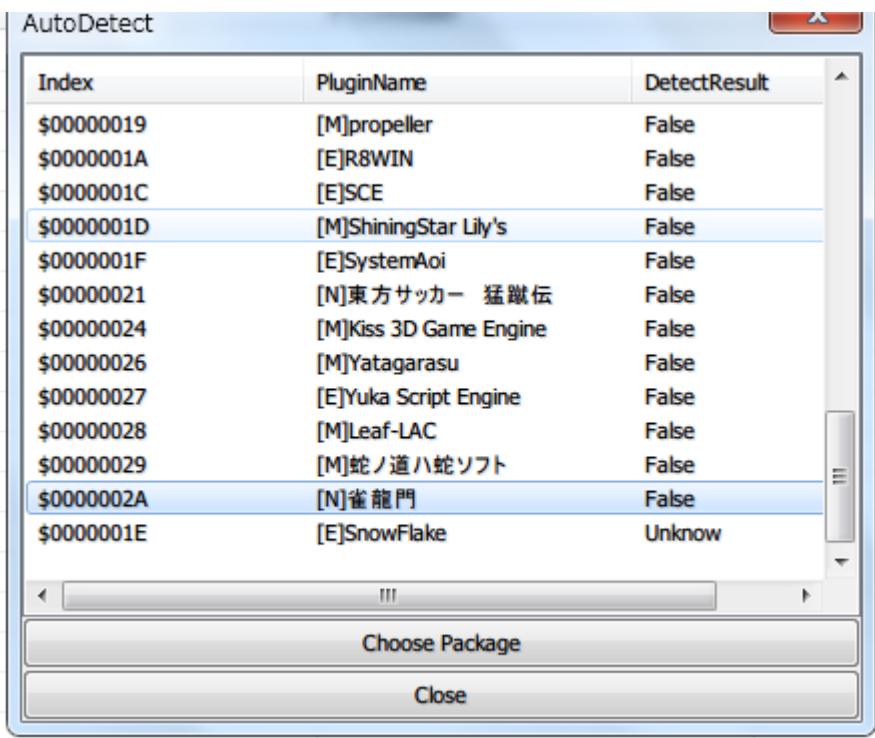
因为 Unknown 的判断标准很模糊，当出现多个插件报告时，建议多使用几个封包来消除误判。举个例子。以 hourglass 的この歌が終わったら为例。

实际上这个游戏是用的 SnowFlake (AXL), 而如果选取 Instdata 作为样本, 则会导致误判:



Index	PluginName	DetectResult
\$00000019	[M]propeller	False
\$0000001A	[E]R8WIN	False
\$0000001C	[E]SCE	False
\$0000001D	[M]ShiningStar Lily's	False
\$0000001F	[E]SystemAoi	False
\$00000024	[M]Kiss 3D Game Engine	False
\$00000026	[M]Yatagarasu	False
\$00000027	[E]Yuka Script Engine	False
\$00000028	[M]Leaf-LAC	False
\$00000029	[M]蛇ノ道ハ蛇ソフト	False
\$0000002A	[N]雀龍門	False
\$00000004	[M]AIL Team Riva	Unknow
\$0000001E	[E]SnowFlake	Unknow

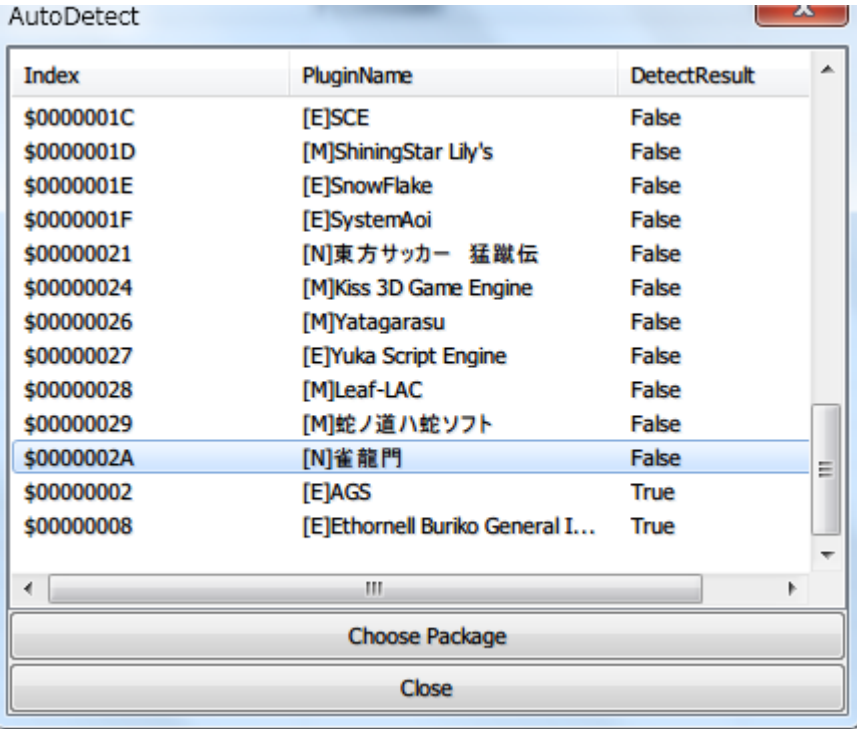
可以看到 Ail 的引擎在内, 然而换用 gra 作为样本:



Index	PluginName	DetectResult
\$00000019	[M]propeller	False
\$0000001A	[E]R8WIN	False
\$0000001C	[E]SCE	False
\$0000001D	[M]ShiningStar Lily's	False
\$0000001F	[E]SystemAoi	False
\$00000021	[N]東方サッカー 猛獣伝	False
\$00000024	[M]Kiss 3D Game Engine	False
\$00000026	[M]Yatagarasu	False
\$00000027	[E]Yuka Script Engine	False
\$00000028	[M]Leaf-LAC	False
\$00000029	[M]蛇ノ道ハ蛇ソフト	False
\$0000002A	[N]雀龍門	False
\$0000001E	[E]SnowFlake	Unknow

可以看到只有一个引擎给出了结果。

又比如操心術 0，对某些封包，会有：

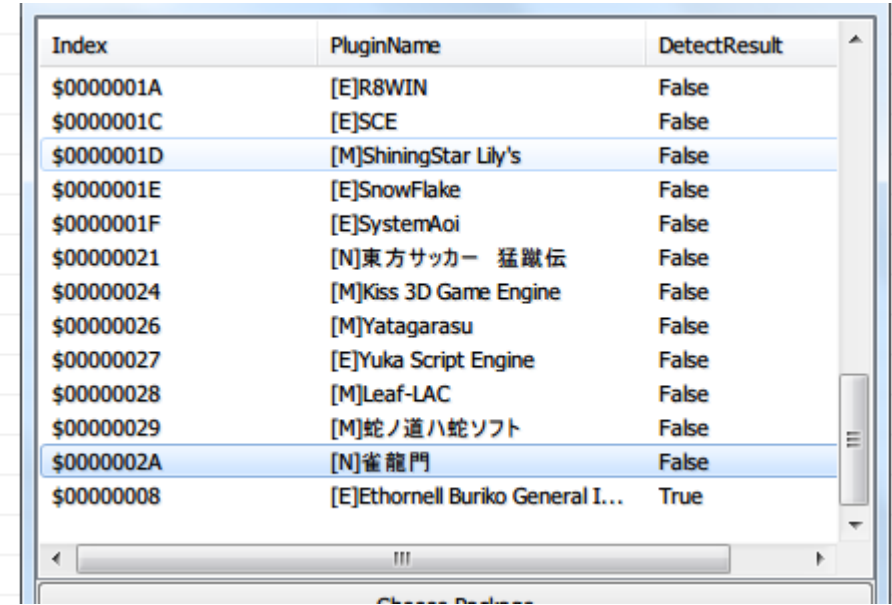


The screenshot shows a window titled "AutoDetect" with a table of plugins. The table has three columns: Index, PluginName, and DetectResult. The following table represents the data shown in the image:

Index	PluginName	DetectResult
\$0000001C	[E]SCE	False
\$0000001D	[M]ShiningStar Lily's	False
\$0000001E	[E]SnowFlake	False
\$0000001F	[E]SystemAoi	False
\$00000021	[N]東方サッカー 猛蹴伝	False
\$00000024	[M]Kiss 3D Game Engine	False
\$00000026	[M]Yatagarasu	False
\$00000027	[E]Yuka Script Engine	False
\$00000028	[M]Leaf-LAC	False
\$00000029	[M]蛇ノ道ハ蛇ソフト	False
\$0000002A	[N]雀龍門	False
\$00000002	[E]AGS	True
\$00000008	[E]Ethornell Buriko General I...	True

Below the table, there are two buttons: "Choose Package" and "Close".

AGS 和 BGI 都给出了结果，然而换用其他封包：



The screenshot shows the same "AutoDetect" window but with a different set of plugins. The following table represents the data shown in the image:

Index	PluginName	DetectResult
\$0000001A	[E]R8WIN	False
\$0000001C	[E]SCE	False
\$0000001D	[M]ShiningStar Lily's	False
\$0000001E	[E]SnowFlake	False
\$0000001F	[E]SystemAoi	False
\$00000021	[N]東方サッカー 猛蹴伝	False
\$00000024	[M]Kiss 3D Game Engine	False
\$00000026	[M]Yatagarasu	False
\$00000027	[E]Yuka Script Engine	False
\$00000028	[M]Leaf-LAC	False
\$00000029	[M]蛇ノ道ハ蛇ソフト	False
\$0000002A	[N]雀龍門	False
\$00000008	[E]Ethornell Buriko General I...	True

Below the table, the "Choose Package" button is visible.

可以看到只有一个结果。

再看青空の見える丘：

Index	PluginName	DetectResult
\$0000001A	[E]R8WIN	False
\$0000001C	[E]SCE	False
\$0000001D	[M]ShiningStar Lily's	False
\$0000001E	[E]SnowFlake	False
\$0000001F	[E]SystemAoi	False
\$00000021	[N]東方サッカー 猛獣伝	False
\$00000024	[M]Kiss 3D Game Engine	False
\$00000026	[M]Yatagarasu	False
\$00000028	[M]Leaf-LAC	False
\$00000029	[M]蛇ノ道ハ蛇ソフト	False
\$0000002A	[N]雀龍門	False
\$00000027	[E]Yuka Script Engine	True
\$00000017	[E]OverTure Advanced...	Unknow

Choose Package

Close

可以看到 OTA 给出了模糊匹配，而 Yuka 则是强匹配，此时则要选取 Yuka。

而目前对于 Ail 这种标志很弱的封包也可以正常识别：

Index	PluginName	DetectResult
\$0000001A	[E]R8WIN	False
\$0000001C	[E]SCE	False
\$0000001D	[M]ShiningStar Lily's	False
\$0000001E	[E]SnowFlake	False
\$0000001F	[E]SystemAoi	False
\$00000021	[N]東方サッカー 猛獣伝	False
\$00000024	[M]Kiss 3D Game Engine	False
\$00000026	[M]Yatagarasu	False
\$00000027	[E]Yuka Script Engine	False
\$00000028	[M]Leaf-LAC	False
\$00000029	[M]蛇ノ道ハ蛇ソフト	False
\$0000002A	[N]雀龍門	False
\$00000004	[M]AIL Team Riva	Unknow

Choose Package

Close



关于 Bug:

在非 Win7 环境测试中出现了少数奇怪的问题，比如偶尔会出现不明原因的异常。这是目前暂时没解决的，如果遇到了此类问题请将系统环境、游戏名和相应封包、以及对应插件名称发给我，以便于更好的查明原因。

因为之前在用 delphi2010 时发现 win7 和 xp 下编译的 dll 有些差别，并且在少数情况下不能通用，不知道到了 delphix64 里有没有解决这个问题。

插件开发说明:

如果希望 Halte3 支持自己希望的格式，那么请遵循下面的说明。

综述:

首先想让 Halte3 能够识别插件，必须满足下面的条件:

1. 后缀为 GP
2. 至少导出 GetPluginName、GenerateEntries、ReadData 三个函数
3. 符合 DelphiDll 标准

大部分扩展功能都是由主程序实现的，插件只需要简单的处理下面几个过程:

GetPluginName 是必须导出的函数，它指明本插件的名称。

GenerateEntries 是必须导出的函数，它处理封包的文件头和 Indexs。

ReadData 是必须导出的函数，它对提取的数据作处理。

在导出这三个必要函数之后，插件便可以实现抽取，预览等功能。

AutoDetect 是可选的函数，它用来提供封包匹配检测功能。

在导出这个函数之后，插件便可以参与封包匹配检测。

PackData 是可选的函数，用来回封（模式 1）。

PackProcessData、PackProcessIndex、MakePackage 这三个函数必须一起出现，单独出现是没有意义的，它们用来回封（模式 2）。注意：这三个函数的优先级高于模式 1。

在导出这个函数之后，插件便可以支持回封。

最初版本预留的扩展函数 Decyption 已经不存在了，解密总是和 ReadData 在一起。

接口:

下面是 ADV+++的实例，借此做简单说明。

{ 代码开始 }

library ADVPPP;

uses

FastMM4 in '..\..\..\FastMM\FastMM4.pas', //需要引用 FastMM

SysUtils,

Classes,

CustomStream in '..\..\..\Core\CustomStream.pas', //核心类集合

PluginFunc in '..\..\..\Core\PluginFunc.pas', //常数定义

StrUtils,

FastMM4Messages in '..\..\..\FastMM\FastMM4Messages.pas',

Windows; //如果喜欢 dw 而不是 cardinal 可以引用这个，不过没什么用

//另外如果需要交互请引用 dialogs，但是会使插件大 500K 左右。

{ \$E gp } //扩展名 gp

{ \$R \*.res }

{ \$DEFINE SP } //是否支持回封？

Type //结构定义

Pack\_Head = Packed Record

MAGIC: array [0 .. 3] of AnsiChar;

RESERVED: Cardinal;

ENTRIES\_OFFSET: Cardinal;

ENTRIES\_COUNT: Cardinal;

TIME: TSystemTime;

End;

Pack\_Entries = packed Record

START\_OFFSET: Cardinal;

Length: Cardinal;

End;

data\_head = packed record

MAGIC: array [0 .. 3] of AnsiChar;

flag: Cardinal;

uncomprlen: Cardinal;

RESERVED: Cardinal;

```

end;

function GetPluginName: PChar; //传回插件名
begin
    Result := PChar(FormatPluginName
        ('ADV+++(Professional Adventure-Game System)', clEngine));
end;

function GenerateEntries(IndexStream: TIndexStream;
    FileBuffer: TFileBufferStream): Boolean; //抽取 Indexs
var
    Head: Pack_Head;
    indexs: Pack_Entries;
    i: Integer;
    Rlentry: TRlEntry;
begin
    Result := True;
    try
        FileBuffer.Seek(0, soFromBeginning);
        FileBuffer.Read(Head.MAGIC[0], SizeOf(Pack_Head));
        FileBuffer.Seek(Head.ENTRIES_OFFSET, soFromBeginning);
        if not SameText(Trim(Head.MAGIC), 'YOX') then
            raise Exception.Create('Magic Error.');
```

for i := 0 to Head.ENTRIES\_COUNT - 1 do

```

    begin
        FileBuffer.Read(indexs.START_OFFSET, SizeOf(Pack_Entries));
        Rlentry := TRlEntry.Create;
        Rlentry.AOffset := indexs.START_OFFSET;
        Rlentry.AOrigin := soFromBeginning;
        Rlentry.AStoreLength := indexs.Length;
        Rlentry.ARLength := 0;
        Rlentry.AName := '$' + IntToHex(i, 8);
        IndexStream.Add(Rlentry); //只需要将填好的 Entry 加入到 list 里
    end;
except
    Result := False;
end;

```

```

    end;
end;

procedure Decyption(); //留空，但必须导出
begin
    { do nothing }
end;

function ReadData(FileBuffer: TFileBufferStream;
    OutBuffer: TStream): LongInt; //处理数据
var
    tmpbuffer: TGICompressBuffer;
    Head: data_head;
begin
    Result := FileBuffer.ReadData(OutBuffer); //内置的方法，可以直接读取数据
    OutBuffer.Seek(0, soFromBeginning);
    OutBuffer.Read(Head.MAGIC[0], SizeOf(data_head));
    if SameText(Trim(Head.MAGIC), 'YOX') and
        (Head.flag and 2 <> 0) then
    begin
        tmpbuffer := TGICompressBuffer.Create; //泛用压缩解压类
        tmpbuffer.CopyFrom(OutBuffer, OutBuffer.Size -
            SizeOf(data_head));
        tmpbuffer.Seek(0, soFromBeginning);
        tmpbuffer.Output.Seek(0, soFromBeginning);
        tmpbuffer.ZlibDecompress(Head.uncomprlen);
        tmpbuffer.Output.Seek(0, soFromBeginning);
        OutBuffer.Size := 0;
        OutBuffer.Seek(0, soFromBeginning);
        Head.flag := 0;
        OutBuffer.Write(Head.MAGIC[0], SizeOf(data_head));
        OutBuffer.CopyFrom(tmpbuffer.Output, Head.uncomprlen);
        tmpbuffer.Destroy;
    end;
end;
end;

```

```

const
    MAGIC: AnsiString = 'YOX'#0;

function PackData(IndexStream: TIndexStream;
    FileBuffer: TFileBufferStream; var Percent: Word): Boolean; //回封只有一个过程，可
    以不导出表示不支持
var
    i: Integer;
    Head: Pack_Head;
    indexs: array of Pack_Entries;
    tmpbuffer: TFileStream;
    zero: Byte;
begin
    Result := True;
    try
        Percent := 0; //需要自己控制进度
        FileBuffer.Seek(0, soFromBeginning);
        Head.ENTRIES_COUNT := IndexStream.Count;
        Head.ENTRIES_OFFSET := $800;
        Head.RESERVED := 0;
        DateTimeToSystemTime(gettime, Head.TIME);
        Move(Pbyte(MAGIC)^, Head.MAGIC[0], 4);
        FileBuffer.Write(Head.MAGIC[0], SizeOf(Pack_Head));
        zero := 0;
        SetLength(indexs, IndexStream.Count);
        while FileBuffer.Size < $800 do
            FileBuffer.Write(zero, 1);
        for i := 0 to IndexStream.Count - 1 do
            begin
                tmpbuffer := TFileStream.Create(IndexStream.ParentPath +
                    IndexStream.Rlentry[i].AName, fmOpenRead);
                indexs[i].START_OFFSET := FileBuffer.Position;
                indexs[i].Length := tmpbuffer.Size;
                tmpbuffer.Seek(0, soFromBeginning);
                FileBuffer.CopyFrom(tmpbuffer, tmpbuffer.Size);
                tmpbuffer.Free;
            end
        end
    end
    try
    except
        Result := False;
    end
end;

```

```

        Percent := 90 * (i + 1) div IndexStream.Count;
    end;
    Head.ENTRIES_OFFSET := FileBuffer.Position;
    for i := 0 to IndexStream.Count - 1 do
    begin
        FileBuffer.Write(indexs[i].START_OFFSET, 8);
        Percent := 90 + 10 * (i + 1) div IndexStream.Count;
    end;
    FileBuffer.Seek(0, soFromBeginning);
    FileBuffer.Write(Head.MAGIC[0], SizeOf(Pack_Head));
except
    Result := False;
end;
end;

exports GetPluginName;

exports GenerateEntries;

exports Decyption;

exports ReadData;

{$IFDEF SP}
exports PackData;
{$ENDIF}

begin
    //留空即可
end.
{ 代码结束 }

```

其中用到的核心文件请单独向我发邮件索要，并请务必说清楚。  
 （如果您不习惯使用 **delphi**，我可能不会提供代码）

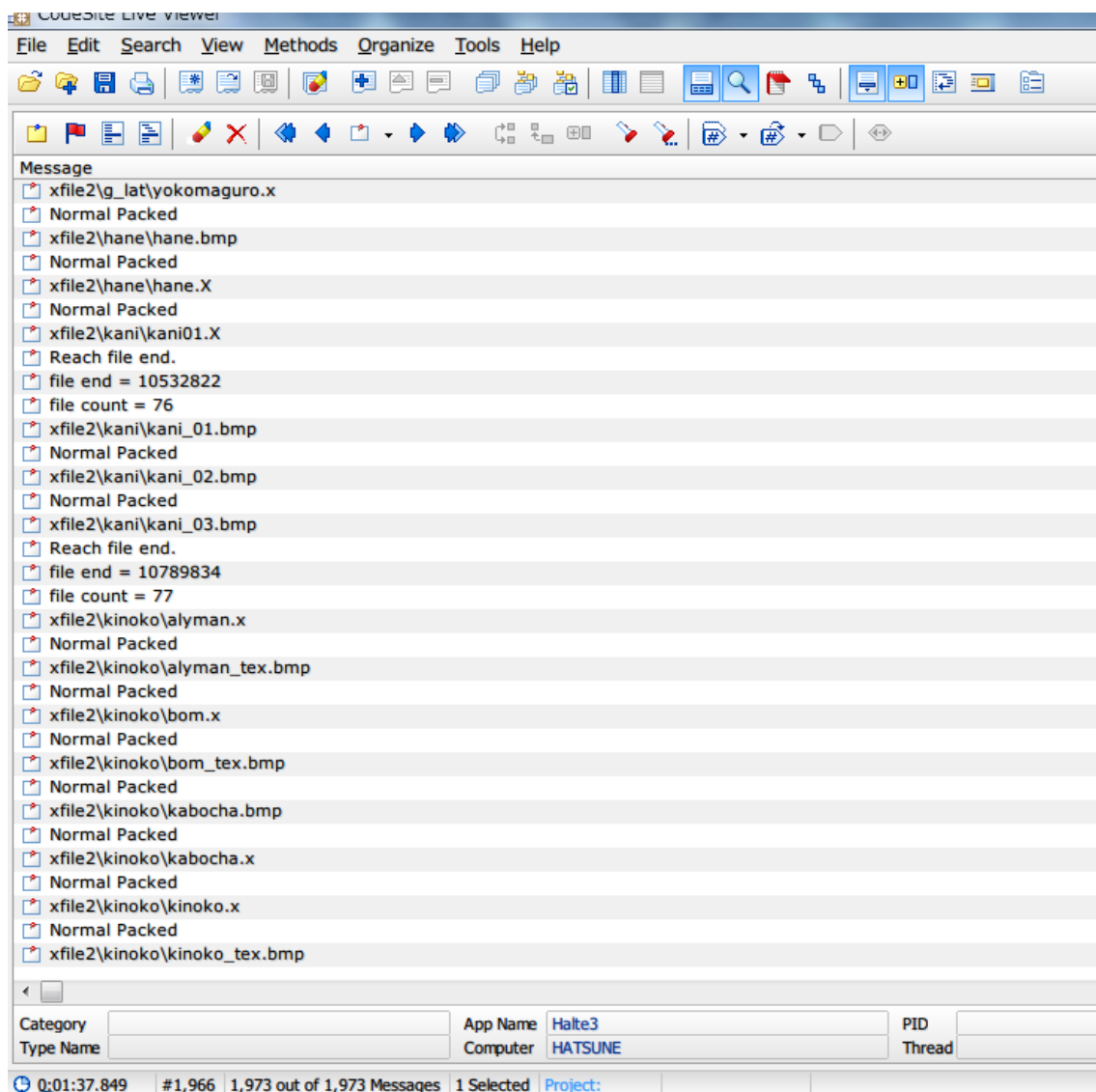
然而 **delphi** 到目前为止对 Dll 调试依然做得不好，因此建议使用 **codesite** 来发送调试信息。  
 比如以新公司 **Chariot** 的 **IndexsDecode** 为例：



```

procedure DecodeIndex(Buffer: Pbyte; Indexs: cardinal);
var
    i: Integer;
    Pos: Integer;
begin
    Pos := 0;
    {$IFDEF DEBUG}
        CodeSite.Send('Decode Index');
        CodeSite.Send('Loop count : ', Indexs);
    {$ENDIF}
    for i := 0 to Indexs - 1 do
        begin
            Decode(@Buffer[Pos], 0, 4);
            Decode(@Buffer[Pos + 4], 4, PDword(@Buffer[Pos])^ - 4);
            Pos := Pos + PDword(@Buffer[Pos])^;
        end;
    {$IFDEF DEBUG}
        CodeSite.Send('Decode Index Finish. ');
    {$ENDIF}
end;

```



接口说明 2:

在最新版本的 Halte3 里，Core 版本由 2.0 升为了 2.0Extended，增加了更加方便的回封，并且优先级高于老接口。

实际上，到目前为止 Halte 的内核和 GUI 依然没有完全分开，我只好增加一个中间模块来平衡这方面的问题。因此主要改动的并非内核，而是过渡模块。

下面用 EGS 的例子来作说明：

{ 前面略 }

```
function PackProcessData(Rlentry: TRIEntry;
```

```
    InputBuffer, FileOutBuffer: TStream): Boolean; //处理单个文件
```

```

var
  Compr: TMemoryStream;
begin
  Compr := TMemoryStream.Create;
  Lz77Compress(InputBuffer, Compr);
  Compr.Seek(0, soFromBeginning);
  with Rlentry do
  begin
    Rlentry.ARLength := InputBuffer.Size;
    Rlentry.AStoreLength := Compr.Size;
    Rlentry.AOffset := FileOutBuffer.Position;
  end;
  FileOutBuffer.CopyFrom(Compr, Compr.Size);
  Compr.Free;
end;

function PackProcessIndex(IndexStream: TIndexStream;
  IndexOutBuffer: TStream): Boolean; //整理 indexs
var
  indexs: array of Pack_Entries;
  i, k: integer;
begin
  SetLength(indexs, IndexStream.Count);
  ZeroMemory(@indexs[0].filename[1], IndexStream.Count *
    SizeOf(Pack_Entries));
  for i := 0 to IndexStream.Count - 1 do
  begin
    if length(IndexStream.Rlentry[i].AName) > 128 then
      raise Exception.Create('Filename too long. ');
    Move(Pbyte(AnsiString(IndexStream.Rlentry[i].AName))^,
      indexs[i].filename[1],
      length(IndexStream.Rlentry[i].AName));
    for k := 1 to $80 do
    begin
      if indexs[i].filename[k] = '\' then
        indexs[i].filename[k] := '/';
    end;
  end;
end;

```

```

end;
indexs[i].DecompressLen :=
    EndianSwap(IndexStream.Rlentry[i].ARLength);
indexs[i].start_offset :=
    EndianSwap(IndexStream.Rlentry[i].AOffset);
indexs[i].length :=
    EndianSwap(IndexStream.Rlentry[i].AStoreLength);
end;
IndexOutBuffer.Write(indexs[0].filename[1], IndexStream.Count *
    SizeOf(Pack_Entries));
end;

```

```

function MakePackage(IndexStream: TIndexStream;
    IndexOutBuffer, FileOutBuffer: TStream;
    Package: TFileBufferStream): Boolean; //生成封包
var
    Head: Pack_Head;
    i: integer;
begin
    Head.indexs := EndianSwap(IndexStream.Count);
    Package.Write(Head.indexs, 4);
    IndexOutBuffer.Seek(0, soFromBeginning);
    Package.CopyFrom(IndexOutBuffer, IndexOutBuffer.Size);
    FileOutBuffer.Seek(0, soFromBeginning);
    i := 0;
    while FileOutBuffer.Position < FileOutBuffer.Size do
        begin
            DivStream(FileOutBuffer, flen,
                Format(ExtractFilePath(Package.filename) + sformat, [i]));
            i := i + 1;
        end;
    end;
end;

```

```

exports GetPluginName;

```

```

exports GenerateEntries;

```

```
exports Decryption;
```

```
exports ReadData;
```

```
exports PackData;
```

```
exports PackProcessData; //这三个优先级高于 Packdada 只要有它们存在就不会使用  
Packdata
```

```
exports PackProcessIndex;
```

```
exports MakePackage;
```

```
{ 后略 }
```

新的接口细分了回封的处理，使得写起来更加方便，因为考虑到可能有特殊的封包不能采用这种形式，所以依然保留老式的接口。

不过这种借口也存在一个问题，如果是内存流，那么对于封包则会短时间占掉大量内存，而选择文件流则会导致速度变慢。这样使得两种模式都不如老式的平衡。

接口说明 3:

在最近的更新中 GUI 从 3.0 更新到了 3.0Ext。由此带来了内核的升级，同时支持了 **AutoDetect**。这是需要插件支持的，下面以 EGS 的插件为例：

```
//这里仅作为例子，因此用文件名简单的做检测。
```

```
//在实际的代码里实际并非如此。
```

```
function AutoDetect(FileBuffer: TFileBufferStream): TDetectState;
```

```
begin
```

```
    Result := dsFalse;
```

```
    try
```

```
        if SameText(ExtractFileName(FileBuffer.filename),
```

```
            'lnkdatas.bin') then
```

```
        begin
```

```

        Result := dsTrue;
        Exit;
    end;
    if CompareExt(FileBuffer.filename, '.bin') and
        SameText(LeftStr(ExtractFileName(FileBuffer.filename), 4),
            'data') then
    begin
        Result := dsUnknown;
        Exit;
    end;
except
    Result := dsError;
end;
end;
end;

```

\*\*\*\*\*中略\*\*\*\*\*

exports AutoDetect; //注意导出这个函数

exports GetPluginName;

exports GenerateEntries;

exports Decryption; //这个函数已经可以不导出了，在 Ext 里去掉了这个没用的函数

exports ReadData;

exports PackData;

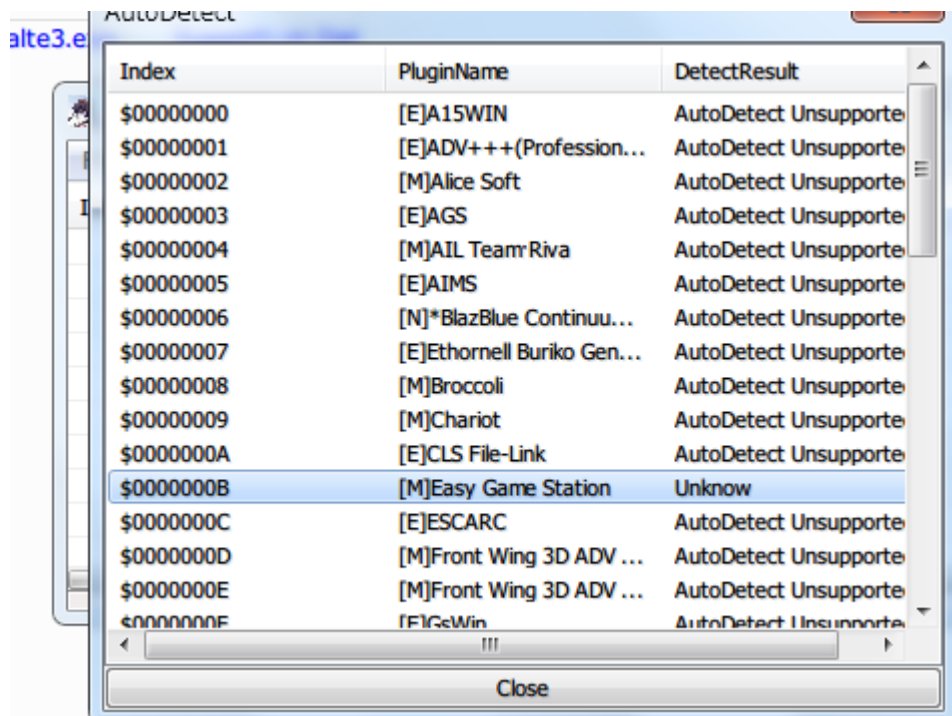
exports PackProcessData;

exports PackProcessIndex;

exports MakePackage;

可以看到检测结果：





最后是一个完整的例子，InnocentGrey 的抽取/回封（模式 2），可以以此作为模版：

```
library InnocentGrey;
```

```
uses
```

```
FastMM4 in '..\..\..\FastMM\FastMM4.pas',
SysUtils,
Classes,
CustomStream in '..\..\..\core\CustomStream.pas',
PluginFunc in '..\..\..\core\PluginFunc.pas',
Windows,
FastMM4Messages in '..\..\..\FastMM\FastMM4Messages.pas',
StrUtils;
```

```
{ $E GP }
```

```
{ $R *.res }
```

```
Type
```

```
Pack_Head = Packed Record
```

```
    Magic: array [0 .. 7] of AnsiChar;  
    Dummy_Indexs: Cardinal;  
    indexs: Cardinal;  
End;
```

```
Pack_Entries = packed Record  
    filename: array [1 .. 32] of AnsiChar;  
    start_offset: Cardinal;  
    EncodeFlag: Cardinal; // always $20000000?  
    DecompressLen: Cardinal;  
    length: Cardinal;  
End;
```

```
const
```

```
    Magic: AnsiString = 'PACKDAT.';
```

```
function AutoDetect(FileBuffer: TFileBufferStream): TDetectState;
```

```
var
```

```
    head: Pack_Head;
```

```
begin
```

```
    Result := dsFalse;
```

```
    try
```

```
        FileBuffer.Read(head.Magic[0], SizeOf(Pack_Head));
```

```
        if SameText(Trim(head.Magic), Magic) then
```

```
        begin
```

```
            if head.indexs < FileBuffer.Size then
```

```
            begin
```

```
                Result := dsTrue;
```

```
                Exit;
```

```
            end;
```

```
        end;
```

```
    except
```

```
        Result := dsError;
```

```
    end;
```

```
end;
```

```

function GetPluginName: PChar;
begin
    Result := PChar(FormatPluginName('Innocent Grey',
        clGameMaker));
end;

function GenerateEntries(IndexStream: TIndexStream;
    FileBuffer: TFileBufferStream): Boolean;
var
    head: Pack_Head;
    indexs: Pack_Entries;
    i: integer;
    Rlentry: TRlEntry;
begin
    Result := True;
    try
        FileBuffer.Seek(0, soFromBeginning);
        FileBuffer.Read(head.Magic[0], SizeOf(Pack_Head));
        if not SameText(Trim(head.Magic), Magic) then
            raise Exception.Create('Magic Error.');
```

for i := 0 to head.indexs - 1 do

```

        begin
            FileBuffer.Read(indexs.filename[1], SizeOf(Pack_Entries));
            Rlentry := TRlEntry.Create;
            Rlentry.AName := Trim(indexs.filename);
            Rlentry.AOffset := indexs.start_offset;
            Rlentry.AOrigin := soFromBeginning;
            Rlentry.AStoreLength := indexs.length;
            Rlentry.ARLength := indexs.DecompressLen;
            Rlentry.AKey := indexs.EncodeFlag;
            IndexStream.ADD(Rlentry);
        end;
    except
        Result := False;
    end;
end;
end;

```

```

procedure Decode(Input, Output: TStream);
var
    key, tmp: Cardinal;
    shifts: Byte;
    i: integer;
begin
    key := Input.Size div 4;
    key := (key shl ((key and 7) + 8)) xor key;
    while Input.Position < Input.Size do
    begin
        Input.Read(tmp, 4);
        tmp := tmp xor key;
        shifts := Byte(tmp mod 24);
        key := (key shl shifts) or (key shr (32 - shifts));
        Output.Write(tmp, 4);
    end;
end;

```

```

procedure Decode2(Input: TStream);
var
    tmp: Byte;
    i: integer;
begin
    while Input.Position < Input.Size do
    begin
        Input.Read(tmp, 1);
        tmp := not tmp;
        Input.Seek(-1, soFromCurrent);
        Input.Write(tmp, 1);
    end;
end;

```

```

function ReadData(FileBuffer: TFileBufferStream;
    OutBuffer: TStream): LongInt;
var

```

```

    Buffer: TGlCompressBuffer;
begin
    Buffer := TGlCompressBuffer.Create;
    try
        FileBuffer.ReadData(Buffer);
        Buffer.Seek(0, soFromBeginning);
        if (FileBuffer.PrevEntry.AKey and $FFFFFF) <> 0 then
            begin
                if (FileBuffer.PrevEntry.AKey and $10000) <> 0 then
                    begin
                        Decode(Buffer, Buffer.Output);
                    end;
                end
            end
        else
            Buffer.Output.CopyFrom(Buffer, Buffer.Size);
            if CompareExt(FileBuffer.PrevEntry.AName, '.s') then
                begin
                    Buffer.Output.Seek(0, soFromBeginning);
                    Decode2(Buffer.Output);
                end;
            Buffer.Output.Seek(0, soFromBeginning);
            OutBuffer.CopyFrom(Buffer.Output, Buffer.Output.Size);
        finally
            Buffer.Free;
        end;
    end;
end;

```

```

function PackProcessData(Rlentry: TRlEntry;
    InputBuffer, FileOutBuffer: TStream): Boolean;
var
    Compr: TGlCompressBuffer;
begin
    Compr := TGlCompressBuffer.Create;
    try
        InputBuffer.Seek(0, soFromBeginning);
        Compr.CopyFrom(InputBuffer, InputBuffer.Size);
    end;
end;

```

```

    if CompareExt(Rlentry.AName, '.s') then
    begin
        Compr.Seek(0, soFromBeginning);
        Decode2(Compr);
    end;
    Compr.Seek(0, soFromBeginning);
    with Rlentry do
    begin
        Rlentry.ARLength := Compr.Size;
        Rlentry.AStoreLength := Rlentry.ARLength;
        Rlentry.AOffset := FileOutBuffer.Position;
    end;
    FileOutBuffer.CopyFrom(Compr, Compr.Size);
finally
    Compr.Free;
end;
end;

function PackProcessIndex(IndexStream: TIndexStream;
    IndexOutBuffer: TStream): Boolean;
var
    indexs: array of Pack_Entries;
    i: integer;
begin
    SetLength(indexs, IndexStream.Count);
    ZeroMemory(@indexs[0].filename[1], IndexStream.Count *
        SizeOf(Pack_Entries));
    for i := 0 to IndexStream.Count - 1 do
    begin
        if length(IndexStream.Rlentry[i].AName) > 32 then
            raise Exception.Create('Filename too long.');
```

```

        Move(PByte(AnsiString(IndexStream.Rlentry[i].AName))^,
            indexs[i].filename[1],
            length(IndexStream.Rlentry[i].AName));
        indexs[i].DecompressLen := IndexStream.Rlentry[i].ARLength;
        indexs[i].start_offset := IndexStream.Rlentry[i].AOffset +
```



```

        IndexStream.Count * SizeOf(Pack_Entries) + SizeOf(Pack_head);
    indexs[i].length := IndexStream.Rlentry[i].AStoreLength;
    indexs[i].EncodeFlag := $20000000;
end;
IndexOutBuffer.Write(indexs[0].filename[1], IndexStream.Count *
    SizeOf(Pack_Entries));
end;

```

```

function MakePackage(IndexStream: TIndexStream;
    IndexOutBuffer, FileOutBuffer: TStream;
    Package: TFileBufferStream): Boolean;
var
    head: Pack_Head;
begin
    Move(PByte(Magic)^, head.Magic[0], length(Magic));
    head.indexs := IndexStream.Count;
    head.Dummy_Indexs := head.indexs;
    Package.Write(head.Magic[0], SizeOf(Pack_Head));
    IndexOutBuffer.Seek(0, soFromBeginning);
    Package.CopyFrom(IndexOutBuffer, IndexOutBuffer.Size);
    FileOutBuffer.Seek(0, soFromBeginning);
    Package.CopyFrom(FileOutBuffer, FileOutBuffer.Size);
end;

```

```

exports AutoDetect;

```

```

exports GetPluginName;

```

```

exports GenerateEntries;

```

```

exports ReadData;

```

```

exports PackProcessData;

```

```

exports PackProcessIndex;

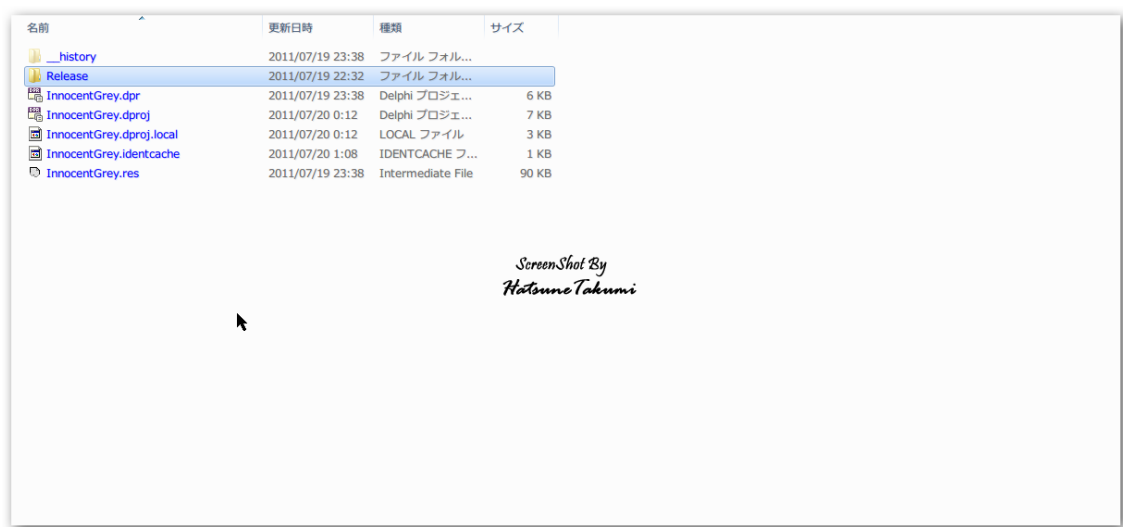
```

exports MakePackage;

begin

end.

相关截图:



Index	PluginName	DetectResult
\$0000000A	[M]Chariot	False
\$0000000B	[E]CLS File-Link	False
\$0000000C	[M]Noesis	False
\$0000000D	[M]Easy Game Station	False
\$0000000E	[E]ESCARC	False
\$0000000F	[M]Front Wing 3D AD...	False
\$00000010	[M]Front Wing 3D AD...	False
\$00000011	[E]GsWin	False
\$00000012	[M]Innocent Grey	True
\$00000013	[E]HyPack	False
\$00000014	[E]LC-ScriptEngine	False
\$00000015	[E]Malie	AutoDetect Unsuppor...
\$00000016	[E]MPF2.0	False
\$00000017	[E]NUG System	False
\$00000018	[E]OverTure Advance...	False

Index	Filename	Offset	StoreLength	Reallength
<input checked="" type="checkbox"/> \$00000001	bgm043.ogg	\$00000940	\$002F3FFE	\$002F3FFE
<input checked="" type="checkbox"/> \$00000002	bgm044.ogg	\$002F4940	\$002960E6	\$002960E6
<input checked="" type="checkbox"/> \$00000003	bgm045.ogg		\$0029A127	\$0029A127
<input checked="" type="checkbox"/> \$00000004	bgm046.ogg		\$003515B9	\$003515B9
<input checked="" type="checkbox"/> \$00000005	bgm046b.ogg		\$00340A80	\$00340A80
<input checked="" type="checkbox"/> \$00000006	bgm047.ogg		\$000C8604	\$000C8604
<input checked="" type="checkbox"/> \$00000007	bgm001.ogg		\$001BFF0A	\$001BFF0A
<input checked="" type="checkbox"/> \$00000008	bgm002.ogg		\$002717C3	\$002717C3
<input checked="" type="checkbox"/> \$00000009	bgm003.ogg		\$00299754	\$00299754
<input checked="" type="checkbox"/> \$0000000A	bgm004.ogg		\$00204C95	\$00204C95
<input checked="" type="checkbox"/> \$0000000B	bgm005.ogg		\$002E89B6	\$002E89B6
<input checked="" type="checkbox"/> \$0000000C	bgm006.ogg		\$00332B40	\$00332B40
<input checked="" type="checkbox"/> \$0000000D	bgm007.ogg		\$0027AE6E	\$0027AE6E
<input checked="" type="checkbox"/> \$0000000E	bgm008.ogg		\$0028F2E6	\$0028F2E6
<input checked="" type="checkbox"/> \$0000000F	bgm009.ogg		\$003384F2	\$003384F2
<input checked="" type="checkbox"/> \$00000010	bgm010.ogg		\$0025A5C8	\$0025A5C8
<input checked="" type="checkbox"/> \$00000011	bgm011.ogg		\$001FA66B	\$001FA66B
<input checked="" type="checkbox"/> \$00000012	bgm012.ogg		\$0034EE81	\$0034EE81
<input checked="" type="checkbox"/> \$00000013	bgm013.ogg		\$00111FAF	\$00111FAF
<input checked="" type="checkbox"/> \$00000014	bgm014.ogg	\$02F621F8	\$001B040D	\$001B040D
<input checked="" type="checkbox"/> \$00000015	bgm015.ogg	\$03112608	\$0021AB14	\$0021AB14
<input checked="" type="checkbox"/> \$00000016	bgm016.ogg	\$0332D11C	\$001E6E64	\$001E6E64
<input checked="" type="checkbox"/> \$00000017	bgm016a.ogg	\$03513F80	\$001B827A	\$001B827A
<input checked="" type="checkbox"/> \$00000018	bgm017.ogg	\$036CC1FC	\$003EDF20	\$003EDF20
<input checked="" type="checkbox"/> \$00000019	bgm018.ogg	\$03ABA11C	\$000B80FC	\$000B80FC

