Programmers should perform the Frog Developer Setup before taking the steps on this page. You will also need to install Visual Studio 2013, as explained on the Visual Studio page. The DirectX SDK is not currently necessary for Frog.

Debugging with Data Structures

On its own, the Visual Studio debugger doesn't know how to interpret and display the Frog data structures in a way that's generally useful. For example, to see the contents of a Map, you'd need to traverse the tree manually by following the pointers. However, if you use STL containers, the contents are displayed as convenient lists, regardless of the implementation details. To make the debugger display Frog's data structures in a useful way, you will need to make some adjustments to your IDE.

Visual Studio 2013

This version of the IDE uses an XML-based system called Natvis to customize the appearance of items in the debugger. Create a text file named "Webfoot.natvis" in the "Common7\Packages\Debugger\Visualizers" folder of the IDE installation. On this computer, the full path is "C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Packages\Debugger\Visualizers\Webfoot.natvis". If you don't have write access to this folder, you can alternatively put the file in "My Documents\Visual Studio 2013\Visualizers". Fill the file with the following content, and it should take effect the next time you start a debugger session.

```
<?xml version="1.0" encoding="utf-8"?>
<AutoVisualizer xmlns="http://schemas.microsoft.com/vstudio/debugger/natvis/2010">

<Type Name="Webfoot::Table&lt;*&gt;">
   <DisplayString>{{size = {size}}}</DisplayString>
   <Expand>
      <Item Name="size">size</Item>
      <Item Name="capacity">capacity</Item>
      <ArrayItems>
         <Size>size</Size>
         <ValuePointer>data</ValuePointer>
      </ArrayItems>
   </Expand>
</Type>

<Type Name="Webfoot::TableStatic&lt;*,*&gt;">
   <DisplayString>{{size = {size}}}</DisplayString>
   <Expand>
      <Item Name="size">size</Item>
      <Item Name="capacity">$T2</Item>
      <ArrayItems>
         <Size>size</Size>
         <ValuePointer>data</ValuePointer>
      </ArrayItems>
   </Expand>
</Type>

<Type Name="Webfoot::List&lt;*&gt;">
   <DisplayString>{{size = {size}}}</DisplayString>
   <Expand>
      <Item Name="size">size</Item>
      <Item Name="capacity">size + availableNodeCount</Item>
      <LinkedListItems>
         <Size>size</Size>
         <HeadPointer>head.next</HeadPointer>
         <NextPointer>next</NextPointer>
         <ValueNode>data</ValueNode>
      </LinkedListItems>
   </Expand>
</Type>

<Type Name="Webfoot::ListStatic&lt;*,*&gt;">
   <DisplayString>{{size = {size}}}</DisplayString>
   <Expand>
      <Item Name="size">size</Item>
      <Item Name="capacity">$T2</Item>
      <LinkedListItems>
         <Size>size</Size>
         <HeadPointer>head.next</HeadPointer>
         <NextPointer>next</NextPointer>
         <ValueNode>data</ValueNode>
      </LinkedListItems>
   </Expand>
</Type>

<Type Name="Webfoot::Map&lt;*&gt;">
   <DisplayString>{{size = {size}}}</DisplayString>
   <Expand>
      <Item Name="size">size</Item>
      <TreeItems>
         <Size>size</Size>
         <HeadPointer>root</HeadPointer>
         <LeftPointer>left</LeftPointer>
         <RightPointer>right</RightPointer>
         <ValueNode Condition="level &gt; 0">this</ValueNode>
      </TreeItems>
   </Expand>
</Type>

<Type Name="Webfoot::MapNode&lt;*&gt;">
   <DisplayString>{{key = {key}, value = {value}}}</DisplayString>
   <Expand>
      <Item Name="key">key</Item>
      <Item Name="value">value</Item>
   </Expand>
</Type>

<Type Name="Webfoot::JSONValue">
   <DisplayString Condition="_valueType == 0">value = empty</DisplayString>
   <DisplayString Condition="_valueType == 1">value = {booleanValue}</DisplayString>
   <DisplayString Condition="_valueType == 2">value = {numberValue}</DisplayString>
   <DisplayString Condition="_valueType == 3">value = {stringValue}</DisplayString>
   <DisplayString Condition="_valueType == 4">value = {objectValue}</DisplayString>
   <DisplayString Condition="_valueType == 5">value = {arrayValue}</DisplayString>
   <Expand>
      <Item Name="value" Condition="_valueType == 1">booleanValue</Item>
      <Item Name="value" Condition="_valueType == 2">numberValue</Item>
      <Item Name="value" Condition="_valueType == 3">stringValue</Item>
      <Item Name="value" Condition="_valueType == 4">objectValue</Item>
      <Item Name="value" Condition="_valueType == 5">arrayValue</Item>
   </Expand>
</Type>

</AutoVisualizer>
```

Visual Studio 2005

Microsoft has a little language for this, which is used in a file named autoexp.dat. The file is in the "Common7\Packages\Debugger" subfolder of Visual Studio 2005's folder. On this computer, the full path is "C:\Program Files\Programming\Microsoft Visual Studio 8\Common7\Packages\Debugger\autoexp.dat". I put together some scripts for this file to make debugging with Frog data structures just as easy. Getting this to work involves adding to the "Visualizer" section of the file, which is labeled "DO NOT MODIFY". However, I suspect they just don't want to document their narrow-purpose language, ensure backwards compatability, or fix any minor shortcomings of the system. For example, it seemed to have trouble with the fact that there was both a JSONValue::ValueType and a JSONValue::valueType. Also, this syntax works…

```
Webfoot::Table<*>{
   ...
}
```

… but the following syntax does not…

```
Webfoot::Table<*>
{
   ...
}
```

While this type of visualizer customization is evidently not supported, it's worked for me so far in

Visual Studio 2005 RTM. I've found other people on forums, including the MSDN forum, that have used this system to add add visualizers for their objects. I haven't tested it with other versions of Visual Studio though.

To add the Frog visualizers to your setup…

- Back up autoexp.dat
- Open autoexp.dat in a text editor.
- Copy and paste the following scripts to the end of the [Visualizer] section. On this computer, it is toward the end of the file, just before the [hresult] section.

If you want to debug the data structures themselves, just go back to the normal autoexp.dat.

Scripts

```
;------------------------------------------------------------------------------
; Frog Data Structures
;------------------------------------------------------------------------------

Webfoot::Table<*>{
   children
   (
      #array
      (
         expr : ($c.data)[$i],
         size : $c.size
      )
   )

   preview
   (
      #(
         "[", $c.size, "](",

         #array
         (
            expr : ($c.data)[$i],
            size : $c.size
         ),

         ")"
      )
   )
}

Webfoot::TableStatic<*>{
   children
   (
      #array
      (
         expr : ($c.data)[$i],
         size : $c.size
      )
   )

   preview
   (
      #(
         "[", $c.size, "](",

         #array
         (
            expr : ($c.data)[$i],
            size : $c.size
         ),

         ")"
      )
   )
}

Webfoot::List<*>{
   children
   (
      #list
      (
         head : $c.head.next,
         skip : &$c.tail,
         next : next
      )
   )

   preview
   (
      #(
         "[", $c.size, "](",

         #list
         (
            head : $c.head.next,
            skip : &$c.tail,
            next : next
         ),

         ")"
      )
   )
}

Webfoot::ListStatic<*>{
   children
   (
      #list
      (
         head : $c.head.next,
         skip : &$c.tail,
         next : next
      )
   )

   preview
   (
      #(
         "[", $c.size, "](",

         #list
         (
            head : $c.head.next,
            skip : &$c.tail,
            next : next
         ),

         ")"
      )
   )
}

Webfoot::ListNode<*>{
   children
   (
      $c.data
   )

   preview
   (
      $c.data
   )
}

Webfoot::Map<*>{
   children
   (
      #tree
      (
         head : $c.root,
         skip : $c.nullNode,
         size : $c.size,
         left : left,
         right : right
      )
   )

   preview
   (
      #(
         "[", $c.size, "](",

         #tree
         (
            head : $c.root,
            skip : $c.nullNode,
            size : $c.size,
            left : left,
            right : right
         ),

         ")"
      )
   )
}

Webfoot::MapNode<*>{
   children
   (
      #(
         key : $c.key,
         value : $c.value
      )
   )

   preview
   (
      #( "key=", $c.key, ", value=", $c.value )
   )
}

Webfoot::JSONValue{
   children
   (
      #(
         type : $c._valueType,
         #if($c._valueType == 1) (
            #( value : $c.booleanValue )
         ) #elif($c._valueType == 2) (
            #( value : $c.numberValue )
         ) #elif($c._valueType == 3) (
            #( value : $c.stringValue )
         ) #elif($c._valueType == 4) (
            #( value : $c.objectValue )
         ) #elif($c._valueType == 5) (
            #( value : $c.arrayValue )
         ) #else (
            #( value : "empty" )
         )
      )
   )

   preview
   (
      #(
         "type=", $c._valueType, ", value=",
         #if($c._valueType == 1) (
            #( $c.booleanValue )
         ) #elif($c._valueType == 2) (
            #( $c.numberValue )
         ) #elif($c._valueType == 3) (
            #( $c.stringValue )
         ) #elif($c._valueType == 4) (
            #( $c.objectValue )
         ) #elif($c._valueType == 5) (
            #( $c.arrayValue )
         ) #else (
            #( "empty" )
         )
      )
   )
}
```

Xcode

By default, Xcode 4 puts the output of builds in a temporary location far from the sources. To fix this, open Xcode, and click Xcode | Preferences. Select the “Locations” tab and click the “Advanced” button by “Derived Data”. Select “Legacy”, and click “Done”.

3ds Max

When working on the 3ds Max 9 exporter using Visual Studio 2013, you'll need to make some changes locally.

- VS2013 does not appear to handle the “3DSMAXSDK_PATH” environment variables correctly. To work around this, duplicate this environment variable, and give it the name “MAXSDK_PATH”.
- When compiling the 3ds max exporter using VS2013, there are a couple items in a 3ds Max SDK header file that conflict with the newer C++11 standard. To resolve this, you'll need to manually edit “3ds Max 9 SDK\maxsdk\include\texutil.h”. Open the file and comment out the definition/declaration for fmin and fmax.

Frog Engine

/share/Web/wiki/dokuwiki/data/pages/frog_programmer_setup.txt · Last modified: 2014/09/25 22:39 by mpellegrini