Subject: AngelScript - AngelCode Scripting Library
Posted by Sender Ghost on Fri, 18 Nov 2011 09:37:26 GMT
View Forum Message <> Reply to Message

Homepage:
http://www.angelcode.com/angelscript/

License:
zlib

Version:
2.30.0 (February 22nd, 2015)

Description:
The AngelCode Scripting Library, or AngelScript as it is also known, is an extremely flexible cross-platform scripting library designed to allow applications to extend their functionality through external scripts. It has been designed from the beginning to be an easy to use component, both for the application programmer and the script writer.

Efforts have been made to let it call standard C functions and C++ methods with little to no need for proxy functions. The application simply registers the functions, objects, and methods that the scripts should be able to work with and nothing more has to be done with your code. The same functions used by the application internally can also be used by the scripting engine, which eliminates the need to duplicate functionality.

For the script writer the scripting language follows the widely known syntax of C/C++, but without the need to worry about pointers and memory leaks. Contrary to most scripting languages, AngelScript uses the common C/C++ datatypes for more efficient communication with the host application.

Documentation:
http://www.angelcode.com/angelscript/documentation.html

In the attachments you could find AngelScript source code, add-ons, samples, converted to U++ packages and documentation.

To note:
There were minor changes for include files for original sources to adapt for U++ package structure.

Edit: Updated to 2.30.0 version.

File Attachments

1) AngelScript_v2.30.0.7z, downloaded 378 times

Subject: Re: AngelScript - AngelCode Scripting Library

Posted by koldo on Fri, 18 Nov 2011 11:12:31 GMT
View Forum Message <> Reply to Message

Hello Sender

It seems really interesting.

It seems worthwhile to include it in Bazaar:
- Package
- A sample project
- Original code in plugin/Bazaar

It would also be very useful to have a benchmark to compare it with this example from Mirek. I would like to add to it the TCC (Tiny C Compiler) part.

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by Sender Ghost on Fri, 18 Nov 2011 14:38:05 GMT
View Forum Message <> Reply to Message

Hello, Koldo.

koldo wrote on Fri, 18 November 2011 12:12
It seems worthwhile to include it in Bazaar

For now, I just uploaded it here. Feel free to test it.
koldo wrote on Fri, 18 November 2011 12:12
It would also be very useful to have a benchmark to compare it with this example from Mirek. I would like to add to it the TCC (Tiny C Compiler) part.

Based on AngelScript samples, I added the same calculation functions for Mirek source code, with following results (for MSC9 compiler in optimal mode):

1 / (1 - x * y + x - y) = -0.01851851852
fn->Execute() = -0.01851851852
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
TIMING AngelScript (fully interpreted): 110.00 ms - 110.00 ms (110.00 ms / 1), min: 110.00 ms, max: 110.00 ms, nesting: 1 - 1
TIMING AngelScript (interpreted): 332.00 ms - 332.00 ms (332.00 ms / 1), min: 332.00 ms, max: 332.00 ms, nesting: 1 - 1
TIMING Direct : 14.00 ms - 14.00 ms (14.00 ms / 1), min: 14.00 ms, max: 14.00 ms, nesting: 1 - 1
TIMING Compiled : 58.00 ms - 58.00 ms (58.00 ms / 1), min: 58.00 ms, max: 58.00 ms, nesting: 1 - 1

TIMING Interpreted : 807.00 ms - 807.00 ms (807.00 ms / 1), min: 807.00 ms, max: 807.00 ms, nesting: 1 - 1

But need to note, that it is also possible to bind compiled functions to script functions, cache compiled functions, etc. Therefore, the real optimized results might be different.

The changed package could be found in the attachments.

File Attachments

1) UppCompiler.zip, downloaded 499 times

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by Sender Ghost on Fri, 18 Nov 2011 16:37:18 GMT
View Forum Message <> Reply to Message

Also, I tested AngelScript with bindings for two cases:
1:
Toggle Spoiler

```
double calculate()
{
 double x, y, sum = 0;
 for(x = 0; x < 1; x += 0.001)
  for(y = 0; y < 1; y += 0.001)
   sum += 1 / (1 - x * y + x - y);
 return sum;
}

CONSOLE_APP_MAIN
{
// ...
 // Create the AngelScript engine
 asIScriptEngine *engine = asCreateScriptEngine(ANGELSCRIPT_VERSION);
 if(engine == 0)
 {
  Cout() << "Failed to create AngelScript engine.\n";
  SetExitCode(1);
  return;
 }

 engine->RegisterGlobalFunction("double calculate()", asFUNCTION(calculate),
asCALL_CDECL);
  const String script =
  "double compute() {\n"
  " return calculate();\n"
  "}";
```

```
asIScriptModule *mod = engine->GetModule("script", asGM_ALWAYS_CREATE);
if(mod->AddScriptSection("script", ~script, script.GetCount()) < 0) {
 Cout() << "AddScriptSection() failed\n";
 engine->Release();
 SetExitCode(1);
 return;
}

if(mod->Build() < 0) {
 Cout() << "Build() failed\n";
 engine->Release();
 SetExitCode(1);
 return;
}

// Create a context that will execute the script.
asIScriptContext *ctx = engine->CreateContext();
if(ctx == 0) {
 Cout() << "Failed to create the context.\n";
 ctx->Release();
 engine->Release();
 SetExitCode(1);
 return;
}

// Find the function id for the function we want to execute.
int funcId = engine->GetModule("script")->GetFunctionIdByDecl("double compute()");
if(funcId < 0) {
 Cout() << "The function \"compute\" wasn't found.\n";
 ctx->Release();
 engine->Release();
 SetExitCode(1);
 return;
}

{
 RTIMING("AngelScript (interpreted with binding)");
 double sum = 0;
 if(ctx->Prepare(funcId) < 0) {
  Cout() << "Failed to prepare the context.\n";
  SetExitCode(1);
  ctx->Release();
  engine->Release();
  return;
 }

 if(ctx->Execute() == asEXECUTION_FINISHED)
  sum = ctx->GetReturnDouble();
```

```
  RDUMP(sum);
 }

 ctx->Release();
 engine->Release();
}
```

With following results:

```
1 / (1 - x * y + x - y) = -0.01851851852
fn->Execute() = -0.01851851852
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
TIMING AngelScript (interpreted with binding): 16.00 ms - 16.00 ms (16.00 ms / 1 ), min: 16.00 ms, max: 16.00 ms, nesting: 1 - 1
TIMING Direct         : 13.00 ms - 13.00 ms (13.00 ms / 1 ), min: 13.00 ms, max: 13.00 ms, nesting: 1 - 1
TIMING Compiled       : 53.00 ms - 53.00 ms (53.00 ms / 1 ), min: 53.00 ms, max: 53.00 ms, nesting: 1 - 1
TIMING Interpreted    : 800.00 ms - 800.00 ms (800.00 ms / 1 ), min: 800.00 ms, max: 800.00 ms, nesting: 1 - 1
```

2:
Toggle Spoiler

```
double calculate(double x, double y)
{
 return 1 / (1 - x * y + x - y);
}

CONSOLE_APP_MAIN
{
// ...
 // Create the AngelScript engine
 asIScriptEngine *engine = asCreateScriptEngine(ANGELSCRIPT_VERSION);
 if(engine == 0)
 {
  Cout() << "Failed to create AngelScript engine.\n";
  SetExitCode(1);
  return;
 }

 engine->RegisterGlobalFunction("double calculate(double, double)", asFUNCTION(calculate), asCALL_CDECL);

 const String script =
```

```
 "double compute() {\n"
 " double x, y, sum = 0;\n"
 " for (x = 0; x < 1; x += 0.001)\n"
 "  for (y = 0; y < 1; y += 0.001)\n"
 "   sum += calculate(x, y);\n"
 " return sum;\n"
 "}";

asIScriptModule *mod = engine->GetModule("script", asGM_ALWAYS_CREATE);
if(mod->AddScriptSection("script", ~script, script.GetCount()) < 0) {
 Cout() << "AddScriptSection() failed\n";
 engine->Release();
 SetExitCode(1);
 return;
}

if(mod->Build() < 0) {
 Cout() << "Build() failed\n";
 engine->Release();
 SetExitCode(1);
 return;
}

// Create a context that will execute the script.
asIScriptContext *ctx = engine->CreateContext();
if(ctx == 0) {
 Cout() << "Failed to create the context.\n";
 ctx->Release();
 engine->Release();
 SetExitCode(1);
 return;
}

// Find the function id for the function we want to execute.
int funcId = engine->GetModule("script")->GetFunctionIdByDecl("double compute()");
if(funcId < 0) {
 Cout() << "The function \"compute\" wasn't found.\n";
 ctx->Release();
 engine->Release();
 SetExitCode(1);
 return;
}

{
 RTIMING("AngelScript (fully interpreted with binding)");
 double sum = 0;
 if(ctx->Prepare(funcId) < 0) {
  Cout() << "Failed to prepare the context.\n";
```

```
   SetExitCode(1);
   ctx->Release();
   engine->Release();
   return;
  }

  if(ctx->Execute() == asEXECUTION_FINISHED)
   sum = ctx->GetReturnDouble();

  RDUMP(sum);
 }

 ctx->Release();
 engine->Release();
}
```

With following results:

```
1 / (1 - x * y + x - y) = -0.01851851852
fn->Execute() = -0.01851851852
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
TIMING AngelScript (fully interpreted with binding): 170.00 ms - 170.00 ms (170.00 ms / 1 ), min: 170.00 ms, max: 170.00 ms, nesting: 1 - 1
TIMING Direct         : 14.00 ms - 14.00 ms (14.00 ms / 1 ), min: 14.00 ms, max: 14.00 ms, nesting: 1 - 1
TIMING Compiled       : 54.00 ms - 54.00 ms (54.00 ms / 1 ), min: 54.00 ms, max: 54.00 ms, nesting: 1 - 1
TIMING Interpreted    : 793.00 ms - 793.00 ms (793.00 ms / 1 ), min: 793.00 ms, max: 793.00 ms, nesting: 1 - 1
```

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by koldo on Sat, 19 Nov 2011 00:35:57 GMT
View Forum Message <> Reply to Message

Hello Sender and Mirek

I have added Tcc code and the results are:

```
TIMING Tcc (fully interpreted, included compiling): 23.00 ms - 23.00 ms (23.00 ms / 1 ), min: 23.00 ms, max: 23.00 ms, nesting: 1 - 1
TIMING Tcc (interpreted): 20.00 ms - 20.00 ms (20.00 ms / 1 ), min: 20.00 ms, max: 20.00 ms, nesting: 1 - 1
TIMING AngelScript (fully interpreted): 169.00 ms - 169.00 ms (169.00 ms / 1 ), min: 169.00 ms,
```

max: 169.00 ms, nesting: 1 - 1
TIMING AngelScript (interpreted): 485.00 ms - 485.00 ms (485.00 ms / 1), min: 485.00 ms, max: 485.00 ms, nesting: 1 - 1
TIMING Direct : 20.00 ms - 20.00 ms (20.00 ms / 1), min: 20.00 ms, max: 20.00 ms, nesting: 1 - 1
TIMING Compiled : 89.00 ms - 89.00 ms (89.00 ms / 1), min: 89.00 ms, max: 89.00 ms, nesting: 1 - 1
TIMING Interpreted : 2.20 s - 2.20 s (2.20 s / 1), min: 2.20 s , max: 2.20 s , nesting: 1 - 1

The code is:

```
 try {
  Tcc tcc;
  tcc.Init();
  //tcc.AddIncludePath(includePath);
  //tcc.AddLibraryPath(libsPath);
  tcc.Compile(script);
  tcc.Link();
  double (*calculate)(double, double) = (double (*)(double, double))tcc.GetSymbol("calculate");

  RTIMING("Tcc (interpreted)");
  double sum = 0;
  for(x = 0; x < 1; x += 0.001)
   for(y = 0; y < 1; y += 0.001)
    sum += calculate(x, y);
  RDUMP(sum);
 } catch(Exc err){
  Cout() << err;
 }
 try {
  RTIMING("Tcc (fully interpreted, included compiling)");
  Tcc tcc;
  tcc.Init();
  //tcc.AddIncludePath(includePath);
  //tcc.AddLibraryPath(libsPath);
  tcc.Compile(script2);
  tcc.Link();
  double (*calculate)() = (double (*)())tcc.GetSymbol("calculate");

  double sum = calculate();
  RDUMP(sum);
 } catch(Exc err){
  Cout() << err;
 }
```

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by Sender Ghost on Sat, 19 Nov 2011 05:19:52 GMT

View Forum Message <> Reply to Message

Thanks Koldo, for testing.
The TCC results looks quite faster.

I didn't want to intersect various possibilities/disadvantages of TCC and AngelScript engines, but in result they are different: support for various processors, platforms (32/64 bits), possibility to do C++ object oriented programming (and bind them vise versa), etc.

The purpose of Mirek's "little experiment", as I understood, is to develop something new for U++, with comparable quality and speed.
There are many other script engines and even whole compilers with jit capabilities, like LLVM/Clang.

Edit: Updated link.

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by koldo on Sat, 19 Nov 2011 14:24:34 GMT
View Forum Message <> Reply to Message

Hello Sender

Yes, TCC applicability is limited and in fact the project seems to be slightly stopped. In addition it is only C.

In fact I use Mirek parsing technology to analyze a "special" language I use, and with that TCC C code is generated and run.

It is less smart but it is rather fast, so I get the best of both.

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by mdelfede on Sat, 19 Nov 2011 15:33:19 GMT
View Forum Message <> Reply to Message

What about squirrel ? www.squirrel-lang.org.
I looked at it some time ago and seems quite interesting.

Max

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by Sender Ghost on Thu, 24 Nov 2011 23:37:21 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sat, 19 November 2011 16:33What about squirrel ? www.squirrel-lang.org.

I looked at it some time ago and seems quite interesting.

After private conversation with Massimo, I decided to post Squirrel package(s) here.
Feel free to test it (or change it to fit your needs).

Edit: Updated link.

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by koldo on Tue, 27 Mar 2012 09:41:50 GMT
View Forum Message <> Reply to Message

Hello Sender

Thank you for your package. I think it should be in Bazaar .

I have begun to work with it in a project where a scripting language with operator overload is needed..

Subject: Re: AngelScript - AngelCode Scripting Library
Posted by Sender Ghost on Tue, 03 Jul 2012 02:30:20 GMT
View Forum Message <> Reply to Message

The AngelScript archive updated to 2.24.0a version.

Also I updated UppCompiler archive (in attachment) with changes of 2.24.0a version and ScopeRelease helper class template for releasing resources on scope completion (e.g. for asIScriptEngine, asIScriptContext objects, which have Release virtual methods):
Toggle source code

```
template<class T>
class ScopeRelease {
protected:
 T *object;
public:
 ScopeRelease() : object(0)     { }
 ScopeRelease(T *x) : object(x) { }
 ~ScopeRelease()               { if (object) object->Release(); }
 ScopeRelease& operator=(T *x)  { object = x; return *this; }
 T *GetObject()                { return object; }
 T *operator->()               { return object; }
 T *operator~()                { return object; }
 bool operator==(T *x)          { return object == x; }
 bool operator!=(T *x)         { return object != x; }
 virtual bool Release()
{ // In case you need to release the object before the scope completion.
  if (object) {
```

```
   object->Release();
   object = 0;
   return true;
  }

  return false;
 }
};
```

With following results for 2.24.0a version:

```
1 / (1 - x * y + x - y) = -0.01851851852
fn->Execute() = -0.01851851852
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
sum = 5190404.858
TIMING AngelScript (fully interpreted): 78.00 ms - 78.00 ms (78.00 ms / 1 ), min: 78.00 ms, max: 78.00 ms, nesting: 1 - 1
TIMING AngelScript (interpreted): 321.00 ms - 321.00 ms (321.00 ms / 1 ), min: 321.00 ms, max: 321.00 ms, nesting: 1 - 1
TIMING Direct         : 13.00 ms - 13.00 ms (13.00 ms / 1 ), min: 13.00 ms, max: 13.00 ms, nesting: 1 - 1
TIMING Compiled       : 53.00 ms - 53.00 ms (53.00 ms / 1 ), min: 53.00 ms, max: 53.00 ms, nesting: 1 - 1
TIMING Interpreted    : 837.00 ms - 837.00 ms (837.00 ms / 1 ), min: 837.00 ms, max: 837.00 ms, nesting: 1 - 1
```

File Attachments

1) UppCompiler.zip, downloaded 384 times