Microsoft

PDC2008

PROFESSIONAL DEVELOPERS CONFERENCE

PDC2008
PROFESSIONAL DEVELOPERS CONFERENCE

Mono And .NET

→ Miguel de Icaza
VP Developer Platform
Novell, Inc.

Agenda

- Introduction to Mono
- Mono’s Customizable CLR
- Mono’s C# Eval
- Assembly binary reshaping
- Turbo charging games and graphics
- Static Compilation
- Others

Mono 2.0

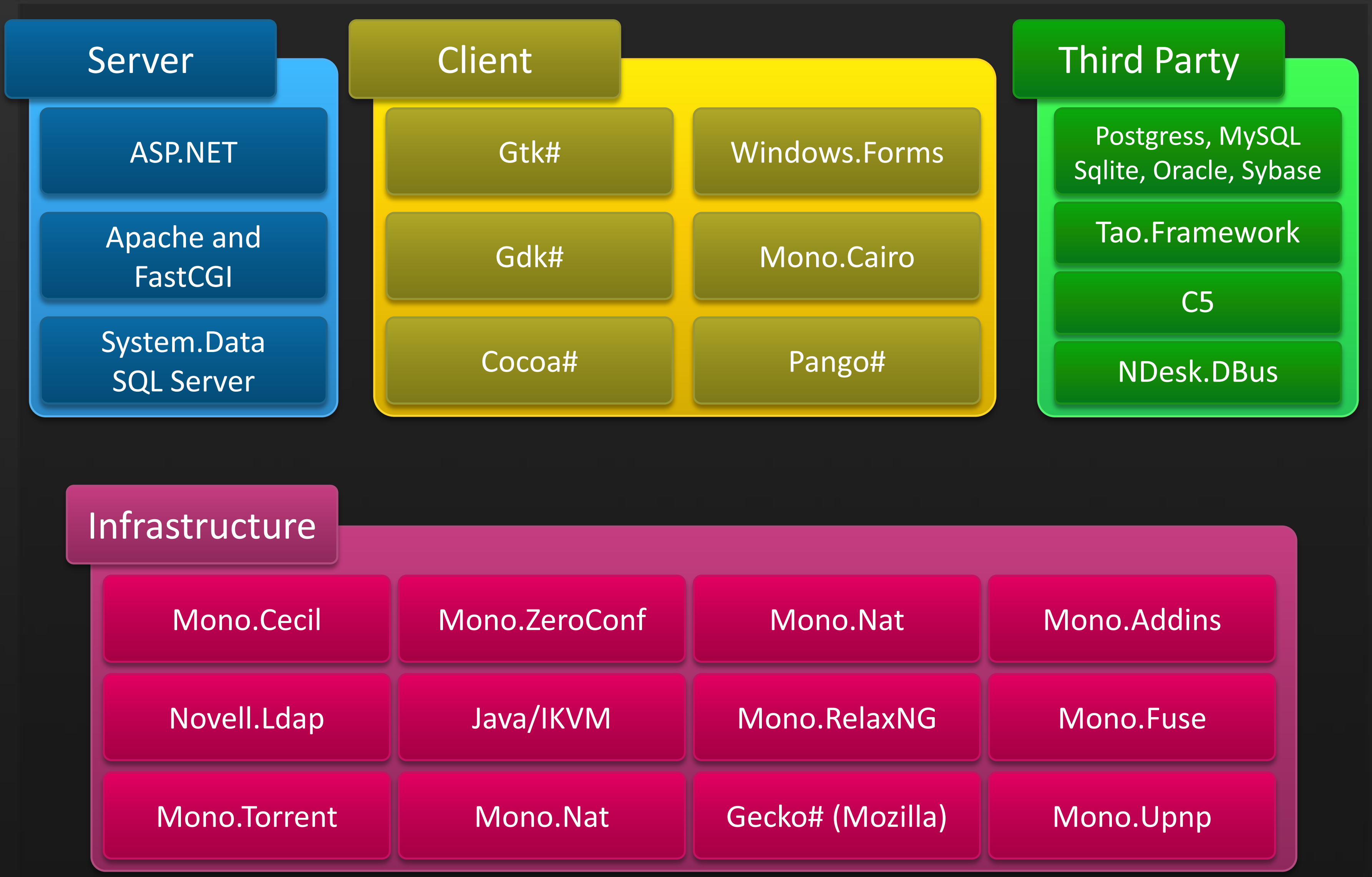
Just released!

- An open source .NET implementation:
  - A subset of .NET
  - Sponsored by Novell
  - ~120 non-affiliated contributors (1.2 -> 2.0)
- Direction driven by contributors

Compatibility

- Our goal is to have a compatible runtime to the CLR
    - ECMA specifications make it possible
    - Develop, build, debug on Visual Studio or Unix
    - Deploy on Linux, Mac OSX and embedded

APIs



Server

- ASP.NET
- Apache and FastCGI
- System.Data SQL Server

Client

- Gtk#
- Windows.Forms
- Gdk#
- Mono.Cairo
- Cocoa#
- Pango#

Third Party

- Postgress, MySQL Sqlite, Oracle, Sybase
- Tao.Framework
- C5
- NDesk.DBus

Infrastructure

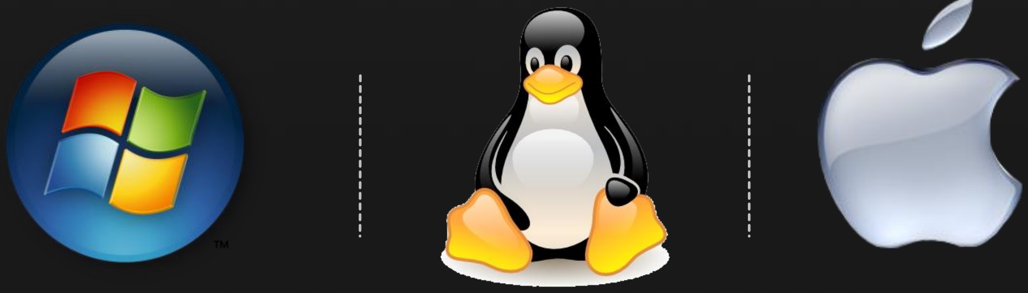
- Mono.Cecil
- Mono.ZeroConf
- Mono.Nat
- Mono.Addins
- Novell.Ldap
- Java/IKVM
- Mono.RelaxNG
- Mono.Fuse
- Mono.Torrent
- Mono.Nat
- Gecko# (Mozilla)
- Mono.Upnp

Mono's CLI Implementation

- We can offer a few bonuses
  - Take .NET where no .NET has gone before
  - Offering new forward-compatible features
  - Support special scenarios

Mono's CLI Implementation

- We can offer a few bonuses
  - Take .NET where no .NET has gone before
  - Offering new forward-compatible features
  - Support special scenarios



CLRs For Everyone

- Microsoft has the CLR, CF and the CoreCLR
  - CoreCLR is a small version of CLR
  - CoreCLR used in Mesh and Silverlight
  - Compact Framework used in XNA

CLRs For Everyone

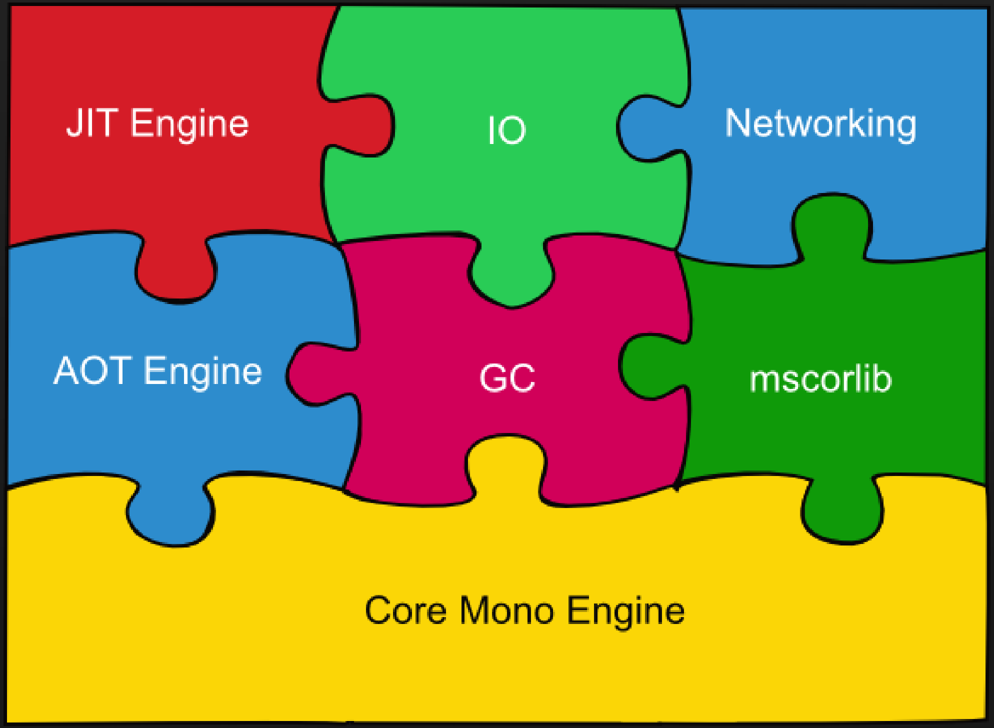
- Microsoft has the CLR, CF and the CoreCLR
  - CoreCLR is a small version of CLR
  - CoreCLR used in Mesh and Silverlight
  - Compact Framework used in XNA

- For everyone else, there is Mono

Mono Adaptability

From full framework to tailored framework

- Full framework is 100 megs (uncompressed)
- Minimal setup is 2 megs (uncompressed)
- Modular runtime can be shrunk/grown:



PDC2008

PROFESSIONAL DEVELOPERS CONFERENCE

The Evolution of a Compiler

(or, C# 5 today)

Mono's C# 3.0 Compiler

- C# compiler written in C#
  - Originally, a project to learn to write C# code

Mono's C# 3.0 Compiler

- C# compiler written in C#
  - Originally, a project to learn to write C# code
- First bootstrap (2001)
  - 17 seconds to bootstrap, 10,000 lines
  - csc compiled it in a second

Mono's C# 3.0 Compiler

- C# compiler written in C#
  - Originally, a project to learn to write C# code
- First bootstrap (2001)
  - 17 seconds to bootstrap 10,000 lines
  - csc compiled it in a second
- Speed is no longer a problem
  - Today 82,000 lines in 2.2 seconds
  - 1.6x slower than csc

Mono.CSharp.dll – Compiler Service

- Mono.CSharp.Evaluator
  - Encapsulates the compiler in one class
  - Provides C# Eval and C# Run:

```
using System;
using Mono.CSharp;

class MyFirstCSharpInterpreter {
    static void Main (string [] args)
    {
        object r = Evaluator.Evaluate (args [0]);
        Console.WriteLine (r);
    }
}
```

Mono.CSharp – Applications

- Read-Eval-Print-Loop (repl)
- Script applications with C#
- Rapid prototyping in target language
- Automation

- Would be cool to have this on every app!

The csharp Command

- Python and Ruby have interactive shells
- Read-Eval-Print Loop
- Expressions and Statements:

```
csharp> 1;
1;
csharp> “Hello, World”.IndexOf (“,”);
5;
csharp> 1 +
      > 2;
3
csharp> var a = Console.ReadLine ();
```

LINQ From The Command Line

```
$ csharp
Mono C# Shell, type “help;” for help

Enter statements below.
csharp> using System.IO;
csharp> var last_week = DateTime.Now – TimeSpan.FromDays (7);
csharp> from f in Directory.GetFiles (“/etc”)
      >   let fi = new FileInfo (f)
      >   where fi.LastWriteTime < last_week
      >   select f;
{ “/etc/adjtime”, “/etc/asound.state”,
  “/etc/ld.so.cache”, “/etc/mtab”,
  “/etc/printcap”, “/etc/resolv.conf” }
csharp>
```

Interactive LINQ To XML

```
csharp> LoadLibrary (“System.Xml.Linq”);
csharp> using System.Xml.Linq;
csharp> var xml = new XElement("CompilerSources",
      >   from f in Directory.GetFiles ("/cvs/mcs/mcs")
      >   let fi = new FileInfo (f)
      >   orderby fi.Length
      >   select new XElement ("file",
      >     new XAttribute ("name", f),
      >     new XAttribute ("size", fi.Length)));
csharp> xml;
<CompilerSources>
  <file name="/cvs/mcs/mcs/mcs.exe.config" size="395" />
  <file name="/cvs/mcs/mcs/gmcs.exe.config" size="464" />
  <file name="/cvs/mcs/mcs/OPTIMIZE" size="498" />
  <file name="/cvs/mcs/mcs/lambda.todo" size="658" />
  [...]
</CompilerSources>
```

GUI Shell

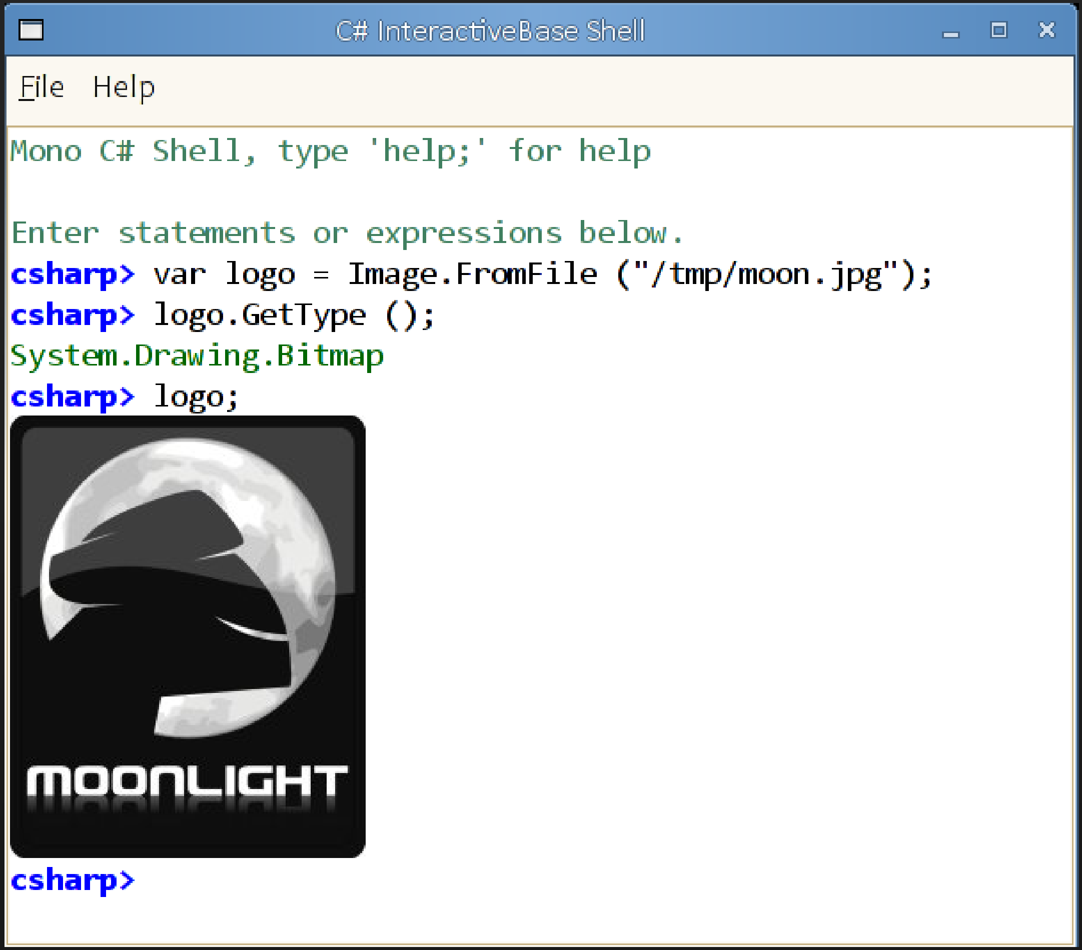
C# eval hosted in a GUI

- Replace base class, with GUI base class:

GUI Shell

C# eval hosted in a GUI

- Replace base class, with GUI base class:



```
Mono C# Shell, type 'help;' for help

Enter statements or expressions below.
csharp> var logo = Image.FromFile ("/tmp/moon.jpg");
csharp> logo.GetType ();
System.Drawing.Bitmap
csharp> logo;
```

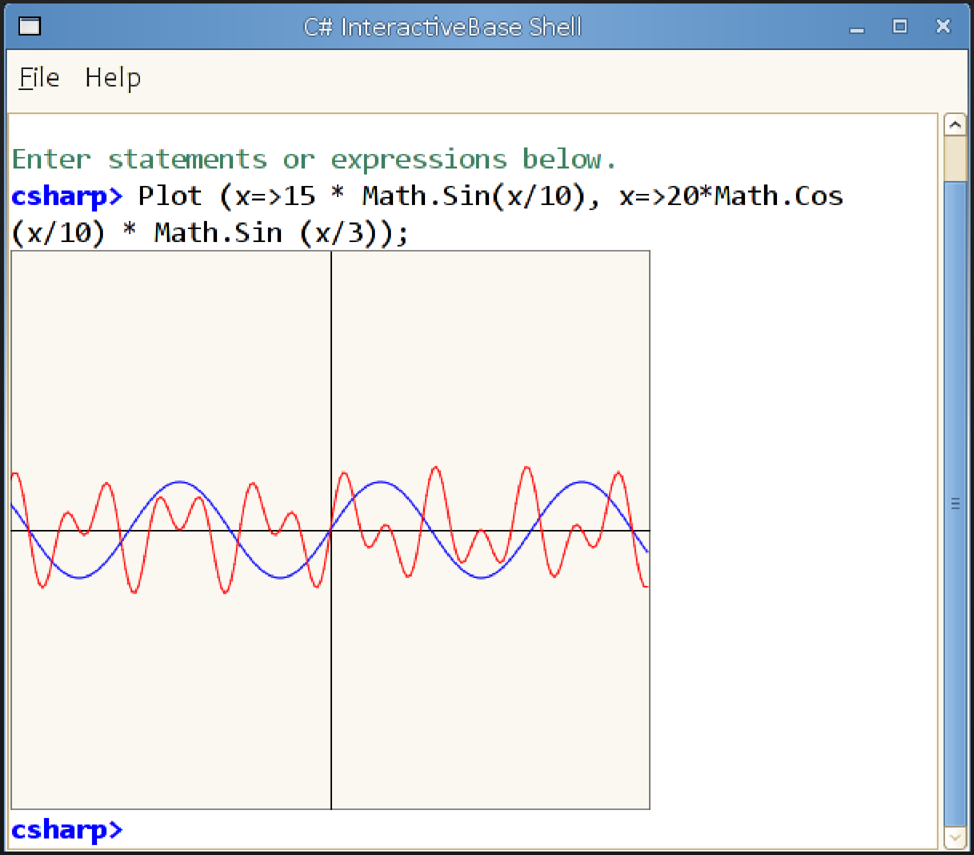
```
csharp>
```

GUI Shell, Quick Plot Method

- Plot (Func<double,double>);

GUI Shell, Quick Plot Method

- Plot (Func<double,double>);



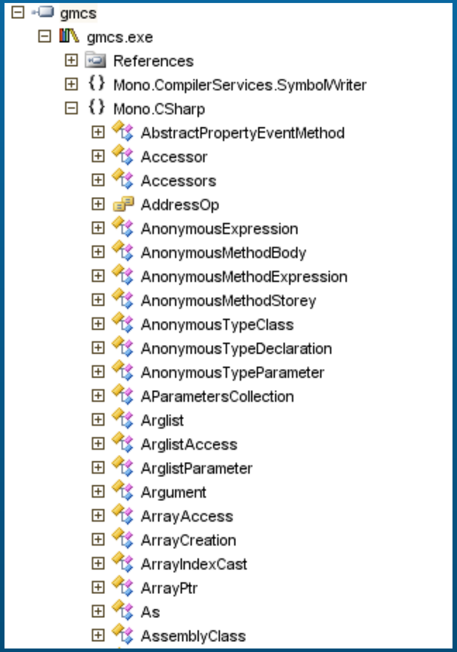
```
Enter statements or expressions below.
csharp> Plot (x=>15 * Math.Sin(x/10), x=>20*Math.Cos
(x/10) * Math.Sin (x/3));
csharp>
```

Reshaping The API

Turning the compiler into a library

Complete C# Compiler

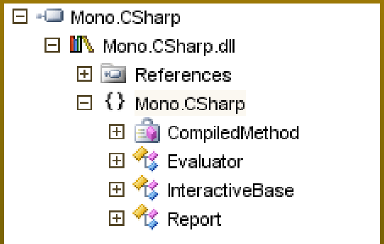
- Everything public



```
⊟ gmcs
  ⊟ gmcs.exe
    ⊞ References
    ⊞ {} Mono.CompilerServices.SymbolWriter
    ⊟ {} Mono.CSharp
      ⊞ AbstractPropertyEventMethod
      ⊞ Accessor
      ⊞ Accessors
      ⊞ AddressOp
      ⊞ AnonymousExpression
      ⊞ AnonymousMethodBody
      ⊞ AnonymousMethodExpression
      ⊞ AnonymousMethodStorey
      ⊞ AnonymousTypeClass
      ⊞ AnonymousTypeDeclaration
      ⊞ AnonymousTypeParameter
      ⊞ AParametersCollection
      ⊞ Arglist
      ⊞ ArglistAccess
      ⊞ ArglistParameter
      ⊞ Argument
      ⊞ ArrayAccess
      ⊞ ArrayCreation
      ⊞ ArrayIndexCast
      ⊞ ArrayPtr
      ⊞ As
      ⊞ AssemblyClass
```

Mono.CSharp

- Minimal API



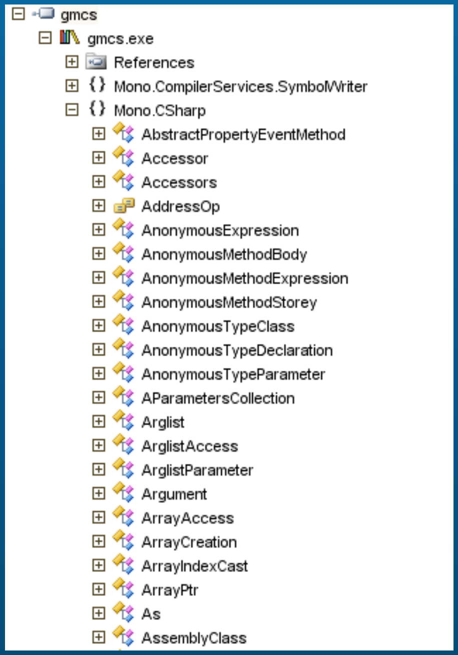
```
⊟ Mono.CSharp
  ⊟ Mono.CSharp.dll
    ⊞ References
    ⊟ {} Mono.CSharp
      ⊞ CompiledMethod
      ⊞ Evaluator
      ⊞ InteractiveBase
      ⊞ Report
```

Reshaping The API

Avoid manual work, reusing Mono.Cecil and Mono.Linker

Complete C# Compiler

- Everything public



```
gmcs
  gmcs.exe
    References
    Mono.CompilerServices.SymbolWriter
    Mono.CSharp
      AbstractPropertyEventMethod
      Accessor
      Accessors
      AddressOp
      AnonymousExpression
      AnonymousMethodBody
      AnonymousMethodExpression
      AnonymousMethodStorey
      AnonymousTypeClass
      AnonymousTypeDeclaration
      AnonymousTypeParameter
      AParametersCollection
      Arglist
      ArglistAccess
      ArglistParameter
      Argument
      ArrayAccess
      ArrayCreation
      ArrayIndexCast
      ArrayPtr
      As
      AssemblyClass
```

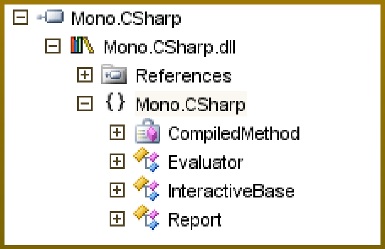
Mono Linker

- Uses Mono.Cecil
- Reshapes code
- Desired API
- Removes or hides

link.xml

Mono.CSharp

- Minimal API



```
Mono.CSharp
  Mono.CSharp.dll
    References
    Mono.CSharp
      CompiledMethod
      Evaluator
      InteractiveBase
      Report
```

Mono Linker Use Cases

- Shrinking Assemblies
  - Shipping only what is required
  - Simplify deployment
- Create your own Compact Framework
  - What you need from the superset
- .NET 3.5 to Silverlight
  - We reshape our assemblies.
  - Minimal hand-editing/tuning.

Beyond the CLR: Innovating on a Solid Foundation

Virtual machines are fascinating

- Great innovations are possible
  - Build on an existing large ecosystem
  - Instrument, expand, innovate
  - Special code generation

Beyond the CLR: Innovating on a Solid Foundation

Virtual machines are fascinating

- Great innovations are possible
  - Build on an existing large ecosystem
  - Instrument, expand, innovate
  - Special code generation
- VM potential limited by vendor realities
  - Provider scarcity
  - Shipping dates
  - Staffing
  - Product Management
  - Feature prioritization

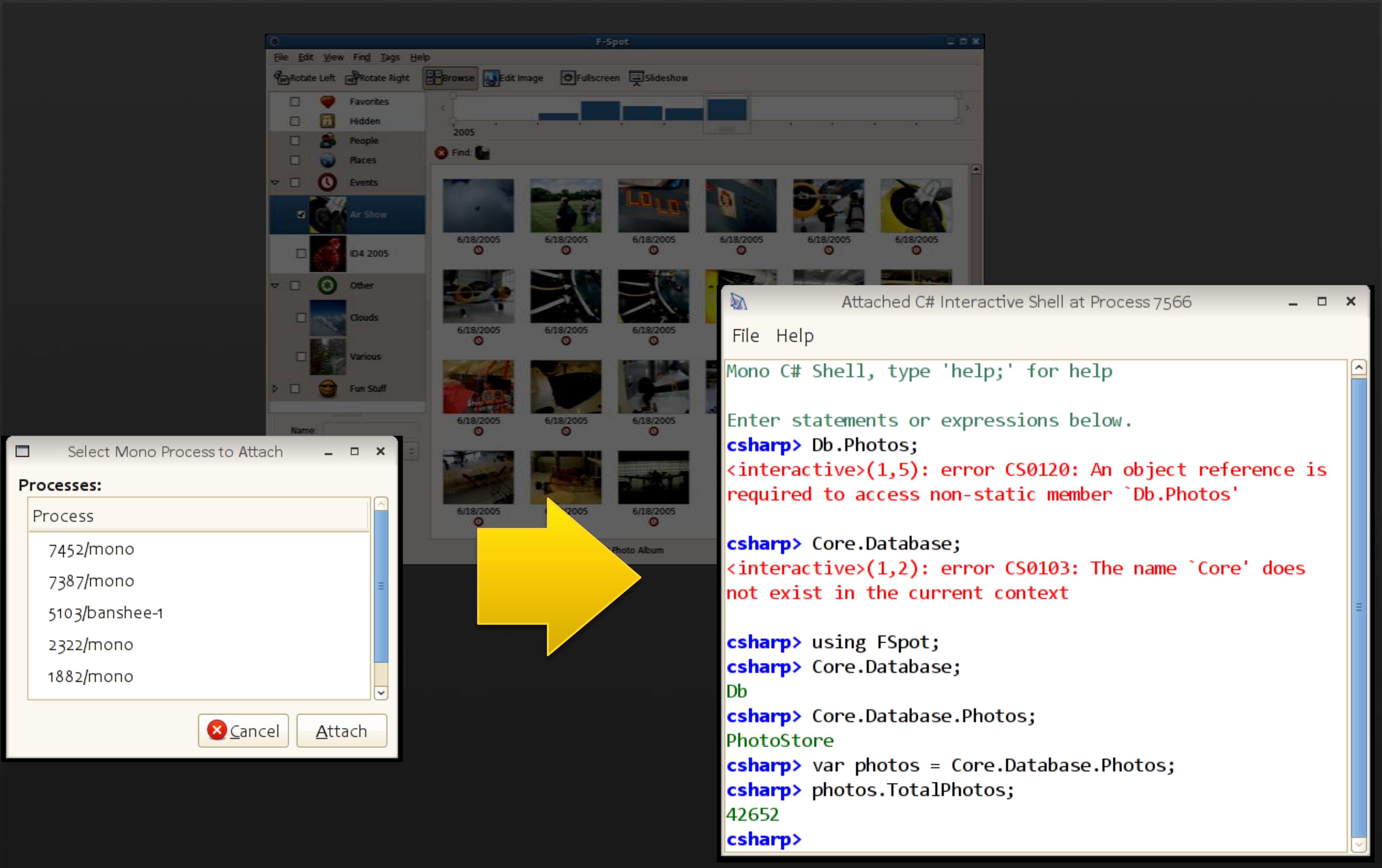
Injecting Code Into A Live Process

The `Mono.Attach.VirtualMachine` API

- On the root `AppDomain`, on a new thread

Injecting GUI Interactive C#

Consoles for everyone!



Select Mono Process to Attach

Processes:

Process
7452/mono
7387/mono
5103/banshee-1
2322/mono
1882/mono

Attached C# Interactive Shell at Process 7566

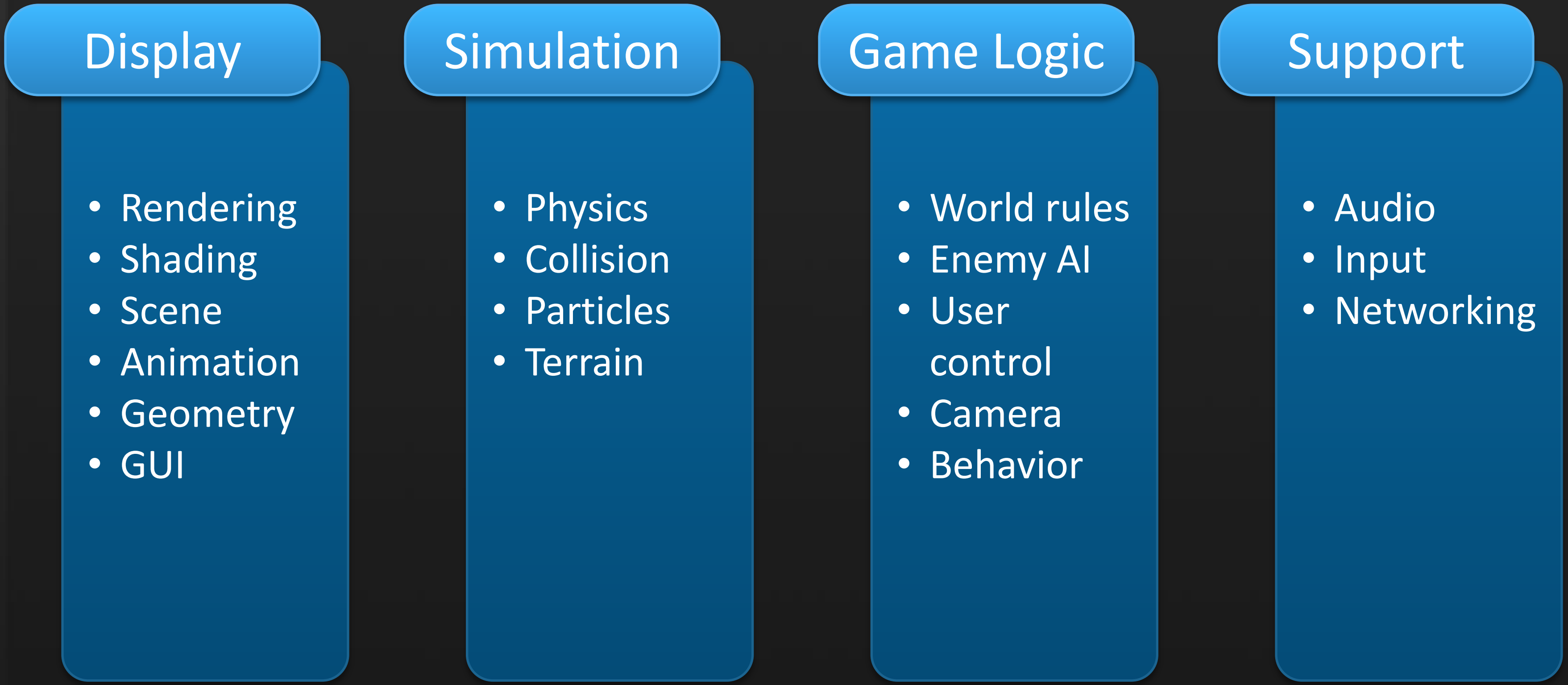
```
Mono C# Shell, type 'help;' for help

Enter statements or expressions below.
csharp> Db.Photos;
<interactive>(1,5): error CS0120: An object reference is
required to access non-static member `Db.Photos'

csharp> Core.Database;
<interactive>(1,2): error CS0103: The name `Core' does
not exist in the current context

csharp> using FSpot;
csharp> Core.Database;
Db
csharp> Core.Database.Photos;
PhotoStore
csharp> var photos = Core.Database.Photos;
csharp> photos.TotalPhotos;
42652
csharp>
```

PDC2008
PROFESSIONAL DEVELOPERS CONFERENCE

Turbocharging Games

Fast, Productive, Safe.
Pick all three.

Game Software Components



Display

- Rendering
- Shading
- Scene
- Animation
- Geometry
- GUI

Simulation

- Physics
- Collision
- Particles
- Terrain

Game Logic

- World rules
- Enemy AI
- User control
- Camera
- Behavior

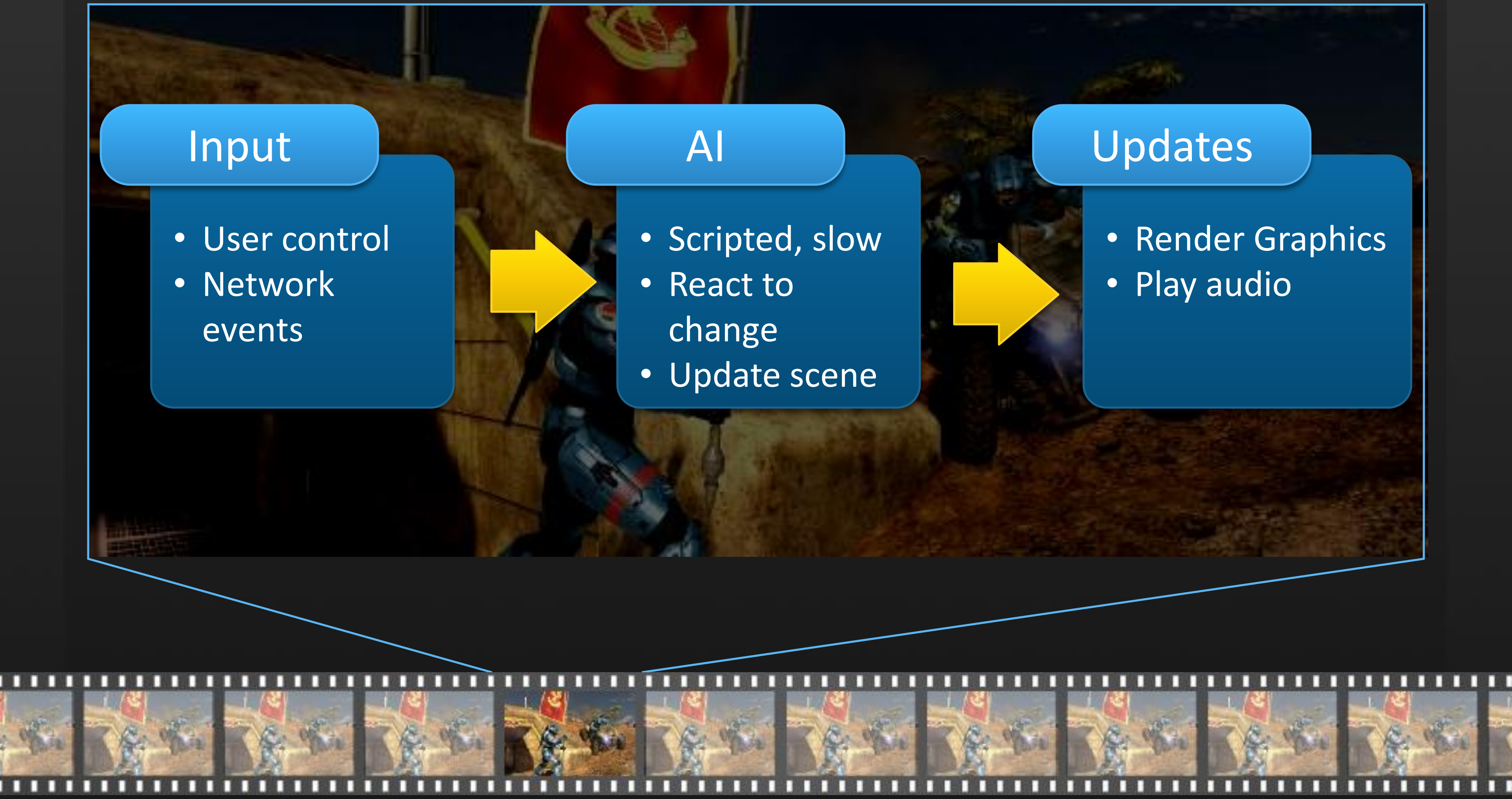
Support

- Audio
- Input
- Networking

The Problem

Games are real-time programs

- 30 to 60 frames per second (0.016 seconds)



Input

- User control
- Network events

AI

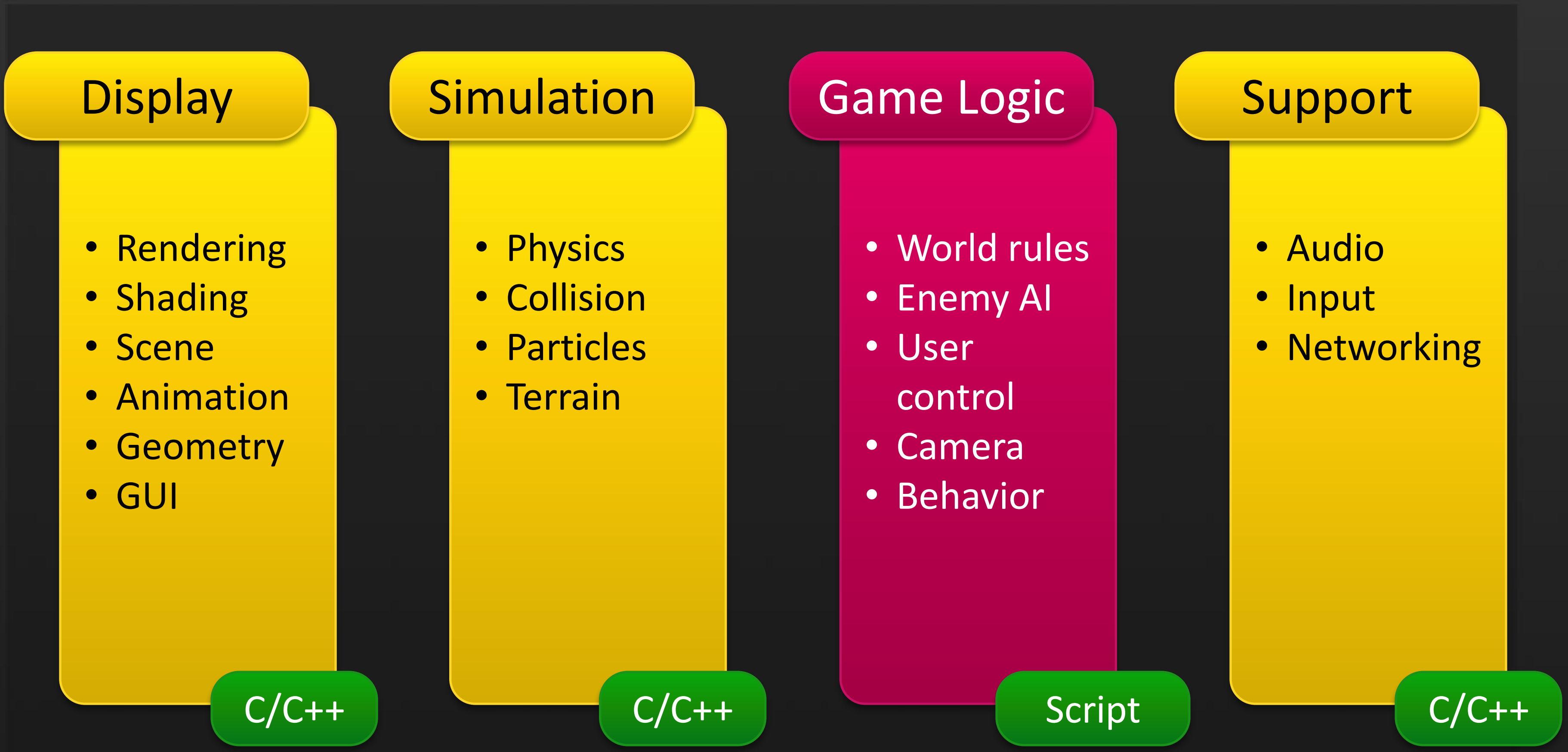
- Scripted, slow
- React to change
- Update scene

Updates

- Render Graphics
- Play audio

Problem: Scripting Is A Bottleneck

Gaming's Achilles' Heel



Display

- Rendering
- Shading
- Scene
- Animation
- Geometry
- GUI

C/C++

Simulation

- Physics
- Collision
- Particles
- Terrain

C/C++

Game Logic

- World rules
- Enemy AI
- User control
- Camera
- Behavior

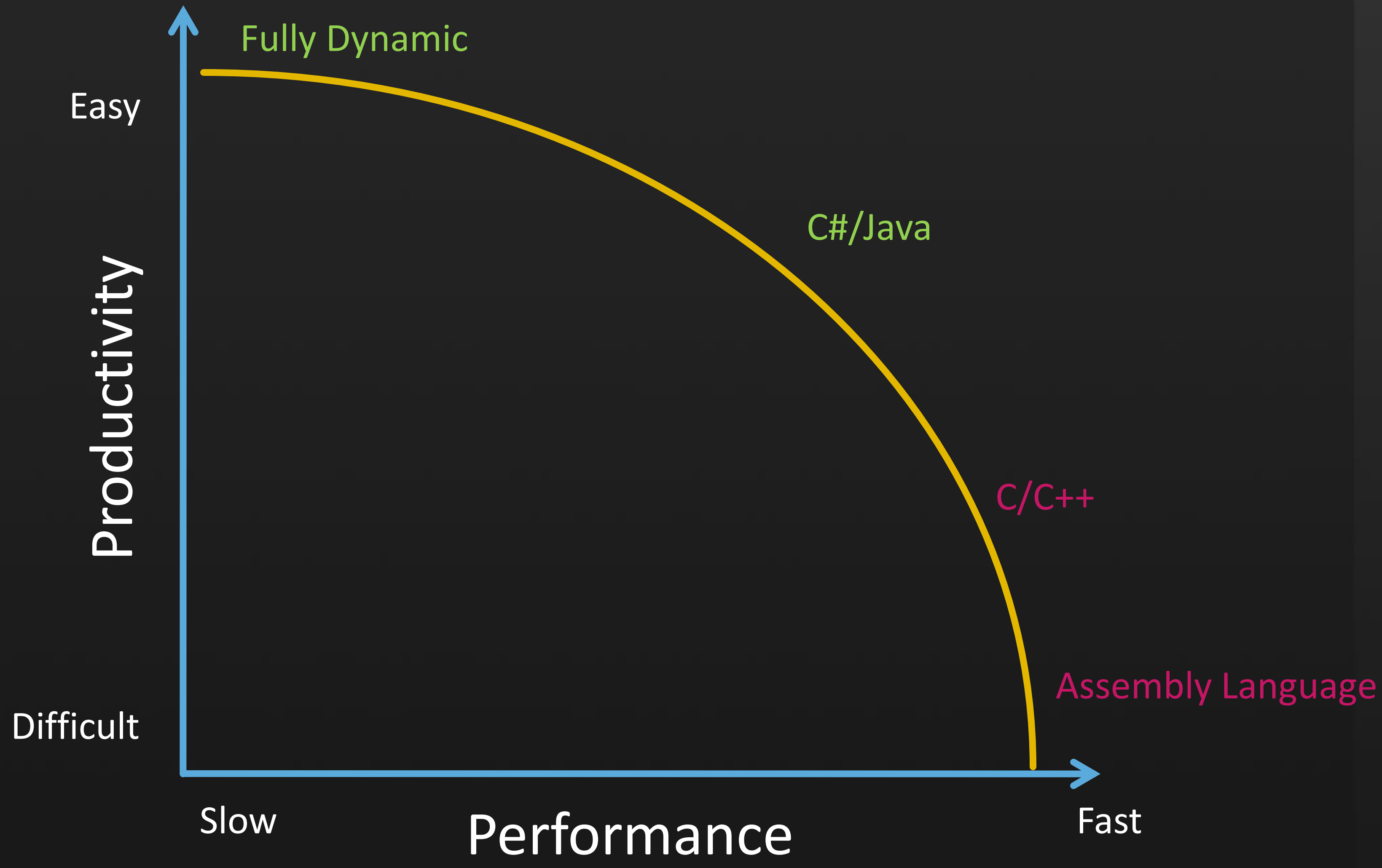
Script

Support

- Audio
- Input
- Networking

C/C++

Language Choices



Mono in Gaming Today

Moving from scripting to static/compiled

- Mono’s CLR is ideal for embedding

Mono in Gaming Today

Moving from scripting to static/compiled

- Mono’s CLR is ideal for embedding
- Some examples
  - SecondLife: Switched from LSL to Mono
    - 50x to 300x performance increase

Mono in Gaming Today

Moving from scripting to static/compiled

- Mono’s CLR is ideal for embedding
- Some examples
  - SecondLife: Switched from LSL to Mono
    - 50x to 300x performance increase
  - Unity3D: Powers Cartoon Network’s FusionFall
    - Uses C#, UnityScript and Boo
    - UnityScript is a strongly typed Javascript

Demo

Managed Code In Gaming

Improving developer productivity while maintaining program speed

	Traditional
Game AI	Scripted Slow/easy
Game Engines	Compiled Fast/Hard
Graphics Engine	Compiled Fast/Hard

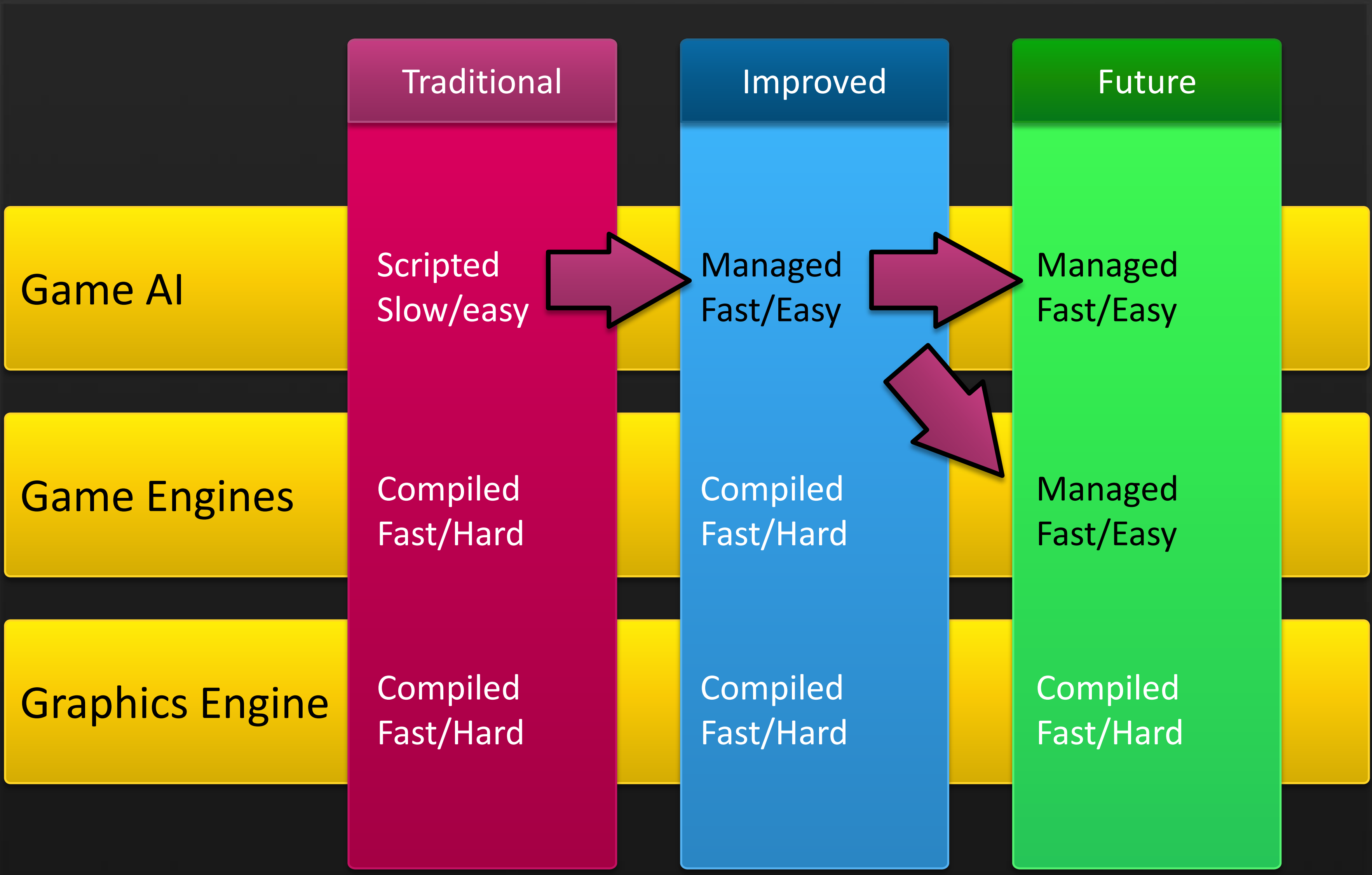
Managed Code In Gaming

Improving developer productivity while maintaining program speed

	Traditional	Improved
Game AI	Scripted Slow/easy	Managed Fast/Easy
Game Engines	Compiled Fast/Hard	Compiled Fast/Hard
Graphics Engine	Compiled Fast/Hard	Compiled Fast/Hard

Managed Code In Gaming

Improving developer productivity while maintaining program speed



	Traditional	Improved	Future
Game AI	Scripted Slow/easy	Managed Fast/Easy	Managed Fast/Easy
Game Engines	Compiled Fast/Hard	Compiled Fast/Hard	Managed Fast/Easy
Graphics Engine	Compiled Fast/Hard	Compiled Fast/Hard	Compiled Fast/Hard

3D Floating Point Vector Operations

At the core of gaming engines

- Exploring an innocent looking loop in C#:

```
UpdatePos (Vector3f [] points, ref Vector3f delta)
{
    for (int i = 0; i < points.Length; i++)
        points [i] += delta;
}
```

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} = \begin{bmatrix} x_i+\delta_x \\ y_i+\delta_y \\ z_i+\delta_z \end{bmatrix}$$

```
Vector3f static operator + (Vector3f a, Vector3f b)
{
    return new Vector3f (a.x+b.x, a.y+b.y, a.z+b.z);
}
```

UpdatePos Method In IL

The code that does the addition

```
.method private static  hidebysig
           default void UpdatePos (valuetype [Mono.Simd]Mono.Simd.Vector4f[] points, valuetype
[Mono.Simd]Mono.Simd.Vector4f& delta)  cil managed
    {
        // Method begins at RVA 0x2144
             // Code size 50 (0x32)
             .maxstack 4
             .locals init (int32        V_0)
             IL_0000:  ldc.i4.0
             IL_0001:  stloc.0
             IL_0002:  br IL_0028

             IL_0007:  ldarg.0
             IL_0008:  ldloc.0
             IL_0009:  ldelema [Mono.Simd]Mono.Simd.Vector4f
             IL_000e:  dup
             IL_000f:  ldobj [Mono.Simd]Mono.Simd.Vector4f
             IL_0014:  ldarg.1
             IL_0015:  ldobj [Mono.Simd]Mono.Simd.Vector4f
             IL_001a:  call valuetype [Mono.Simd]Mono.Simd.Vector4f valuetype
                          [Mono.Simd]Mono.Simd.Vector4f::op_Addition(valuetype [Mono.Simd]Mono.Simd.Vector4f,
                          valuetype [Mono.Simd]Mono.Simd.Vector4f)
             IL_001f:  stobj [Mono.Simd]Mono.Simd.Vector4f
             IL_0024:  ldloc.0
             IL_0025:  ldc.i4.1
             IL_0026:  add
             IL_0027:  stloc.0
             IL_0028:  ldloc.0
             IL_0029:  ldarg.0
             IL_002a:  ldlen
             IL_002b:  conv.i4
             IL_002c:  blt IL_0007

             IL_0031:  ret
    } // end of method X::UpdatePos
```

Vector4f.op_Addition in IL

The IL implementation

```
 // method line 24
    .method public static  hidebysig  specialname
           default valuetype Mono.Simd.Vector4f op_Addition (valuetype Mono.Simd.Vector4f v1, valuetype
Mono.Simd.Vector4f v2)  cil managed
    {
        // Method begins at RVA 0x24ac
        // Code size 69 (0x45)
        .maxstack 7
        .locals init (
                valuetype Mono.Simd.Vector4f    V_0)
        IL_0000:  ldloca.s 0
        IL_0002:  ldarga.s 0
        IL_0004:  ldfld float32 Mono.Simd.Vector4f::x
        IL_0009:  ldarga.s 1
        IL_000b:  ldfld float32 Mono.Simd.Vector4f::x
        IL_0010:  add
        IL_0011:  ldarga.s 0
        IL_0013:  ldfld float32 Mono.Simd.Vector4f::y
        IL_0018:  ldarga.s 1
        IL_001a:  ldfld float32 Mono.Simd.Vector4f::y
        IL_001f:  add
        IL_0020:  ldarga.s 0
        IL_0022:  ldfld float32 Mono.Simd.Vector4f::z
        IL_0027:  ldarga.s 1
        IL_0029:  ldfld float32 Mono.Simd.Vector4f::z
        IL_002e:  add
        IL_002f:  ldarga.s 0
        IL_0031:  ldfld float32 Mono.Simd.Vector4f::w
        IL_0036:  ldarga.s 1
        IL_0038:  ldfld float32 Mono.Simd.Vector4f::w
        IL_003d:  add
        IL_003e:  call instance void valuetype Mono.Simd.Vector4f::'.ctor'(float32, float32, float32, float32)
        IL_0043:  ldloc.0
        IL_0044:  ret
    } // end of method Vector4f::op_Addition
```

UpdatePos in x86 code

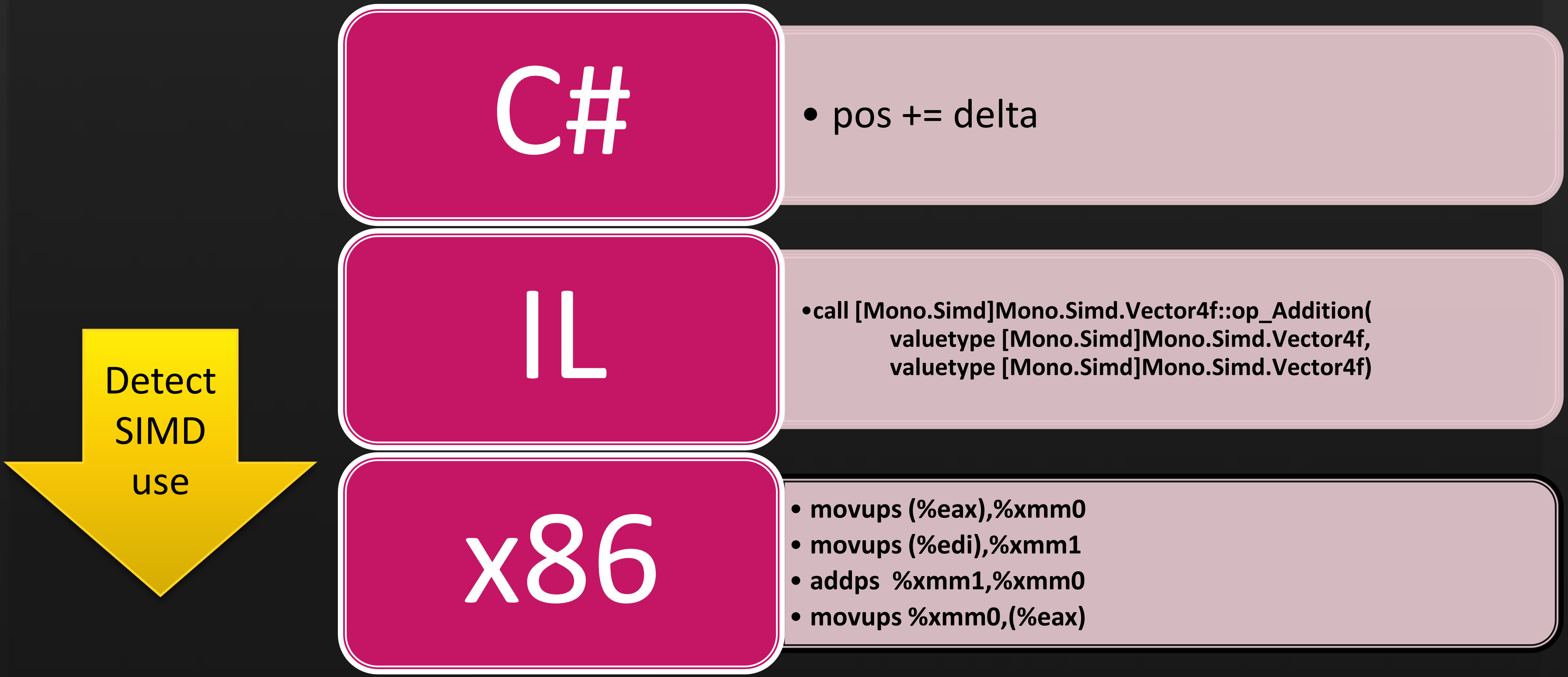
Generated asssembly code

```
00000000 <X_UpdatePos>:
   0:   55                      push   %ebp
   1:   8b ec                   mov    %esp,%ebp
   3:   53                      push   %ebx
   4:   57                      push   %edi
   5:   56                      push   %esi
   6:   83 ec 38                sub    $0x38,%esp
   9:   8b 75 08                mov    0x8(%ebp),%esi
   c:   8b 7d 0c                mov    0xc(%ebp),%edi
   f:   33 db                   xor    %ebx,%ebx
  11:   e9 ad 00 00 00          jmp    c3 <X_UpdatePos+0xc3>
  16:   8b c0                   mov    %eax,%eax
  18:   39 5e 0c                cmp    %ebx,0xc(%esi)
  1b:   0f 86 b5 00 00 00       jbe    d6 <X_UpdatePos+0xd6>
  21:   8b cb                   mov    %ebx,%ecx
  23:   c1 e1 04                shl    $0x4,%ecx
  26:   8b c6                   mov    %esi,%eax
  28:   03 c1                   add    %ecx,%eax
  2a:   05 10 00 00 00          add    $0x10,%eax
  2f:   89 45 bc                mov    %eax,-0x44(%ebp)
  32:   8b 08                   mov    (%eax),%ecx
  34:   89 4d c4                mov    %ecx,-0x3c(%ebp)
  37:   8b 48 04                mov    0x4(%eax),%ecx
  3a:   89 4d c8                mov    %ecx,-0x38(%ebp)
  3d:   8b 48 08                mov    0x8(%eax),%ecx
  40:   89 4d cc                mov    %ecx,-0x34(%ebp)
  43:   8b 40 0c                mov    0xc(%eax),%eax
  46:   89 45 d0                mov    %eax,-0x30(%ebp)
  49:   8b 07                   mov    (%edi),%eax
  4b:   89 45 d4                mov    %eax,-0x2c(%ebp)
  4e:   8b 47 04                mov    0x4(%edi),%eax
  51:   89 45 d8                mov    %eax,-0x28(%ebp)
  54:   8b 47 08                mov    0x8(%edi),%eax
  57:   89 45 dc                mov    %eax,-0x24(%ebp)
  5a:   8b 47 0c                mov    0xc(%edi),%eax
  5d:   89 45 e0                mov    %eax,-0x20(%ebp)
  60:   8d 45 e4                lea    -0x1c(%ebp),%eax
  63:   83 ec 10                sub    $0x10,%esp
  66:   8b 4d d4                mov    -0x2c(%ebp),%ecx
  69:   89 0c 24                mov    %ecx,(%esp)
  6c:   8b 4d d8                mov    -0x28(%ebp),%ecx
  6f:   89 4c 24 04             mov    %ecx,0x4(%esp)
  73:   8b 4d dc                mov    -0x24(%ebp),%ecx
  76:   89 4c 24 08             mov    %ecx,0x8(%esp)
  7a:   8b 4d e0                mov    -0x20(%ebp),%ecx
  7d:   89 4c 24 0c             mov    %ecx,0xc(%esp)
  81:   83 ec 10                sub    $0x10,%esp
  84:   8b 4d c4                mov    -0x3c(%ebp),%ecx
  87:   89 0c 24                mov    %ecx,(%esp)
  8a:   8b 4d c8                mov    -0x38(%ebp),%ecx
  8d:   89 4c 24 04             mov    %ecx,0x4(%esp)
  91:   8b 4d cc                mov    -0x34(%ebp),%ecx
  94:   89 4c 24 08             mov    %ecx,0x8(%esp)
  98:   8b 4d d0                mov    -0x30(%ebp),%ecx
  9b:   89 4c 24 0c             mov    %ecx,0xc(%esp)
  9f:   50                      push   %eax
  a0:   e8 43 00 00 00          call   op_Addition
  a5:   83 c4 20                add    $0x20,%esp
  a8:   8b 45 bc                mov    -0x44(%ebp),%eax
  ab:   8b 4d e4                mov    -0x1c(%ebp),%ecx
  ae:   89 08                   mov    %ecx,(%eax)
  b0:   8b 4d e8                mov    -0x18(%ebp),%ecx
  b3:   89 48 04                mov    %ecx,0x4(%eax)
  b6:   8b 4d ec                mov    -0x14(%ebp),%ecx
  b9:   89 48 08                mov    %ecx,0x8(%eax)
  bc:   8b 4d f0                mov    -0x10(%ebp),%ecx
  bf:   89 48 0c                mov    %ecx,0xc(%eax)
  c2:   43                      inc    %ebx
  c3:   8b 46 0c                mov    0xc(%esi),%eax
  c6:   3b d8                   cmp    %eax,%ebx
  c8:   0f 8c 4a ff ff ff       jl     18 <X_UpdatePos+0x18>
  ce:   8d 65 f4                lea    -0xc(%ebp),%esp
  d1:   5e                      pop    %esi
  d2:   5f                      pop    %edi
  d3:   5b                      pop    %ebx
  d4:   c9                      leave
  d5:   c3                      ret
```

Mono.SIMD: Mapping To Native Instructions

SIMD aware runtime

- Object-oriented APIs for Vector processing
  - Vector4f, Vector4i, Vector2d, Vector16b, etc
  - Mapped to hardware operations



C#

- pos += delta

IL

```
•call [Mono.Simd]Mono.Simd.Vector4f::op_Addition(
          valuetype [Mono.Simd]Mono.Simd.Vector4f,
          valuetype [Mono.Simd]Mono.Simd.Vector4f)
```

x86

```
• movups (%eax),%xmm0
• movups (%edi),%xmm1
• addps  %xmm1,%xmm0
• movups %xmm0,(%eax)
```

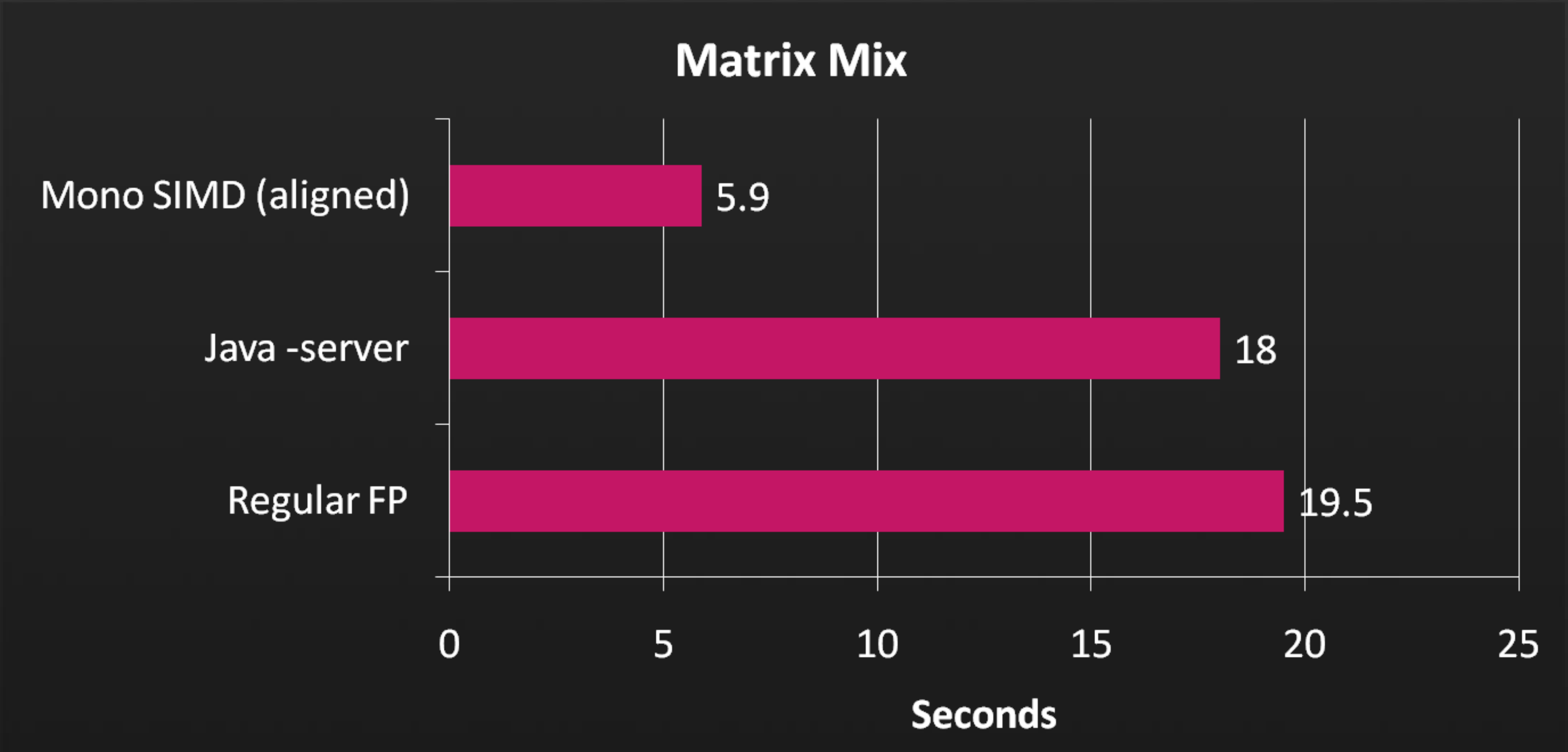
Detect SIMD use

UpdatePos With Mono’s SIMD

```
00000000 <X_UpdatePos>:
   0:   55                      push   %ebp
   1:   8b ec                   mov    %esp,%ebp
   3:   53                      push   %ebx
   4:   57                      push   %edi
   5:   56                      push   %esi
   6:   83 ec 04                sub    $0x4,%esp
   9:   8b 75 08                mov    0x8(%ebp),%esi
   c:   8b 7d 0c                mov    0xc(%ebp),%edi
   f:   33 db                   xor    %ebx,%ebx
  11:   eb 29                   jmp    3c <X_UpdatePos+0x3c>
  13:   8d 64 24 00             lea    0x0(%esp),%esp
  17:   90                      nop
  18:   39 5e 0c                cmp    %ebx,0xc(%esi)
  1b:   0f 86 2a 00 00 00       jbe    4b <X_UpdatePos+0x4b>
  21:   8b cb                   mov    %ebx,%ecx
  23:   c1 e1 04                shl    $0x4,%ecx
  26:   8b c6                   mov    %esi,%eax
  28:   03 c1                   add    %ecx,%eax
  2a:   05 10 00 00 00          add    $0x10,%eax
  2f:   0f 10 00                movups (%eax),%xmm0
  32:   0f 10 0f                movups (%edi),%xmm1
  35:   0f 58 c1                addps  %xmm1,%xmm0
  38:   0f 11 00                movups %xmm0,(%eax)
  3b:   43                      inc    %ebx
  3c:   8b 46 0c                mov    0xc(%esi),%eax
  3f:   3b d8                   cmp    %eax,%ebx
  41:   7c d5                   jl     18 <X_UpdatePos+0x18>
  43:   8d 65 f4                lea    -0xc(%ebp),%esp
  46:   5e                      pop    %esi
  47:   5f                      pop    %edi
  48:   5b                      pop    %ebx
  49:   c9                      leave
  4a:   c3                      ret
```

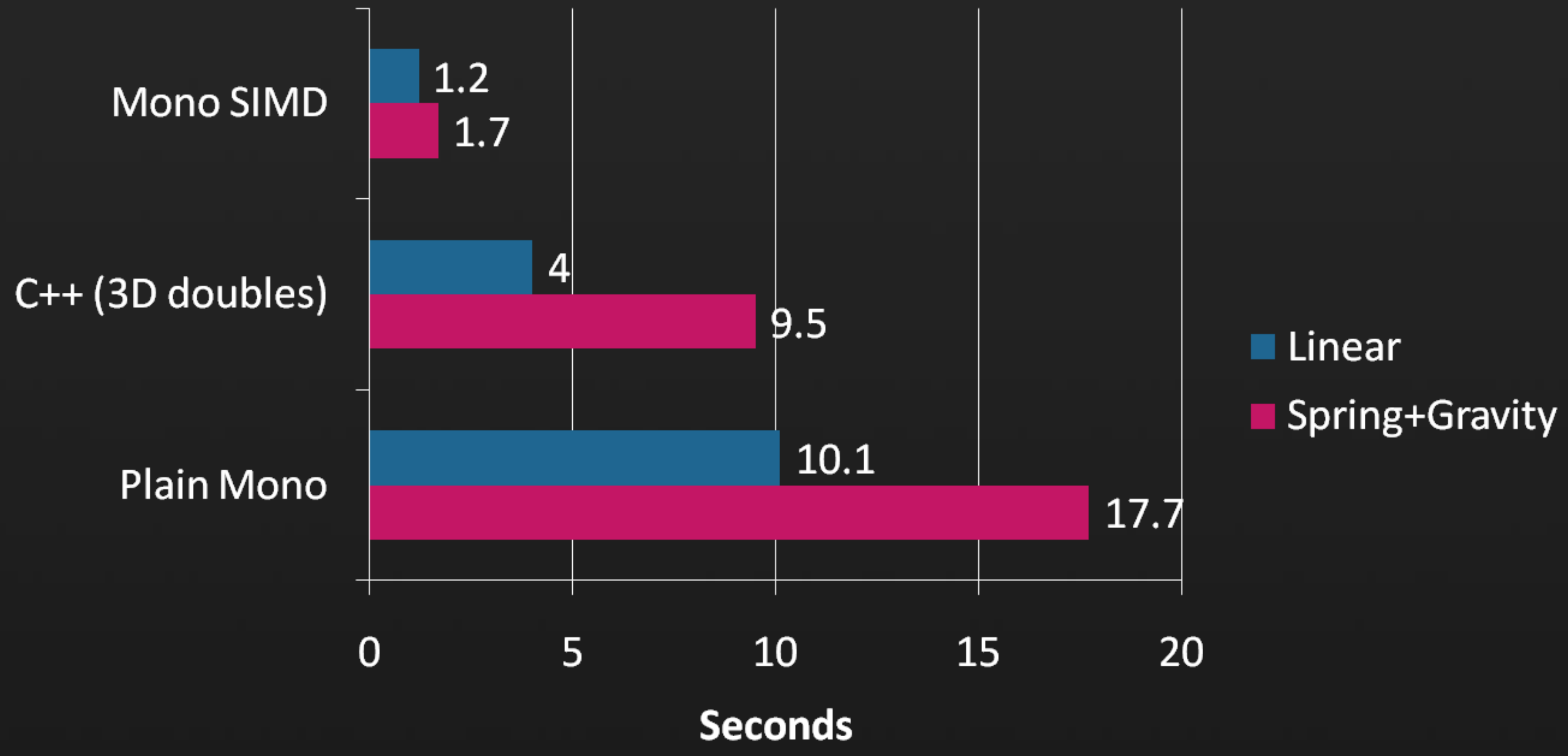
SIMD Operations Mix

Developer created tests



Mono.SIMD: Speedups

Physics simulations, no optimizations



Based on the C++ simulation code at
sharp-gamedev.blogspot.com/2008/09/updated-c-version.html

Ahead Of Time Compilation

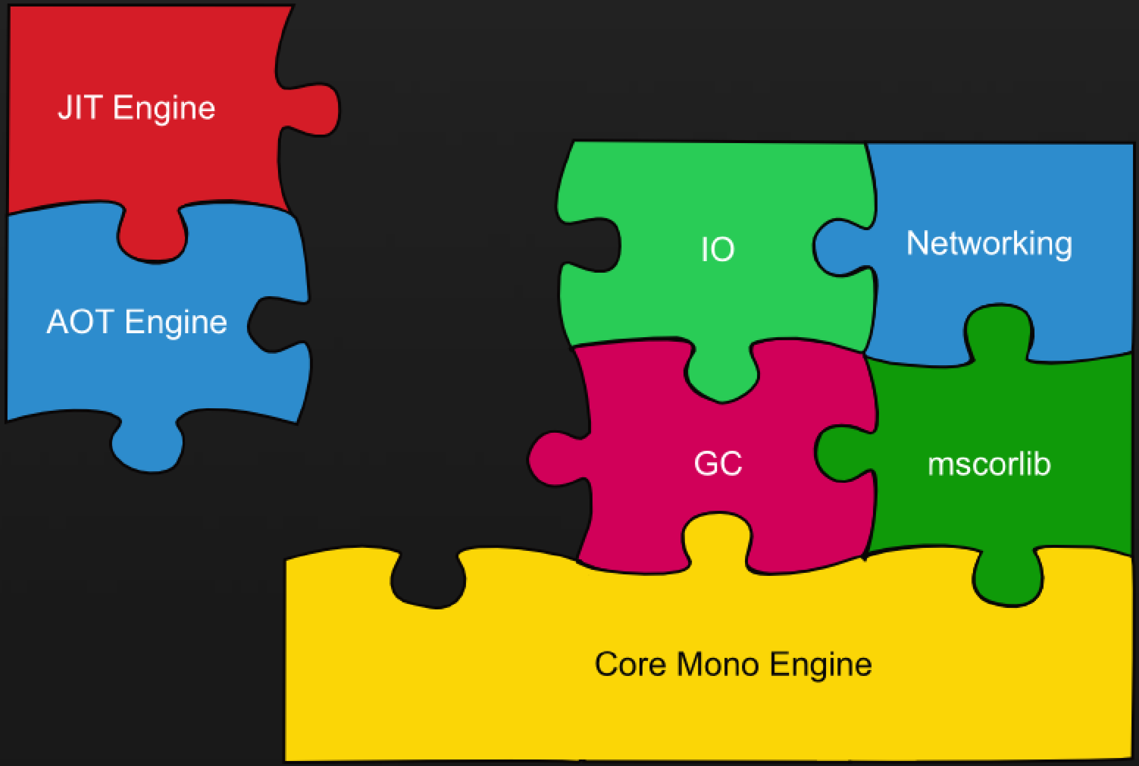
Batch compilation of CIL/.NET code

- Ahead of Time compilation (AOT):
  - “ngen” in the .NET world
  - Precompiled IL code to native code
- Visible effects
  - Saves on startup time
  - Decreases footprint across multiple processes
  - Produces slower code
- Not complete
  - Can handle most of the JIT generated code
  - A few bits are not AOTed

Full Ahead Of Time Compilation

Entirely static compilation of CIL/.NET code

- Some devices disable on the fly codegen:
  - iPhone OS 2.x, XBox360
- Full AOT: Does AOT for the missing bits



Demo – Mono on iPhone.

Other Topics

Much more

- Mono Continuations.
  - Like Stackless-Python
  - Cooperative multi-threading
  - Avoids concurrency bugs
  - Concurrency achieved with processes
- Supercomputing Mono
  - 64 bit arrays

Learning More About Mono

- http://www.mono-project.com
- Getting Started
  - http://www.mono-project.com/Start
- Community blogs
  - http://www.go-mono.com/monologue
- Miguel’s blog
  - http://tirania.org/blog

PDC2008
PROFESSIONAL DEVELOPERS CONFERENCE

Microsoft®

Your potential. Our passion.™