

**What it actually is (Now):**

- o An extended SDK building on top of the Microsoft XNA game engine for the PC, Xbox & Zune.
- o Builds upon the Microsoft Robotics studio to support a wide variety of (as yet undefined) services and to exploit features of the robotics studio - this was not shown in the beta version I have explored.

**What it is planned to be:**

- o The easiest to use editing system for a game engine [release some time in 2010] ?!

**Good points:**

- o Builds on a reasonable engine [Microsoft XNA engine [http://en.wikipedia.org/wiki/Microsoft\\_XNA](http://en.wikipedia.org/wiki/Microsoft_XNA)]
- o Supports all of the widely used physics engines - the first engine I have seen to do this
- o SDK seems well written (as is the XNA engine) and comes with a wide range of good SDK Sample projects
- o Supports .FBX models
- o Supports a large range of post processing & lighting effects

**Bad points:**

- o As with the XNA engine I am not sure how scalable the engine will be nor how extensible the low level SDK is.
- o Building an SDK on top of another SDK is powerful but I fear that it may be constrictive & not scalable? Although it is likely to speed up the development process considerably :)

**Future:**

- o Interested to see what the editor looks like when it is released
- o I'd like to find out more about how the Microsoft CCR and DSS toolkits are exploited in the product to manage concurrency.
- o When the toolkit is more stable it will be interesting to try this out on a some large geometry.

**Conclusions:**

The tool claims to be "The easiest tool to create real-time 3d software", this beta release is a first step at this. It provides a good SDK for a rendering engine with physics. This provides some novel features such as the support for multiple physics engines in the same application and a nice Day/Night lighting system. The SDK looks like a good stepping stone to an intuitive user interface. Which is due in a future release. The engine is meant to support robotics and simulations however the beta currently gives limited intuition on how will it fulfils this role. Though the video above looks pretty interesting.

In summary this is an interesting product since it leverages and extends a number technically competent products from Microsoft namely the .Net framework in C#, Microsoft's Robotics Studio and the XNA Games Studio. This provides a good starting point and it will be interesting to see how this product develops.

**Mapped Points @** <http://www.doc.ic.ac.uk/~db805/GameEngines/>

```
<Cost value="500">Unknown</Cost>
<Features value="40">below average</Features>
<Usability value="30">No editor but good SDK</Usability>
<Rendering_Capacity value="200">Unknown</Rendering_Capacity>
<Fidelity value="60">easy to use lighting effects</Fidelity>
<Cross_Platform value="1">Windows only?</Cross_Platform>
<Web_Player value="25">False</Web_Player>
<Customizability value="80">C# scripting</Customizability>
<Source value="20">unknown</Source>
<Scalability value="20">Unknown likely low</Scalability>
```

<b>Real-Time 3D</b> <ul style="list-style-type: none"> <li>3D software design</li> <li>Ease of use</li> <li>Performances</li> </ul>	<b>Physics</b> <ul style="list-style-type: none"> <li>Rigid bodies</li> <li>Joints, materials...</li> <li>Forces, torques...</li> </ul>	<b>Advanced Graphics</b> <ul style="list-style-type: none"> <li>Latest 3D techniques</li> <li>Built-in shaders</li> <li>Post processing effects</li> </ul>	<b>Multi Physics Engine</b> <ul style="list-style-type: none"> <li>PhysX, Newton, ODE...</li> <li>Flexibility</li> <li>More functionalities</li> </ul>	<b>Software Interactions</b> <ul style="list-style-type: none"> <li>Service architecture</li> <li>Remote control</li> <li>Software compatibility</li> </ul>
<b>Scene Management</b> <ul style="list-style-type: none"> <li>Scripting</li> <li>Optimization</li> <li>XML and .NET</li> </ul>	<b>3D Environments</b> <ul style="list-style-type: none"> <li>Wide &amp; realistic</li> <li>Ground and aerial</li> <li>Underwater</li> </ul>	<b>Audio &amp; Video</b> <ul style="list-style-type: none"> <li>3D sounds</li> <li>Integrated videos</li> <li>Interactivity</li> </ul>	<b>Robotics Elements</b> <ul style="list-style-type: none"> <li>Sensors &amp; Actuators</li> <li>MSRDS compatible</li> <li>Robotics samples</li> </ul>	<b>3D Editors</b> <ul style="list-style-type: none"> <li>Graphic tools</li> <li>Simplycity</li> <li>Efficiency</li> </ul>
<b>Accessible Resources</b> <ul style="list-style-type: none"> <li>3D object libraries</li> <li>Full environments</li> <li>3D model import</li> </ul>	<b>Learning Materials</b> <ul style="list-style-type: none"> <li>User guides</li> <li>Documentation</li> <li>PDF &amp; Video tutorials</li> </ul>	<b>Evolve Platform</b> <ul style="list-style-type: none"> <li>One-year license</li> <li>Free automatic updates</li> <li>New features regularly</li> </ul>		

**Easy and rapid programming: Microsoft .NET**

As we explain in our [introduction post](#) on our product, one of our goals is to offer you a product that is easy to use, to enable any developer to benefit from real-time 3D simulation technologies. As developer ourselves, we already had an idea of what "ease of use" can mean for a developer, and that led us quite immediately to the choice of [Microsoft .NET](#).

As we wanted performance and quality 3D rendering we didn't had many option: standard C++ or .NET (java was clearly not an option). Then the choice was easy: .NET had two major advantages, the first was for our team: .NET meant that we could develop more rapidly and efficiently (and Visual Studio is an awesome IDE). The second and probably most important advantage is simply that it would be the same for our customers! Plus they can choose their programming language in the variety that the .NET framework offers (as for us, we use and recommend C#).

**Graphic engine: Microsoft XNA**

The choice of XNA as a graphic engine came quite naturally too, it's using direct X 9.0 guaranteeing performance and nice 3D rendering and it's on .NET. Plus we liked their asset managing utility. We've tested many other 3D engines in many programming languages before, and XNA is, for now, our choice.

**Editors interface: Microsoft WPF**

An important part of the "ease of use" (or "simply spirit" as we like to call it) that we wanted in our products meant that we had to design efficient user interfaces for our editors. That led us to [WPF](#), mainly because we liked the freedom we had to design our interface combined to the ease of use of the technology.

**Software interaction: Microsoft CCR/DSS**

We first discovered [Microsoft CCR/DSS](#) when we started to use [Microsoft Robotics Developer Studio](#), and we just love this technology. We had already played a bit with the concept of "service oriented" or "component oriented" architecture before and DSS is a really nice implementation of these concepts, while CCR offers a very smart way to solve concurrency and coordination problems. We'll discuss more about this in a dedicated post, but to sum up the main advantage of this technology for us it to enable our 3D simulation to be really easy to interface with any other software application.

**Physic engine: the multi-engine option**

As you'll have understood if you watched our [first demo of our UAV simulation](#), physic realism is really important in our product. So we've had a look at all the different physic engine available on the market (PhysX, Newton, ODE, Havok, and a few others) and after an in depth review, we decided not to choose. In fact these physic engines have not been created for the same purpose, and they have different strong points. Some have very good performance with a lot of objects, some are really focusing on the accuracy of the movements, others are designed for fast moving objects... and we felt that the choice of the physic engine really depended on the type of application. So we've built the SimplyDynamics, a software library that gives you the opportunity to choose the physic engine you're application is using. And as it might not be evident which one to choose, you can make this choice at the very last moment: at run time!

Pasted from <http://simplysim.wordpress.com/2009/12/03/an-inside-look-the-technology-used-in-our-3d-simulation/>