Breathing New Life into Accessibility

The MindGames group is an interdisciplinary team at MIT Media Lab Europe that integrates intelligent biofeedback, computer gaming and sensory immersion to develop techniques for positively affecting a participant's state of mind. For over a year, the group has developed an extensive signal processing framework and game engine that leverage the combined power of the C# language, the .NET Framework, and associated technologies like Managed DirectX.

Still Life: The Butterfly Soars

Still Life is one of the MindGames projects being developed in conjunction with physiotherapists at the Central Remedial Clinic in Dublin. The project uses a movement interface designed to creatively reward a participant for controlling their physical movements in a calm and relaxed way. It can be customized so that a patient is rewarded for practicing a movement over and over again and getting it correct, thereby turning previously monotonous exercises into an engaging interaction. The program is also able to track progress, so that a physiotherapist needn't always be present to monitor improvement during daily exercises.

Still Life leapt from the world of physiotherapy into the performing arts in early September as part of the *Féileacán project*, a performance that had its debut at the closing ceremonies of the 2003 conference for the Association for the Advancement of Assistive Technology in Europe (AAATE). (*Féileacán* is the Irish word for 'butterfly.')

Féileacán united the teenagers from four Special Schools in Ireland who both inspired and expressed the project's narrative. The Féileacán itself is a symbol for rebirth that provides the focus of the narrative: the children of the world, fed up with prejudice and poverty, build a spaceship to take them to a better world. The ship evolves into the Féileacán that guides them to a new space where they can breathe new life, encounter new friends, and overcome their fears.

The Féileacán story is told through a dance performance that stars the children themselves alongside Dublin's *Counterbalance* dance troupe. The performers are supported by a variety of innovative technologies, including Still Life, that facilitate new forms of expression. Still Life is presented on a wall-sized screen behind the performers, effectively turning that wall into a "magic mirror" in which the performers are reflected. The program tracks their movements and responds in a variety of ways – for example, by causing dancers to glow, orbs to radiate energy, and elements of the new world to be revealed as though they were being painted by the movement of the performers.

Still Life is just one of the projects the MindGames group has worked on as part of their contribution to the European Year of People with Disabilities. Other projects include *Mind Balance* – a game that uses EEG brain wave activity as a control interface – and *Biomelodics*, which uses musical biofeedback to help teach a participant to control their heart rate.

Top to bottom: Mind Balance, Still Life, Relax to Win on the PocketPC, and Still Life in rehearsal for the Féileacán Project.

MINDGAMES

Media Lab Europe
European Research Partner of MIT Media Lab

Mind Balance: The Control System You Have In Mind

Imagine a world where your brainwaves offer you another degree of freedom in a control system – and think how useful that freedom would be for someone who can't use a conventional controller like a mouse. We can't lift starfighters with Jedi Mind Tricks just yet, but the MindGames group is taking strides in that direction through the non-invasive real-time analysis of human brainwaves. *Mind Balance* was the first application developed as part of an ambitious collaboration with researchers at University College Dublin to implement new brain-computer control interfaces.

Taking the Mawg for a Walk

In *Mind Balance,* a participant must assist a tightrope-walking (apparently Scottish) behemoth known as the Mawg, by helping him keep his balance as he totters across a cosmic tightrope. All in a day's work for a typical computer gamer – but a participant at the helm of Mind Balance has no joystick, no mouse, and not even a camera – only a brain cap that non-invasively measures signals from the back of the head.

Specifically, the cap monitors electrical signals from the surface of the scalp over the occipital lobes (just above the neck), the home of the brain's visual processing, and sporting an effectively direct connection to the eyes via the brain's optical nerve. When the participant stares at regions on the screen that are blinking at known frequencies, their brain processes that blinking in enigmatically complex ways. But one side-effect of that processing – an increase in electrical activity at the same frequency as the blinking orb – is sufficiently pronounced that it can be detected in the electromagnetic soup at the surface of the head. These detectable artifacts are called Visually Evoked Potentials, or VEPs.

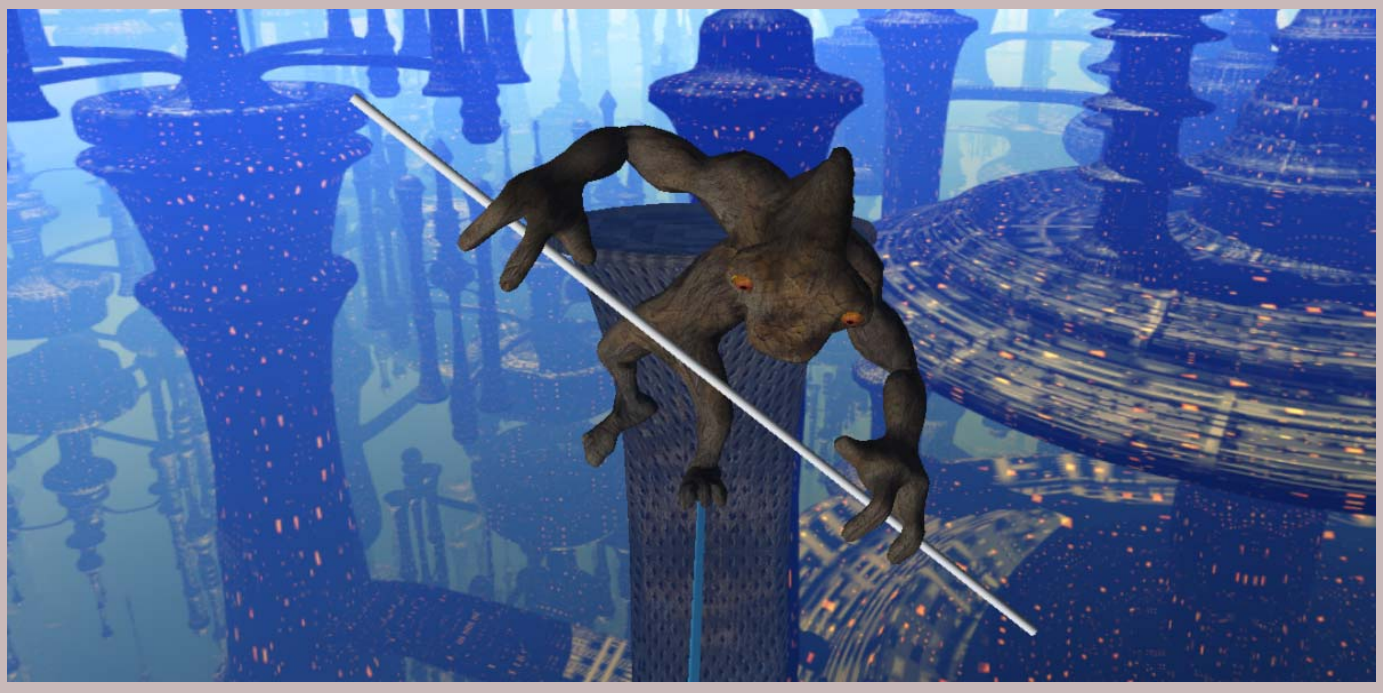
If the Mawg slips to the right, the participant can help shift the creature's balance back to the left by staring at the orb flickering on the left-hand side of the screen. The subsequent change in brainwave electrical activity is detected by the system as a VEP, and transformed into a one-dimensional analog control axis that can be used to get the Mawg back on track.

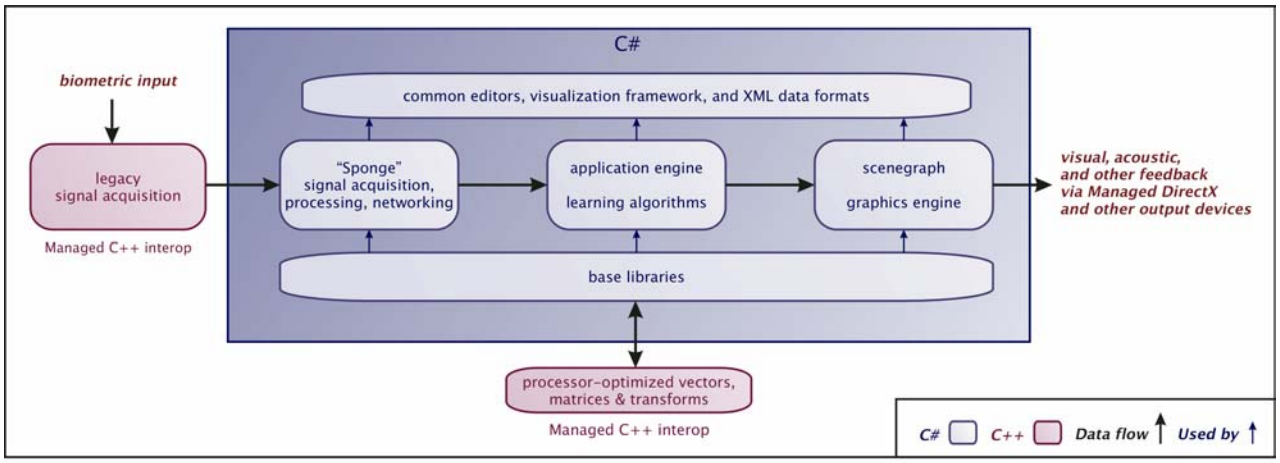
The Brain behind the Brain

All of this requires some fairly fancy graphical and signal-processing footwork. In order that the blinking of the orbs produces a signal that can be reliably detected, the orbs must be rendered at a consistent 60 frames-per-second or more. Our in-house C# graphics engine and scenegraph, *Puck*, is capable of rendering the orbs, together with the animated Mawg and his environment, at over 100 fps on conventional hardware running Windows XP.

It would certainly be possible, given performance figures like these, to perform signal acquisition and processing on the same PC that is rendering the graphics. But in order to facilitate rapid development, and decouple the signal acquisition and processing steps from the actual gameplay, we used the *Sponge* signal processing framework to offload signal processing to another PC. On that signal acquisition PC, the electrical signals are acquired from the brain, VEPs are detected, and the left-right feature is extracted. That simple feature is then sent across the network to the computer that controls the Mawg and renders the Mind Balance world.

It's a comparably simple step to take the *Mind Balance* technology and add another axis, thereby turning it into a two-dimensional controller. And another technique currently under development involves the observation of *imagined muscle movements*: instead of staring at the blinking orb, you just *imagine* moving your left hand, and the character moves left. So although today we're just taking the Mawg out for a walk, tomorrow we may be making Jedi Mind Tricks a reality!





The Need for Speed

MindGames has been using C# and the .NET Framework for over a year to develop the engine that provides the foundation for all of these projects. The software architecture that the group has developed in-house includes *Sponge*, a real-time signal processing engine that can dynamically adapt to absorb available resources on a LAN; and *Puck,* a 3D scenegraph and graphics library that utilize many advanced features of Managed DirectX 9.

The group has discovered firsthand that when working with the .NET framework technologies, rapid development doesn't come at the expense of rapid execution. The *Mind Balance* project, for example, relies on the extremely precise presentation of visual signals, which are processed by the brain and then reliably detected as electrical signals acquired non-invasively from the surface of the head. *Mind Balance* uses the *Puck* and *Sponge* libraries, is written entirely in C#, and runs robustly at over 100 frames per second. (See the *Mind Balance* sidebar for more information.)

As another example of how fast the engine runs, some of *Still Life*'s particle effects produce and destroy literally thousands of new objects a second – an egregious abuse of memory which is handled with great panache (and without hiccups) by the Common Language Runtime's garbage collector.

All the .NET's a Sponge

An important component of the MindGames codebase architecture is a real-time signal processing framework, named *Sponge*, that has been used in our applications to process everything from video signals to brainwaves. Sponge provides a visual interface that allows a designer to drag and drop atomic Signal Processors that perform operations ranging from Fourier transforms and windowing functions, to color tracking and image differencing. These can be assembled into real-time Signal Processing Networks that are used by an application.

One particularly useful feature of Sponge is its ability to dynamically harvest available resources on a Local Area Network. Sponge uses WMI technology to query other computers on the LAN that are running a sentinel program, and determine what processor and memory resources they have available. Then, using .NET Remoting, Sponge can connect to chosen machines and deploy signal processing tasks.

The original intention of Sponge was to facilitate system designs where signal acquisition and processing could be performed on unspecified computers, in a potentially dynamic LAN environment. Sponge has allowed for this, and also offered some tremendous fringe benefits. For example, *Still Life* can use Sponge to record a live video of a performance by streaming screenshots to another computer over the LAN in real-time. These can be processed off-line to turn them into a moving video file. Trying to stream the screenshots to the hard disk of the computer rendering Still Life was prohibitively expensive, but Sponge does the job with only a moderate framerate hit.

A number of features of the C# language made Sponge's development straightforward. The language's unified type system allows our Signal Processors to acquire Objects as Inputs and Outputs, and thereby accept boxed primitive types (and arrays of primitive types) in addition to more complex Objects. Reflection is heavily used by the visualization framework, for example, to automatically make Signal Processor subclasses available for interactive construction. And Serialization – both binary and using a custom XML format – is used to store Signal Processing Networks, and even transmit them over the net. All of this has meant rapid development, the integration of powerful editors and visualizers, and the ability to leverage powerful existing tools in our architecture.

With the nascent Compact .NET Framework, we're not even tied to our desktop PCs any more. In a matter of days, a miniaturized version of the architecture was used to port another major project, *Relax to Win,* to the PocketPC. *Relax to Win* is a two-player competitive racing game that is controlled by each player's level of relaxation, as measured by their Galvanic Skin Response, which is a technology similar to the one used in lie detectors. As each player relaxes, their dragon moves faster. Therefore, unlike in most competitive games, in *Relax to Win* it is the player who can most overcome their tendency towards increased tension and stress that will win the race.

MindGames has been thoroughly impressed with how the .NET technologies have not only allowed us to develop a strong foundation, but also to rapidly build and deploy such a wide variety of applications. Through our work with the Central Remedial Clinic and other partners, these technologies have helped us create tools for empowering people with special needs – with applications as far-reaching as physiotherapy and the performing arts!

Robert Burke, MindGames Group, Media Lab Europe

… who would like to acknowledge his advisor, Gary McDarby, and all the members of the MindGames group for their immense contributions to these projects, as well as our collaborators on the Féileacán and Mind Balance projects: the Central Remedial Clinic, University College Dublin, SMARTlab UK, the Fluxis group and Counterbalance, and New York University's Center for Advanced Technologies.

(A version of this article originally appeared in CodeZone Magazine, December 2003.)