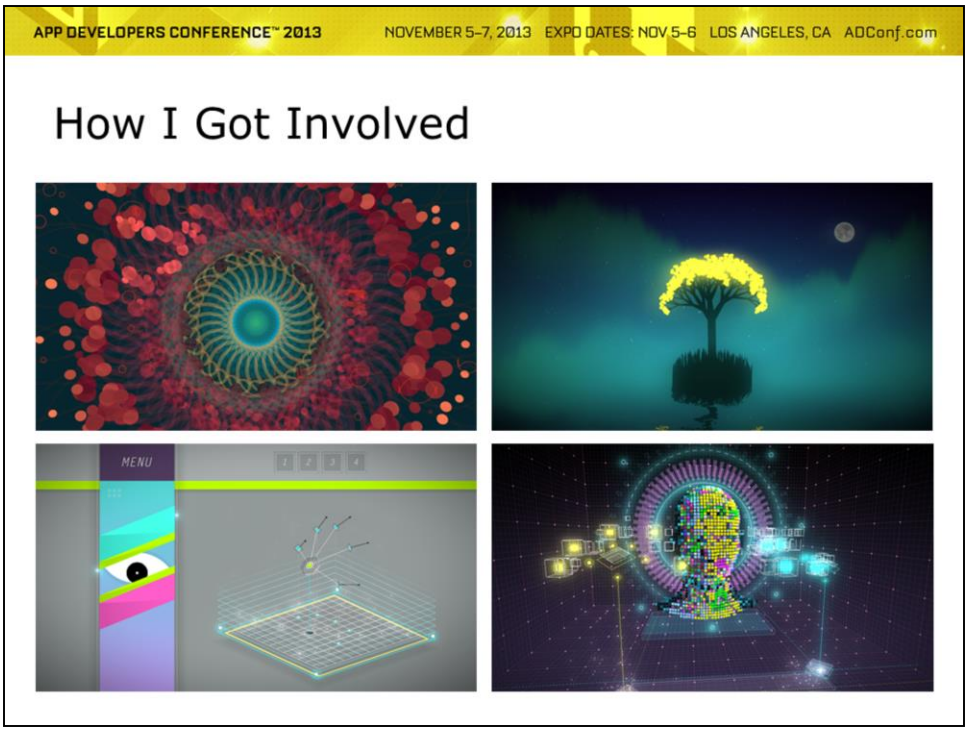
Developing Apps in the 3rd Dimension

Eddie Lee
Founder, Funktronic Labs

HI, my name is Eddie Lee and I am currently the founder of a small independent game studio called Funktronic Labs.

Previously to founding the studio, I was a graphics programmer at a Japanese game studio called Q-Games, where I worked on the Pixeljunk series on PlayStation 3 and PC.



So my first experience with 3D motion controls is when I started working on a title called "PixelJunk 4am" when I was still a programmer at Q-Games. PixelJunk 4am is music creation tool/visualizer, on the Playstation 3, where the player must use the Playstation Move controller to create and weave music andviuals . It was nominated for an IGF award last year.

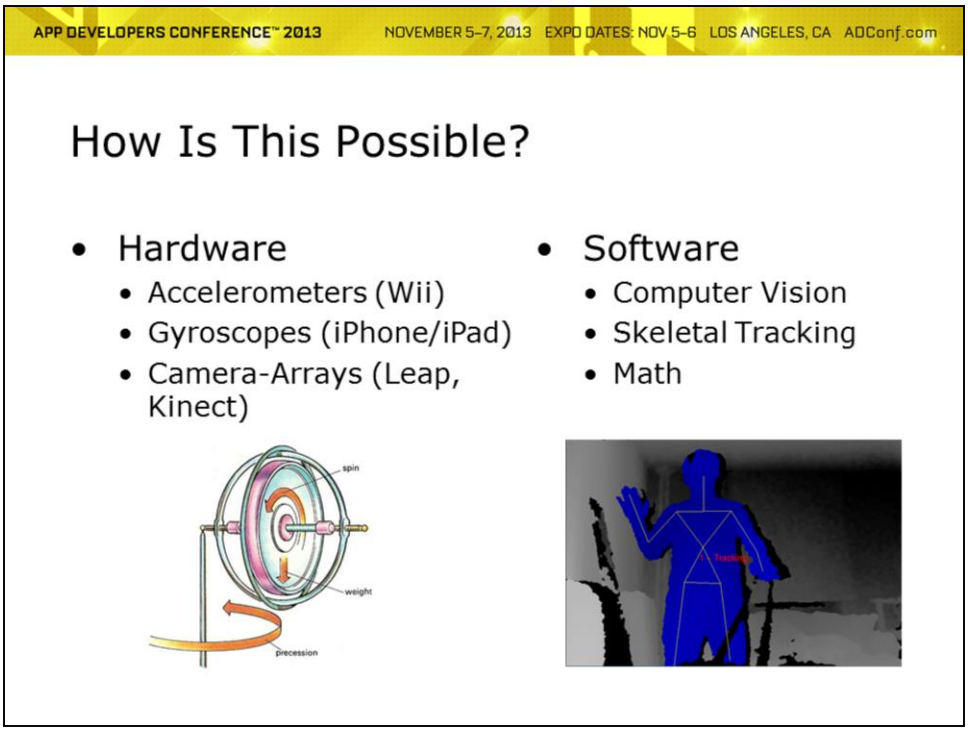
After I left Q-Games, I founded my own studio and was contracted by Leap Motion to help develop apps for their launch during this July (2013). I created two games for Leap Motion, one of which is bundled with the device!

Because of my experience with games, I just want to say that I will be speaking mostly from the perspective of developing motion controlled games. However, these principles can easily be carried over to apps.

3d Motion controls are defined as human interfaces that allow users to interact in 3D space.

With the **previous generation**, who had most famously the WiiMote and the Playstation move controllers. We also had Microsoft **Kinect** which allowed for skeletal tracking.

As we move into the **modern era**, we have devices like the Leap Motion as well as the next generation of Microsoft Kinect that will bring about even more accurate and responsive motion interfaces.



From the **hardware side**, motion sensing technology have become widespread due the lowered cost of hardware modules such as the gyroscope and accelerometer. Even from a DIY perspective, you can just even hack in your own motion sensing device for cheap.

The **accelerometer**, which is used in the WiiMote records the **3d direction** and strength of the **force** that is being applied on the device. It can detect exactly which direction the device is traveling as well as two tilt directions: bank (rotating z-axis) and pitch (rotating x-axis).

Gyroscopes, which in inside the iPhone, tracks the **modular angular rate in 3-axes** (yaw, pitch, and roll), as well as rotation around gravity. The iPhone/iPad actually combines the gyro and accelerometer to provide 6-axis motion sensing.

But not only it is hardware that made this possible, advancements in **software** how also helped usher motion detection into the modern era.

Using dual camera array have allowed technologies such as the Kinect and Leap to exist by triangulating positions using infared sensors and running statistical algorithms to detect skeletal or positional data.

As developers, we are savy to the fact, but when you mention motion controls to the average Joe, they would probably joke about being like Tom Cruise in Minority report. Don't get me wrong, Minority Report was a great movie, but one downside is that it has clouded people's perceptions of what motion controls actually is and what it can be.

Visually, it is impressive and stunning, but in practice it is not very applicable, at least not for long term use. If you ever played a Wii game, depending on your physical fitness level, you will notice that your arms get quite tired quite fast. In the movie, Tom Cruise waves his hands dramatically to shift through GUI windows and swimming through documents. Looks cool, but for navigating information, on a daily basis, it is impractical, because if you just do this for 5 minutes, and you will quickly be out of breath.

In the movie, Tom Cruise drags windows windows physically from left to right in an almost 1-to-1 mapping in physical space. But I feel these actions can be abstracted to reduce the

amount of energy required on the user. Why not, instead of having to use your entire upper body just to zoom in or out of a window, why not abstract it just pinch gestures using one hand.

Motion controls aren't just a 3-dimensional substitution or replacement for your mouse and keyboard, it is a **completely new interface paradigm**.

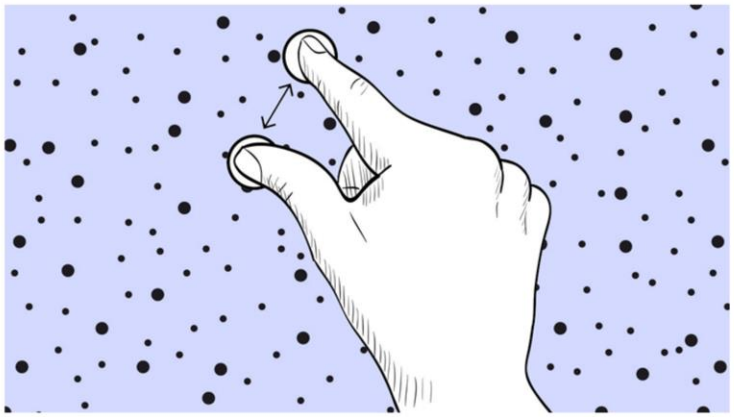
For example, people on the internets have been quite **vocal and frustrated** when they attempt to **replace their PC/Mac's mouse and keyboard** with motion controls like the Leap Motion. The reason why motion controls doesn't translate well is because the whole Windows Graphical interface was built to be used with mouse and keyboard, and thus simply plopping motion controls, which it was absolutely not intended for, makes it feel very clumsy and awkward.

It is like when a developer ports a **D-Pad controlled game** onto a mobile platform by simply adding a virtual D-Pad, it just doesn't work; it feels wrong, it feels clumsy, and simply unusable, generally. Just like the best games on the mobile platform makes the best use of touch mechanics, I feel the best interface paradigms for motion controlled apps need to be built from the ground-up with motion controls in mind.

Don’t limit yourself to the conventions of previous generation interfaces,. Motion controls is a wonderfully new and adventurous frontier, let’s embrace the new!

Best Practices for Apps/Games

- Intuitive and Natural



(Image from FastCoDesign)

So anyways, how do we apply some best practices when integrating motion controls into your app?

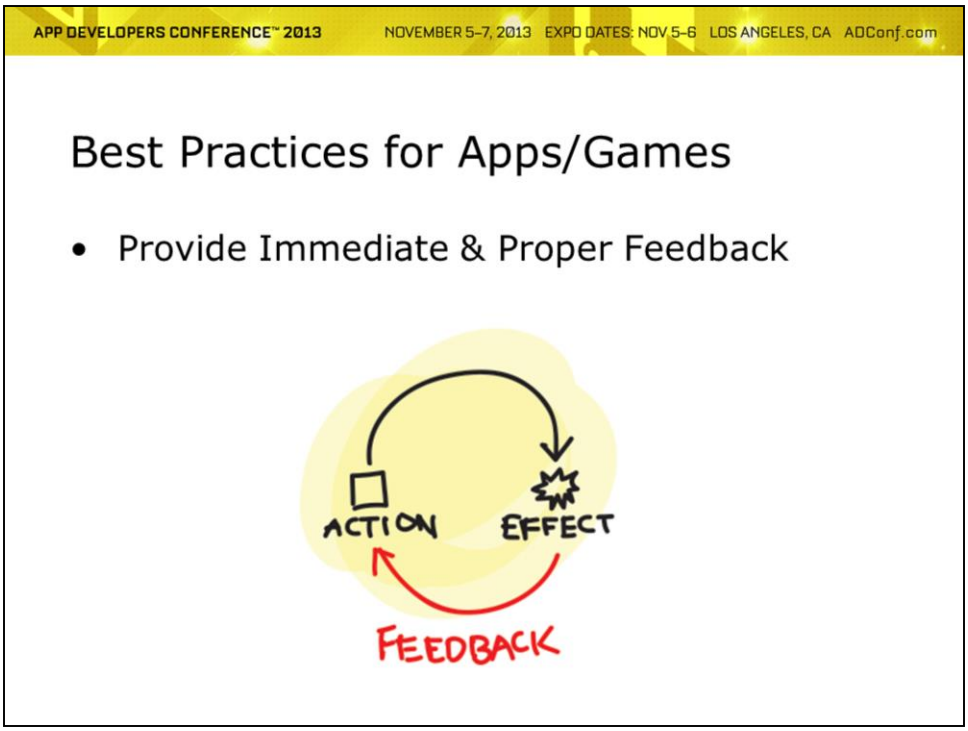
Motion controls in 3D-space is definitely a **new interface paradigm**, so you cannot assume that people will understand motion-controlled mechanics as well as people understand other human interfaces like the keyboard or mouse. For example, left-click is widely understood as the primary action and right-click as secondary; also, double-clicking and all that have become standardized and commonplace over decades of using the mouse.

Thus, given the relative novelty of motion controls, it is best to draw inspiration from **real-world physical behaviors**, such as swiping, turning, throwing, as they are relatable way for users to accustom themselves to motion controls and to immediately connect themselves to the experience.

Sure, the user has never played with motion controls before, but they know how to throw, how to swipe, how to spin

things… By using real-world behaviors, it can better bridge the gap between the new (motion controls) with the familiar.

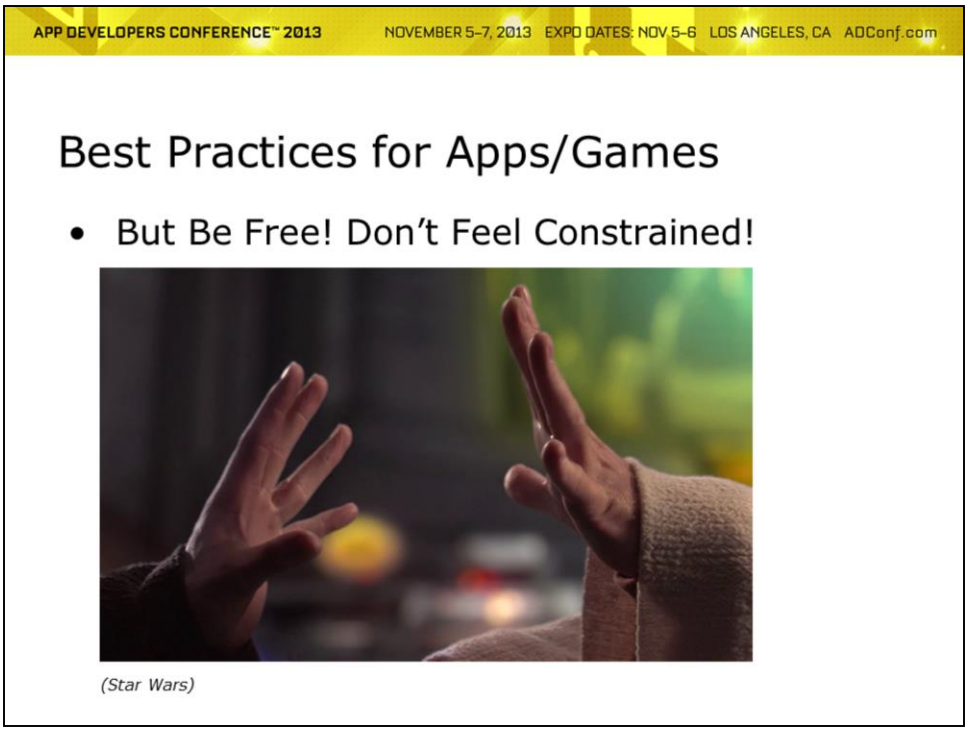
Basically, the more intuitive and nature the interface, the less training is required for them to play your game. The ultimate goal is to make the controls so natural and intuitive that the user requires **no tutorials or instructions** to be able to play your game.



Deliberate actions and gestures should be **immediately recognized** (with visuals and/or audio) so that the user could immediately understand that their actions were registered and if they are doing the right actions correctly.

The player comes into your game **fresh and new**, not really knowing the controls or mechanics, but if you provide good feedback, it also allows the player to quickly learn and understand what they are doing right or wrong and ultimately can figure out how to play your game.

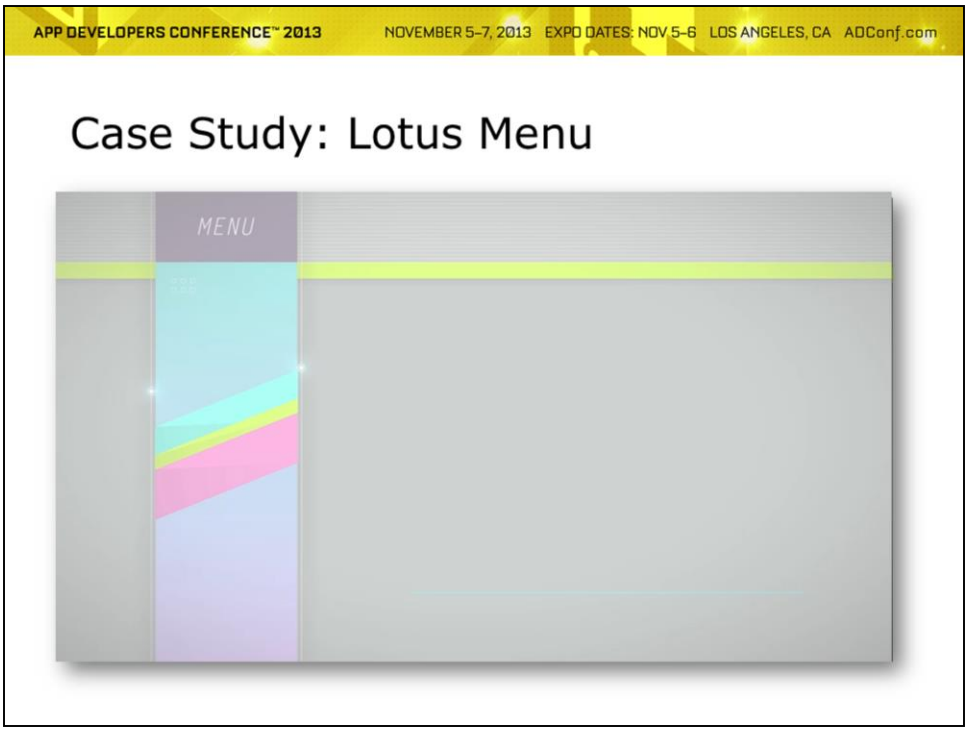
The worst thing you can do is to make the user wonder if their actions were properly executed; providing not enough feedback will immediately disconnect the player from the experience and cause the player to simply check out, so make sure the feedback is **immediate** and **crystal clear**!



So, as I said earlier, draw inspiration from physical interactions the real world, but that doesn't mean that everything needs to be **100% physically correct**! "Don't feel constrained by the limitations or inconveniences of the real-world — this is your world."

An example would be if there is an **box** far off in the distance, why make the user walk all the way over or reach all the way over to pick it up? Why not give the player "the force" to bring the box to the player! Imbue the player with the power of magic and let them feel like a digital magician!

As long as actions are relatable and aren't too abstract, the user will be able to pick it up and will enjoy their newfound sense of power. Motion control technology is meant to empower us, so don't just simulate the real world.



So to vibe on the "don't be constrained" tip, I just wanted to give some insight on a menu system that I have developed for the app I've developed called "LOTUS".

In Lotus, there are four scenes that the player can choose between, and I wanted to figure out a way for the player to intuitively choose between these four scenes. My first idea to come to mind was to kick it old school with four labelled buttons that you have to press with your finger.

But after I've implemented it, I realized that it just wasn't very fun. Pushing 2D interface buttons with motion controls felt awkward and frustrating and I definitely needed something new and refreshing from motion controls to inject into this application. I thought, well, why not enable the user to choose their app with their fingers. One fingers loads scene 1, four fingers loads scene 4. It is natural, it related the user to how they would generally gesture numbered choices with their fingers.

When you use are using our **mouse**, just moving the mouse a few centimeters across your mouse-pad can span your entire monitor. Otherwise, if it was exactly 1-to-1 physical mapping, your arm would get really tired from having to move your mouse a foot-or-so just to move your pointer from the left to right position on your monitor.

Same goes with motion controls: **you don't need to do a 1-to-1 mapping** of your physical hand position to what happens on screen. Give the player some power, and exaggerate their motions.

However, **beware of over-amplication**, as over-exaggeration might make the player feel like it is too sensitive and that they do not have any control over their actions.

Subtle gestures should be reserved for subtle actions.

Conversely, an act such as closing an application or deleting a file can be a non-reversible event requiring a more deliberate action. You don't want the user to accidently erase a file, so make sure the gesture required to do so is as important as the action.

Conversely, you do not want basic actions to require big gestures, as it will quickly frustrate the user.

As a developer, you may have been developed your motion controlled game for months and are really used to the control scheme, but it may not be intuitive for everyone!

The best way to determine if your motion controls "feel right" is to have others try out your game and observe how the playtester reacts. Most people have not experienced motion controls before, so you cannot assume they understand motion-controlled mechanics very well. Thus, your goal should be trying to make the controls feel as natural as possible and you would tweak your values accordingly.

You should get into a loop of testing, observing, and tweaking; and you repeat this until you feel that users (especially new users) are no longer frustrated but having fun!

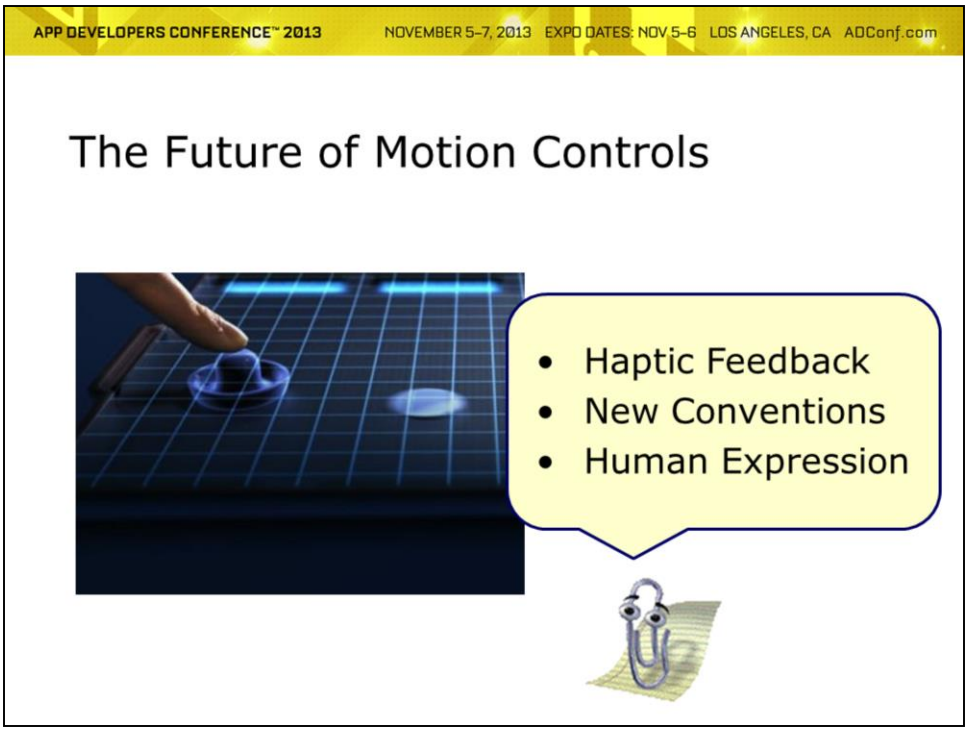
Maybe your game is perfect for motion controls. But maybe not!

But gamers are savvy and will definitely notice when a game has blatantly "**shoehorned-in**" a control-scheme at the last-minute. A shoehorned-in controls scheme usually ends up feeling very gimmicky and is a total disrespect for the art as well as the gamer.

And just like you cannot simply port a **D-pad** controlled game onto mobile by adding on-screen virtual D-Pad controls, you can't assume that plopping in motion controls into a game will be a good idea.

Take time to evaluate if your game will benefit from motion controls, as some games might be better off *not* having motion controls. For example, a heavily menu-based game might not be a good idea as motion controls does not fare well in navigating menus. However, physically interactive games such as *Cut The Rope* or *Fruit Ninja* have transitioned very

well to motion controls.



Haptic Feedback – One problem with motion controls is that you do not receive the proper tactile feedback as you do with other successful devices like the mouse or keyboard. So sure, you may have used your finger to "press down" on a virtual button, but without any proper tactile feedback, the motion may leave the player feeling empty and disconnected.

Haptic feedback is the use of the sense of touch in a user interface design to provide information to an end user. There are already several companies doing amazing things with haptic feedback. Specifically, Valve with their controller is going to be the one that brings it mainstream, is my prediction.

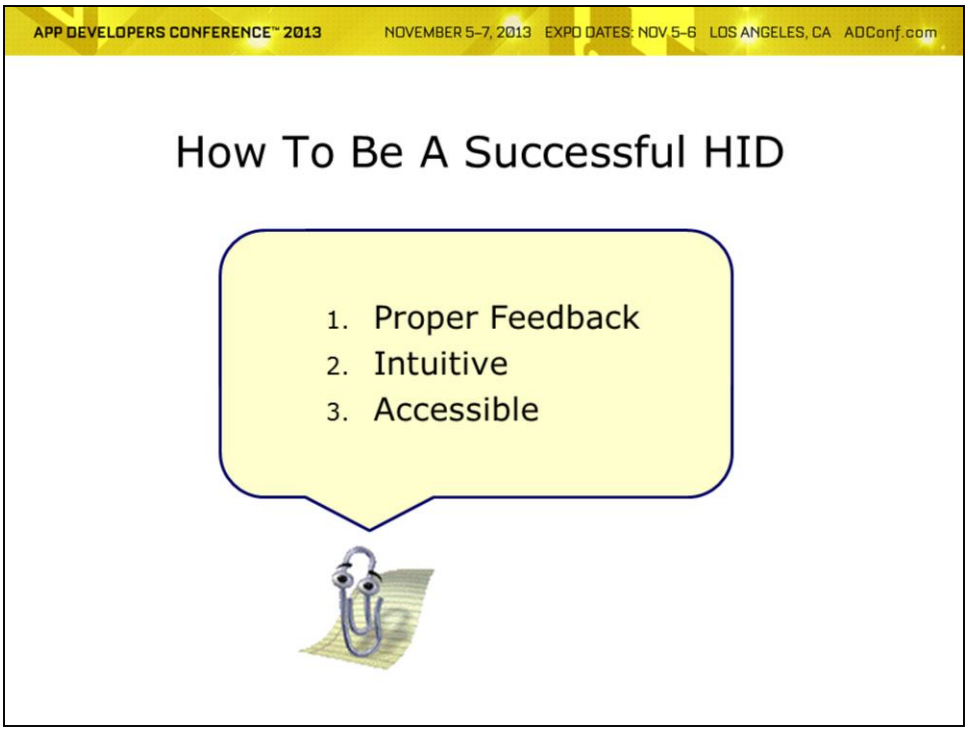
New Conventions – As I mentioned previously, as motion controls become more standard, new conventions will be born and people will build from these conventions in new and interesting ways.

Human Expression – The evolution of human interface devices has gradually evolved from mechanical to more natural, and now, from **2D to 3D**. From trackballs and mouse to touchscreen and motion controls, the interfaces has evolved to allow for **more expressive gestures** and more **meaningful interactions**, **bridging the gap** of the real and virtual world, **merging** man and machine and **removing the barrier** of human expression in digital space.

I think the future is extremely exciting and I am definitely looking to see how motion controls will embed itself in our society and what wonderful things people will build with this technology!

Thanks! Questions?

(Follow me: @**eddietree**)



HID stands for **Human interface device** -- such as keyboards, mouse, touchpads, joysticks, gamepads, etc…

In order for 3d motion controls to be widely accepted as a standard form of interfacing with your computer and not just a gimmicky fad, we need to consider and evaluate previously successful interface paradigms and observe how we can apply what we learned to motion controls.

Proper Feedback – The mouse and keyboard are successful because they provide great feedback, specifically tactile feedback. When using a mouse, everytime you execute a click, the feedback is very immediate, clear and responsive. Same with typing on the keyboard; in fact, you can probably type a whole essay with your eyes closed, the feedback is immediate and you can trust that actions were registered. Same as touchscreens, you feel the screen when you swipe your fingers on it.

Of course, there are physical feedbacks, but it can be **virtual**

too, like visuals or audio.

Intuitive – The mouse is intuitive because it maps to the 2D layout of the computer screen. Touchscreen phones/tablets are intuitive to use because it combines the touch interface with physical behaviors like elastic bouncing on the windows makes it really relatable and easy to get into.

(**Keyboards**, not really that intuitive, the learning curve is high and the path from beginner to productive typist takes a lot of muscle training, but its popularity and success is born out the necessity for humans to input characters into the computer.)

Accessible – It has to be usable. Like I mentioned earlier, it has to be able to be used for an good amount of time without physical or mental fatigue. I can go a very long time with using my mouse and keyboard without feeling tired. Also, it has to be affordable in order to become mainstream.