# Buttons, Simplicity, and Natural Interfaces

**J.R. Parker**
Department of Drama
Faculty of Fine Arts
University of Calgary
jparker@ucalgary.ca

## Abstract

Because the push button was an inexpensive way to send control signals to a computer from a user or game player, it became the most common aspect of game controllers. The fact that a pressing motion is not in any way connected to typical game activities makes a button inappropriate, and damage may have been done to playability and to user expectations in the long term. Although there was no option in the early days of video games, the push button must be abandoned as soon as possible and be replaced by more natural interfaces, which are largely implemented in software.

## Introduction

It has been recently said that "the button is a central feature in the short history of video games" (Griffin, 2005). If true, this is an unfortunate state of affairs. Objects and ideas are often used in ways unanticipated by the designers, and the use of the button in game controllers is certainly one of those ways. The button is a cheap way in a video game to control electrical current, and therefore is an attractive control option in many circumstances. It is, however, sadly inflexible, and shows none of the ability of a modern computer to express user intention. It is, in spite of its firm insertion in popular and technical culture, not a natural device. It rarely reflects a normal activity for a person using it in a video game situation.

It has become a part of our culture to use a doorbell, for example, although it was not always so. To press a button and hear a bell or buzzer is normal for people today. This is an evolution; a hundred years ago it was not so. To push a button and expect a character on a screen to punch another, or to have a vehicle on that screen accelerate as a result is not completely natural. It does not correspond with our experience when fighting or driving a car, but it is becoming natural when playing a video game. It seems clear that what is culturally normal, what can be expected from devices and other people, evolves with time. This sort of evolution is to be expected.

On the other hand, the culture of the video game is not very old, and memes can still be changed. The button can be thought of as a hindrance in the modern context of games, and can be likened to the semicolon in programming languages. A semicolon in the Pascal language is a statement separator, in that one places them in between statements. In the C language semicolons are statement terminators, to be placed at the end of every statement. The same symbol means different things. Similarly, the action of pushing a button means different things in different games. For the same action to mean vastly different things seems a little bizarre when looked at carefully. It is normal for different buttons to do different things, but controller buttons have a different mapping in each game genre, if not each game. Humans can easily learn these, but it is an unnecessary complication from a machine that is supposed to adapt to us, rather than the other way around.

## Buttons and Historical Context

The button is, in an electrical sense, the simplest form of switch. It has been used since the 19th century to control (on and off only) the flow of electrical current by creating a physical interruption. Electricity requires a complete loop from positive source to negative for current to flow and for work to be done, and breaking that loop will turn off any appliances in the circuit. Alternatively, if the button completes a circuit then any devices in that circuit will be activated. It is in this latter form that the button found function in the 1890s as a doorbell activator. The first portable button was likely the type used on flashlights in the 1920s.

Thus it was that the button came into the public consciousness. Because they activated electrical labour-saving devices, of which there were suddenly a large number, buttons became synonymous with automation and leisure, and later with all things modern and futuristic. This 'push-button' world was illustrated in advertisements and became a key aspect of popular culture, as seen, for instance, in Disneyland and on television programs like the Jetsons (see *Figure 1*). That George Jetson lived in the future was obvious, and that his job was to push a button was a reference to that future in the minds of 1950's viewers. The fact that nobody knew what the button actually did was not relevant, and rarely did anyone give it any thought – it was the button itself that was thought to be important. However, the button would in fact have been an interface to another device, and its use would have normally required some judgment or skill. This context was absent from the narrative, and the button became merely an abstraction.

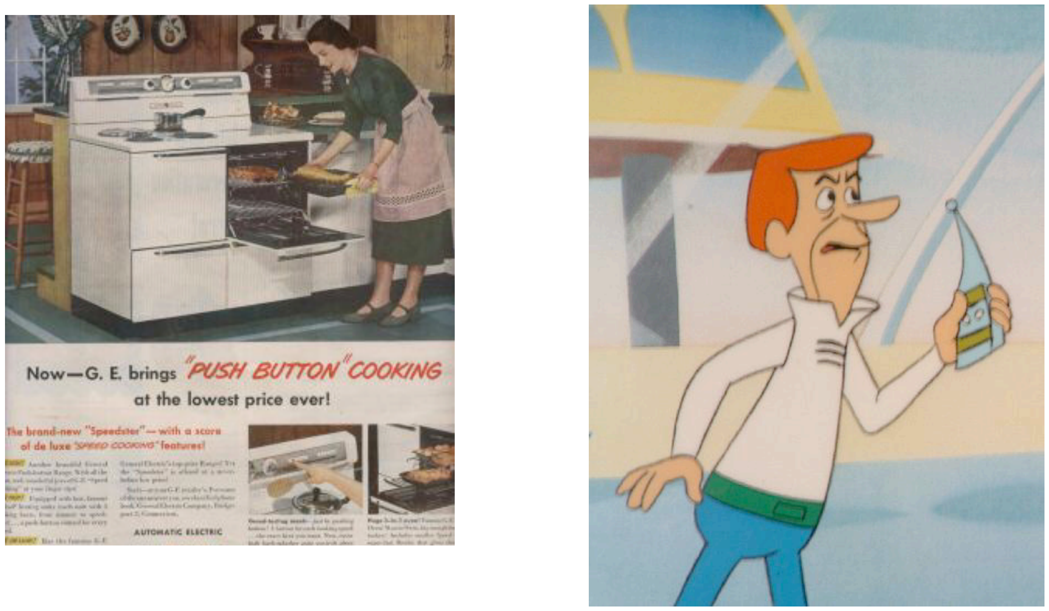

Now—G. E. brings "PUSH BUTTON" COOKING
at the lowest price ever!
The brand-new "Speedster"—with a score of de luxe "SPEED COOKING" features!
AUTOMATIC ELECTRIC

*Figure 1*: Left: The labour-saving push-button world of the future seen by advertisers in the 1950's. Right: Cartoon character of the 1960's George Jetson had a 'futuristic' job as a 'digital index operator' (button-pusher).

## Natural Interfaces

An interface is a connection between two distinct things. The human skin, for instance, is the interface between a person and the world. It goes further than that, though, because interfaces can be *modal* - ears are present as an *audio* interface, allowing sounds to impinge on the animal consciousness, and the eyes are a *visual* one. An interface with a computer is a sensory or communication link between a human (usually) and a computer. A keyboard is a simple example, and an appropriate one here because it consists of a set of buttons. A computer interface usually has a hardware component and a software one (such as a driver). A keyboard has become a natural interface to a computer, mainly because it permits the transmission of text to the machine. Text is a very common way to communicate even between people, and a keyboard is a natural device for entering or recording text. A mouse is a better interface only when it is used in the context of a graphical display having widgets designed for a mouse to manipulate – it's awful for sending text to a computer. Thus a mouse is not necessarily a more natural interface than a keyboard. It depends on the mode of operation.

Consider the actual case of a novice user who could deal easily with the mouse when it was only moving the screen cursor left and right. However, she lifted the mouse up from the desk when she wanted to make the cursor move up the screen. This was the natural way to make it work, in her opinion. One wonders what she would have done to make the cursor move down the screen.

The interface used by a game is *natural* if the actions used to play it have a natural interpretation based on the context of the game. A fighting game would naturally use punching, jumping, and kicking motions, and little in the way of stroking, tapping and sliding. A soccer game would naturally involve running and kicking, but no punching. A non-natural interface causes an interruption in the *flow* (Csikszentmihalyi, 1997) of the activity being performed. This interruption is the result of the disconnect between an activity being portrayed in the game, which the player is participating in indirectly, and the activity required by the player to perform the interaction. This results in a splitting of attention that is not productive or amenable to the effective completion of either task being performed.

Natural interfaces that can deal with actual human motions in real time are not yet particularly common or effective. Such interfaces must either use special purpose devices or rely on computer vision technology. Computer vision is a largely software-based technology that is far from being perfect in general, but it is usable for specific tasks in restricted domains, so-called *tactical* machine vision. Computer vision is not generally able to be used reliably to detect human pose or identity activities, and is especially unreliable when used in real time applications. Monitoring human activity in a video game context must be done in a way that implicitly eliminates most of the ambiguity that is usual in vision methods. Special sensors must be evolved to be used in this context. Sensors of this type are generally expensive, so it may be necessary to develop new ways to use cheap sensors in the game environment.

There are a few ways in which interfaces can be natural: through speech and sound, motion and position sensing, and through identifying gestures. Any of these can, and have, been replaced in game interfaces by the use of buttons. Speech and positional interfaces require complex software for implementation. The technology now exists to eliminate buttons and use a more natural and expressive interface.

## Speech Interfaces

A natural computer interface uses the activity or motion that is involved with a real activity as the same one used to communicate with the computer or other device (Parker, 2006). Humans often speak to one another so a good example of a natural interface is one that uses speech. It may well be, to be more specific, that a speech interface is natural only if it speaks and understands spoken language. If it only speaks, this can be confusing. So a car or refrigerator than can speak but that can not be spoken to can, in fact, be irritating, which may explain why the fad of having speaking appliances has died away for the time being.

Voice recognition is presently at the technical level where it can be effective in some circumstances, although it is still weak in the general case. Like a fingerprint scanner used to log in to a computer, voice recognition systems require training of the user's voice to be really useful. Most applications of voice recognition take advantage of a restricted domain of discourse; this is sometimes referred to as *tactical* speech recognition (Johnson, 2005). The system is useful in a small domain of human endeavor that has a small vocabulary associated with it. There are two reasons for this. First, the restricted vocabulary makes it easier to match the input audio signals against the known speech examples. The second issue is more difficult to deal with. It turns out that computer algorithms have had little success in parsing human speech in interactive, real environments. That is, even if the voice recognizer succeeds in converting a sentence into text, the correct interpretation of that sentence is still impossible in the general case. Tactical voice recognition domain helps here, too.

When giving the computer an oral command, it can be structured so as to provide contextual aid. The military is good at this. Experience with the Canadian Air Force has taught that a command on a parade ground has four important parts: the identifier, precautionary, cautionary, and executive parts. This was reduced to a simple identifier/ acknowledge/ command protocol used in the television show *Star Trek* when speaking to their ship's computer.

The important parts of the command, and the two that nearly always occur, are the *identifier* and *executive*. The identifier states who is to execute and gives warning of a command to come. The executive allows a detailed command and indicates that it is to be done *now*. The military command *'squad stand-at ease'* has *squad* as the identifier and *ease* as the executive (Parker, 2006).

The *Star Trek* version is directly relevant to our situation. The command is normally a dialog, as follows:

> *Spock*: Computer.
> *Computer*: Working.
> *Spock*: Calculate, to the last digit, the value of PI.

The idea is to alert the computer that it is to pay attention to the next spoken sequence and execute it as a command. This is done by first saying the word 'computer' before issuing a command. The alternative is that the computer listens to everything, and may execute commands that are spoken but not intended. For example, in a typical family household simply yelling at the dog to 'stop' could result in the video recorder taking the obvious action, and someone may miss their favorite show. Precautionary and cautionary commands can be used to further restrict the domain. After 'computer' the user may say 'video', meaning to search the TV/video function of the computer for the next command and to execute it.

Basic voice recognition is being used in some games at this time. The Nintendo *DS* game *Nintendogs* for example, has a speech recognition system that is trained by the player. The Nintendo *DS* is a portable game platform with limited processing power, which makes this functionality all the more impressive. There is also the *Game Commander* software for converting speech into keystroke sequences for PC games, making speech input possible with existing games.

## Audio Input

Karaoke is two Japanese words: *Kara* = missing and *Oke* = orchestra. Most people know what it is about, and the appeal of the pastime is that it can be social among friends, or it can be used as singing practice for the shy types. In the game *Karaoke Revolution*, the player has access to a selection of songs. When a song is chosen it starts to play, and the lyrics are displayed on the screen. The player is expected to sing along, and the computer scores according to how well the melody is followed in a tonal sense. The system extracts the main frequency in the player's singing and matches it to that of the original song.

This can be a lot of fun, and the display is really not needed since most folks know the words for the song they're going to sing. One problem is that of *ornamentation*. If the song is not reproduced faithfully then a penalty will result. This leaves no room for creativity, although the activity can still be fun.

There are other examples of the use of sound as input. The *DS* also has a game that allows the player to blow out a candle by actually blowing on the screen (the hardware features a microphone). It should not be necessary to enumerate all of the possibilities of sound input to make the point, however (Igarashi & Hughes, 2001).

## Gesture Input

A gestural interface can identify simple user motions and connect them to specific activities. In literature on natural interfaces, more entries than would normally be expected concern the use of gestures (Moeslund et al., 2001; Sato et al., 2007). This works well if the user motion or gesture being recognized is the same one as would be performed in the real activity modeled by the game. The game of solitaire is one of the most commonly executed programs on a PC, and is a game that was originally played with real cards on a flat surface. The natural interface to the game would be to somehow pick up and move cards to their desired position with motions of the player's hand. The cards, of course, would be virtual and the cards could either be projected onto the playing surface or simply be displayed on a screen.

This game needs two gestures. One gesture asks for information about the tableau (*hand open*), another asks to move a card or card stack (*hand closed*). Input of hand locations and poses can be accomplished using computer vision methods. A simple video camera can be positioned above the table looking down, so that the hand is in clear view. A frame grabber captures video of the tableau, which is sent one frame at a time to the vision software for interpretation. This is essentially the same scheme used for playing standard computer solitaire, except that the usual game uses a mouse and a button click instead of hand motions. This gesture-based solitaire game has been built (Parker & Baumback, 2003) and works very well indeed. With the addition of speech output it can be used by players with visual impairment.

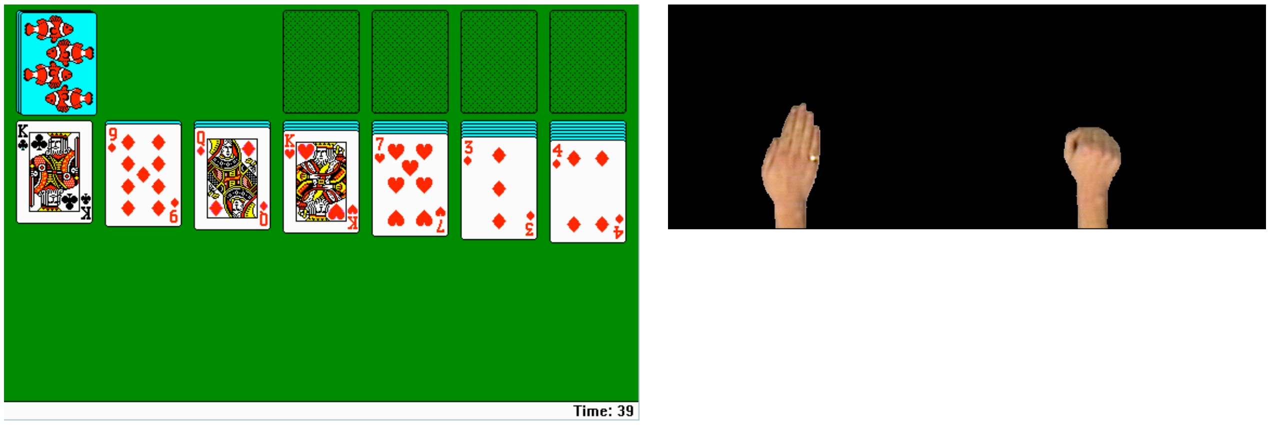


*Figure 2*: *ERSolitaire* takes a standard PC game (left) and controls it using gestures. Two hand positions, open (center) and closed (right) are recognized. Positioning an open hand over a card and then closing the hand 'grabs' a virtual card and allows the player to move it.

## Touch and Multi-touch

The Nintendo *DS* currently has a working touch screen and a well-sized stable of games that make use of it. ***DS*** stands for **D**ouble **S**creen, and the lower screen of the two is touch sensitive. What can be done with a touch screen? One can pop bubbles, poke and prod characters into action, tap things, move object around the screens, and much more. Since the screen is touch sensitive, a camera and computer vision software are not needed. This permits a small, portable game device to be able to implement such an otherwise complex interface.

Touch interfaces are not unlike gesture-based interfaces, but the third dimension is not present since the screen represents a constraint in the y or z axis (depending on how one draws the axes). Multi-touch screens also have this constraint, but permit more than one contact on the screen at a time. These devices frequently use cameras and have a more complicated tracking algorithm that requires more computational power than a standard touch screen interface. Also, because of the cameras these multi-touch screens only work when there is sufficient illumination.

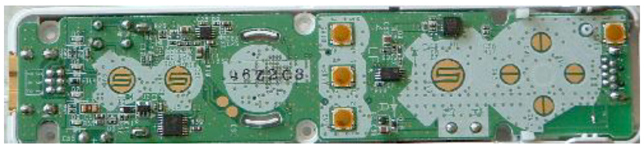
Interestingly, the actions possible on a touch screen are almost the same as those possible with a mouse. It is the method of use, the nature of the interaction, that makes the touch screen a more natural device. Touch screens are already being use with video games. The prospects for game interfaces utilizing multi-touch screens are enormous, but for the near future their expense will limit their use.

## Pressure Sensors

*Dance, Dance Revolution* (*DDR*) may be the most studied game available partly because of its possible exercise benefits (Katz et al., 2006; Lieberman, 2005). The *DDR* dance pad is a collection of primitive pressure sensors; primitive in the sense that they are buttons, of course, and detect pressure in only an on/off sense. An impact on a specific part of the sensor is recognized and coded for transmission by the game console. In both cases the result is either on or off: the sensor does not detect a degree of pressure, only that contact was made. This is a very natural use of a button, probably the most suitable one in modern video games. Contrariwise, it is plain that in a dance the landing of a foot is not a binary thing. Position and pressure are key elements.

The degree of pressure can be measured using piezoelectricity. Such a device uses the electrical charge generated when a plastic-like film (polyvinylidene fluoride) is stressed by an impact: is bent or squeezed or struck. As the magnitude of the pressing motion increases, so does the voltage created by the film. This voltage is sampled and converted into pressure (pounds or pascals) by a computer.

It is possible to build a small grid of these pressure sensors and place them inside the shoes of a player (see Paradiso et al., 2000). The films are thin enough that this will cause no discomfort. As a result of this installation, a player can participate in *DDR* without a dance pad. Jumping onto a foot creates a distinctive pressure pattern that can be detected by a computer. The entire grid of pressures is sent to a PC many times each second and is converted into a pressure *image*. A pattern recognition and machine learning algorithm is used to match the player's pattern against a 'training' pattern for evaluation. Yes, the game requires a bit of start-up help, which can be portrayed as a game tutorial.



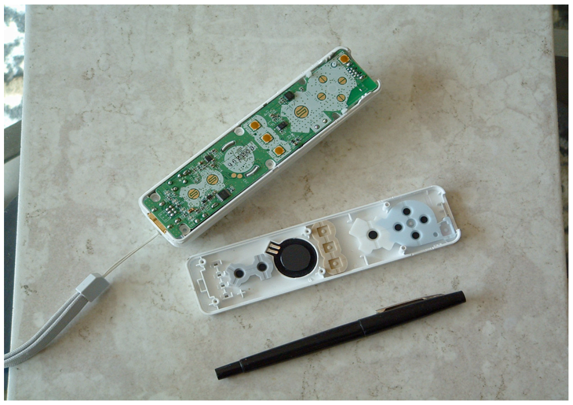


*Figure 3*: The *Wii* controller, inside-out. Left: The famous accelerometer does not occupy a central location in the controller. Right: broken open we see a (4-layer) circuit board and some standard switches (buttons!).

These in-shoe systems are used now as orthotic calibration devices. They are used in medical applications for such things as diabetes screening, monitoring the results of surgery, fitting orthotic devices, and even for the analysis of athletic performance and for shoe design. An example of such a system is the F-Scan system[1]. This device has 960 sensors in a 60 x 21 grid, or about four sensors per square centimeter. There are other schemes for this system, and the price will drop as the market grows. It is not clear how many sensors are required for effective use in game applications, nor how often they need to be sampled or how accurate the data needs to be. The duration of the training regimen will also be important, as few gamers are willing to undertake any more than a minimal training session before game play.

It is possible to retrofit these pressure sensors to existing games, and this will work for multiplayer games as well. Many may have noticed that some Massively Multiplayer Online Role-Playing Games (MMORPGs) involve a lot of running from place to place in the virtual world. In fact, the time involved is used by the game to perform computations and to synchronize events. As a result, the virtual running can be converted into *actual* running. If we add a current generation VR display goggle, which can be had for a few hundred dollars today, then the direction of running in-game can be changed by altering the orientation of the player's body.

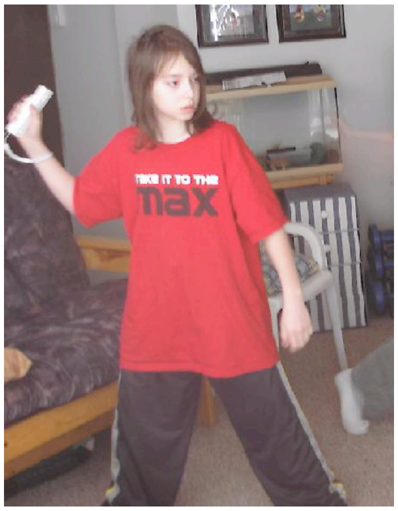
## Sports, Motion, and the *Wii*

The discussion of natural interfaces for use in video games is not moot. There is an existing game console that permits novel interface technologies to partly replace the traditional controller: the Nintendo *Wii*. It has been available since November 2006 and, from the beginning, the excitement was over its controller, not the graphics or the sound (Parker, 2007). The *Wii* controller is wireless, using a built-in Bluetooth link. It has a distinct attachment that plugs into it, called the *Nunchuk,* that acts as a controller for the other hand. For example, in the *Wii Sports* boxing game the *Nunchuk* allows the player to register blows with both hands. The controller contains a small speaker that can reproduce sounds from the main sound system, and a small motor that is used to convey impacts. In addition, there are two major new aspects to the *Wii* controller. One is the use of infrared lights and sensors that permit the controller to be used as a pointer. Thus, it can be a gun, a lifting or pushing tool, a light, and many other things that a game can take advantage of. In practical effect this functionality is the same as that of a mouse, but wireless and with no need for a surface. Traditionally, game consoles have not used a mouse or a keyboard, choosing to instead use a combination of buttons and levers on the controller.

The second, and most important, new aspect to the controller is that of motion sensing. The motion of the controller is detected by a special accelerometer chip and is transmitted to the *Wii*, where the game software can make use of it. An *accelerometer* measures changes in velocity. The most well known use of these devices is for automobile air bag deployment, but they are also commonly used for inertial navigation and condition (wear) sensing for machines. In the *Wii* context they can be thought of as motion detectors. These devices detect accelerations in a particular direction, or *axis*, by detecting the force that the acceleration applies to the sensing element. To detect motion in an arbitrary direction, each of three accelerometers are placed at right angles to the others so that motion in any direction will create a detectable force on at least one of the sensors at all times. This is called a *3-axis accelerometer*, and can be obtained as a unit. The ADXL330 device, for example, is a 3-axis accelerometer which can be purchased for about six dollars.

A key use of accelerometers in the *Wii* is in estimating the motion and orientation of the sensor by *dead reckoning*. Since the controller is in the player's hand, the game will have a good estimate of the hand's position during each frame, as well as direction and speed of motion across multiple recent frames. This information can be fed to the game so graphics can be updated to display a view of the virtual world in response to player actions. The graphics are not high resolution or realistic, but they reflects the play and are 'fun'. A typical use is to have the controller play the role of an instrument that is held or moved by the player. In a baseball game the controller acts as a bat, in a tennis game it is the racquet, in a shooter it's the gun, and in a driving game the controller stands in as the steering wheel. One treats the controller in much the way one would treat a real device in a real situation.

So assume that the player is holding a bat. As the player swings the controller, the avatar on the screen can be made to perform a similar motion. Since the avatar is holding a virtual bat, it can interact with virtual objects in the game world (like other players, and baseballs). This interface encourages physical activity, not just in the sense that players need to move their hands, but, in fact, that players can be encouraged to immerse themselves in the simulation and put their whole body into a game. The standard *Wii* games involve sports: tennis, bowling, boxing, and baseball. Network play allows people to play virtual sports with other people in distant locations (Mueller & Agamanolis, 2005). This happens now with traditional controllers and can be very entertaining, but bowling and baseball are normally social games played with and against friends. This is, again, a natural interaction. It's also a long way from a simple button, even if there are a dozen of the dreaded things still present on the *Wii* controller.



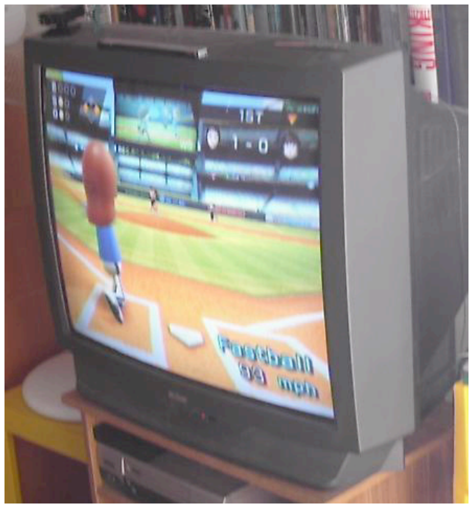


*Figure 4*: Gameplay in the *Wii Sports* game. Left: The player is playing baseball. Although a smaller motion would work the same way, players tend to adopt a batting stance when playing. Left: The graphics are not high resolution or realistic, but they reflects the play and are 'fun'.

## Conclusions

The statement of Griffin (2005) that, "only the controller can lead to action in the game space," is clearly incorrect. User actions can obviously lead to actions in game spaces, and this is a preferred mode. Mechanical controllers like the button are actually a drag on game design, limiting the degree of immersion by requiring that the player recall button press sequences for each game that they play. This hindrance has likely had a negative impact on game design since the medium's conception, and the cumulative effect can't be determined but is likely to be significant. Since the development of the *Playstation*, video games have been built knowing that their controllers would restrict interaction and dictate game play.

Natural interfaces are software intensive and could not have been used with the games and computers of 20 years ago for that very reason. Most of the technology involved is more than a decade old, and yet it seems as though all of the really novel interface ideas in games have been introduced only in the past few years. It’s hard not to imagine that the existence of the inexpensive button-based controllers, coupled with institutional inertia of the console developers, has actually retarded the creation of many truly novel designs.

The immense popularity of the *Wii* almost certainly speaks to the benefits of natural interface and the nature of the gameplay that it encourages.

## References

Csikszentmihalyi, M. (1997). Finding flow. New York: Basic Books.

Griffin, S. N. (2005). *Push. Play. An examination of the gameplay button*. Proceedings from DiGRA 2005: Changing Views - Worlds in Play, Vancouver.

Igarashi, T., & Hughes, J. (2001). *Voice as sound: Using non-verbal voice input for interactive control*. Proceedings from UIST-01: 14th Annual Symposium on User Interface Software and Technology Available online at www-ui.is.s.u-tokyo.ac.jp/~takeo/papers/voice.pdf.

Johnson, L. (2005). *Tactical serious games development! The creation and usage of the tactical Iraqi language trainer*. Proceedings from Serious Games Summit, Washington.

Katz, L., Parker, J., Tyreman, H., Kopp, G., & Chang, E. (2006). Virtual reality in sport and wellness: Promise and reality. *International Journal of Computer Science in Sport, 4*(1), 4-16.

Lieberman, D. (2005*). Dance Dance Revolution: The most researched serious game ever. Why, and what have we learned?* Proceedings from Serious Games Summit, Washington.

Moeslund, T., Störring, M., & Granum, E. (2001). *A natural interface to a virtual environment through computer vision-estimated pointing gestures*. Lecture Notes In Computer Science, Vol. 2298 (Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction). 59-63.

Mueller, F., & Agamanolis, S. (2005). Pervasive gaming: Sports over a distance. *Computers in Entertainment, 3*(3), 1-11.

Paradiso, J., Hsiao, K., Benbasat, A., & Teegarden, Z. (2000). Design and implementation of expressive footwear. *IBM Systems Journal, 39* (3, 4), 511-529.

Parker, J. R. (2006). *Human motion as input and control in kinetic games*. Proceedings from FuturePlay 2006, London.

Parker, J. R. (2007). *Games for physical activity: A preliminary examination of the Nintendo Wii*. Proceedings from 6th International Symposium on Computer Science in Sport, Calgary.

Parker, J. R., & Baumback, M. (2003). *Creating an enhanced reality user interface – ERSolitaire*. Proceedings from CHI 2003, Ft. Lauderdale.

Sato, E., Yamaguchi, T., & Harashima, F. (2007). Natural interface using pointing behavior for human–robot gestural interaction. *IEEE Transactions on Industrial Electronics, 54*(2), 1105-1112.

---

[1] Tekscan Inc, F-Scan In-Shoe Pressure Measurement System, http://www.tekscan.com