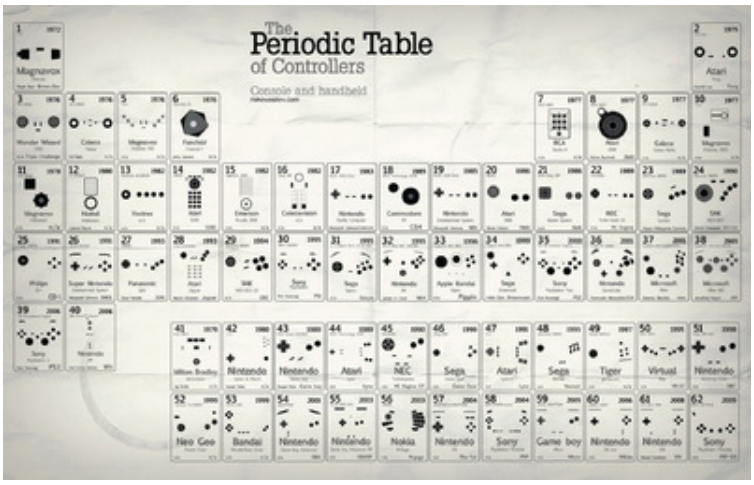
Controller Design: Buttons pt.1

Monday, August 8, 2011 at 11:08PM

Richard Terrell (KirbyKid) in Controller Design, Super Mario Bros.

Before we consider genre. Before **emergence**. Before **gambits, strategy, and tactics**. And before **level design** we consider **mechanics** (player actions). This is why early in the life of this blog I did a series titled **Mechanics & Abstractions**. In it I broke down the foundation of gameplay. Now, I want to take a step back and analyze the foundation of interactive design; controller input devices. From buttons to analog sticks, touch screens to motion controls, and mics to mice we'll take a detailed look at how input design shapes all player actions and expectations.



Click to enlarge. Image by **Ev Turn**

The series will be broken up into articles each examining a particular input device. In each article I'll discuss the pros and cons of the digital information the device can transmit, various examples of the device, how designers have commonly worked with and around the device limitations, and any notable accomplishments.

> *"Whenever we make a new game console, we've done it without throwing away buttons and the directional pad. The reason for that... it's better to have them, because buttons and directional pads benefit gameplay response." ~ Satoru Iwata*

ButtON - ButtOFF

We begin with buttons, a staple in the world of machines both analog and digital. Buttons are so simple that it's easy to overlook why they are so effective. Buttons are a bridge between man and machine. On the one side we have man; an infinitely complex and variable being who interacts and responds to the world with his/her squishy organics. And on the other side we have machine. Cool. Clean. Functionally oriented yet also capable of intricate and complex tasks. These two sides do not speak the same language. To bridge this gap so man and machine can communicate, we can use

buttons.

The simplest button is a one way communication line with only two states; on and off. It's hard to get lost in translation over such binary potential. This button is squishy (like us) yet transmits a clean signal to the machine. If using a machine is like working as a team, then clear communication is key (**read more about team skills here**). Between humans verbal, written, and body languages are effective yet they introduce possible communication errors. However, with buttons between humans and machines, it's clear what messages get through to the machine. When the machine only looks for a button to be on or off we can significantly narrow down variables in our learning experiments. This simplifies the process of understanding the functions of the entire system (man + machine).

Visit the orignal artists' page.

Clear communication is the nature of the "gameplay response" that Iwata referred to. Basically, **trial and error** is the most widely used learning method. When we learn game systems we observe actions, form hypothesis, conduct experiments, analyze the data, and repeat as necessary. The less confusion about the input (experimental variables) and the feedback (data/results), the better we learn.

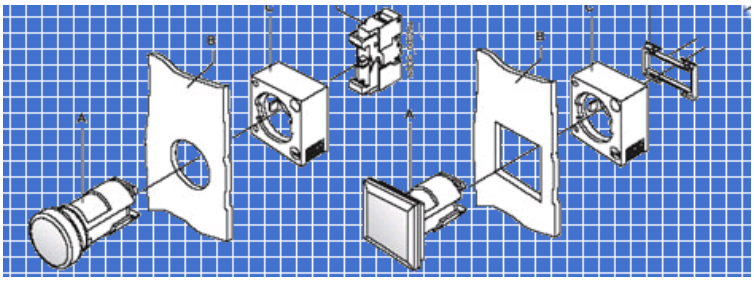
Controller Examples

I don't think it's necessary to go over examples of video game controllers that feature conventional buttons. Almost all of them do. So I present examples of gaming buttons that are less obvious.

- **Gamecube/Wii Classic Controller L/R Shoulder Buttons.** Deep within each shoulder trigger is a shoulder button. Though it is strange that you have to completely depress the analog shoulder triggers to reach these buttons, because of the distinct snappy feel and independent on/off state, they are still buttons. Super Mario Sunshine, Luigi's Mansion, and Super Smash Brothers Melee use the shoulder trigger/buttons well.
- **PS3/360 Controller L3/R3.** Similar to the previous example, underneath the analog stick of these controllers is a button. If you press into the sticks you'll hear and feel a distinct click. Though these buttons are generally within reach, they're typically mapped to infrequent actions like crouching (Halo) or knife attacks (Call of Duty).
- **D-pad.** The direction pad and nearly all of its variants are actually a collection of at least 4 individual buttons resting under the plus shaped pad. The D-pad was designed to control movement. Because of this, the pad doesn't allow players to press buttons on opposite sides simultaneously. By rocking the pad around you can only press a maximum of two adjacent buttons at a time, thus allowing 8 total directions.
- **Arcade Stick.** Like the D-pad, the arcade stick is a clever construction of buttons. Instead of 4 sub-buttons like with the D-pad, some arcade sticks have 8 individual buttons, one for each cardinal and intercardinal direction. Instead of the diagonal directions being the combination

of two buttons the arcade stick clicks into the "diagonal chamber" distinctly and easily.

- **Guitar Hero Controller.** The strum bar and the colored buttons are all buttons with springy releases. Like the arcade stick, the strum bar has a distinct click feel helping players know exactly when the internal button is pressed. The Rock Band guitars do not feature this click.
- **PS3 Pressure Sensitive Buttons.** The 4 face buttons on the PS3 controller probably cannot be classified as buttons. Though they look and feel like normal buttons, instead of just an on and off state, these buttons respond to hundreds of different degrees of pressure. This makes them analog which means they're more like triggers than buttons. More on triggers in part 2.
- **Virtual Buttons.** All touch screen, on screen, or motion based buttons are not true buttons because they do not have the shape/physicality or the spring action.



Mechanics Examples

Let's look at how various developers have designed mechanics around buttons. This will showcase the versatility of the button as well as its limitations. I've explained above why the button is such a great interface device. The buttons is as black and white as you get. But while solely having an on and off state is ideal for doorbells, for anything more complicated the software side of mechanics design must compensate.

Pac-Man is about as simple as it gets. When you move the arcade stick , Pac-Man moves. When you release, he stops. From what I remember, the arcade stick is a 4-way stick for the cardinal directions. Likewise, Pac-Man can only move in those 4 directions. As long as Pac-Man lives, this interactivity remains consistent.

Mario is more complicated. Even looking at just the horizontal motion, there is momentum. Run right and then try to move left and Mario will skid, stop, and then move left. This design isn't too complicated. Instead of thinking that left/right on the D-pad makes Mario WALK left and right, we automatically make the adjustment mentally and think of the left/right buttons as Mario willing himself to move in a particular direction. Sometimes this makes Mario skid to a stop. Other times he can just start running. Still, the most important part about moving Mario is that players can do it at almost all times while Mario is playable whether in the air or on the ground.

Mario's JUMP mechanic is more complex still because the system looks to Mario's position to determine if Mario can JUMP or not. After all, how can Mario JUMP if he isn't standing on anything? It makes sense to us in our heads because we can tell the story and relate to virtual Mario using our real world experiences. But knowing this doesn't explain what happens when we press the JUMP button while Mario is in the air.

If you do press the JUMP button mid air, nothing happens. In fact, a lot of mechanics are designed to give the player no feedback when the mechanic doesn't activate. This is because of the expectation we have of buttons and mechanical/digital systems. Because electricity is so fast we commonly

expect immediate actions to follow a button press. Videogames are generally designed with a fast action-reaction combination as well. So, in an odd way the instant lack of feedback can work as a clear indication that the button did not work. For this reason games tend to keep their feedback design quick and clear.

It would be neat if the physical controller had a mechanism that prevented the JUMP button from being pressed when Mario was airborne. That way we would never have to make up an excuse/special case to explain why the JUMP button doesn't always make Mario JUMP when pressed. It may seem like I'm making a big deal over this minor limitation. However, understanding why this minute design detail is potentially troublesome is the key to understanding controller design. In other words, **controller design takes into account player expectation, input devices, gameplay/mechanics design, and gameplay feedback**.

Because developers sought to make increasingly complex gameplay interactions and mechanics button design has been thoroughly experimented with. On the complex end we have games like **Poto & Cabenga**. In the middle there's the game **One Button Bob**. And on the extremely simple end there's **Linear RPG** and **Super PSTW Action RPG**. Notice the different ways these games use taps, holds, and releases. With these designs along with button combinations (e.g. A + B), mechanics can be designed with a wide variety considering their limited vocabulary, so to speak.

In **Melee**, performing a short jump with Fox requires players to hit the jump button and release in about 3/60th of a second. The Street Fighter games are designed with a feature called "negative edge." Basically, when you press a button in this game your character can do an action. However, if you release the button your character can also do certain actions. So even if you think you're letting go of all the buttons and therefore turning all potential actions off, you may be turning them on. **Read more about the disconnect between player input and game output here**.

And it only gets more complicated from here. This is why I've always stressed **clean design** from the mechanics upward. When buttons are cleanly design, the added clarity of the gameplay feedback helps the player learn and relax. When the disconnect between input and action is large, players resort various technqiues like **button-mashing** or **double-tapping**. Basically, when in a stressed situation where there's no time to learn mechanical intricacies, players commonly mash buttons hoping that some actions do come out and that they're the right ones. Such players probably think it's better to survive blinding than to die learning. A similar thing happens when players don't know the exact timing of an action. They increase their chances by inputting the desired input (generally a button) multiple times quickly in succession so that if the first one is too early, the second one might do the trick.

To close, it's important to understand that as great as buttons are, they cannot do it all. The more complex a button based game is the more complex the software side of the mechanics design must be to compensate. The more complex mechanies are on the digital side, the more important feedback is to the player experience. In part 2 we'll look at analog sticks.

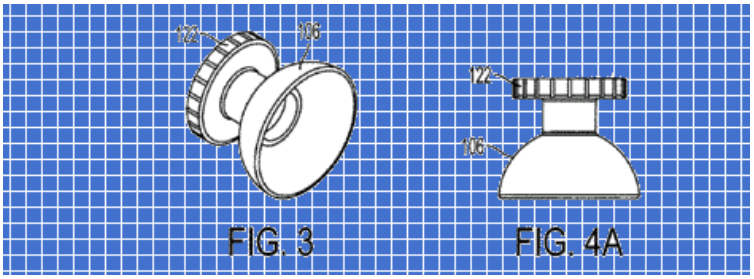
Controller Design: Analog Stick pt.2

Thursday, August 11, 2011 at 10:22PM

Richard Terrell (KirbyKid) in Controller Design

part 1.

After the advent of 3D graphics technology the industry as a whole began to transition into games with 3D environments. I've talked about 2D/3D game design before on this blog in a series called **2D + 2D = 3D**. It t urns out that 3D graphics are not that complicated. It's when 3D graphics and interactivity are combined that things become very complicated. Like never before, 3D games offered another whole dimension to explore. With the world open before us, we first sought a way to move through the 3D worlds intuitively. The 8-way control of the D-pad wasn't good enough. Thus analog control stick technology was embraced.



Sticking Around

We have to be clear about what we mean by "analog." Some stick to the true definition meaning a continuous range of values like the rainbow colors on a light spectrum. A more commonly accepted meaning is simply a wide range of values. How few values a device can have and still be considered analog is generally handled on a case by case basis.

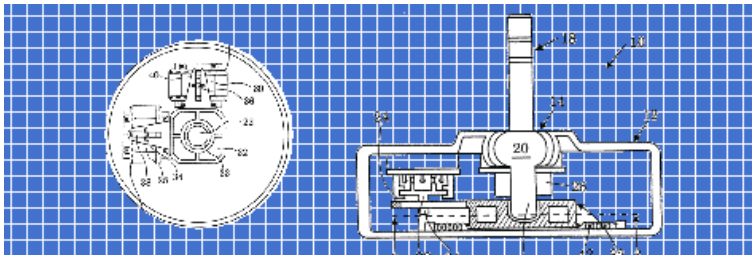
Take the Nintendo 64's controller for example. This device was the forerunner that popularize analog stick controls for video game consoles. With the analog stick and the flagship title Super Mario 64, players coudl stand, craw, tip-toe, walk, and run in seemingly any 2D direction (360 degrees) simply by moving the stick in different ways. Technically the N64 control stick is a clever construction of digital buttons like the D-pad. However, because it offers enough levels of sensitivity it functions as an analog stick for all practical purposes.

While the D-pad has 8 different directional possibilities total, an analog stick has thousands. And while buttons are one dimensional (either off or on), analog sticks are 2D. To qualify as an analog stick or analog pad there needs to be a springy recentering of the device, a high degree of variables, and two axes of movement. The following are a few unique examples:

-  **The N64, GameCube, Nunchuck, and Classic Controller** analog sticks all feature an octagonal 8-way rim (see image right). Before reaching the outer limit of the control stick's physical range, players can move the stick smoothly in circles. Press all the way to the outer edge and the stick will be guided into one of 8 directions. With this design you can distinctly feel the cardinal and intercardinal directions much like you would a D-pad.
- **The 3DS, Wii U, and PSP** controllers feature an analog sliding device. These pads have all of the basic functionality of an analog stick but feel different because they lie flat. Instead of pushing around a stick pivoting in a sphere like motion, players slide the pad around along a flat plane.
- **Pointing sticks (found on laptops)**. Don't worry. I didn't know the name for these little green or red nubs either. Though they respond to varying degrees of pressure to control the mouse position, these nubs are technically not analog sticks. They don't move enough from their center position and therefore they do not snap back either.

The pros of analog stick/pad controls are allowing highly variable, intuitive 2D movement. In the same way that controlling **3D actions in more intuitive with a 3D controller**, controlling 2D actions is more intuitive with a 2D analog stick. Unfortunately, because the analog stick is a more complex mechanical device more complex drawbacks emerge. While buttons are hard to break due to their small size, short movement range, and rubbery spring back mechanisms, analog sticks are somewhat fragile. The outcome we fear most when we accidentally sit on our game controllers is that we'll break the sticks. In addition to forcing the sticks out of wack, general wear and tear has a noticeable effect on analog sticks. Over time many sticks become much looser. I can't be the only gamer with a controller with drifting sticks.

When designing mechanics to use analog sticks, most developers know to factor in a **dead zone**. Because the analog stick is so sensitive, the slightest touch can send a signal to the system. But the slightest touch is too slight. To eliminate false positives from natural stick looseness or slight motions from the player, mechanics are designed not to activate until after the player pushes the stick outside of the dead zone. Another issue that's related to how the dead zone design is when the analog stick sends an unintended signal to the system from the snap back motion the stick makes when recentering. It can be confusing to get a reaction from the game when one lets go of the control stick in the same way that a **negative edge** button action can be confusing (explained in part 1).



Aside from the loosening of the stick and the erosion of the thumb grips, analog stick controls are very reliable input devices. To highlight its versatility here are a few examples of games that use the stick very well.

Super Monkey Ball 1 & 2. The core gameplay of Monkey Ball consists entirely of guiding a ball through a 3D obstacle course with only the analog stick. The first two games in the console series present challenges that require an extremely high level of sensitivity from the stick and control from the player (**see dexterity skills**). Additionally, the party games are designed to stress analog stick dexterity over many other facets of skill.

Super Smash Brothers Series. Unlike nearly every other fighting game, Smash Bros is a 2D fighter designed around analog movement. Instead of having to double tap to dash, in Smash players can simply smash (press quickly) on the stick. In addition to dashing players can crawl, tip-toe, walk, and run. In the air players can also control their horizontal movement to a fine degree even while attacking. This allows for highly variable spacing (aiming) which is the most evident in Brawl with its increase focus on aerial battles.

Geometry Wars Series. Some games are designed with mechanics that ignore parts of the analog stick's 2D range. In Geometry Wars (360/Wii Classic Controller), the right analog stick shoots based on the angle or direction you press. It doesn't matter how far you press in that direction, it only matters what direction. It's also important not to forget how the physicality of input devices shapes our expectations and therefore what feels right. In his article **Prototyping Game Feel v.2**, Steve Swink talks about the turn radius in Geometry Wars and how it was designed to match the radius and rotation speed of the Xbox 360 analog stick.

Everyday Shooter is an interesting case. I believe this shmup was originally designed for the PC using WASD controls to move around and the arrow keys to shoot. After Sony picked up the game for release on the Playstation Network, the controls were adapted to the PS3 controller. Though you can choose to use any combination of the D-pad or left stick to move and the right stick or the buttons to shoot, the mechanics remain the same. Essentially, the right stick is just a D-pad with 8 possible directions. Even if you try to aim to a finer degree, the ship will adjust the aim to fire in one of the 8 directions.

Skate. Though most designers use analog sticks for movement, aiming, or camera control, the developers of Skate decided to replace the more traditional button based actions of a skating game entirely with the right analog stick. The video below explains it well. Using the 2D range of the stick, command motions could be designed in ways that reflect the motions of actual skating. I wrote about this inward innovation and others **here**.

Gamers intuitively make connections and expectations between how we manipulate the controller and the gameplay mechanics (player actions) that follow. The more the input method matches the the virtual actions the more direct the particular mechanic. In other words, we get a sense that what we do to the controller directly affects what happens in the game. Mario's JUMP is designed with much directness. Tap the button and you get a small JUMP. Tap and hold and Mario JUMPs higher. I've known gamers who thought that the NES controller could actually sense how much harder the buttons are pressed to make Mario JUMP higher. We know now that the NES buttons are digital. With only an on and off state the system cannot tell how hard you press. Still, Mario's JUMP is expertly designed so that players make that intuitive connection anyway.

Expectations are similar for the analog stick. The harder you press in a direction the more effort you exert. So if you're exerting more effort the game actions should as well. This kind of thinking works great when designing movement mechanics for characters that can tip-toe, walk, and run based on the analog stick position. However, the design of FPS aiming is much trickier.

The bottom line is thumb based analog stick control doesn't allow the finest degree of control. Just try writing your name in cursive using a machine gun against a wall in your favorite console FPS to see what I mean. To help, most console FPSs are designed with some kind of sticky aim. Basically, the system knows, or assumes, what you're aiming at and adjusts the aim movement speeds to help you stay on target. When done well, most players don't even notice there's aim assist.

Super Mario Galaxy. The analog stick is a relative control device. Depending on the game state or how the character is oriented, pressing up on the stick may result in proceeding in any one of 360 directions. Designing good relativistic controls requires understanding how quality, feedback design and the other facets of mechanics design come together. Take the Mario Galaxy series. Never before has a game let players explore such a unique and 3D space that is frequently redefined by the gravitation pull of nearby objects. Even still, in space one's sense of direction is even more relative than ever. Running down the side of a planetoid is the same as running up it depending on how you look at it. A key point I must make here is "how you look at it" is largely the result of how the camera is designed. Galaxy's movement controls are built in with a wider turn radius than in Super Mario 64 and with a bit of smart-aim-assist like functionality. As long as you don't let go of the stick and let it recenter, you can push the system far enough that holding down makes Mario move up. As backward as it sounds, it feels completely natural.

To close, analog sticks work best for controlling 2D continuous actions like moving

To close, analog sticks work best for controlling 2D continuous actions like moving. This is why few gamers spam motions into the control stick. Unlike buttons which tend to activate non continuous, primary mechanics, spamming the stick will most likely make the on screen character shiver shake in place. So if you want to reduce spammy tendencies, go with sticks. In part 3 we'll look at touch screens and triggers.

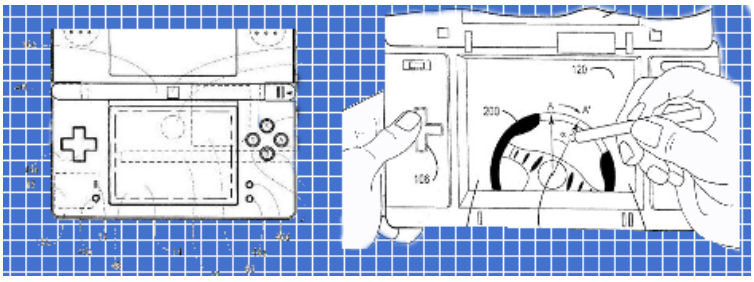
Controller Design: Touch Screen pt.3

Friday, August 12, 2011 at 9:24PM

Richard Terrell (KirbyKid) in Controller Design

part 1. part 2.

It may seem odd now to use a handheld gaming device that doesn't have a touch screen. The 3DS, Playstation Vita, the iPhone/iPad, and all droid phones now feature touch displays. Yet when Nintendo first announced the DS with its odd dual-screen-single-touch design the gaming industry had trouble seeing the point. It's been over six years since the launch of the DS and the legitimacy of the touch screen has been proven.



Reach Out And Touch

The touch screen is an analog 2D input and display combination where the visuals on the screen can have a direct, one-to-one relationship with the touch sensitive areas. As long as you have screen to touch you have potential interactivity. Before I go into detail, there are three types of touch controls that I want to focus on; **touch pads, touch screens,** and **multi-touch screens**.

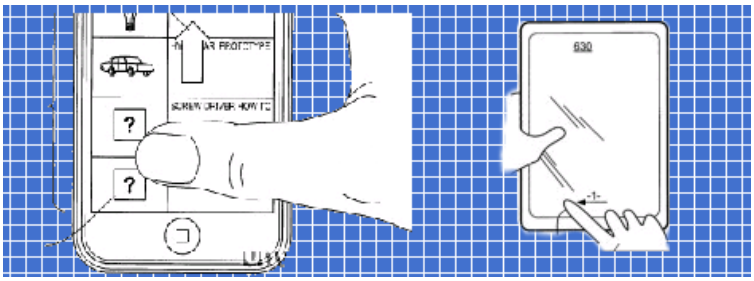
Touch pads are touch sensitive areas that have no display. The degree of sensitivity varies. Sometimes styli are used. Other times, just fingers. The Guitar Hero 4 guitar controller features a touch pad. With it players could play by pressing on the sections like buttons or by sliding their fingers up and down. Most laptops feature a touch pad to control the pointer. And the Playstation Vita will innovate by featuring a rear touch pad that is the same size as the screen display on front.

Touch screens are displays that are sensitive to touch. In this case the screens are pressure sensitive so a stylus or a finger will work. However, the resistive screen technology is limited to accurately tracking one touch input at a time. The Nintendo DS, 3DS, and upcoming Wii U all feature touch screen with stylus control. Using the pencil like a stylus draws on one's fine motor skills from writing or drawing. With pixel fidelity stylus control uses the muscles of multiple fingers, your wrist, and your arm. Such controls would be quite a dexterity barrier if we weren't already trained. There is, without a doubt, no input device that gives players a higher level of 2D analog control and precision. One downside to stylus control is how the player's hand and stylus can obscure the view of the screen.

Popularized by the success of the iPhone, capacitive touch screens are commonly referred to as **multi-touch screens**. Capacitive touch screens react to anything that conducts electricity. So a plastic stylus doesn't work on these screens. Instead, special capacitive styli can be used though they

are not widespread. With the norm being finger tip interactions a few issues emerge. A finger tends to obscure much more of the screen than a stylus. The finger has less fine control than a stylus. And it's more difficult to tell where exactly a finger touches the screen because of how large and soft it is. Though the multi-touch technology allows for more complex actions like pinch zoom, more fingers on the screen obscures even more of the view.

Regardless of the type of touch screen, the biggest limitation developers face is dealing with the lack of tactile feedback. Though one can feel when a stylus or a finger makes contact with the touch screen, when interacting with virtual elements there is no tactile differentiation if you miss. It all feels the same. Fortunately, good audio and visual feedback can help make up for the lacking tactile feedback. When interacting with or manipulating a touch screen element the player tends to focus on the visual object. After this point if the object moves or the context changes the player can react accordingly. It's when players are encouraged or required to look at a location other than where they are touching that the chances of missing the target increase. With traditional gaming controllers players quickly learn to feel their away around the controller so they can focus on the game screen. The more complex the game and inputs (like multi-touch) the more players will have to focus on inputting. Yet, if the virtual buttons are in the same place relative to the screen, muscle memory can help.

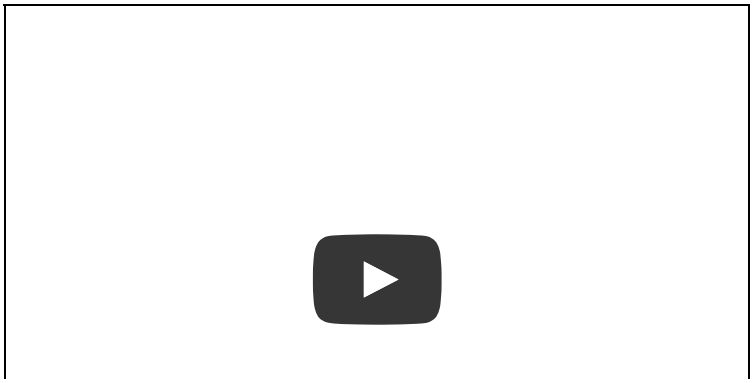


Regardless of input device limitations we can always design smarter. Design (the software part) is the most flexible part of game design. A few years ago I wrote a post on the great design features for the DS that utilize the touch screen. I suggest taking a look **here**. The following are a few examples I want to highlight.

- **Kirby Canvas Curse.** While Yoshi Touch & Go introduced all-touch-screen, indirect-control design, Kirby Canvas Curse perfected it (and at the beginning of the DS's life too). You can tap Kirby, obstacles, switches, and enemies to help you progress. To effectively move around players must draw rainbow pathways that have a built in treadmill effect. Depending on the curve of your lines you can make Kirby bounce off, crawl vertically, speed boost, or even travel upside down. This kind of gameplay is only possible with the fidelity, accuracy, and directness of touch screen controls.
- **Zelda Phantom Hourglass and Spirit Tracks.**These games (both on my GOTY lists) can be played entirely with the touch screen. Just with contextual touching players can read a sign, move Link, or attack an enemy. With a tap of the virtual button, Link can change weapons and prepare to use them. While at the ready, Link can aim but not move (**stop and pop makes for a cleaner game**). This design simplifies the gameplay, allows players to plant Link at a spot to focus on other elements, and keeps the controls from overlapping. Such a design is a perfect way to work around the limitations of touch. These controls held up well through the scrutiny of multiplayer competition too (**read about the battle mode here**)!

of multiplayer competition too (**Read about the battle mode here**).

- **Rhythm Heaven.** With this near flawless music-rhythm game players hold their DS book style. To avoid the lack of tactile feedback issue for virtual buttons, the developers made the entire touch screen a single button. Not only does this design make inputting hard to mess up, but it also utilizes the edges of the screen as a tactile barrier. With this game you can feel the edges of the touch screen to help you stay on target even when you're not looking. The visual feedback on the touch screen also clear indication where you tap, where you hold, and in what direction you flick.
- **Metroid Prime Hunters & Geometry Wars: Galaxies.** These games offer a variety of control schemes. If you prefer the direct interactivity of touching exactly where and what you want to shoot, you can set the game view on the touch screen and tap away. If you'd rather not obscure your view, you can move the game view to the top screen and use the touch screen like a virtual analog stick or mouse. Personally, I prefer the latter option. In fact, I played Metroid competitively online using the thumb stylus. This small piece of plastic was fashioned over your thumb so you could slide it around the touch screen much in the same way you would an analog stick.
- **Ninja Gaiden: Dragon Sword.** Relativistic controls can become difficult to manage in games with frequently changing camera angles. This game solves the problem with by focusing on direct touch screen controls. It doesn't matter if you spot an enemy deep into the background, to the side of you, or closer to the camera. Just tap on them and Ryu Hayabusa will throw a shuriken at the target. To move Ryu through side scrolling, isometric, or overhead environments, just tap and hold where you want him to go and he'll path find there. Attack combos are also context senstive to the enemy, Ryu's position, and how the player swipes with the stylus.
- **Eliss.** Multi-touch can become quite complicated. Technically, like Street Fighter commands, we can design multi-touch commands to require specific multi-finger motions and timings. Generally, this is not a good idea. Even a pianist/violinist/artist like myself doesn't have too much simultaneous finger dexterity (using one hand). Keeping it simple and intuitive works best so far and Eliss does just that. Simply use your fingers to pull part or push together the colored circles to decrease or increase their size respectively (see video below).



Triggers Down

The trigger is basically an analog button. Or perhaps it's like a button and a lever in one. Still one dimensional, a trigger has a much wider range than just on and off. To clarify, a trigger must have a springy push back (the button like quality) and a wide range of sensitivity. While most console controllers have triggers the Wii Nunchuck and Wii U controller do not. More examples include the Wii Balance Board, which is made up of 4 triggers like a D-pad for your feet. The Guitar Hero/Rock Band drum pedals are also triggers. Similar examples that can't quite be categorized as triggers are the DJ Hero controller turn table, the Arcanoid paddel, some racing wheels, and the mouse wheel.

The hardest part about designing mechanics that utilize the analog trigger is coming up with fitting ideas. As I've explained, buttons are great for turning things on and off (binary 1D). And because countless mechanical and electronic devices are activated by buttons, in video games buttons cover a wide array of creative possibilities. Analog sticks are great for controlling actions like motion especially in 2D. Simply point in the direction you want to move and the game actions generally follow suit. But how many things in real life operate by a pedal (analog 1D)? The easy answer and possibly the only answer are motor vehicles. The bottom line is that a trigger is a specialized input device best used for controlling variable continuous actions that deactivate once you release the pressure. There aren't many functions and devices that need this.

After racking my brain and reading through a few NeoGaf threads, I couldn't come up with many mechanics that work best (or even work well) with an analog trigger outside of racing/acceleration mechanics. Here are the examples I found.

- Racing games use it for acceleration and break pedals (Forza. GT. GTA. Burnout. etc.).
- **Grand Theft Auto 4**. Apparently you can switch between free aiming and lock on based on how far you press and hold the trigger.
- **Super Mario Sunshine**, **Luigi's Mansion**, and **Super Smash Brothers Melee** each used the trigger to control the strength of the F.L.U.D.D. (water gun), Poltergeist 3000 (vacuum), and shields respectively.
- **F-Zero GX**: Players can side strafe at different speeds using the analog triggers.
- In **Halo 3**, the Spartan Laser requires a charge before firing. Players can control the length of charge up time with the analog trigger.

I believe most developers simply design trigger mechanics as if triggers are buttons. With such mechanics, developers have to be mindful of the "deadzone" design like analog stick mechanics. For me, the actions mapped to the triggers in Street Fighter 4 for the PS3 are too sensitive. And for Brawl, because light shielding was removed, players have to press all the way to the button at the bottom of the triggers to activate a shield or air dodge. This means when I hit the trigger like a button, my timing gets off. Adjusting to mismatches like these between software and hardware design takes a bit of time. After all, the mismatch exist because of a clash between long established learned and intuited behavior

learned and intuited behavior.

In part 4 we'll cover pointing devices from mice to Wiimotes. We'll also tackle microphones as inpute devices.

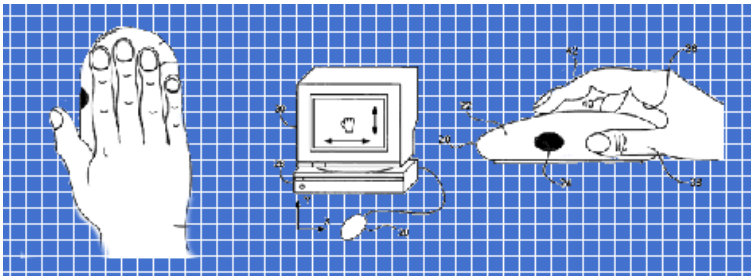
Controller Design: Pointers pt.4

Sunday, August 14, 2011 at 2:22PM

Richard Terrell (KirbyKid) in Controller Design

part 1. **part 2**. **part 3**.

Personal computers wouldn't be the same without the mouse. Instead of navigating via key strokes, with a mouse users can simply point to the on screen interface to indicate which actions to take. It's hard to imagine using a PC without a mouse, especially for PC games. The mouse's unique features put it into a class of few. It along with the keyboard is the main reason why there is such a large disparity between the types of games that are made for PCs versus consoles. But the mouse isn't the only pointing device common to gaming.



.

That's the Point

A pointing device either directly or indirectly uses light or other mechanisms to relay positional coordinates to a cursor via 2D or 3D movements. The cursor movement is always proportional to real life movements. Also, there is no springy action or snap back with pointers. The modern mouse is a 2D pointing device with high sensitivity. Ignoring its buttons and other features, the mouse is controlled by multiple fingers, the wrist, and the rest of the arm. With so many muscles and joints involved mouse control is great at tracing straight lines in various directions, tracing circles, making very slight movements, and stopping dead still on a target. The larger the movement the more ease one has with mouse control due to the increased number of muscles involved. Conversely, very small complex movements are not as easy to be precise. In other words, pointers are great at pointing and moving, but not great at fine control like with handwriting or stylus control. Test this out for yourself by playing **Dexterity, a skill testing game by B.E.S**.

The mouse is also a relative pointing device. As long as you have a large enough flat surface (and a long enough cord/connection) you can pick up and reposition the mouse anywhere. It doesn't matter where you place the mouse, all of the pointing is relative to the changes in mouse position. To avoid having to pick up and replace the mouse for larger movements many gamers have adjusted to a high cursor speeds so that moving less in real life moves more on the screen. At the same time these limitations increase the focus players need to keep on the cursor like working with some touch screens.

Pointers don't stop with 2D. Some devices can point in 3D. But to cover these less common devices, I might as well describe them in the following examples.

- **Wiimote**. This device is the most mainstream 3D pointing device ever created. With the

- **Wiimote.** This device is the most mainstream 3D pointing device ever created. With the sensor bar set either above or below the TV, pointing is more direct than with a mouse. In other words, it is important where you hold and point the Wiimote in real space. With this design you can use some muscle memory to increase accuracy and precision. While pointing at the TV the system can not only tell your 2D screen position and 3D depth (closeness to the TV), but it can also tell the angle you point at the TV. Though the pointing is accurate, holding the Wiimote in place tends to cause the aim or cursor to tremble. This issue can be smoothed on the software side. Also, up to 4 Wiimotes can be synced to the same Wii.
- **Playstation Move.** Using the Playstation Vision Cam and the Move controllers the system works as a 3D pointer much like the Wiimote technology.
- **Novint Falcon (see video).** This device is a futuristic piece of hardware. First of all, it's a 3D pointing device. Secondly, using haptic feedback the system can simulate different textures, object densities, and directional forces. For example, if you get shot in an FPS you'll actually feel the controller jerk in the appropriate direction.
- **DS Side Controller (see video)**. Never released outside of Japan, this controller is basically a light mouse for the DS.
- **Kinect.** With vision of the space in front of it, the Kinect camera can sense the players hand or any other object and track it as a pointer. The trouble with controller free gaming is that you lose access to some of the most intuitive and effective input devices like buttons and analog sticks.

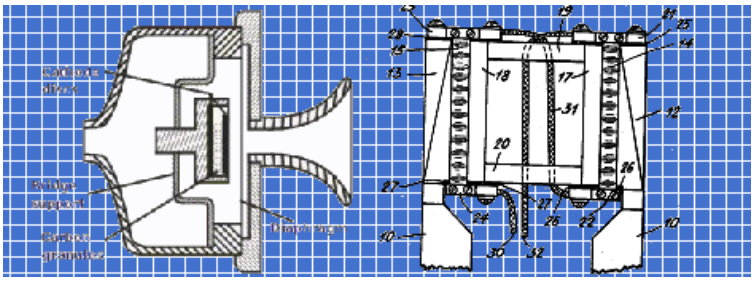
The precision and accuracy of the mouse is great, but the simplicity of control is the key. No matter how you move the mouse, the cursor moves at the same rate that you move it. In other words, you never have to calibrate for the increasing speeds that analog stick movement tends to have. With the mouse and increasingly higher resolution computer monitors many, if not most, PC game developers design their UI (user interface) with tiny text and small virtual buttons. There much more to discuss about PC game design and feel, but this is not the place.

The mouse pointer can do little without mouse buttons. For the most part the Kinect has to work around limitation of controller free interaction. **Dontclick.it** web developers chose to design a website that can be navigated completely without clicking the mouse. This site is stunning, stylish, informative, and entertaining topping the chart of my favorite websites of all time. Experience it for yourself to understand just how the UI and virtual buttons had to be redesigned to compensate.

Input devices become extensions of our hands and our minds. According to MIT research we think with our hands, eyes, arms, and anything else we can manipulate. So while we're holding a controller, it's possible to think through it. When we hold a mouse, we often unconsciously move it to where our minds are focused. Moving the mouse, seeing the cursor, and moving the mouse again creates cognitive positive feedback loops. I'll cover this in greater detail soon on this blog.

Microphone

Voice control. This idea is nothing new. Scifi shows have featured characters conversing with their computers for a long time. I remember when voice recognition technology was spotty at best. This coupled with poor quality microphones didn't make a winning combination. Now, I can can speak just about anything into my phone like "search: jigglypuff serebii.net" and the system recognizes my speech perfectly. To reiterate, voice recognition technology finally works. As far as gaming goes, this technology is slowly gaining steam. First, I'll address the pros and cons of microphone input.



The microphone is a device that converts the sound waves (mechanical waves) into a digital signal. This means all the different ways to alter sound become distinct input variables. This includes frequency, tone, and volume. Even before speech/language recognition comes into play, the mic can function like a hands free button or trigger. Just program a mechanic to react to the loudness of incoming sound and the game can react in an 1D digital or analog fashion. On the complex side, mics can use speech recognition to process language. The range of sensitivity, specificity, and versatility of language is unmatched by any other input method. For a comparison, speech recognition is like the complex move command in Street Fighter times a billion. The best part of all is that we already understand these billions of commands because we've internatized and continually practice using our native languages.

On the downside, mics can pick up ambient noise that you may not be able to control. Noise filtering software and other technologies exists to help focus mics on the sounds intended to be picked up. While voice commands can replace buttons for simpler, non-timed commands, and replace analog triggers for simpler action, mics are generally not suited for these functions. Voice commands work the best with increasingly large numbers of specifics. What I mean by this is that thinking and saying words is the fastest, most intuitive method of identification when there are lots of items to choose from. Think of how simple it is to type in a question into google search. Any term. Any word. Any category. And you get results quickly. Voice commands allow for this kind of precision searching and indentification completley hands free. Just think if you could search your Pokedex for any Pokemon, item, ability, or attack just by speaking. This would be very convenient. Instead we have to push buttons, tap the screen, scroll up and down, and then click again to get to the content we want.

The following are examples that use a microphone. Be sure to check out this article I wrote on **DS microphone design** for more examples.

- **Donkey Kong Jungle Beat.** One of last generations best platformers is controlled via a plastic bongo set. Built into the side of this unusual controller is a small microphone designed like a button to respond to loud sounds. Clapping your hands together is just the type of sound the system was designed to respond to. When you clap, Doneky Kong claps. There are even moments in the game where it's important not to clap. At these times, any loud noise in the room can set the game off.
- **Exclamation Warriors (see video).** This quirky game never made it state side. Use voice commands to punch, transform, and other co-op combinations.
- **GNILLEY.** Scream to survive in this indie game. Noice how the screaming acts as a analog trigger. **See video here**. **Download it here**.
- **Tom Clancy's End War.** This console RTS features optional voice command controls (**see video**). With a list of about 50 voice commands players can control all of the action. Couple

this system with some controller input and the system becomes very accurate and versatile. It also helps that the voice commands have clear visual feedback (**see video**).

- **Brain Age 1 & 2.** These games feature voice recognition with colors, Rock Paper Scissor hands, and the secret phrase "glasses glasses." (**see video**)
- **Kinect.** The DS has a built in mic and uses it often enough. The Wii U controller has a mic as well and will support video chat. But it's the Kinect that's currently pushing voice commands in gaming the hardest. **Menu navigation**, **weapon custimization**, and **ally commands** are all coming.

In part 5 I'll cover motion controls and camera based input systems.

Controller Design: Motion pt.5

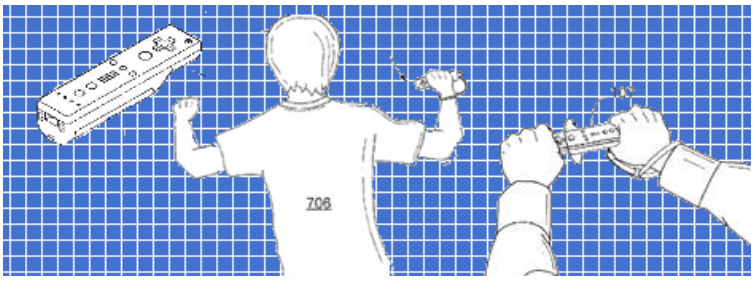
Thursday, August 18, 2011 at 8:25PM
Richard Terrell (KirbyKid) in Controller Design

part 1. part 2. part 3. part 4.

Since the first video games gameplay elements and actions were inspired and modeled after real life. Despite layers of abstractions and processing limitations we used whatever input devices available to bridge the gap between the virtual world and ours. To help in this process we designed mechanics to be intuitive so that our lived experiences would help us better understand the game. Now, over three decades later, the industry now supports two controller technologies that allow players to move and play intuitively; motion controls and camera based controls.

Play Like A Pro

Motion control is a term that applies to devices that use at least one of the following technologies designed to sense motion and position: linear accelerometers, gyroscopic sensors, rotation sensors, rate sensors, and magnetometers. To keep things focused, I won't go into the technical side of these parts.



Both the Wiimote+ (Wii remote with Wii motion plus functionality) and the Playstation Move controller work with their pointer features to most accurately calculate where the controller is in and how it moves through 3D space. Without the pointer technology, both devices are a combination of about 7 sensors or so of various types. In the same way that the D-pad is a combination of 4 digital buttons, the Wiimote+ is a combination of several sensors set in a specific formation to sense motion in any direction. With controllers relaying information back to the system faster than 1/60th of a second there is a high level of sensitivity and precision possible from motion sensing controllers.

Yes, the Wii and the Playstation Move are the most obvious examples, but motion controls have had a much longer history especially with handheld gaming.

- **Kirby Tilt 'n' Tumble (see video).**Using a series of accelerometers, tilting the Gameboy tilts the game world maneuvering the ball like Kirby. Popping the Gameboy upward caused Kirby to jump. Before playing the game needed to be calibrated on a flat surface.
- **WarioWare Twisted (see video).**The WarioWare games are expertly designed. The entire concept of micro games that **accelerate and challenge**the player with **mixups** is solid. But

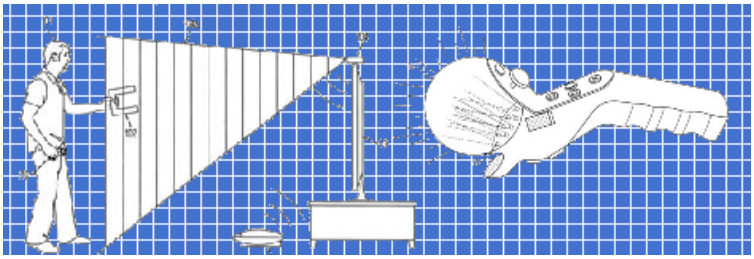
  beyond this, each game in the series is designed to take advantage of a different type of technology. With Twisted, the game features a tilt pack powered by a gyroscope designed to detect angular movement. Calibration is done after starting up the game and more subtly after

each microgame.

- **Wii.**Holding a TV remote like motion controller frees the hand and range of motion in ways that were not possible on the Nintendo handhelds mentioned above. WarioWare Smooth Moves showcases the many stances and motions the Wiimote can support without the motion plus. Wii Sports Resort is a great showcase of the motion plus technology.
- **iPhone.**Like the Wii, the original iphone initially only featured 3 accelerometers to sense tilt. Gyroscope technology was added in later models. Switching between portrait and landscape is a popular function for non-gaming applications while the sensors are mainly used for tilting in games.
- **PlayStation Sixaxis & Move.**In reaction to the announcement of the Wii controller, Sony supposedly introduced motion controls into their controller. This technology was implemented into a games to varying degrees with varied success (e.g. Warhawk, Lair, Uncharted, Flower, Echochrome). In reaction to the Wii's success Sony developed their own brand of pointer-motion control. Though the tech is a bit different than the Wii's, the over all function is mostly the same.
- **3DS/Wii U/Vita.**The next generation of Nintendo and Sony's gaming devices all feature motion control technology. Interestingly, all three of these devices are handhelds featuring buttons, a touch screen, speakers, and an independent power supply.

The great strength of motion controls doubles as its greatest drawback. It doesn't get much more intuitive than performing an action in a video game exactly like you would in real life (assuming you have experience with the action). Likewise, controlling a 3D action in a video game can be the most intuitive using a **controller capable of sensing 3D space or forces**. Instead of breaking apart or abstracting the action so that it's controlled with a set of buttons, triggers, or sticks controlling pitch, yaw, speed, elevation, etc., a 3D controller can control the action in one, complete, and possibly buttonless way.

The downside is that tapping into the wealth of accumulated knowledge via intuitive controller design also runs a greater risk of tapping into stronger expectations. If you hand someone a Wiimote and tell them to swing it like a sword, chances are they'll swing it like a cartoon sword. With no sense of weight, momentum, or clashing physical bodies a disconnect between player input and game output can emerge. It's not like this issue is unique to motion controls. As I said with buttons, it's common to hit buttons or use any other input device and get zero reaction/feedback because of what's happening in the game. The only difference is, with motion controls user often blames the controller. The benefit of abstracting actions and mapping them to buttons and the like is that a unique bridge is created between the user and the game where the user suspends disbelief and calibrates expectation. Such is the nature of dealing with abstractions from video games to story telling. This natural calibration process may not go over so smoothly if the player is convinced that they can do the real life action well (even if they cannot). Instead of meeting the game halfway, such player make excuses and grow frustrated.



Motion controls also have to work with limitations that are similar to touch screen, mouse, and microphone controls. Like touch screen interactions, with motion controls there is no physical limit or tactile feedback for your motions. This means knowing where to start and stop is a matter of understanding the motion/mechanic you perform. This is inherently more complex. Like a mouse, there doesn't have to be a re-centering of the motion device, though many games have certainly developed such features on the software side. Because motions are relative to the device, you may have to work at pre-positioning the device to best prepare for upcoming actions. If a game doesn't feature any kind of motion sensitivity calibration, there's no way to avoid the "pick up and place" issue PC gamers have avoided with high mouse sensitivity. Depending on the game, being out of position can really clash withplayer expectations. And like the microphone there is no clear or common "off" state for motion controls. This can be troublesome when players try to re-center or reposition the device and inadvertently perform another actions. This issue is very similar to buttons designed with a negative edge discussed in part 1.

Furthermore, there are many different ways the system can use the data from the controller. At any point during the input window, the system can look at any individual sensor or a combination of the controller angle in 3D space, direction of motion in 3D space, and speed. Because a motion made by the player can simply serve as a digital button or a fully analog input along 3 axis, knowledge of how the system works is a crucial factor in understanding how player expectation shapes their experience. And if the system isn't calibrated property potential issues increase.

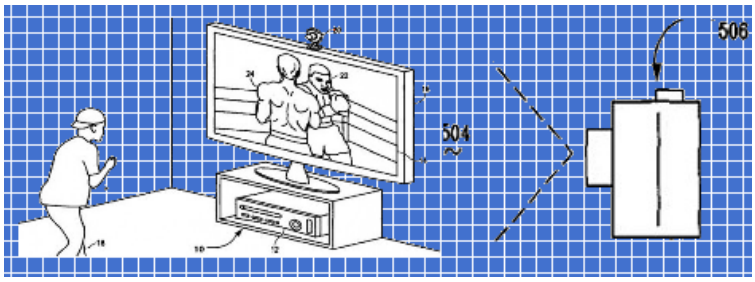
The good news is that motion controls can be more intuitive, engaging, and analog than other input devices. With 3 independent analog axes controllers can be tilted and moved in any direction. The sensitivity and range is great, and motion controls can sit invisibly behind other controller types and functions. For example, in New Super Mario Bros Wii players play with buttons and a D-pad like the original game. However, there are a few elements in the game that are controlled via Wiimote tilt. At these moments, there's no need to switch controller grips or reposition your hand. The motion controls simple kick in and you're already doing it. Combining traditional and motion controls together is a powerful combination. Notice how none of the devices I listed above exclusively feature motion controls. In other words, while playing with buttons, sticks, or even touch screens, motion controls can be at the ready to enhance your experience.

The following are well design and otherwise notable examples of motion controls.

- **Shake to replace a button.** Many have scoffed at motion controls that essentially swap a button press for a motion. I see nothing wrong with this kind of design. In some cases a button might be better, but not all cases. In the simplest cases, the motion-button is more engaging. Twilight Princess sword swing. **Wario Ware Smooth Moves actions**. **Wario Land Shake It actions**. Super Mario Galaxy 1 & 2 spin punch. Pixel Junk Eden dive attack. Mario Strikers Charged tackle. Donkey Kong Country Returns's somewhat controversial shake-to-roll design is actually better than a button press. More on this later.
- **Analog tilt replacing analog sticks.** By using more muscles than those in your thumb, 2D analog tilt controls are generally easier to be more precise. **Monkey Ball Banana Blitz general controls**. **Super Mario Galaxy 1&2 mini game type levels**. Excite Truck, Mario Kart Wii, and Flower for steering.
- **Angle detection.** These games use the sensors to detect the angle the controller is held at instead of sensing motion. Examples include **Samba De Amigo** poses. In Guitar Hero activate star power by tilting the guitar. To charge missiles in Metroid: Other M hold the Wiimote

straight up then hold the A button.

- **3D analog motions.** When I first realized that I had full 3D control over how I returned volleys in tennis, I was blown away. The height, the angel of the hit, the power behind it, and the top-side-back spin were all the result of how I swung the Wiimote. This design is far more advanced than most understand or give the game credit for. I had to buckle down and really focus on my technique to earn a platinum metal in the training mode. And this technology only got more precise with the Wii Motion Plus and Wii Sports Resort.



Camera

Similar to motion controls, almost every current generation and upcoming gaming device features a built in camera or some kind of camera peripheral. Taking pictures and video chat are common features, but few games use the camera as a control input device. For the final piece of input technology I'll cover in this series, we're looking at camera. This won't take very long.

A camera input device is sensitive to light. By analyzing pictures or video, the software is able to recognize patterns and track movement, shapes, colors, and even depth with infrared technology. Since just about every gaming device supports a camera and they all work about the same, I'll skip right to the pros and cons.

The pros include the potential for controller free gaming. Keep in mind that, like microphone controls, cameras can tap into the vast range of body language and body motions. Also it's possible to scan in objects for the camera to track. Augmented reality is a type of interactive experience where the camera sees into the world, displays what it sees on screen, and alters the image in some way.

Unfortunately, the cons are many for camera control. Real-time image processing and augmentation is much more processor intensive than any other input device. Think of this as the difference in the information pipeline between streaming a video off the internet and flicking on a light switch. Other issues include lighting. If there's too much, too little, or specific kinds of light in the environment (e.g., sunlight) some camera technologies falter. Also, camera resolution is important. The higher the resolution of the incoming image the clearer a picture the system has to work with (at the expense of increased processor consumption).

The following are a few examples of camera control.

- **Playstation Eyetoy.**
- **Wario Ware: Snapped.** This DSiWare game tracks your head and hands to play microgames.
- **Ubisoft Your Shape.**
- **Looksley's Line Up.** This game uses face tracking technology to create the illusion of looking into a window into a game world. 2D, button free, analog controls are possible using

looking into a window into a game world. 2D, button free, analog controls are possible using camera technology like this.

- **3DS & Vita ARGs.** With software designed to recognize specific symbols, the camera is used to modify images of reality. Continually updating the images also turns the handheld into a 3D controller. Move up, down, sideward, or closer and the visual update accordingly. Check out **Ghostwire for a example of an immersive, scary experience**.
- **Playstation Eye.**
- **Microsoft Kinect.** Now you and a friend can play together controller free. Move your arms, your legs, or any other part of your body in the 3D space in front of the device to play. Kinect can even recognize you by how you look. There are many games in development for kinect.

In part 6 we'll examine how ergonomics and expections are to controller design as we beging to wrap things up and put it all together.

Controller Design: Terminology pt.6

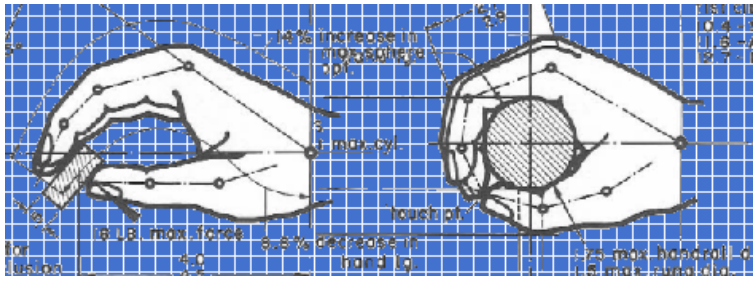
Sunday, August 21, 2011 at 3:17PM

Richard Terrell (KirbyKid) in Controller Design, LittleBigPlanet, Mechanics, Mega Man, Pokemon, Super Mario Bros.

part 1. **part 2**. **part 3**. **part 4**. **part 5**.

Controller design is all about ergonomics, a broad concept that includes the design of inter-working mechanisms, human anatomy, and psychology. To put things into perspective, controller design is a subset of mechanics design. Over three years ago I created a framework to break down and discuss gameplay mechanics (player actions). The four categories I create that seemed to cover all the key aspects of a mechanic are directness, dynamics, intuitiveness, and individuality. It's time to evaluate this system with our new understanding of controller design to see if it needs to be updated.

> *"Ergonomics is the study of designing equipment and devices that fit the human body, its movements, and its cognitive abilities." ~Wikipedia*



In all of my analysis of various controller input technologies, I frequently talked about the muscles and the methods gamers would typically use to manipulate the device. There really is little point in discussing a gaming controller without seriously considering how the player will use it. Will this grip make the analog stick easier to move or will it cut into the player's finger? Can the player even reach this button comfortably? Quickly? How will the player think this device will work before using it? Will these thoughts interfere with the player's expectations? Simple questions like these show that controller design is part mechanical and part psychological.

The psychology of game design is a rich subject that few tackle. Check out the **Psychology of Games blog** by Jamie Madigan for well written and researched articles on the topic. On a broad level, players use their experiences in life (especially with other video games) to shape their expectations. Without even considering what is good or bad game design, a player may like or dislike any particular feature based on their expectations. If you've ever instantly liked a sound effect or song because it sounds similar to one from a favorite game, you know what I'm talking about. In some cases, simply being familiar with another game in the same genre can result in the player having a very difficult time adjusting. And most of this process is unconscious. The player might not consciously understand that the reason they missed so many shots in a new FPS is because of a difference in the auto aim design. All they do know is that their shot is off, they're not winning, and therefore the game doesn't feel right. Because these expectations are made up of countless observations, experiences, and other factors (technical, environmental, personal, etc.) it is impossible to precisely design around the specific expectations of multiple gamers.

On a very close level, our expectations tint and shape our very perception before we consciously

realize it. This means that all of our **trial-and-error experiments** we conduct to learn a game system have biased inceptions and biased results. Like with any experimental process, we can minimize the effect our biases have if we're careful. The problem is, most gamers looking for low work, low stress entertainment simply do not reach this metacognitive level of addressing the faults in their own thinking. **Because most play games by feel, it is that much more important for designers and Critical-Gamers to be aware of how player expectations, ergonomics, and controller design affect gameplay mechanics, the foundation of our interactive medium.**

To best understand how expectations (ergonomics) impact controller design, let's buckle down, establish clear terms, and look at cases in a technical manner. Throughout this article series I've been careful how I used the word "precise." It along with tight, smooth, responsive, and accurate have been used in a very vague or careless way in our community. Like **"mechanics," a word that I use very specifically**, these words will be the most useful only if we stick to very distinct and clear definitions. Fortunately, the scientific world has used some of these terms for measurements for a long time. If their definitions are good enough to launch a man into space, they're good enough for our purposes.

- **Accuracy:** the degree of closeness of an outcome to the declared target. Accuracy is only useful for comparing the ranges of input of different devices of the same type of input. e.g. the value range of pressing all the way up on the analog stick of a 360 controller versus a PS3 controller.
- **Precision:** the degree to which repeated attempts under unchanged conditions produce the same outcome. (also reproducibility / repeatability).
- **Responsiveness:** The time it takes for the system to respond (an expected output) after inputting.
- **Sensitivity:** the smallest change in the input method that produces a response in the system.
- **Bias:** the difference between the mean of the outcomes and the target value. Only relevant when discussing accuracy.
- **Error:** random variability.

So let's look at Mario JUMP as an example. It's precise; the button based input makes repeating attempts easy. It's responsive; Mario springs off the ground so quickly that the button press and the result are practically simultaneous. In terms of sensitivity, Mario's shortest JUMP is **2 bricks high**. His highest JUMP is 5 bricks after getting a running start. In between these extremes Mario can hit a range of heights. Because the game and the input device is so simple, there's little error to consider. In other words, it's hard not to hit the JUMP button correctly and consistently. In conclusion, Mario's JUMP is a very solid mechanic (both input and function).

Mega Man's JUMP is just as solidly designed as Mario's except it's more sensitive. Mega Man does not have a JUMP height minimum. If you tap the button for only a few hundredths of a second, Mega Man will rise for that time and fall immediately when you release. Mega Man does not softly float at the top of his JUMP arc either. This design matches well when mapped to a button. When pressed (on) Mega Man goes up. When release (off) Mega Man falls down immediately.

In LittleBigPlanet, Sackboy and Sackgirl's JUMP mechanic is somewhat decisive. Some deal with it while others cannot stand it. This is a perfect opportunity to explain the mechanic with our new terminology. Because LBP is designed around user generated content and the ability to modify game elements in various ways, LBP mechanics were designed with more complexities to compensate for these possibilities. Where Mario and Mega Man only JUMP off of flat surfaces in their NES style games, in LBP a continuous range of angles are possible. This is inherently a more difficult challenge to design for. Media Molecule's solution was to make everything physics based. To

clarify, the LBP system factors in the 2D vector momentum and other calculations to determine how the Sackpeople will JUMP. So, while Mega Man has no momentum, and Mario mainly has horizontal momentum, Sackpeople have 2D 360 degree momentum. Because the system is more complex, sometimes my Sackboy doesn't JUMP to the expected maximum height even when I hold the button down. Because the physics calculations vary slightly (error), when I think I'm inputting the same way Sackboy reacts differently. This makes the mechanic seem imprecise.

Because Sackboy doesn't have a minimum JUMP height, like Mega Man, and ascends slowly to various heights due to the imprecision, the LBP JUMP mechanic can seem quite unresponsive. Because of these results and how Sackboy doesn't JUMP very high (compared to his height) it's

more difficult to see the range of sensitivity the JUMP. All of these problems stem from the physics based variability or error. Even if these background calculations are indeed consistent, what really matters is the action-reaction expectations of the player. A shaky foundation is a dangerous thing especially for core/primary mechanics.

To conclude this part I'll wanted to make another clarification. It seems that many use the words "controls" and "mechanics" interchangeably. (**Remember when I explained how vague language breeds more vagueness?**) These words do more harm than good if they share the same meaning. For when they do, it's hard to know whether someone refers to the physical controller or the way a gameplay mechanic is programmed to use the controller. So, to be clear, the word **controls** refers to input device hardware (including which moves are mapped to which devices), **mechanics** refers to the hardware and software combination of gameplay actions, and an individual reference to a mechanic (e.g. JUMP) refers to just the software side.

To illustrate the difference between controls, mechanics, and gameplay actions consider the following example. For our purposes, the target goal is successfully landing an attack. Imprecise controls would be a faulty button on the controller. When you press this button the gizmos inside will either send the signal properly or short circuit. For an imprecise mechanic there would be great difficulty in repeatedly hitting the target because, perhaps, of how sensitive the motion controls are (assume that the input device works properly). No matter how careful you are in attempting to repeat the same input, the results vary. The problem is the result of the input device/method and the programming of the move. And finally, an example of an imprecise attack is Fissure, the 1-hit-KO attack from Pokemon. Using this attack produces varied results purely because of how it's programmed. Even though the button or touch screen controls are 100% precise, Fissure only hits 30% of the time.

In part 7 we'll use the terminology to finish our evaluation of my four part mechanics criteria.

Controller Design: Ergonomics pt.7

Monday, August 22, 2011 at 10:59PM

Richard Terrell (KirbyKid) in Clean Design, Dynamics, Learning, Motivation, & The Mind, Mechanics, Skill, Super Smash Brothers, Variation

part 1. **part 2**. **part 3**. **part 4**. **part 5**. **part 6**.

With the terms established in part 6, we can now look at a theoretical player. Let's call this player Theo, who loves to play FPSs (get it? Aiming reticle). To sync up with, focus in, or to become immersed with gameplay, Theo looks for a continuous stream of action-reaction pairs. As soon as Theo feels the buttons depress under his fingers, he looks to the game to give him the feedback. Depending on Theo's knowledge of the game and his expectations, Theo will anticipate certain reactions while filtering out other stimuli. For example, when Theo presses the trigger, he ignores all the HUD elements around the sides of the screen. He does this to focus on where his shots land. Already Theo's interactions are quite complex.

Upon repeated tests, **the biggest determinants of how precise or responsive a mechanic feels to a player is the software side of the mechanics design and feedback design, not the controller design.** Yes, the input devices shape our expectations initially, but mechanics and feedback design make up the vast majority of influential elements. When the feedback we receive goes against what we anticipate, that's when there's a problem. When gameplay interactions have hundreds of complexities (rules) that govern the outcome, there's certainly more potential in the code for the player to mis-anticipate than with the controller. Though the devices I analyzed in this series range from simple (the button) to very complex (voice and motion controls) the complexities involved with these devices are just the tip of the iceburg compared to the complexity of emergent gameplay. Think about it this way, you can hit one button in Smash Brothers and yet activate an attack that has **23+ complexities involved**!

Notice how each number hits differently. 5 = fire. 6 = electric. 7 = food. 8 = ice. 9 = death.

For example, Mr.Game&Watch's side+B attack in Super Smash brothers is the JUDGE hammer. Activate the move, and a random number between 1 and 9 is chosen, which determines how strong the attack is. A player who doesn't understand how JUDGE works may think it's imprecise. After all,

something different happens every time. Furthermore, the various numbers have different shield stun, knock back, and other effects (**see data here**). These differences can go a long way in throwing off your timing and therefore your feelings of how responsive the mechanics of the game are. Refer to the **2nd** and **3rd** part of the article series **Complex Time Simplified** for a detailed analysis of how players can develop a warped sense of time with fighting games based on their expectations and design features like hit stun.

The responsiveness of mechanics inherently depends on a game's **feedback**. Keeping in mind how **reflexes** work, we know that sound effects are the type of stimulus recognized most quickly by the brain. Technically touch (rumble) is faster but this communication channel gets easily overloaded. So, regardless of how fast a controller input registers in the system, it can be more important to the player for some kind of sound effect to play as soon as soon as possible. Without proper audio/visual cues players have to basically memorize scenarios and interactions one by one.

In a similar way, the more complex a game and its mechanics are, the more information players have to analyze and strategize with to continually perform better at increasingly difficult challenges. With any learning task, expectations and attitude are huge factors. Sometimes this information is about how gameplay actions interact with level elements. Sometimes it's about **gameplay dynamics**. But sometimes players have to study and experiment with how they manipulate the controller. While this is a perfectly legitimate way to design a gameplay challenge, many players think otherwise.

For one reason or another some gamers see having to study or practice a game as a flaw. This is especially true for studying and practing controller manipulation, which is negatively referred to as performing "**finger gymnastics**." Such gamers who feel like they have mastered holding and manipulating a particular controller tend to become irritated or offended when a game requires that they upgrade their skills. They also think that because they have mastered some other control scheme for some other type of game, they shouldnt' have to learn any other way for any other game. Such views somewhat foolishly try to separate gameplay and emergent strategy from the foundation that is controller and mechanics design. I explore this topic of controller standards and controller feel in greater detail in part 8.

We all know that just because you hit a button that doesn't mean the corresponding action will activate in a video game. There are all kinds of modes, stances, and limited states (like hit stun) that prevent certain actions from activating at specific times. Though potentially troublesome when trying to design clean games, **cancels** will certainly give a game a more responsive feel simply due to their nature. To reiterate, instead of limiting the use of another move until the first move times out, cancels cut in and activate right when the player inputs the move. The more cancels in a game, the more likely the mechanics will respond exactly when the player inputs and according to what the player expects. You can see how such a design can feed positively back to the player. When players don't have to worry about when and how they'll be locked into animations, the player's expectations are met and feelings of mastery and control are soon to follow. Cancelable mechanics is perhaps why many players feel more competent with a combat system like in God of War or Devil May Cry versus Bomberman or Zelda.

If your goal is to design mechanics so that players quickly feel completely in control, then know that controller input devices shape player expectations first. When we see buttons we anticipate individual actions. With analog sticks we anticipate varying degrees of actions; e.g. with slight touches we look for subtle reactions. With touch screens we anticipate direct control and visuals that respond to our specific touches. With pointers, we expect the system to keep up with our

visuals that respond to our specific touches. With pointers, we expect the system to keep up with our hands which race to keep up with our eyes. We expect microphones to work just as well as human ears. And we expect motion controls to capture real life motions. **Whether you embrace these expectations, ignore them, or meet the player half way, know that regardless more complex games create more complex adjustment and learning processes for the player.**

So to conclude this part of the article series, let's revisit my criteria for evaluating video game mechanics. A mechanic's **directness** is largely an issue of ergonomics in which we examine how the potential physical manipulation of the controller influences player behavior and expectations. The **intuitiveness** of a mechanic is a measure of how ergonomics and learned behavior align with the complexities (specific rules/parameters) of the mechanic. Being believable or relatable are also considerations in this category. A mechanic's **dynamic** quality looks entirely at the software side of a system and the emergent possibilities (how the mechanic interacts with other elements). And a mechanic's **individuality** considers its cleanness, which is important for sustaining clear action-reaction pairs between inputs and outputs so that the trial and error process of self teaching goes more smoothly.

I can't think of anything to add to my 4 part criteria of mechanics design. I was onto the right track over three years ago. There's still much more about controller design to discuss. In part 8 we'll consider controller function versus feel and how controller design innovation isn't always about more sensitive technology.

Controller Design: Function vs Feel pt.8

Thursday, August 25, 2011 at 11:05PM

Richard Terrell (KirbyKid) in Controller Design, Misc Design & Theory

part 1. **part 2**. **part 3**. **part 4**. **part 5**. **part 6**. **part 7**.

The video game industry rides the technology wave of the future. Our industry not only adopts high end technologies, but we push the wave along with our billions of dollars. For example, the demand for PS3s creates a demand for multi-cell processors, which go into mass production, which brings down the price and feeds the industry. From glassessless 3D screens to higher resolution monitors to ultra sensitive controllers, video games help bring the future home. It's time to start asking ourselves some tough questions about this curious relationship.

Do we need to ride this technology tsunami? "Need" is a very loaded term. Certainly Super Mario Bros. only needs NES hardware and it's one of the greatest games of all time. And I personally believe that intelligent design is only minimally limted by hardware. So, maybe we shouldn't ask ourselves if we "need" to ride the wave but what we want from technology. The **hardcore technologist theory** is a unique theory that works to explain the core driving force behind the demands "hardcore" gamers. According to this theory the hardcore gamer is not most devoted to gameplay experiences, quality game design, or artistic expression. Rather, the hardcore gamer is most devoted to technology. The faster, more powerful, thinner, lighter, and brighter, the better. And for controller technology, fewer wires, longer charges, new sensors, and more sensitivity is what the hardcore gamers have gravitated toward. Even if you don't fully accept the theory, it is useful for framing the struggle of two ideologies in the realm of gaming controllers. The issue is between controller function and feel.

Function vs Feel

In the previous article of this series I thoroughly explained how the bulk of mechanics design is in the software. I also explained how player expectations play an important part of the player experience at every step. With this in mind I considered how sensitive controllers have to be. As I've outlined previously, modern games use the sensitivity of various current generation controllers to varying degrees. In some cases analog sticks have been programmed to function like a button. Or in other cases the sensitivity range of input is divided into a few, easy to discern stages (e.g. tip-toe, walk, power walk, run). Put simply, mechanics are often times not designed to use the maximum sensitivity of their controllers. Likewise, there are very few games that use every button and analog trigger, stick, and sensor on a controller to the fullest.

From a hardware designer's perspective, designing controllers with many features with large sensitivity ranges is a great way to create a controller that's compatible with many types of games. But from a game designer's perspective, the sensitivity, speed, and even precision of inputting (**function**) may not be most important. **Controller feel**, the feeling of manipulating a particular

controller, is a frequently overlooked aspect of controller design that can be just as important as function.

image by restlesswillow

In terms of gameplay challenges there are 3 ways to challenge a player; **moral-emotionally, physically,** and **mentally**. Gamers with a preference for non-voilence might have a very difficult time playing a game like God of War. Or, players might struggle with the idea of taking time out of their holidays to play Animal Crossing. These example fall under the moral-emotional category. Inputting into all gaming controllers falls into physical category. The **sub facets of dexterity skill** cover this category thoroughly. And all the rest of one's **DKART skills** fall under the mental category. Any one of these methods is a perfectly legitimate way to design a video game challenge. Yet, some players only want and respect mental challenges. Controller feel falls under physical, yet there is more to it than just ergonomics.

Controller feel includes everything from the physical act of pressing buttons to more engaging motion controls. It includes holding the RUN button and rocking the thumb over to JUMP in Super Mario Bros. It includes spinning the control stick and hitting A to spin attack in Zelda. And it's the distinct feel of inputting Street Fighter special attacks. Controller feel is the result of a designed control scheme and the emergent sequences, combinations, techniques, and grips players develop to play. Some mechanics are designed more intuitively than others so that everything meshes. Flying as racoon Mario in Super Mario Bros. 3 is a great example. Having to repeatedly hit the button to ascend matches the feel of inputting with the fiction.

Mechanics design isn't necessarily about making mechanics that are as easy for players to execute as possible. Games with little to no dexterity challenge (mostly **turn-based games**) are generally designed with easy to execute mechanics, but for most games the potential is wide open for supporting easy to nearly-impossible-to-execute moves. Obviously, game designers have to take input difficulty into account when they balance their games. Sometimes the controller feel is prioritized over function. Guitar Hero, DJ Hero, and Rock Band are perfect examples. These games

can be played with a traditional gaming controllers, which some gamers find easier to use. But most who play these games greatly prefer playing on the specialized instrument controllers. The feel of holding a guitar like object or actually hitting drums cannot be matched on a traditional controller. I feel the same way about Donkey Kong Jungle Beat's bongo controller. Virtual On's twin stick set up. **Steel Battalion's legendary controller**. Or even the classic toy Etch 'n' Sketch. As a **graphite and pen&ink artist**, drawing with this toy severely handicaps my abilities. But there's nothing else that feels quite like the dual knob drawing

feels quite like the dual-knob drawing.

Bringing the conversation to Nintendo, I feel that I have a good understanding of how Miyamoto and his co-workers design games. With a history as toy and card game manufacturers, Nintendo developers put a high priority on the uniqueness and novelty of controller feel and gameplay experiences. In the Iwata Asks interviews, Nintendo developers describe sharing hardware prototypes with each other driven by the "oh, well isn't this neat" factor. Just being a neat interactive experience seems to be enough for Nintendo to develop a game around the feeling. So, it doesn't surprise me that Nintendo has a long history of not adopting the highest end technology modules simply for the sake of keeping up with the tech wave. They act differently because they think differently from the DSi/3DS low resolution cameras, the lack of a "W" button on the Gamecube controller (opposite the Z button), the Virtual Boy, the lack of multi-touch support on the Wii U controller, to the vitality sensor.

Controller Standards

I've never been a big fan of game design "laws" or rules of thumb presented in absolute terms. For example, "never not reward the player for exploring an area." In general I can easily think of counter examples where a successful game breaks such "rules." And secondly, I'm not comfortable with limiting the scope of what a video game can do and how it can do it by default. Even on this blog, I try to explain the effects of design decisions and how they may cause unwanted or unpopular consequences down the line. With that said, I want to address the topic of controller standards.

I do not believe there should be standards to the way games map mechanics to controller inputs. Even if an FPS ends up having the same configuration as Modern Warfare or Halo with a max of two weapons, red exploding barrels, and regenerating health, I would like for the developers to have chosen these conventions because they couldn't find a better alternative. One of the best arguments for controller standards is the control scheme of the best or most popular game becomes a kind of common language. Such a control scheme then influences the actions and expectations of players especially for games in the same genre. So, instead of forcing players to learn a new language for each game, why not tap into what they already know. It's less about being original and more about ergonomics.

My counter argument to controller standards is best addressed in part 9 where we'll look at the topics of customizable controls and controller redundancy.

Controller Design: Customization pt.9

Saturday, August 27, 2011 at 12:41PM

Richard Terrell (KirbyKid) in Controller Design, Planet Puzzle League, Super Smash Brothers

part 1. **part 2**. **part 3**. **part 4**. **part 5**. **part 6**. **part 7**. **part 8**.

At the end of part 8, I described the pro controller standards argument. To reiterate, popular games with well designed controls create familiarity and expectations among many gamers. The idea is, if you're designing a game, you should use these control schemes. Doing so makes it easier for many players to pick up and play your game. Thus standards emerge.

My counter argument is using a controller standard may ease the transition for a player, but beyond the control scheme there is much more game specific information players have to learn. Because the bulk of mechanics design is software based, whether you use controller standards or not hardly makes a dent in the complexities the player must learn overall to play effectively. Another way of putting it is that because every game is different, it's not so big of a deal if the control scheme is also different. There are plenty control schemes and controller feels that work well despite being far from the standard. Treasure's games (see **here** or **here**) are known for having unconventional control schemes that work perfectly for each game. From **Iwata Asks**...

> ***Maegawa (President of Treasure):*** *We started talking about how no one hardly ever used the left position [on the N64 controller], even though you can perform independent actions with it.*
>
> ***Iwata:*** *You can do some unique controls, but no one used it.*
>
> ***Maegawa:*** *Right, no one used it. But we thought it would make for interesting game operation and started development.*

Smash Brawl custom controls menu

Customisable Controls

Customisable controls alleviate most of the pressure on developers regarding control mapping and control standards. In general PC games feature more customization options than other platforms. This largely stems from the open platform nature of the PC. Because anyone can make and sell any kind of controller for the PC (mouse, keyboard, joystick, pad, etc.) the easiest way to ensure compatibility is to give players the ability to map anything to just about anything. And because the PC is such a versatile tool, for any options the developers don't supply modders and hackers can certainly pick up the slack. But for all other platforms controls are are more serious issue. I have mixed feelings about customisable controls. To articulate my stance I'll use a musical metaphor.

I always relate control design issues gamers face by comparing video game controllers to a piano. Pianos are mostly built to the same size standards (I'm not talking about video game control standards right now so work with me). This means that most of the pianos I play feel the same in terms of the distance between keys and the muscle memory I can use to play successfully. Over time I've encountered many piano pieces and passages that were technical (dexterity skill) road blocks. Sometimes my hands weren't large enough. Sometimes my fingers weren't quick enough. And other times I just didn't have enough stamina. Regardless of the issue I enjoyed working through the issues. Because the piano was a fixed, consistent challenge, I molded myself around it. There was hardly a challenge that I couldn't overcome with practice and some ingenuity. I don't know what kind of player I would be today if I could have altered the size or shape of the piano to make it easier to play.

I would want the players of my games to be able to experience the same kind of road blocks and successes that I have from playing the piano. I want them to know that just by applying themselves and doing a bit of problem solving, increasing one's dexterity skill is very unique and satisfying. This was my original reasoning for being somewhat against customisable controls. But this view is short sighted when you put video gaming and even music playing into perspective. Years ago my former piano instructor, Dr. Deforest, gave me permission to play the piano with my nose as long as I could consistently produce the right tone. In other words, playing the piano is about making music, not necessarily finger exercises or challenges.

The "music" of gameplay is generally occurs within the computer system because the bulk of mechanic complexities are software. So it makes little difference if a player creates a custom control scheme by rearranging the buttons. They still have to hit the right buttons at the right times to get the job done. Custom controls don't take the dexterity challenges out of playing games. So outside of competitive fairness, I now fully support customisable controls at least on the most basic level: rearranging mechanics (button mapping). As a designer you have to balance customisable options with **gameplay balance** and controller feel. Once you know your limits, the more options the better.

Control Schemes - Give and Take

While some games are completely designed around a particular input device so that they wouldn't work well with any other type, most game systems are pretty flexible. When good developers design control schemes, they take into account the pros and cons of the input devices. So when these developers design multiple sets of control schemes it is not uncommon for there not to be one superior choice (due to the fundamental differences and limitations of the devices). In other words there's a give and take.

The following is a list of games with multiple control schemes: Planet Puzzle League, Animal Crossing Wild World (DS), DigiDrive (DSi), Advance Wars DS games, Resident Evil 4 (Gamecube vs Wii), New Play Control Metroid Prime 1 & 2, New Play Control Pikmin series, and Ookami (PS2 vs Wii). The more you understand the input devices each scheme uses, the easier you can

understand the pros and cons. I'll present one example here and the rest at the end of this article series.

Planet Puzzle League (**learn the rules of the game from the video here**). The D-pad and the A button work very simply and precisely in this block puzzler. If you press left, the cursor will move one block over to the left. If you hit A, two blocks will switch places very quickly. Moving around the block field is always relative to your cursor position.

But with touch screen controls, players can simply touch any block and drag it around to continually swap positions (as opposed to having to hit left + A + left + A + left + A). Many have praised this new control scheme because it's easier to move around the field and perform what would otherwise be highly dexterous, tight timing button sequences. But there are a few technical issues with the touch screen controls that keep it from being an all around superior choice to traditional buttons. The first is that it introduces **buffering into the design, which clutters the game somewhat**. The second is that because you control the action directly according to the objects on the screen it's easier to miss your targets when the entire field moves. And finally, though the stylus is thin, it does obscure the screen view to some degree.

So the trade off is between very straight forward, clean controls versus more direct controls that are easier to move around with and pull of advanced maneuvers, while adding more complex and potentially confusing reactions and visual elements.

Redudant controls are something like this.

Controller Redundancy

Another topic I wanted to touch on before I bring this article series to a close is **controller input redundancy, which is simply different inputs that are mapped to the same mechanic/function.** Redundancy design is similar to optional control schemes in that it gives players options. However, the key difference is that redundant inputs must be mapped within the same control scheme. Redundancy is less common in older handheld and console games. With only a few buttons on the NES controller or the Gameboy there weren't enough free buttons to work with. Since the SNES and the four-face-button layout, many games were designed with redundant controls. In Super Mario World you can either use X or Y to RUN. PC games have also commonly supported WASD, arrow key, and even num pad redundant movement options.

Redundant controls offer interesting bonuses for a game's design. First of all, redundant controls don't interfere with the individuality of a mechanic and therefore it doesn't hamper the cleanness of a game. For my second point, I'll use another musical metaphor. As much as I love playing the piano knowing that every key plays one note and that note can only be produce by hitting the singular key, I love the violin equally (**see video of me here**). Figuring out my fingering (the exact fingers to use on specific keys and strings) for these instruments is like two different puzzle games. Because the piano keys are a fixed arrangement, I have to figure out which of my 10 fingers will play the note and how doing so will affect the notes that follow. It's similar playing a violin, but the unique part is that the 4 strings on a violin are only 5 note values apart from each other. With enough skill you can play many of the notes from the high strings on the lower strings. This design allows me to shift my hand up and down to work out efficient and accurate fingering. **To reiterate, the more redundancy in a control scheme, the more potential there is for the user to improve their input techniques, which can go a long way in improving precision and control (dexterity).**

My favorite redundant control scheme is the Smash Brothers Gamecube scheme (Melee or Brawl). I made **this video** about a year ago explainin the redundant controls. With this scheme you can JUMP with X, Y, or by pressing up on the control stick. You can perform air attacks by pressing A+direction, Z, or using the C-stick. You can shield with L, R, or by holding Z. You can grab with shield+A or Z. You can use smash attacks with A+direction or the C-stick. Now, I will say that some of these redundancies I do not like. Using the stick to JUMP or hitting Z and getting a shield can be frustrating. Fortunately, Brawl at least includes an option to remove stick jump. **The key take away here is that redundant controls design is generally better for individual mechanics. Things can get over complicated when the mechanics are not individual.**

To avoid problems with redundant controls, we must design smarter. Understanding the pros and cons of the input devices is just the beginning. The following are 2 more examples of games that have done it well.

The Wii Sports menu is fantastic. Large buttons with large text makes for easy pointer based navigation. But if the user doesn't feel like pointing, the D-pad also navigates through the menu. Because pointing is more engaging, the developers set the priority to the pointing controls when pointing at the screen. As soon as the cursor moves off the screen, the menu cursor appears and switches over to D-pad controls.

In The Legend of Zelda: Ocarina of Time 3DS Link's movement is controlled with the analog circle pad. Like with analog sticks, very fine movements are difficult with the circle pad because of the few muscles the thumb uses. Much of Zelda's combat works best using the lock-on mechanic. This mechanic eliminates the need for a second analog stick for aiming while letting players avoid **stop-and-pop tactics**. However, there are times when players will stop and aim in first person. Interestingly, using the 3DS motion controls, players can swivel around in real life to adjust the aim in the game. Because these motion controls are redundant with the circle pad controls, players can use both as needed getting close with the circle pad and fine tune using the motion controls. In other words, use the thumb (3 dorsal forearm muscles + 4 thenar muscles) for convenience and ease. Use the 40 or so muscles in both arms for the motion controls to fine tune your aim as needed.

In part 10, I give a recap and end the debate of many controller arguments.

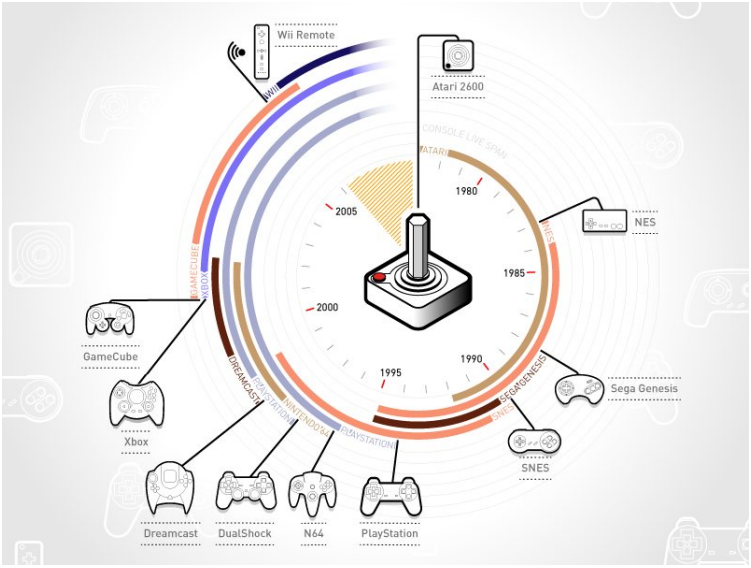
Controller Design: Recap & Rebuttals pt.10

Thursday, September 1, 2011 at 9:44AM

Richard Terrell (KirbyKid) in Balance, Controller Design, Interesting Choices, Interplay, Metagame, Misc Design & Theory, Skill, Variation

part 1. **part 2**. **part 3**. **part 4**. **part 5**. **part 6**. **part 7**. **part 8**. **part 9**.

We've come a long way since the beginning of this article series. A lot was discussed, so here's a recap.



- Controller design is a part of mechanics design, which is the foundation of video gaming (an interactive medium). The word "controls" refers to the physical input device or method. While the word "mechanic" refers to the hardware-software bridge of player actions.
- [Good] controller design takes into account ergonomics (player expectations) and input device pros and cons while significantly influencing the mechanics design and feedback design.

- Buttons are the simplest input device because of their binary on and off states. The spring action helps the user know exactly when the electronic "on" signal is sent. Buttons are relative input devices that have common side effects of button-mashing and double-tapping. Buttons are great for actions that are activated and deactivated from shooting, punching, jumping, and confirming.
- The analog stick is a 2D relative input device that springs back into a centered neutral position. The stick is great for controlling 2D analog movement whether of characters or aiming reticles. Sticks are not so good at controlling 4 or 8 way digital movement because of a disconnect between the expectations of the stick and the actual mechanics. Dead zones, auto aim, and motion tweening are common analog stick design issues.
- Touch screens are direct input devices that allow the user to interact with the visually displayed elements. Relying largely on hand-eye coordination, players have to focus on the elements they manipulate especially because there is no tactile feedback between an interactable elementand a non-interactive section of the screen. Stylus based touch screen controls tap into our handwriting fine motor skills for the most versatile and sensitive input

method. One drawback is that the screen is obscured when in use, which gets worse with multi-touch design.

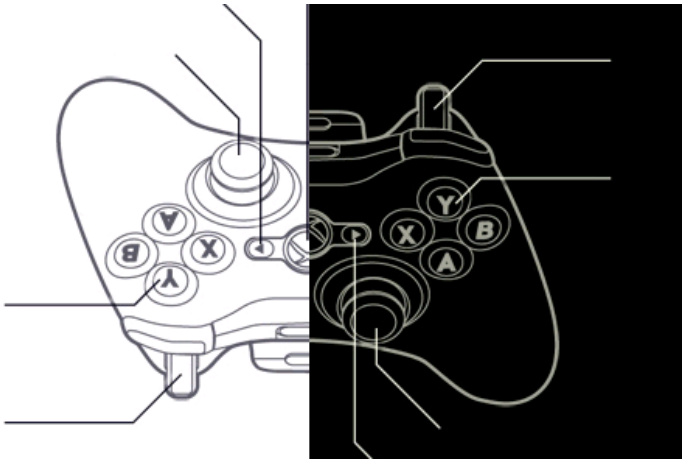
- Triggers are 1D analog buttons. Since their physical design falls somewhere in between a button and an analog stick, the types of mechanics that they are best mapped are somewhat limited. Considering that in real life there are few devices that utilize analog triggers, there are few in gaming.
- The mouse is a 2D relative pointing device with high sensitivity. By using finger and arm muscles to move the mouse, players generally have a lot of control dexterity skills to match the high sensitivity. The Wiimote and the Playstation Move areexamples of 3D direct pointing devices with high sensitivity. Typically, pointer input is coupled with buttons so that the pointer indicates and the buttons selects. The UI for pointer based controls is similar to that of touch screens.
- Motion controls vary in complexity and technology. While tilt motion controls are fairly simple, controls that detect force and direction are harder to develop. The lack of tactile feedback of makes understanding the limits of the motions or what the controller can sense difficult. especially with no tactile feedback. Calibration is another potential issue. The pros include greater control due to more muscle involvement, high accessibility and redundancy potential (ie. you're always in position to make a motion), huge variety for new game feels, and tapping into existing motions intuitively (ie. 3D control of 3D actions and other familiar motions).
- Microphones can tap into a more accurate and descriptive language than motion controls. And it's all hands free. Like with any other analog input device, mechanics can be designed to use a mic simply (detecting just volume or a tone) or in a highly complex way (voice commands). General issues include enunciation, word recognition library, and filtering out ambient noise. Another issue is language requires a lot of mental focus. This human limitation makes designing challenges where players have to simultaneously use voice controls and another method problematic.
- Camera controls, like voice commands, take advanced software and processing power to achieve the same kind of precision and responsiveness as more traditional control devices. Lighting lighting and adequate space are new factors for designers and players to consider. Cameras can recognize faces, objects, motion, or any other visual element. Their versatility runs from a user sign in, to 3D controllers, to augmented reality.

- Precision, responsiveness, and sensitivity are the most common and most important elements of quality mechanics design. Though these elements are complicated by player expectations, the consistency and lack of delay between the player inputs and the game response is the crux of the bridge that allows the player to sync up and become immersed in a game.
- The biggest determinants of how precise or responsive a mechanic feels to a player is the software side of the mechanics design and feedback design, not the controls design.
- Controller feel is the physical sensation of manipulating a particular controller, control scheme, command, sequence, or other input method. It can be just as important to a game's design as precision and responsiveness; or more so.
- Controller feel is also a consideration when tweaking the difficulty of dexterity challenges. Mechanics design isn't all about making inputs as easy for the player as possible. Otherwise we wouldn't have games like Street Fighter.
- Consider controller standards with caution. Overall, the issue is not so important because of customization options and redundancy design.
- More often than not different control schemes for the same game merely offer a different combination of pros and cons. It's really difficult for one scheme to be superior across the boards to another in the categories of controls design (precision, responsiveness, etc.) and game feel. This is especially true because parts of these considerations are the result of player

game feel. This is especially true because parts of these considerations are the result of player preference.

The following are rebuttals and arguments addressing popular topics concerning video game controls.

To Invert or Not to Invert



This is actually quite a sophisticated issue. I've witnessed many gamerstry to debate the topic with little success. To clarify, in many 1st and 3rd person video games (mostly shooters) the camera view and aim is controlled by a 2D input device. Some use sticks (Halo), some mice (Half-Life, TF2), some touch screens (Metroid Prime Hunters), and even buttons (Perfect Dark N64). Regardless of the input technology, there has historically been an option to invert the vertical or y-axis view controls. This means that left and right move the view left and right respectively. But with inverted controls, up moves the view downward, and visa versa. Gamers have argued that one is backwards while the other way is "normal." And despite how the options are worded in option menus (normal vs inverted) this debate is stuck in a bitter standstill.

The answer to this quandary is not that some gamers have gotten used to flight simulator controls where down is up and up is down (as it is with real planes). Nor is the answer as simple as each person just picked a side and has gotten used to it. **Rather, the solution is an issue of the ergonomics of each individual player's reference point.** Imagine holding a sub machine gun with two hands. To aim this imaginary gun you could keep the gun level (parallel to the ground) and move both arms around together. If you aim this way you'll not only waste a lot of energy moving, but

you'll awkwardly swing through a lot of space. This method also has a limited aiming range making it a poor option. To hit targets higher and lower to your position, it's more efficient to rotate the gun. By rotating you can keep your arms from moving so much and achieve a greater aim range. You should understand all of this from first hand experience even if you're not a fan of guns or have never held a gun.

So, here's where player point of reference comes into play. If you visualize yourself holding a gun like the on screen character, you play with inverted controls. If instead, you abstract the action of aiming so that you only focus on your aiming reticle you play with inverted off. **In other words, your preference depends on whether or not you think the input device controls the hand**

holding the gun or the reticle sliding across the 2D screen. There is no right or wrong way. Even if you think inverted is the "right" way because imagining yourself as the game avatar seems to be the best fit, just remember that video games are abstractions. To create a bridge between the abstract virtual world and the real world where the player exists, we learn mechanics and manipulate controllers. Each bridge is different because we make and strengthen connections naturally and somewhat randomly. In other words, we make it up and stick with what clicks for us. What matters most is whether or not the player can control the game well. Whether the player thinks the right analog stick controls the muscles in the avatar's wrist, hand, arm, or the reticle itself hardly matters. This inverted/not-inverted issue falls under ergonomics > psychology > behavior > expectations.

Shooter Controls: Mouse vs Stick

Both the mouse and the analog stick work well for shooters on the PC and consoles respectively. For proof, look to the immense popularity and financial success of the shooter genre. But if it's a question of which input device is better for shooter gameplay, we have to use a full spectrum of critical analysis to form a proper answer. At the foundation of gameplay we have controls design and mechanics design. We already know the limitations of mouse and stick input devices. Ultimately, a mouse allows for players to exert more control dexterity skill because there's no resistance when moving the device and more muscles are used to finely adjust its position. But there's more to it.

As I've explained previously, sensitivity of an input device isn't everything. The feeling of playing a game is also important. Because the feel is largely the result of how the rest of the game is programmed and the emergent possibility it allows, we have to consider how the superior sensitivity/control of a mouse influences the design of a game versus an analog stick.

It's simple really. **All great shooters are tuned for balanced gameplay where, for the most part, one gun or one simple playstyle isn't the dominant strategy.** So designing around an input device with high sensitivity and control, PC shooters are generally designed with quick character movement speeds. After all, if everyone moved too slowly tactics like dodging, running away, and flanking would be severely hampered because of how effectively players would be able to target each other. Though aiming may be lightening quick, shooting (or any other mechanic) isn't necessarily so. There are many factors that slow down the attack rate of players regardless of how

necessarily so. There are many factors that slow down the attack rate of players regardless of how quickly they aim. Reload times, fire rates, recoil, and slow moving ballistics are just a few of the factors that are tuned to give different weapons different timings and feels. In these ways the core gameplay of PC shooters (aiming and shooting) is tuned around the controls to allow a variety of different weapons, tactics, and strategies to exists so that beginners have to practice, experience players have fairly consistent success, and the best are near flawless.

It's the same situation with console shooters. Because aiming is a bit harder with an analog stick, aim assist is generally implemented. By no means does this feature do all the work for the player. To further tune the gameplay, the movement speed of the characters may be turned down or the damage output of the weapons turned up. Like on the PC there are shooters where you can die quickly (Call of Duty, Counter Strike) and there are shooters where you can generally take more hits (Halo, Section 8, Team Fortress). Either way, the core gameplay of these shooters (aiming and shooting) is tuned around the controls to allow a variety of different weapons, tactics, and strategies to exists so that beginners have to practice, experience players have fairly consistent success, and the best are near flawless.

From a **strategy** standpoint, the mouse doesn't necessarily allow any more strategy or depth than an analog stick. It's the game mechanics and the rest of the overall design (**balance**, **interplay**, **variation**, **interesting choices**) that most shapes the emergent gameplay. You can have a very accurate, precise, responsive input device and a very shallow game. Or you can have a very deep game and play on an N64 controller like my favorite FPS Perfect Dark. The more you understand how players learn, develop their skills, and how the **metagame truly evolves**, the easier you'll see that under fair, consistent conditions, the controls (hardware) hardly makes a difference to the strategic gameplay.

The issue of which takes more skill to use is moot as well. The reason it's moot is not because players will eventually get used to playing with either device. It's because the input device differences can only impact the dexterity part of the **skill spectrum**. Even without considering that both games are tuned around the pros and cons of each device, any difference between the skill floors and ceilings between mice and analog sticks would be small. After all, aiming a 2D reticle using a 2D input device isn't complicated. In general, the bulk of what it takes to be competent in an FPS is knowledge skills; then timing, reflex, adaptation, and then dexterity. Though each device tends to stress a different combination of dexterity skills, recall my theory that **all competitive multiplayer games take the same amount of skill to be played continually at a high level**. So between games and between devices in the same genre the differences in skill ceilings are negligible.

Personally, I prefer console style shooters because generally the slower movement and aiming makes for a much cleaner gameplay experience. As great as the mouse is, I believe that most developers leave their gameplay open to support as high a sensitivity and speed as the player wants. The way that many developers tune the gameplay to compensate for mouse controls leads to many design trends and emergent outcomes that I greatly dislike (see **FPShuffle**). I wouldn't design a game with options that would let players adjust their view/aim so quickly the screen it looks like a hard cut is made between looking at one thing and then the next. It's not an issue of what players want. There are many things a player may ask for or want that ultimately work against the quality of their own gaming experience. We know from my An Examination of Skill series that **faster gameplay reduces the skill spectrum variety and the skill ceiling of gameplay**. So I tend to avoid mouse based gameplay that detracts from a game's to cleanness, skill spectrum, and works against the shared mental state between the player and the game.

In the 11th and final part, I'll end with a few more rebuttals.

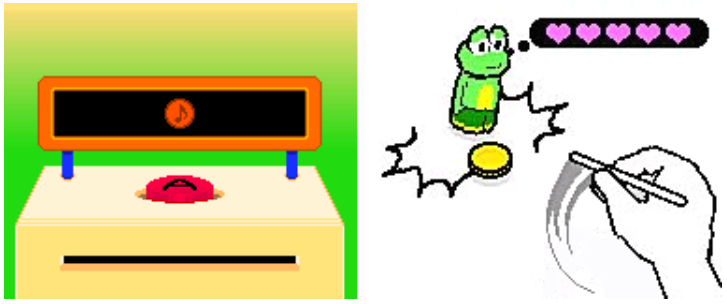
Controller Design: Rebuttals pt.11

Friday, September 2, 2011 at 10:40PM

Richard Terrell (KirbyKid) in Controller Design, Donkey Kong Country Returns, Misc Design & Theory, Rhythm Heaven, Rhythm Tengoku

part 1. **part 2**. **part 3**. **part 4**. **part 5**. **part 6**. **part 7**. **part 8**. **part 9**. **part 10**.

Rhythm Tengoku vs Gold



Rhythm Tengoku and button controls (left). Rhythm Heaven and flick controls (right).

Rhythm Tengoku is a music-rythm game for the GBA that was only released in Japan. Few people state-side have imported the title. But it seems that all who did love the game. Think WarioWare but instead of hundreds of micro games, the game features dozens of musical mini games. And like WarioWare Inc. for GBA, the controls are mapped to very few buttons; most of the time just one button! Years later a DS sequel Rhythm Heaven (Gold) was created and released in the US. Forgoing button controls completely, the DS version uses the touch screen like a giant button. As one of my **GOTY for 2009**, I've written about this excellent game multiple times on this blog. Believe it or not, there has been a debate going on over which of the two versions is better. It's time to put the issue to rest. The following argument does not consider graphics, sound, or other features unrelated to the core gameplay.

Rhythm Tengoku (GBA) is designed to use buttons in the ways that buttons work best; distinct on and off states. Japanese music-rhythm games are known for being more strict about hitting the beats than western music games. And buttons are great for hitting the beat. The argument that the Rhythm Tengoku (GBA) supporters have made is that the touch screen controls in Rhythm Heaven DS are not as precise. To be clear, the GBA version features only button and D-pad controls (remember the D-pad is just a collection of buttons). This means the only functions the game can support are tap, hold, and release. The DS version adds two more functions, flick and scratch, for a total of 5 functions giving the game much more variety.

To compare function to function, tapping, holding, and releasing a button is about the same with a button as it is with a touch screen controlled with a stylus. The real debate has been fought over the flick controls. Basically, instead of simply releasing the hold function on the touch screen, the new

function requires releasing the hold just after quickly sliding the stylus across the screen. It functions and feels just like flicking a coin across a table using the eraser end of a pencil (see right side of image above). Because this action is timing sensitive (ie. you can't slide the stylus too

slowly) it is categorized as an input command. So it the flick function less responsive because of the touch screen hardware? **Technically the flick function is just as responsive as releasing a button or the stylus off the touch screen.**

To understand precision we have to examine how the flick works. Rhythm Heaven (DS) only looks for two things to to detect a proper flick. 1) When exactly the pressure/contact is released from the touch screen. This determines the timing of the flick. And 2) how fast the stylus was moving before the release. That's it. As long as the stylus moves across the touch screen fast enough in any direction before releasing contact, it's a flick. To get the most precision, you have to coordinate a wrist rotation for the speed independently from the stylus lift for the timed release. An easy way to synchronize these actions is to flick faster. Thinking of the flick as one swift, fluid motion is the best way to feel that quick button like release. It's really not very complicated and only takes a bit of thought and practice to work out. Yet, this tiny difference was enough for some gamers who are proficient with buttons to fail with the flick. Having scored a perfect on every level in the game, the precision is there.

Ultimately with equal precision, a wider skill spectrum, and more variety of controller feel Rhythm Heaven DS is the superior version for gameplay.

Button vs Shake-to-ROLL

The single Wiimote is perhaps my favorite gaming controller to date. It can play NES styles games, pointer only games, motion control games, and hydrids of these styles. I think limiting oneself to designing only for the Wiimote is a challenge that brings about elegant, clean, and unique control schemes. Donkey Kong Country Returns is the one of the latest games on the Wii to follow this design path. With the NES grip, the game only uses the 1 and 2 buttons along with the D-pad for RUN, JUMP, and MOVE respectively. The final input is a two handed vertical shake of the controller to activate a ROLL, BLOW, or POUND.

The shake-to-ROLL is so despised by many gamers that hackers have developed a mod that replaces the roll with a button press. Critics and reviewers have also faulted the game for not having a button option. Handicap considerations aside, I'm glad that the Retro Studios stuck with their design choice because the motion controls are superior to a button press for the ROLL mechanic. The reason why is similar to the Rhythm Tengoku v Heaven case.

Here's how the ROLL mechanic works. When holding forward, if you shake the Wiimote vertically with enough speed/force, Donkey Kong will instantly activate a fast moving, jump-cancelable, rolling attack. All the game looks for is two things; 1) if the left/right direction is held down on the D-pad and 2) if the vertical controller speed is beyond a certain value. This design allows for all of your fingers to rest in the same position on the Wiimote while also being ready to shake at any time

your fingers to rest in the same position on the Wiimote while also being ready to shake at any time. But, there's a much better reason why shake-to-ROLL is the superior design. The reason is something I call **manual buffering**.

Remember my article series **The Coefficient of Clean**? In it I explain why clear feedback (cleanness) is so important to the way we play and learn video games and I identify several design features that make the emergent gameplay more or less clean. Buffering is one such feature that makes a game less clean. At the sacrifice of this cleanness, designers opt to include buffering in their games to help players execute moves with frame tight precision without learning strict timings. **The beauty of DKCR's shake-to-ROLL design is that it localizes the buffer to just the ROLL mechanic and it does so without having to design the buffer into the software.** Basically, if you want to ROLL immediately when you land on a platform from a JUMP, you can ensure that the ROLL comes out as soon as possible by performing the motion just before landing. Because the system merely looks for whether you're moving the controller fast enough, you can manually create a larger window of success by making a bigger motion. With this design there's a balance between timing skills and dexterity skills. With so many moving platforms in DKCR, it helps to have a system that I can repeatedly (precision) hit frame tight ROLLs even off the very edges of platforms. If I had to hit a button right when DK landed to get the same results, I wouldn't have been able to gold medal so many time trials (**see video**).

This flexible, elegant design is not possible using a button unless it was designed to activate a roll whenever the button is held down. But with such a button-hold-and-ROLL design, you lose a part of the controller feel because of the disconnect between the kinetic action of rolling and holding a button long ahead of time.

New Play Control vs Old

The New Play Control series is a perfect case to study controller design. The basic idea behind the brand is re-releasing quality GameCube titles on the Wii with retooled control schemes. Some games were outfitted with motion controls, some with pointer functionality, and some with both. Mapping these schemes to a new controller ultimately meant changing the mechanics and, in Donkey Kong Jungle Beat's case, changing the core level design and gameplay as well.

One thing that I like about the original design of some of these games is that it is cleaner. Because the Gamecube C-stick wasn't very suitable for analog 2D aiming, games were designed completely around the left analog stick. Killer 7 and Resident Evil 4 feature stop-and-pop gameplay. Metroid Prime 1 and 2 feature stop-to-free-aim and a lock-on system. And in Pikmin 1 and 2 aiming and moving occur simultaneously. The gameplay of these games isn't necessarily easy or "dumbed down" either. Like I explained with the mouse vs stick case, the gameplay challenges are tuned around the limitations of the input devices.

Using the Wiimote pointer, aiming at Bulborbs, Zombies, or Space Pirates is largely separated from the gameplay in terms of mechanics drawbacks. Instead of being limited by time because of the cursor speed, space because you must move and aim simultaneously, or position because you have to stop-and-pop, aiming is more free than ever. The problem is that gameplay is designed and refined from the bottom up; controls -> mechanics -> level design -> rules. So, for many of these New Play Control games, only the controls were changed. If you're not careful, adjusting the bottom of the "stack" is an easy way to throw the balance off of the entire game.

Many games became much easier to play. In Resident Evil 4 Wii especially, players can aim freely ahead of time so that when they stopped to shoot, everything was already lined up (**see Wii video** versus the **Gamecube video**) Notice how cleaner the look of the game is without a big, white, circular reticle on the screen. Notice how much more suspenseful the action is when the player has to make every shot count because of the slower aiming speed. Notice how the thin laser sight and the small red dot are key feedback elements that require eye focus, tunneling your vision and making the threat of multiple enemies greater.

Quick aiming, cancels, and quick movement are features that can do a lot of damage to the cleanness of a game. But it gets worse. Fast action gameplay limits the skill spectrum of the game. Furthermore, **tension**, suspense, and **spectator sport** moments are all important emergent qualities that require time to develop. The longer the player or the audience dwells on the current predicaments, the more they can weigh how much is at stake. With more to lose, the experience becomes more tense. The better we understand incoming gameplay threats including their methods and counter measures against them, the more we can contextualize each action and each play.

The slower the action, the longer we have to engage and interact with each gameplay action. The more players interact, the more their in-game actions become virtual body language adding another layer to the feedback and communication potential. I like seeing cursors move slowly across the screen like in Final Fantasy Crystal Chronicles (**see video**) because you can really tell what your allies intend to do before it happens. For similar examples see this article titled **Lines of Communication**.

The key take away is quicker, more intuitive, or easier inputting doesn't necessary make for a better game overall. Whatever controller type you design for, tune everything from the controller/mechanics upward. And slower gameplay has many advantages over fast action.

You made it to the end of this lengthy examination of controller design. Congrats and thanks. If you have anything to say or questions to ask regarding controls, feel free to leave a comment.

Peace.
