Controls You Can Feel

Putting Tactility Back Into Touch Controls

Zach Gage - @helvetica - STFJ.net

so this is a talk on why tactility matters and how to bring that to touch games

I am not a scientist

Tuesday, March 6, 12

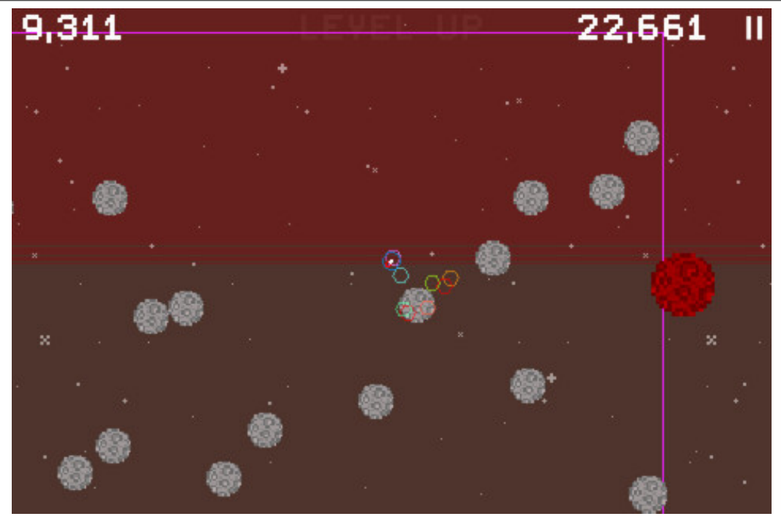
I need to say off the bat that I'm not a scientist.

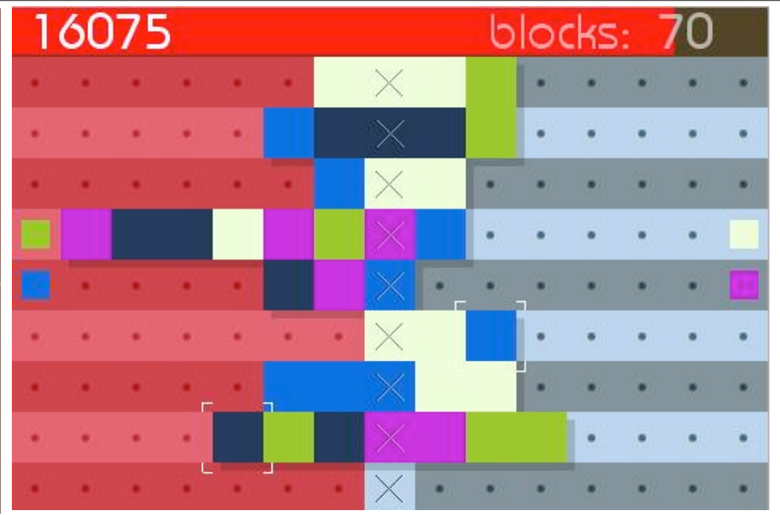
I am an artist

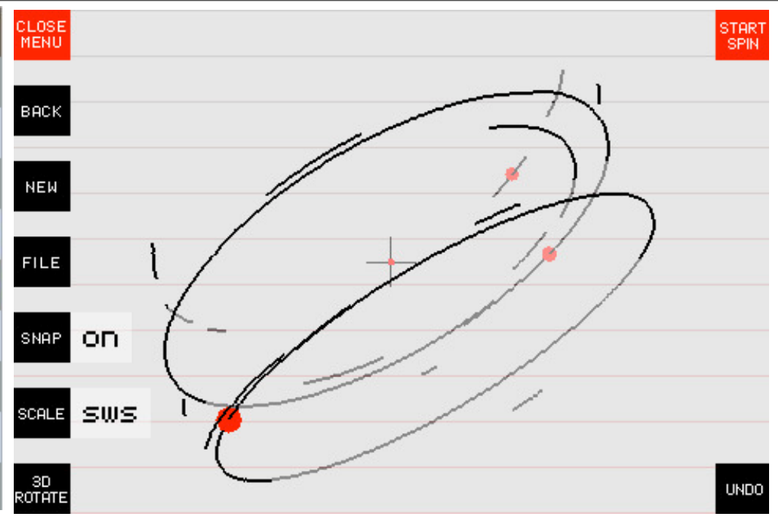
Tuesday, March 6, 12

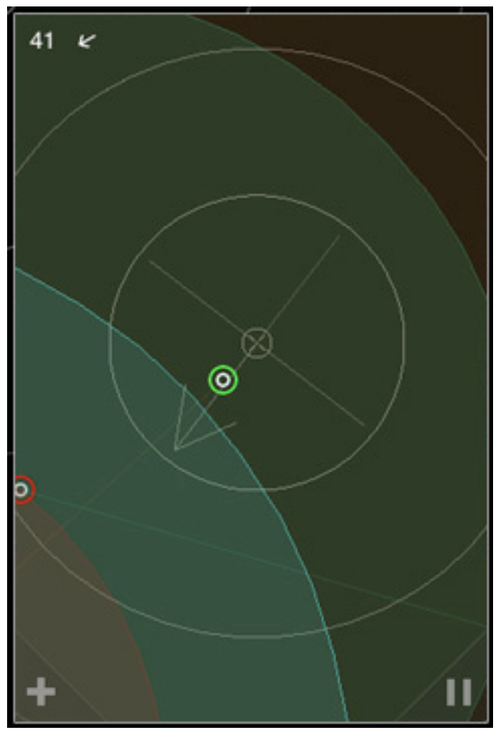
I'm an artist who has spent a lot of time thinking and working with interaction.

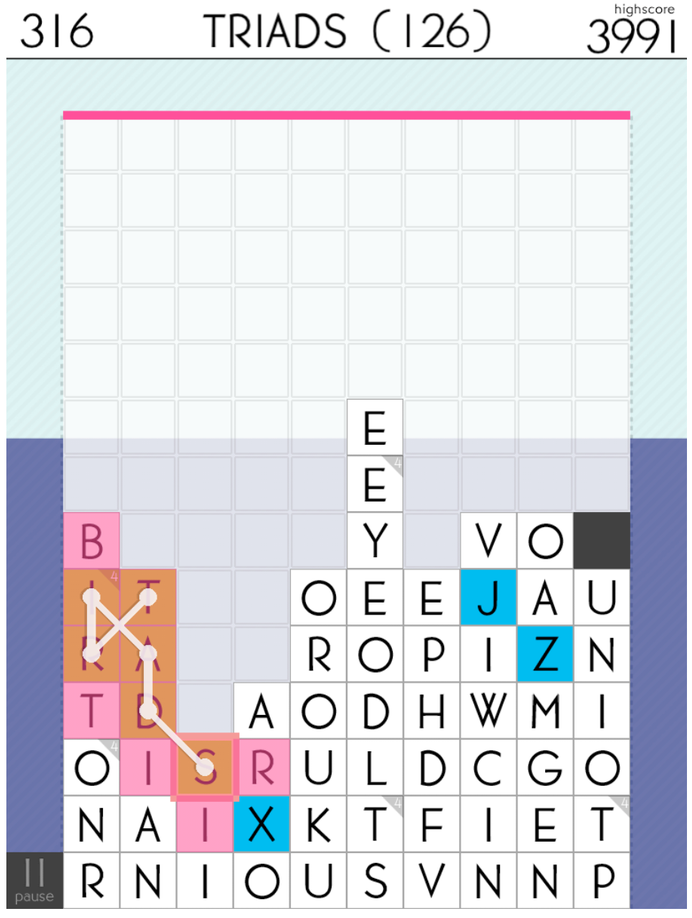
From large scale interactive artworks

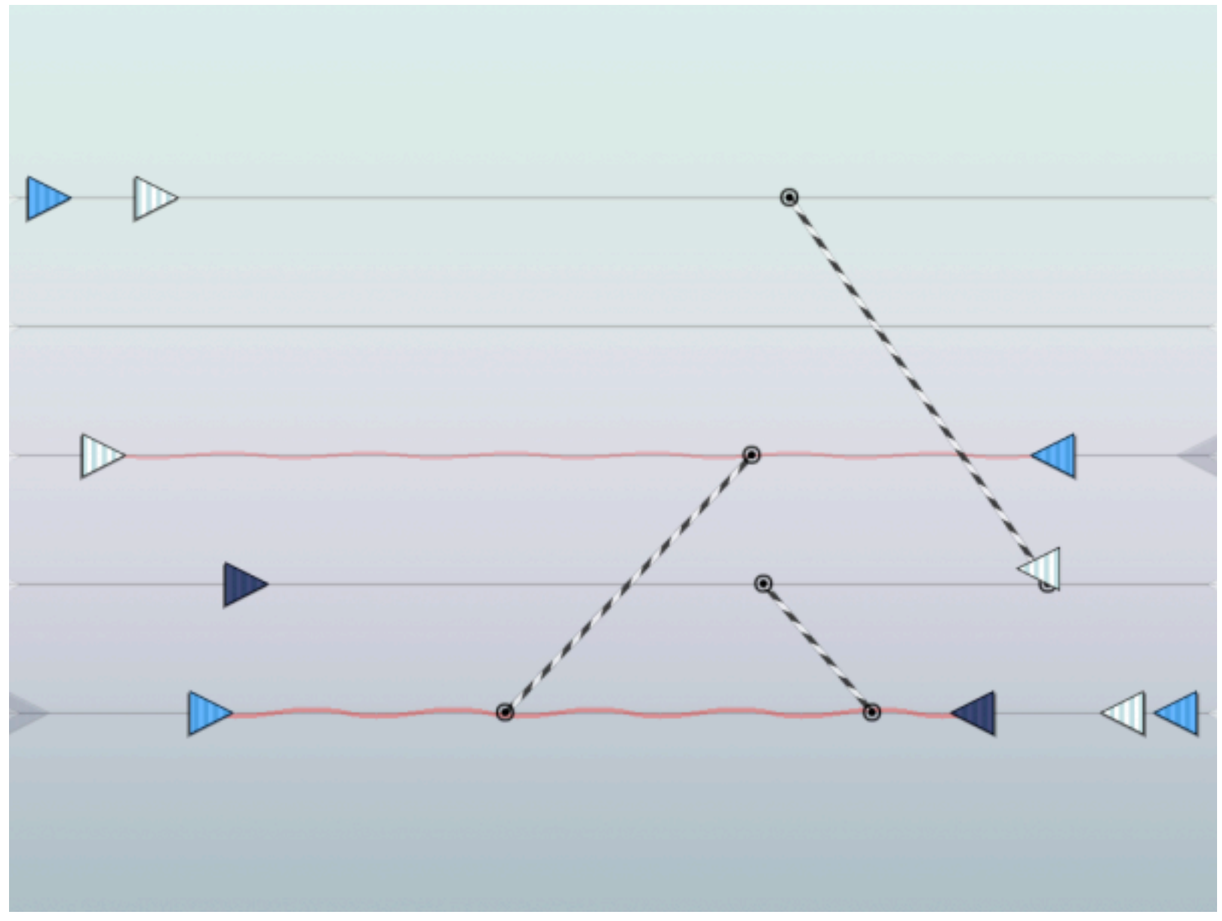






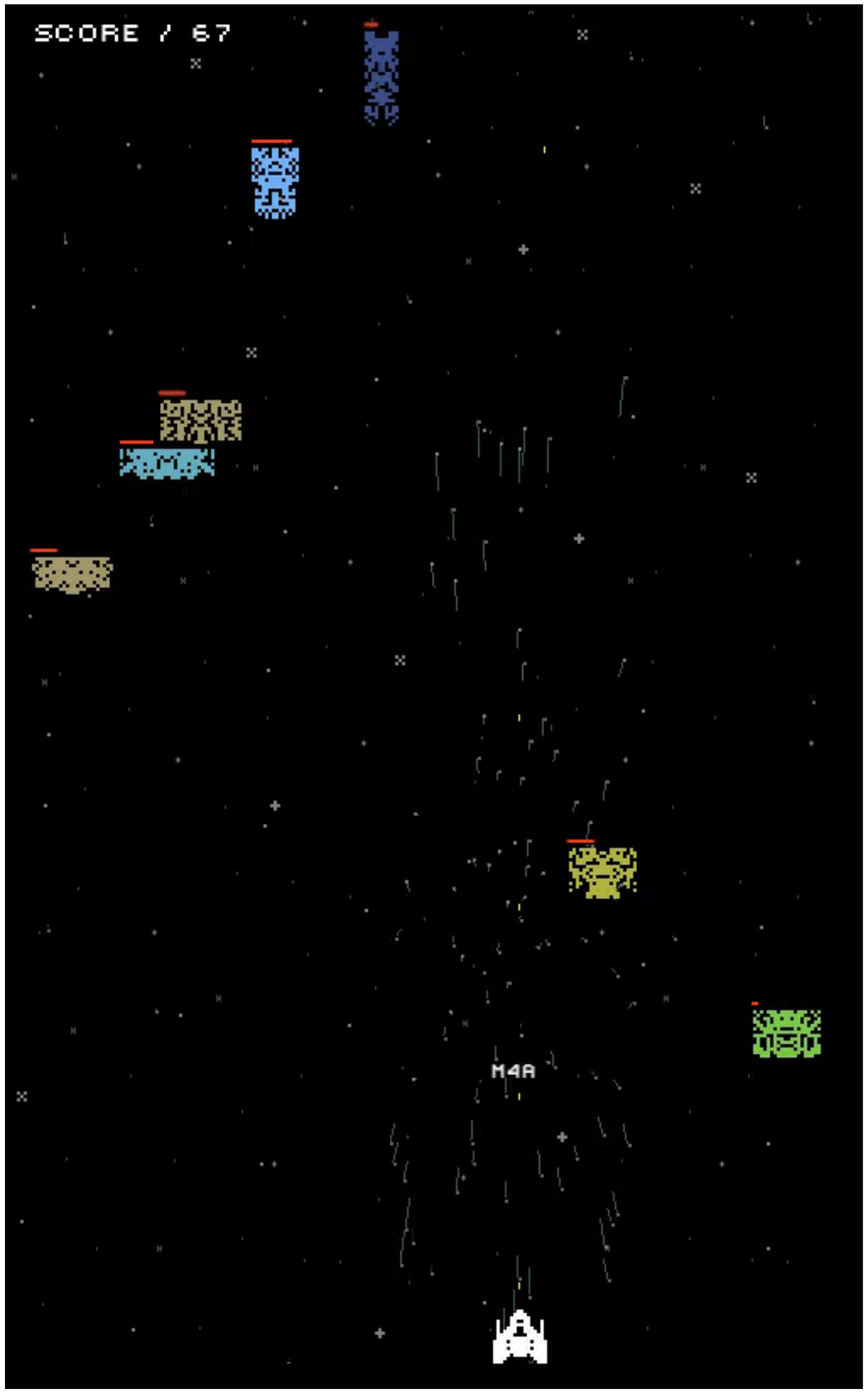






Tuesday, March 6, 12

to small scale critically acclaimed iPhone and iPad games



Tuesday, March 6, 12

to weird art games,

Hey, what did you want to talk about?

Beth:
its about what happened tuesday

Kevin:
??

Beth:
well.... I don't know if you even noticed, but I was pretty upset

Beth:
you were ignoring me like the whole night

Beth:
and you made me get a ride home with charlie, I thought we ewere going to hang out after

Kevin:
oh...

Kevin:
i didn't realize

Beth:
are you even paying attention now

Beth:
its like you don't "see me" anymore

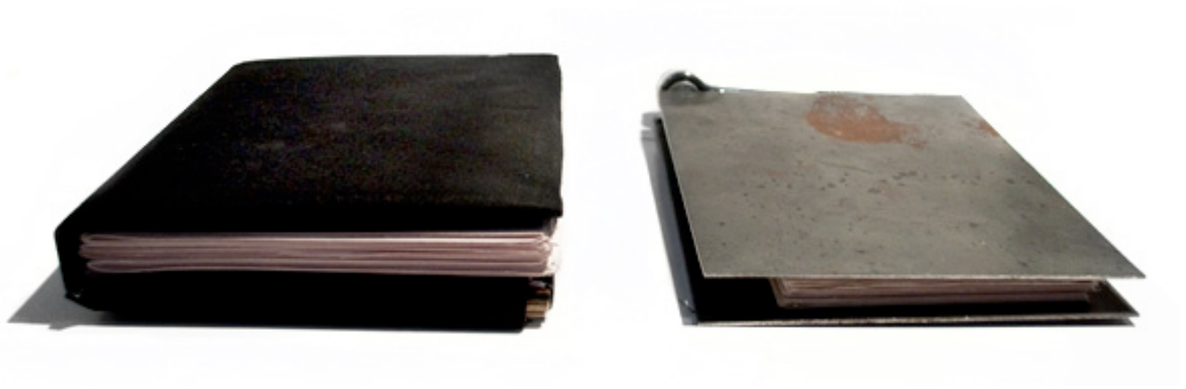
---

well it was j

Tuesday, March 6, 12

to chat
software

to interactive
sculpture



to aggressive books, I’ve designed or thought about interaction in dozens different digital and non-digital contexts.

In my experience, if you know what to think about, designing quality interaction is easy, and on touch systems, quality interaction often means designing for tactility.

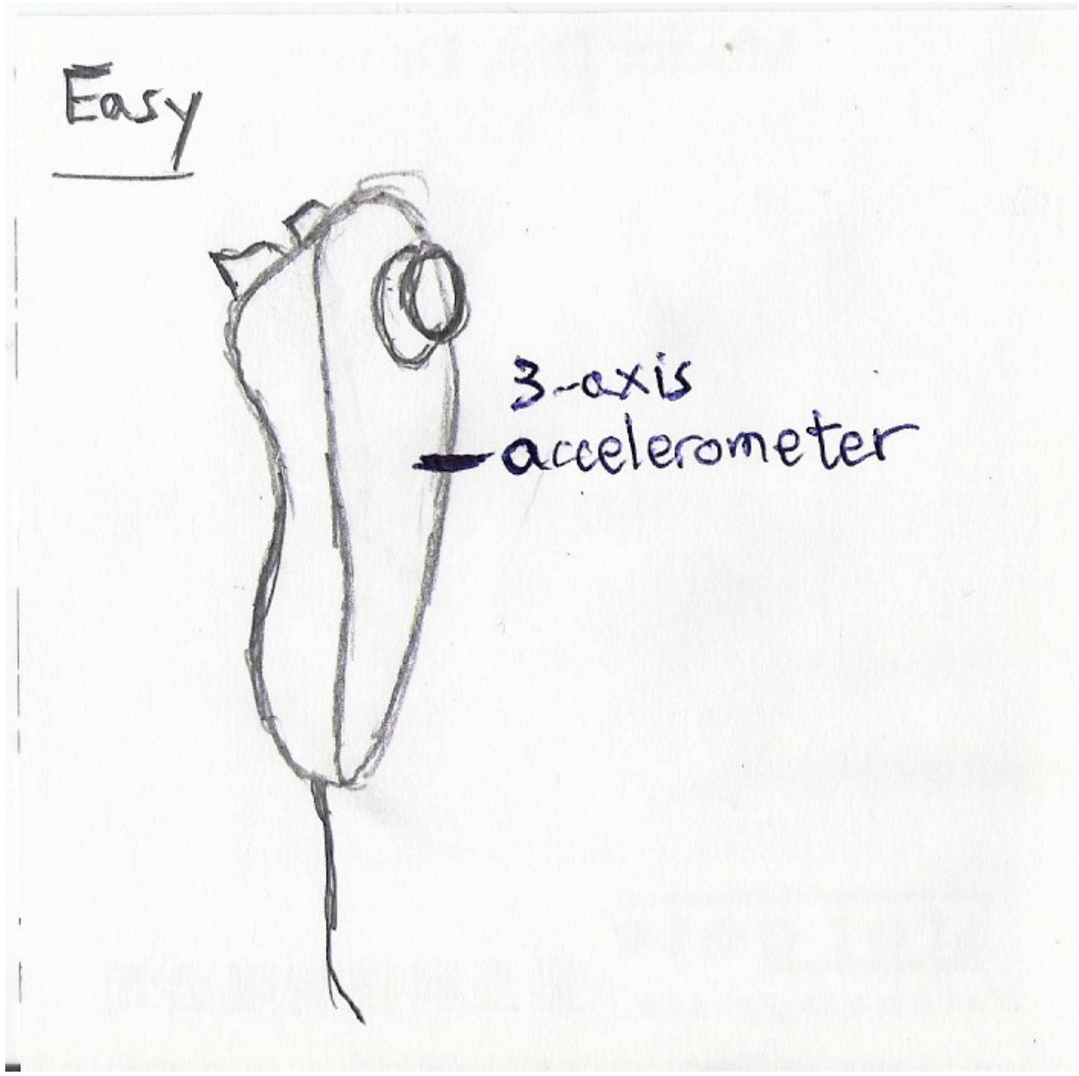
Tactility

When I talk about tactility, I'm not just referring to what you feel with your finger tips, I'm talking about the entire experience of physically interacting with a game. This presentation is an attempt to break my process down into rules that you can use to design your own games either for touch screens or ideally, any unfamiliar system of interaction.

Interaction possibility dictates game possibility.

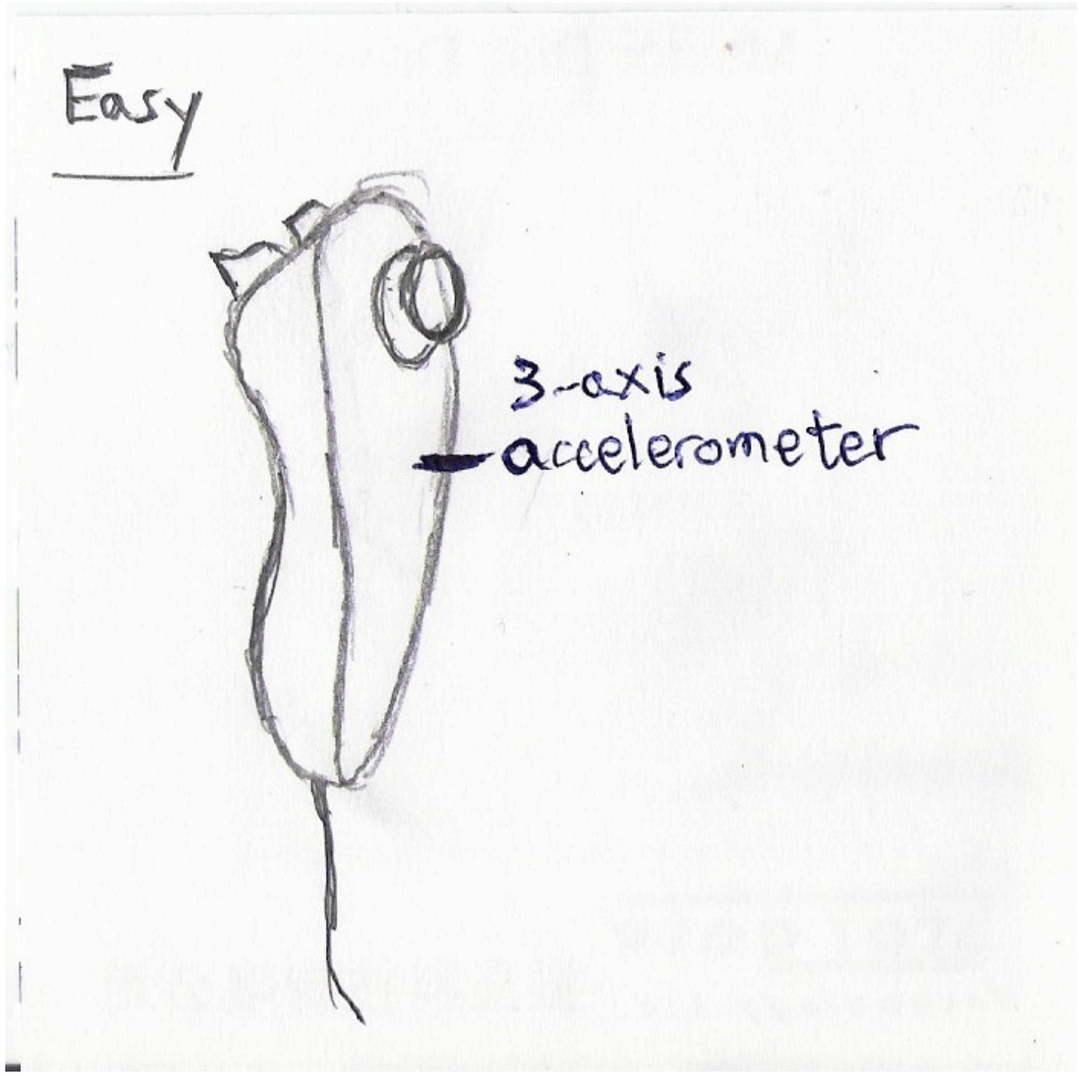
Tuesday, March 6, 12

As far as I can tell, the root of many unsuccessful iPhone control schemes comes from designers not realizing that when you make a game for a touch screen you have to design controls. Beyond that, you must design a game that is supportable by the types of controls that are possible to design on a touch screen. Most of us who have been making games for a long time should be pretty familiar with designing control layouts, as in, which buttons do what on a controller. Because we've been making games for this one interface for so long, its easy to forget that that controller was designed in the first place, and that the types of games we make for it are based as much on that design as they are based on the ideas we bring to the medium.



Tuesday, March 6, 12

Designing for touch interfaces is very different than just deciding what key or button does what. Just as designing a game for the wii or the kinect is different. It's the closest you'll probably come as a game developer to actually designing a physical interface for interaction.



Tuesday, March 6, 12

A long time ago nintendo got to invent things like the B button, the 4 way dpad, the analog stick, and the z-trigger, and just like them, today, when we make games for touch interfaces, we must tackle similar problems. And in case it it's not obvious, I'm not saying we should be using virtual joysticks. that is not a novel or successful solution.

Is it worth it?

And sure, you could say that this isn't worth it, that it doesn't matter, and just in case for some reason you decided to come to my talk but hold that viewpoint, lets just look at the most well known successful iPhone games, and think about how many of them use novel touch interaction mechanics. spoiler alert, all of them.










Tuesday, March 6, 12

Angry Birds, Flight Control, Fruit Ninja, Canabalt, Plants Vs. Zombies, the new Tetris, Cut the Rope, Tiny Wings, Infinity Blade, Temple Run, and Doodle Jump, and these are just a few of them. Nearly every Major iOS success has had either unique controls requiring a touch device, or extremely well thought out controls for the touch interface.

portability scares?

And in case some of you are thinking that building unique to touch controls isn’t a great idea, since that limits portability, consider these three reasons why that might be false.

portability scares?

1. first you must rock

1, it’s much easier to convince users to use sub-standard controls in a port of a successful game then it is to convince them to use sub-standard controls in an original game.

portability scares?

1. first you must rock
2. unique sells

2, games with unique or well thought controls make more money and have more success. Thats why you’ve heard of the games I listed a moment ago, and why a lot of games copy them

portability scares?

1. first you must rock
2. unique sells
3. touch is everywhere

And 3, as the iPhone is right now the mobile console to beat, nearly everyone else is implementing touch sensitivity on their devices.

ok, so how do i do this?

Ok, so designing controls specifically for your game is a good idea. How do we do it? how do we look at something we don't understand and come up with ways for players to interact with it successfully?

Two Rules

I use two rules. Any successful and new control scheme should have two extremely important features:

Two Rules

1. Comfort

1. Comfortable feel.

By this i mean, the gestures or moves that players make, should feel at the very least, comfortable. Better than that, they should actually be pleasurable. Designing a game with uncomfortable gestures will alienate a lot more of your audience than you might expect, and conversely, feel good gestures combined with appropriate gameplay are one of the strongest draws games have. Think pulling back the catapult in angry birds, swinging a racket in wii tennis, or pulling the trigger in any shooter.

Two Rules

1. Comfort
2. Confidence

Tuesday, March 6, 12

2. Confidence.

Your control scheme must provide Primary Feedback that allows practiced players to be 100% confident in their inputs.

Ok this one is a little bit more complicated.

Primary Feedback
vs.
Secondary Feedback

Tuesday, March 6, 12

I like to think that game feedback comes in two categories. Primary (plus optional reenforcement) and Secondary. I split them up like this, because although they are both important, they are each important for different reasons. Having stellar secondary feedback will not solve the problem of having poor primary feedback, and visa-versa.

So what makes up these categories?

Primary Feedback

Tuesday, March 6, 12

Primary feedback is the information players get from their body.

Primary Feedback

Touch, Proprioception

This can be coming in from their sense of touch (at up to 1000fps), and from their muscle memory for body sense (also called proprioception). The success of failure of this feedback is entirely based on your game's input control design, and will be the main focus of this talk.

Primary Feedback

Touch, Proprioception

(opt) Primary Reinforcement

Ears

The optional reenforcement to this generally comes in through players ears, at many hundreds of frames per second, from game sounds specifically cued to input choices (think mario's jump sound).

Primary Feedback

Touch, Proprioception

(opt) Primary Reinforcement

Ears

Secondary Feedback

Secondary feedback comes in

Primary Feedback

Touch, Proprioception

(opt) Primary Reinforcement

Ears

Secondary Feedback

Eyes

through player eyes, often at less than 60fps, and is -never- the solution to fixing poor primary feedback.

invisible

Tuesday, March 6, 12

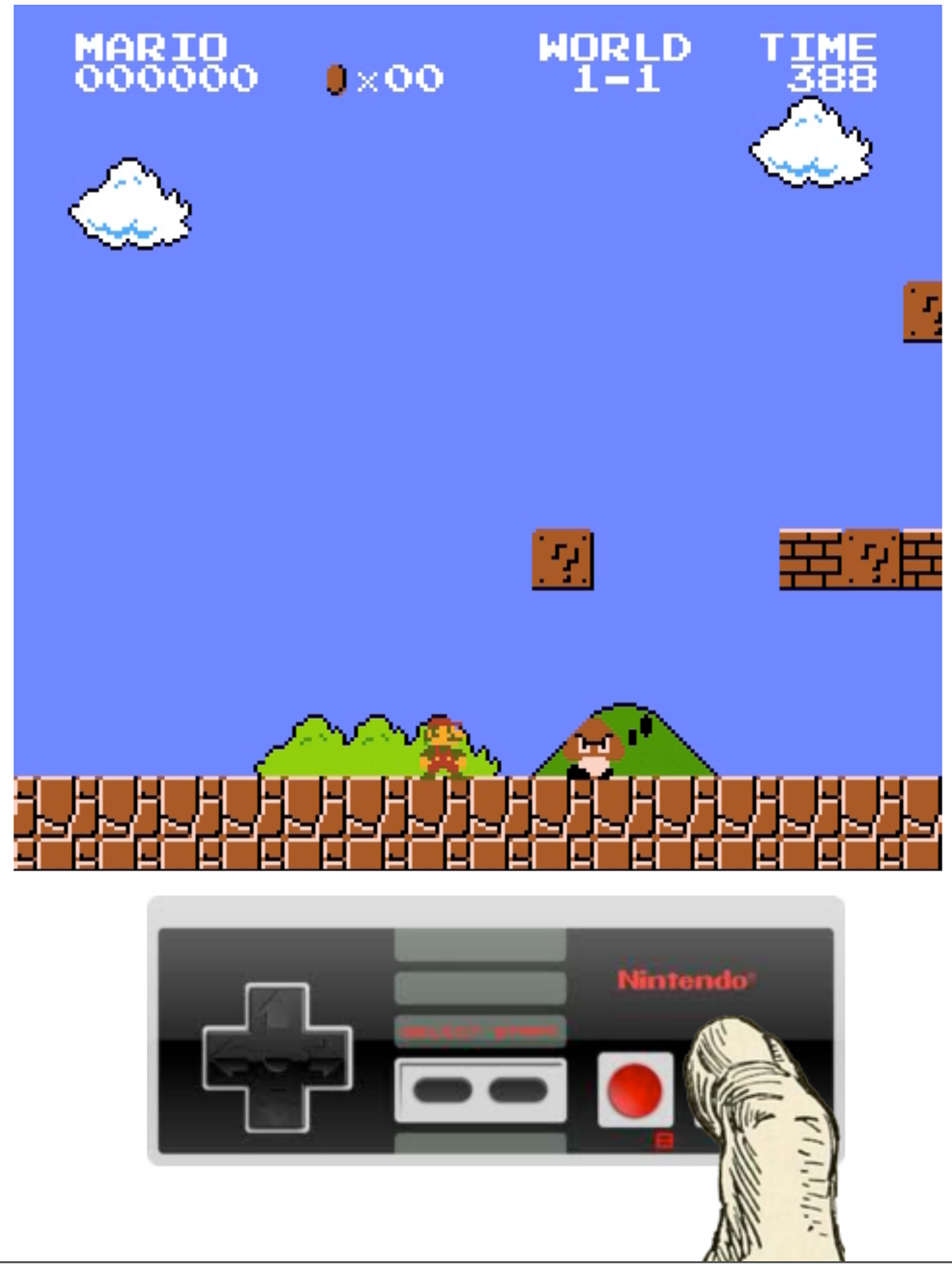
The goal of these feedback mechanisms is to convey the game-state to the player as fast as possible. to make players feel as confident as possible which will drive their ability to embody their character, or if there is no character, to make the virtual world displayed seem as real as possible.

In other words, the goal of these mechanisms is to make the action of interacting with a game entirely invisible to the player.

Tuesday, March 6, 12

To illustrate this clearer lets look at where our information is coming from when we play mario.

If you weren't thinking about controller design, you'd likely play mario and notice that when you push right, mario moves right, when you push A, mario jumps. This is important feedback for new players, as it teaches them how the game functions. You try an input, you see a result, you try another input. But as far as I'm concerned, this visual feedback loop actually represents the second feedback that players are receiving about their in game actions.



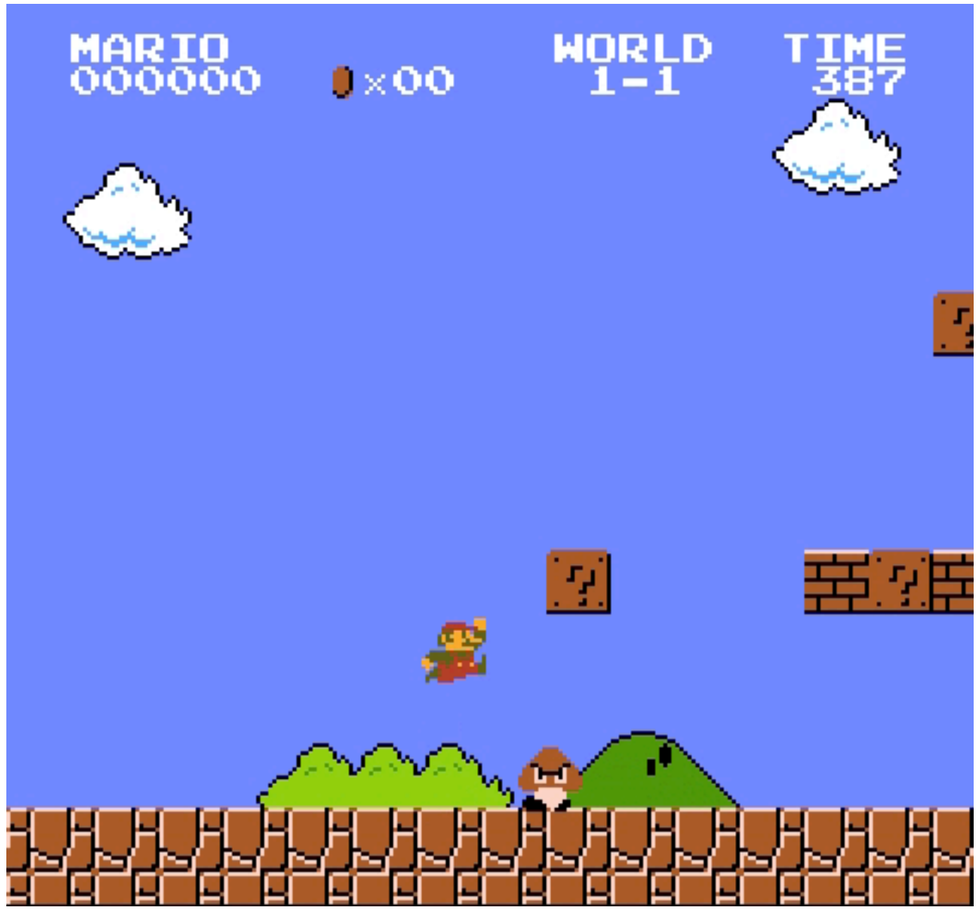
Tuesday, March 6, 12

The first feedback their receiving is through their fingers.

When a practiced player plays mario, she presses right and knows that mario will move right because she feels the button go down, and has an understanding of the state of the game prior to the press.

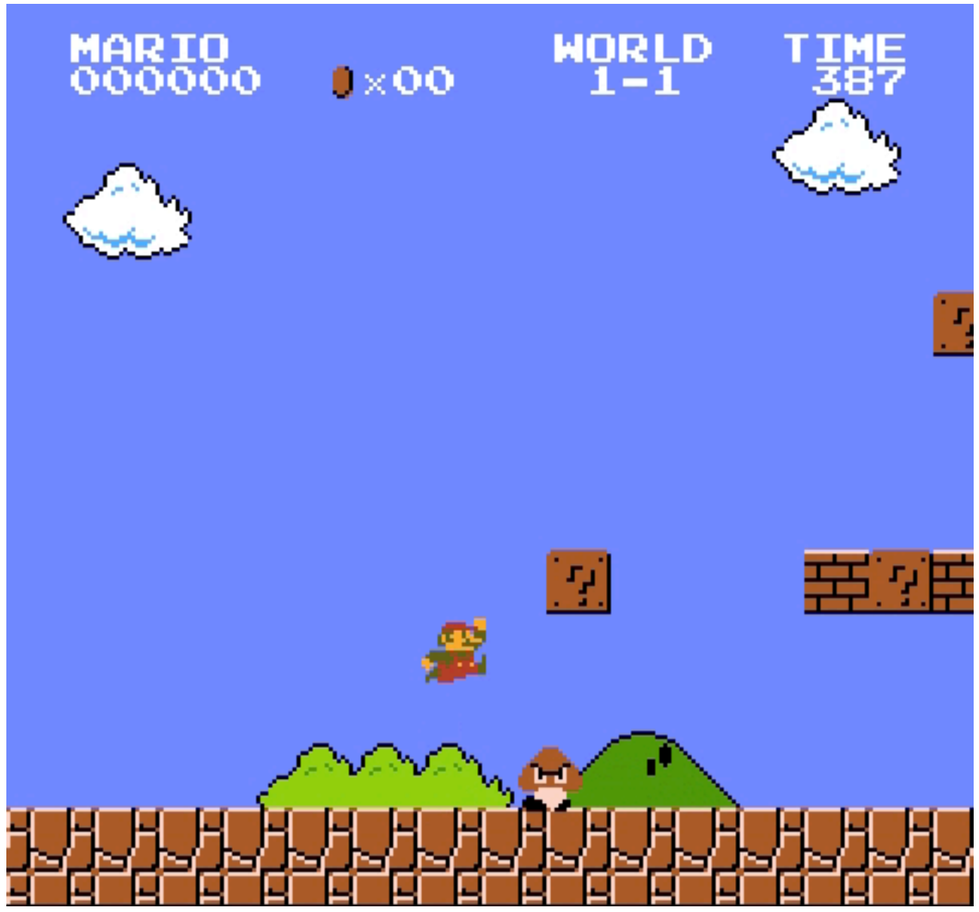
This is easy to demonstrate, heres a screenshot of mario, imagine I'm pressing the A button, what will happen next?

Clearly mario will jump. I don't need to see the next frame of the game to understand that, I have an intuitive understanding because I can feel the A button go down, and my proprioceptive system is ensuring me that my fingers are in the right places. (And as a side note here, the reason you get a re-enforcing sound when mario jumps is because the A button is often context sensitive. While the right button will almost always make mario go right unless he's obstructed which is an obvious situation, the A button will only make him jump if he's on the ground, which can be non-obvious when playing at speed).



Tuesday, March 6, 12

oh heres the next screenshot in case you werent sure if he would jump.



Tuesday, March 6, 12

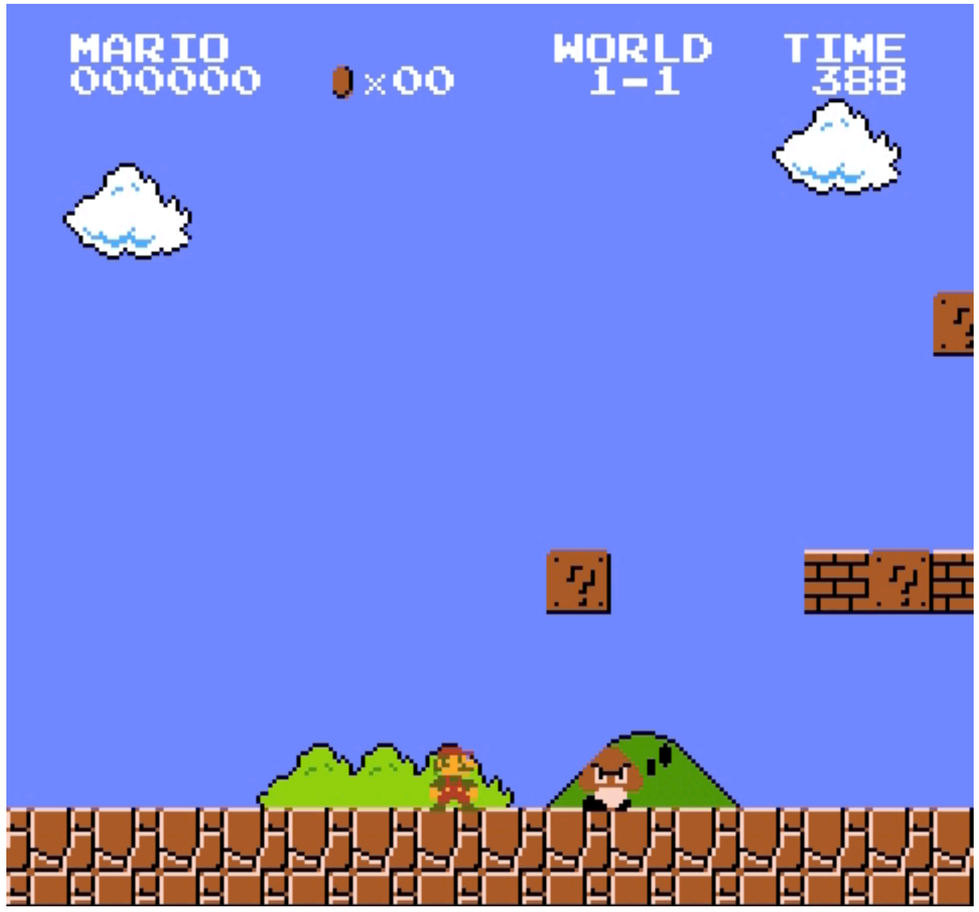
When a novice plays mario, they may not be familiar with the controller, they might not know how to tell the difference between an A and B button from their proprioception yet. They might have to look at the controller a lot when they want to jump as opposed to throw a fireball. This is normal.

bad control design

=

novice play forever

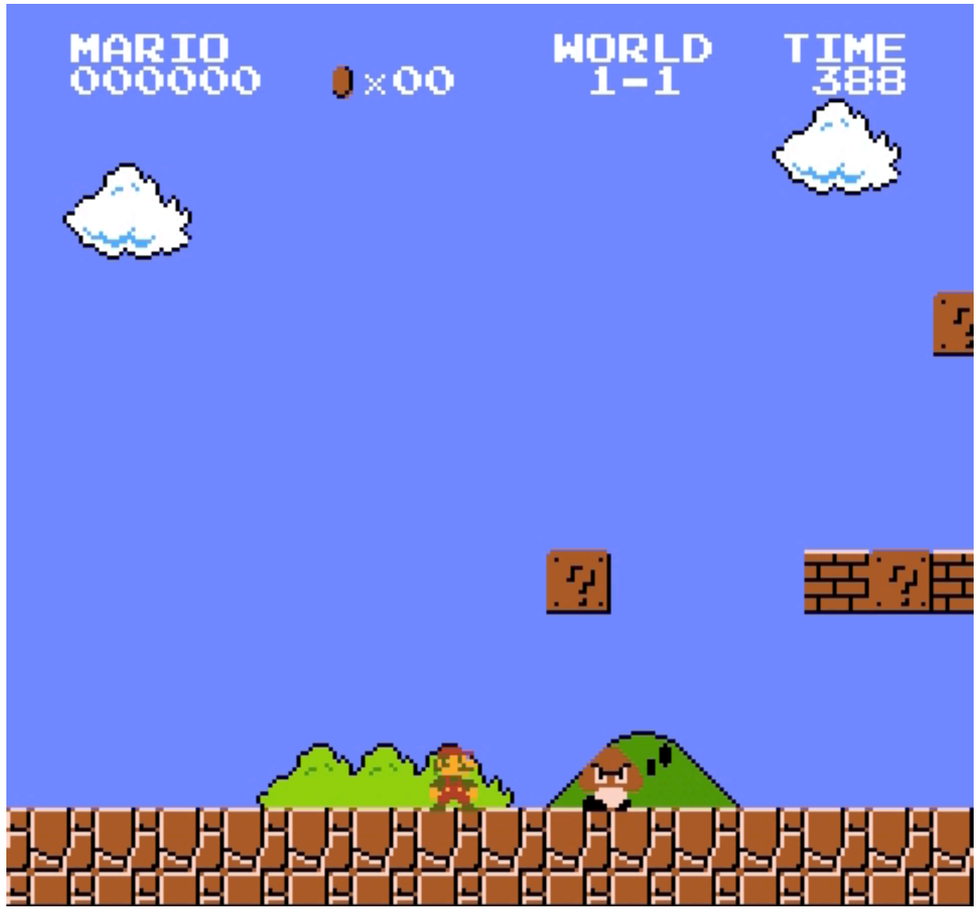
The difference between a good controller design and a bad one, is that with a bad controller, the player never becomes confident at the position of the buttons.



Also easy to demonstrate. Imagine I am a novice player, I'm pressing one of the buttons on the right. what will happen in the next frame? We have no idea. The only way we can find out is by using secondary information, essentially waiting for that frame to render, and determining what has changed. Now all this probably seems incredibly obvious, but the thing about primary feedback, the thing that makes it tricky, is that you can't do it half way.

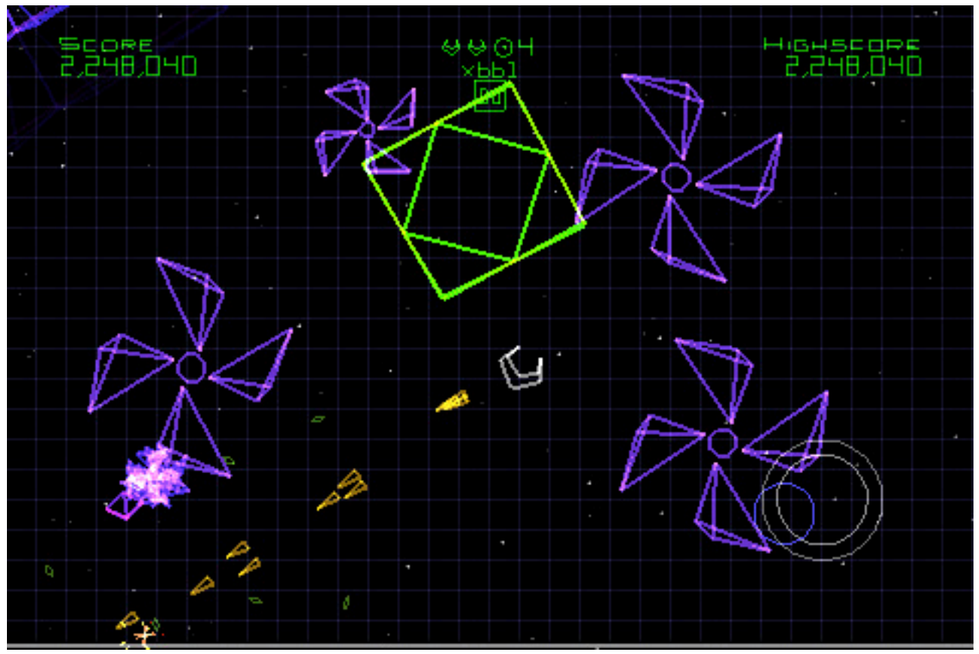
100% confidence

It has to be 100% accurate or players will not feel confident with their controls, and confidence is the goal of primary feedback. To go back to the screenshot,

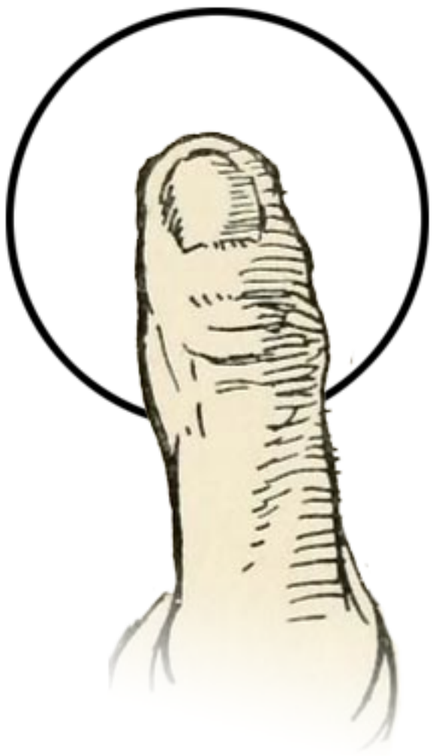


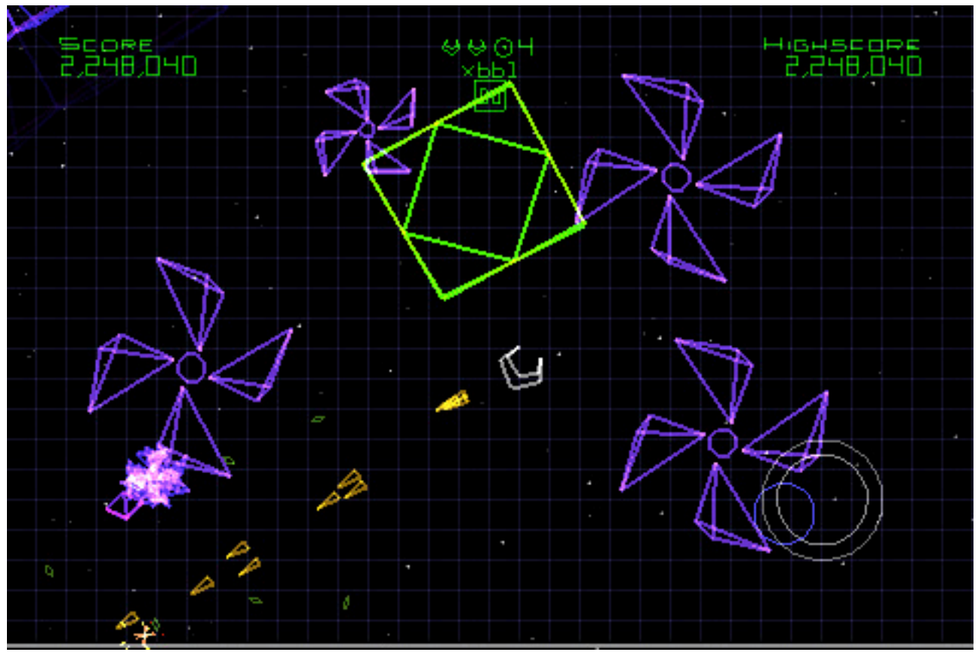
Tuesday, March 6, 12

imagine I’m 80% sure I’m pressing the jump button, what will mario do next frame? Not good enough. Even 90% is not good enough for an action game like mario. If the player is wrong here, mario could die.



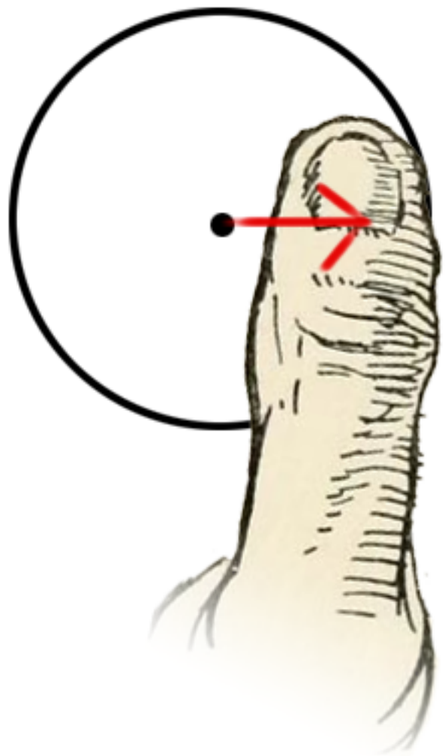
When we take it back to the iPhone context, you would be shocked at how many games fall into this ‘good enough’ trap. Here’s a screenshot of geometry wars, a game that uses a virtual joystick that centers based on player thumb placement.

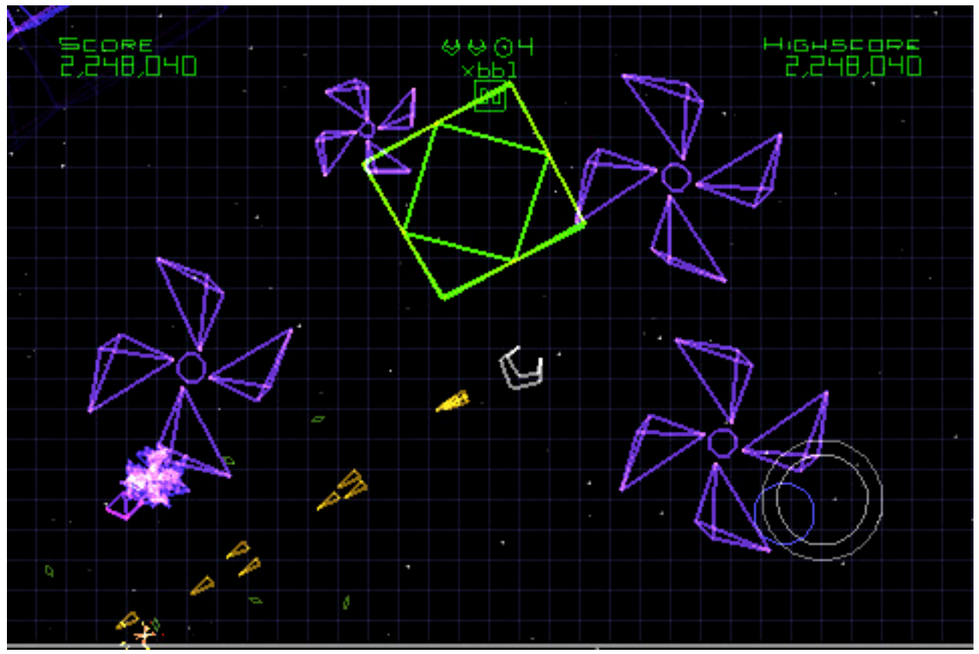




Tuesday, March 6, 12

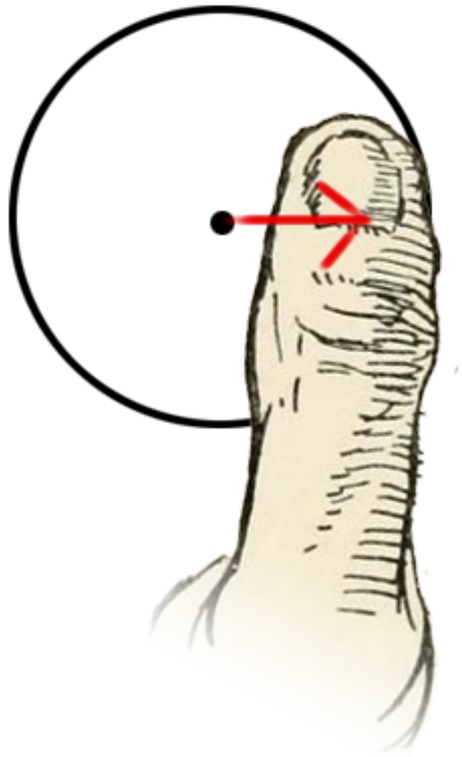
So imagine, I put my thumb down, it’s centered, thats easy.

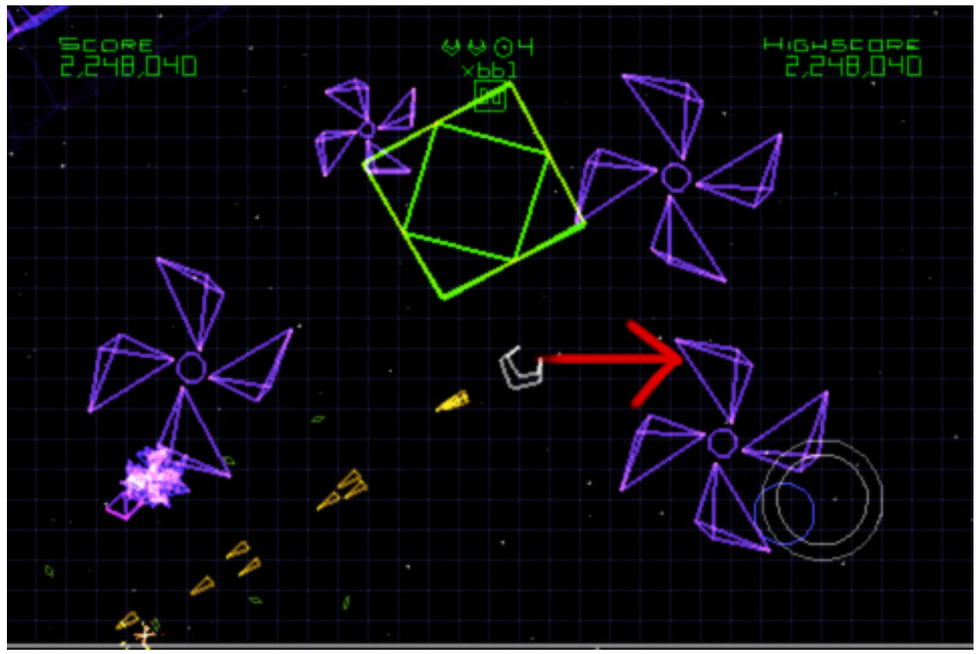




Tuesday, March 6, 12

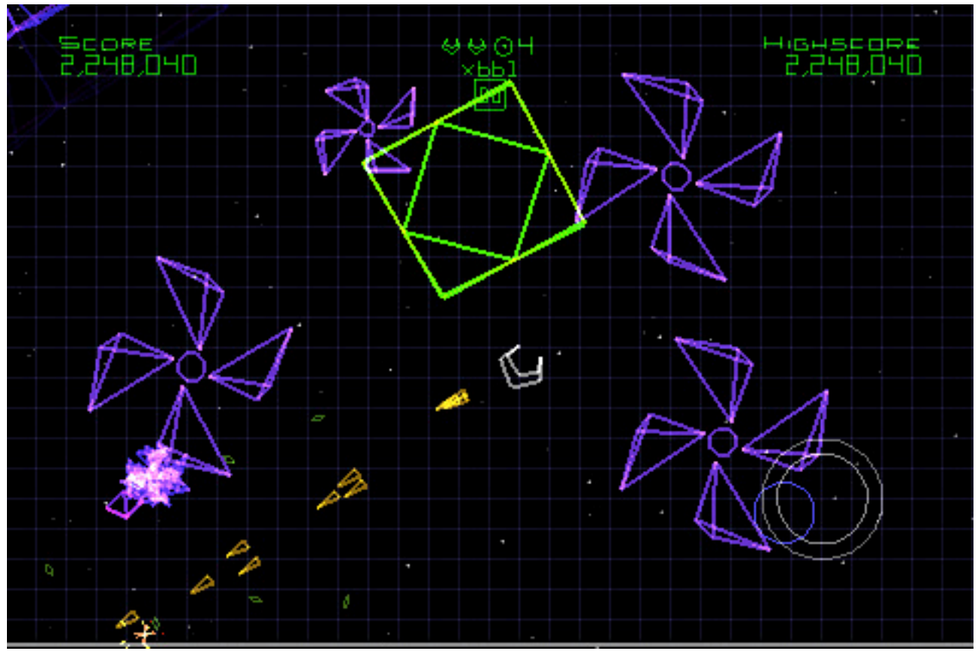
I move it right, this is a gesture, its easy to learn because of my proprioception, and the ship follows my movement. Easy to guess. Lets look at a picture, which way will the ship go?





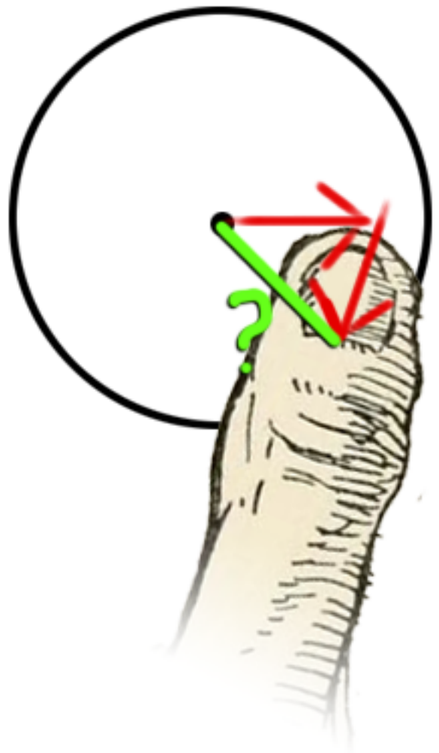
Tuesday, March 6, 12

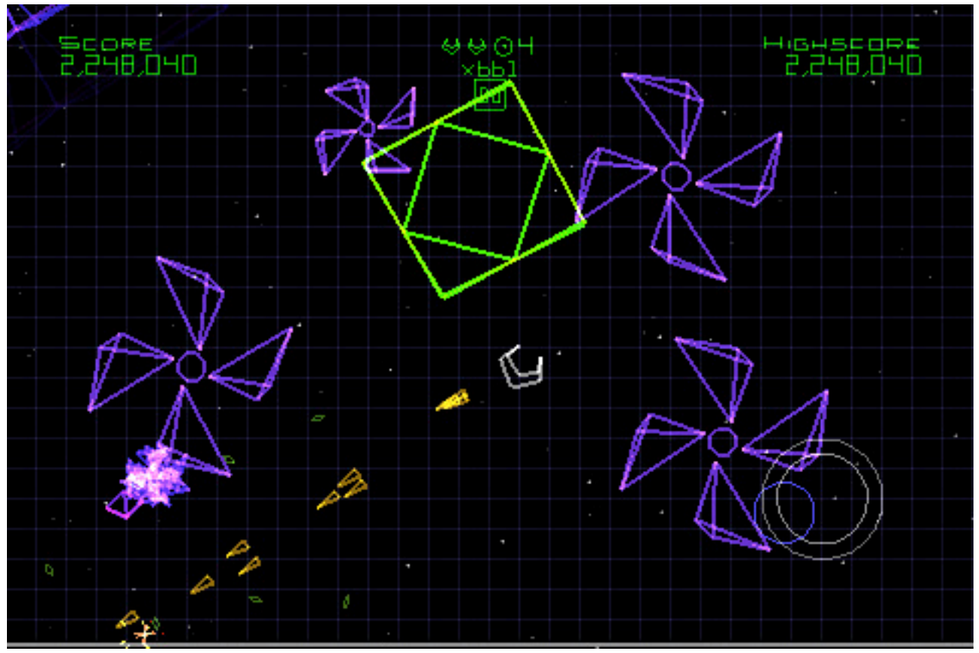
Easy.



Tuesday, March 6, 12

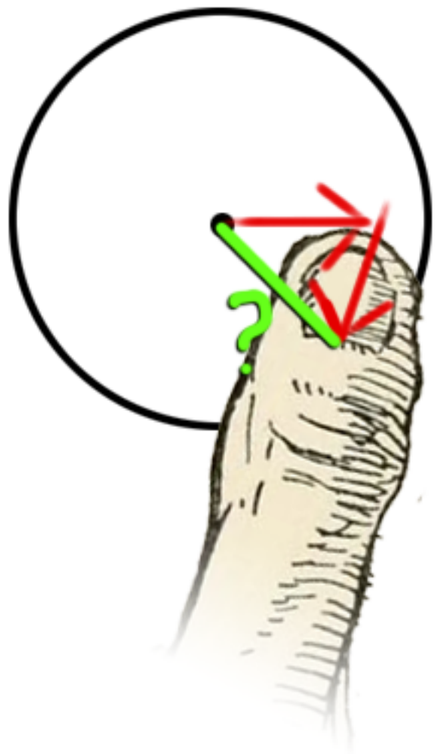
Now, I move my thumb down to go a different direction, look at that picture, which way will the ship go?

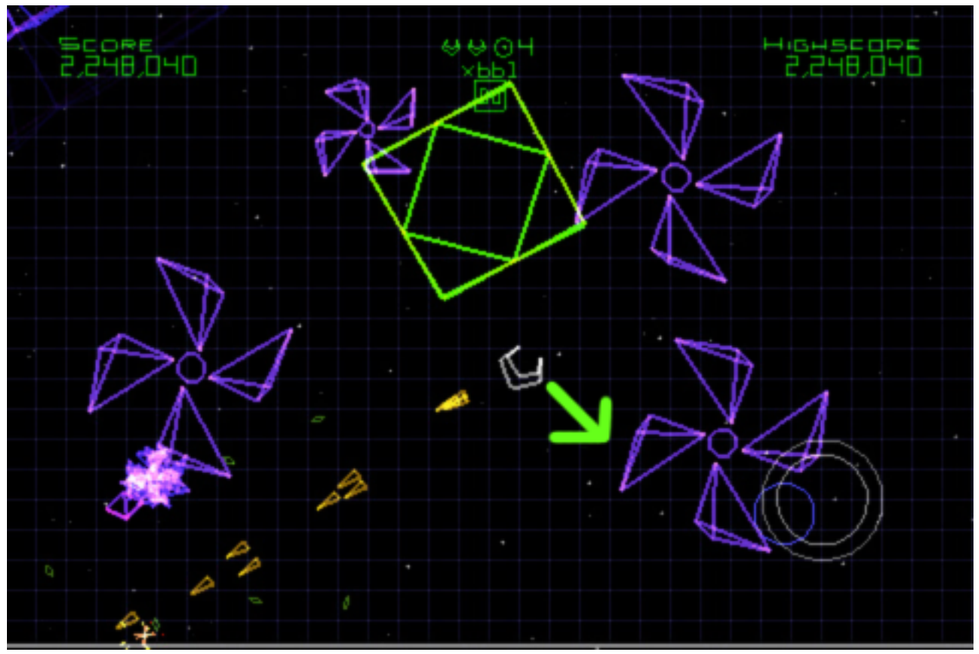




Tuesday, March 6, 12

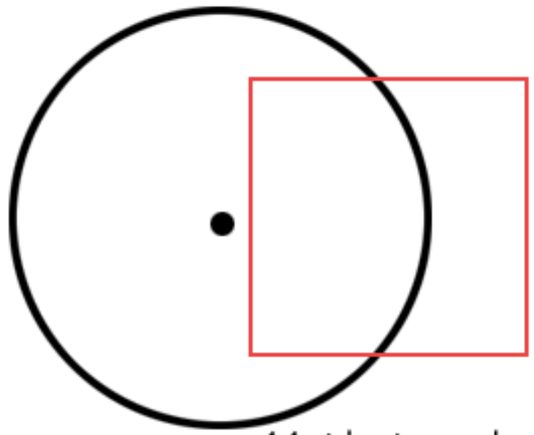
Notice how much longer it takes you figure out the ship orientation, try it with your finger in the air.



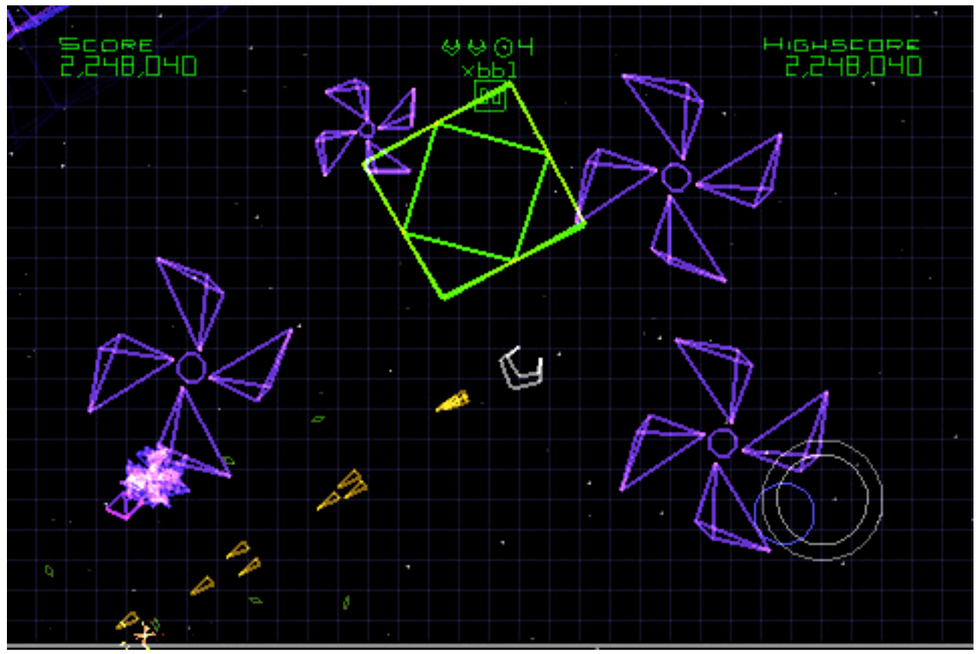


The problem here is that you're taking the benefits players gain from their proprioception, our internal understanding of directed motions, and mixing them with another system, spatial comparative reasoning. Instead of understanding how our thumb moved, we're trying to figure out it's relative position compared to an arbitrary point in space. Suddenly this system is not fast enough to provide primary feedback, and players will fall back on their secondary systems of watching where the ship goes, and basing their inputs on that. This is not a successful input system, and LOTS of iphone games use it.

And the problem here isn't because of the virtual joystick being mobile. Even with a fixed virtual joystick, the amount of space that your thumb moves in, compared to the size of your thumb, doesn't give an accurate reading,

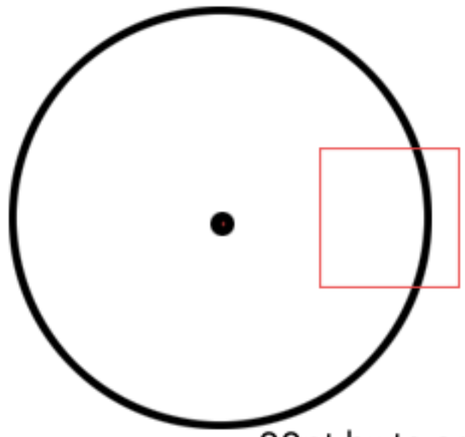


44pt bx to scale
with GW joystick

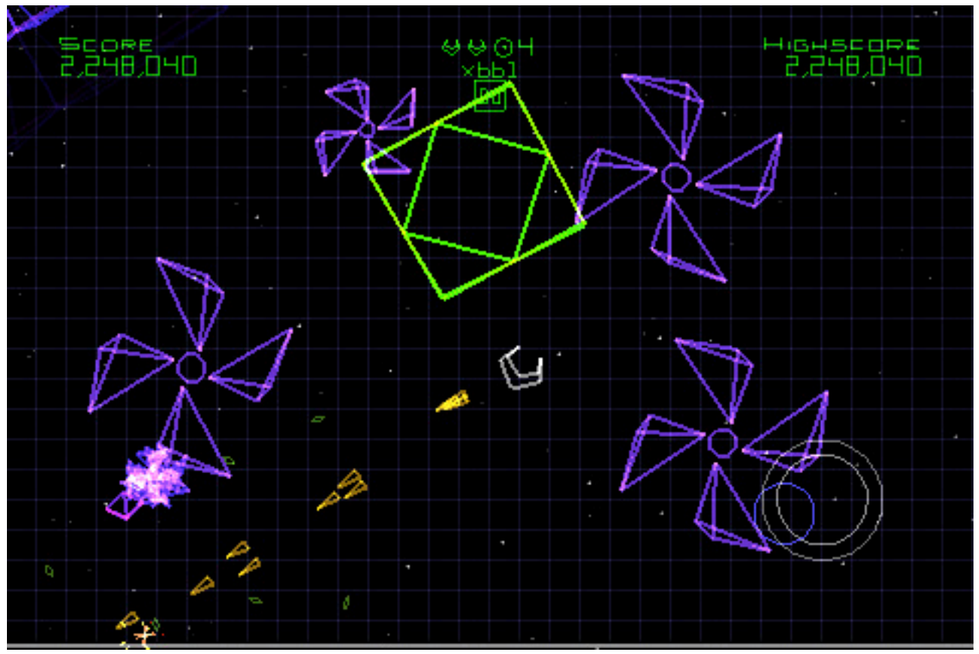


Tuesday, March 6, 12

for example, here is a joystick with a thumb box, what direction am I going? Apple, who has done a lot of research into touch, recommends that your buttons be at least 44p by 44p (points are sort of like pixels based on dpi. So 44 points is 44px on a non retina screen, 88px on a retina), anyway, so if we assume that a very player looking at their thumb

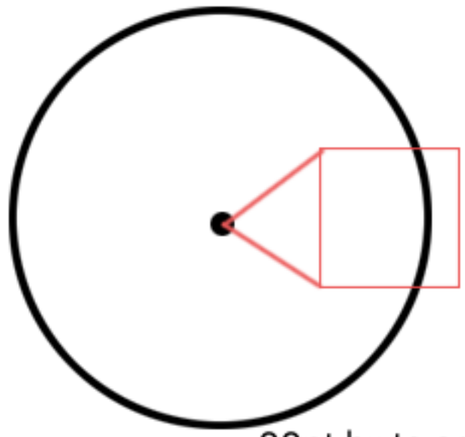


22pt bx to scale
with GW joystick

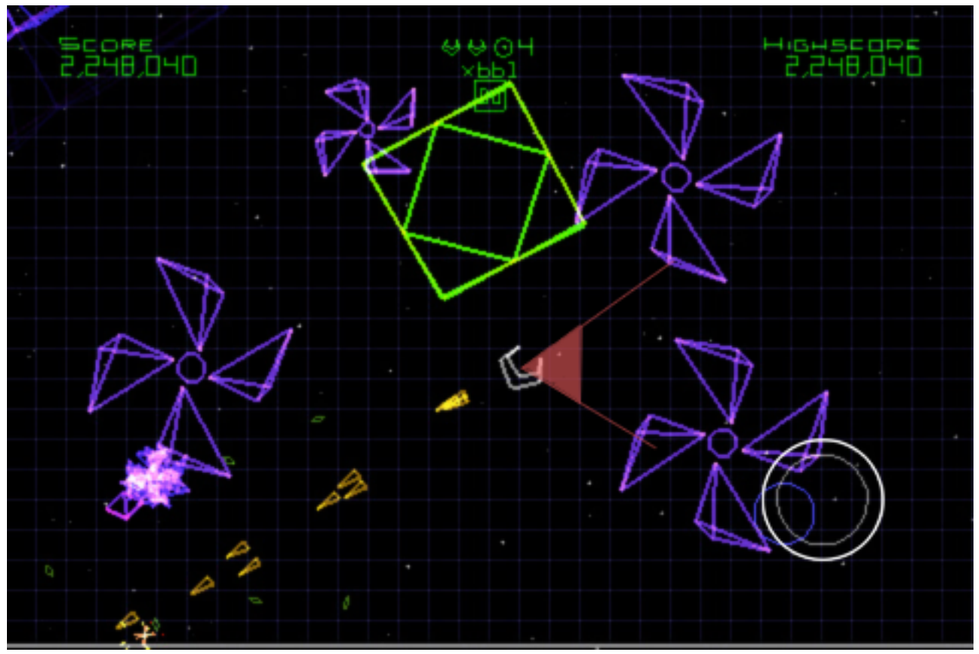


Tuesday, March 6, 12

could have up to maybe a 22 point variance, half of apple’s researched number, thats a lot of directional options for this ship.



22pt bx to scale
with GW joystick



Tuesday, March 6, 12

You can see that its roughly a slice of a pie chart. Pretty unacceptable considering the top and bottom of this will put me into an enemy. And this is assuming with all my movement that I still remember the *exact* location on the screen of the joystick. Imagine I'm off by only half a centimeter, that would produce an even larger variance.

how come it works with physical analog sticks?

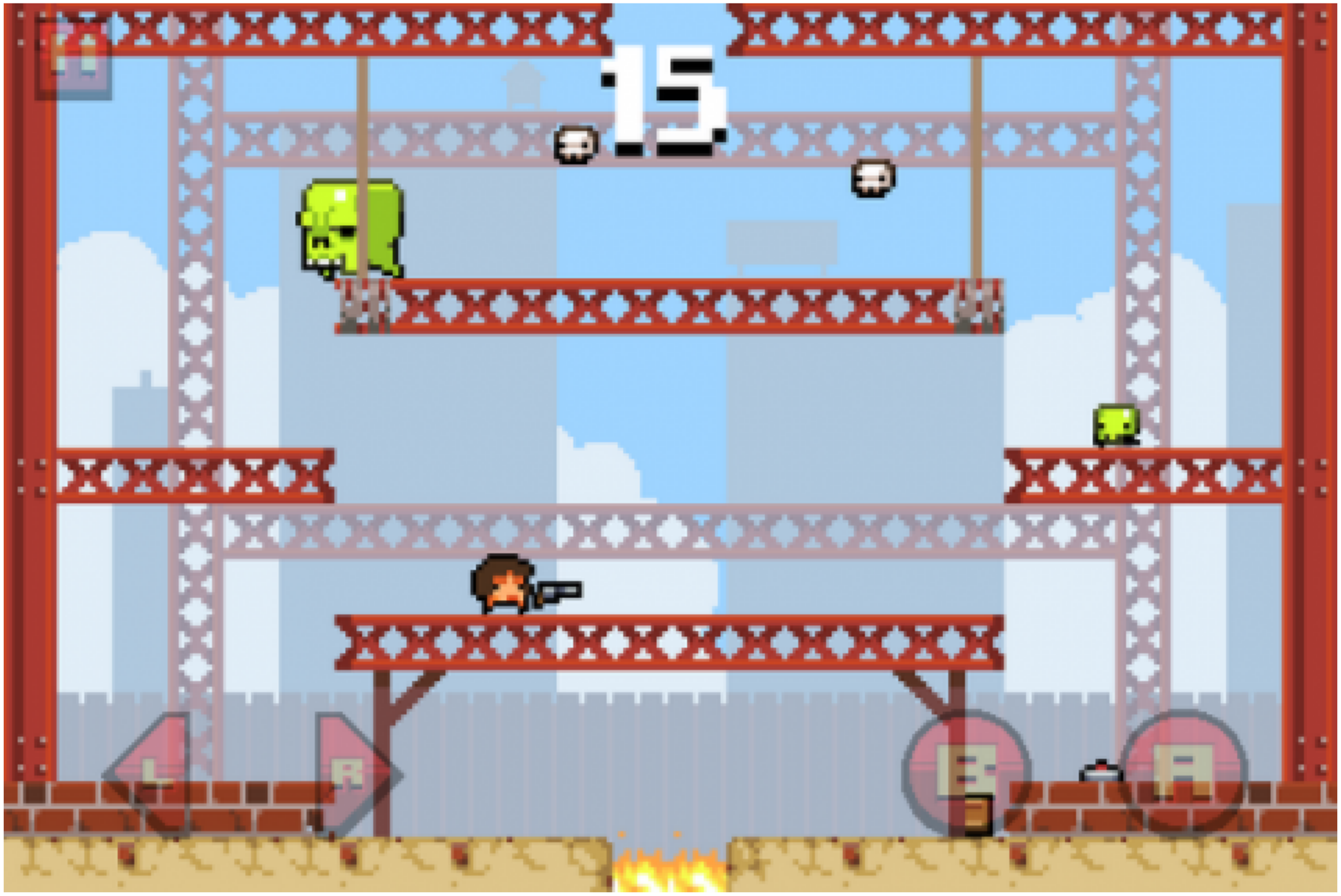
You might now argue that well physical joysticks have a similar problem, but think about the last time you used one, you know how physical joysticks have that spring pull back to the center? Thats to let your thumb know where the center is at all times, so you can understand the exact direction that you're going, and you can understand the degree to which you're going in that direction by the amount of pull. It's no mistake why physical analog joysticks are designed precisely how they are. We have none of these affordances with a virtual input.

virtual d-pads, impossible?

Now this isn’t to say that virtual controls can’t work.

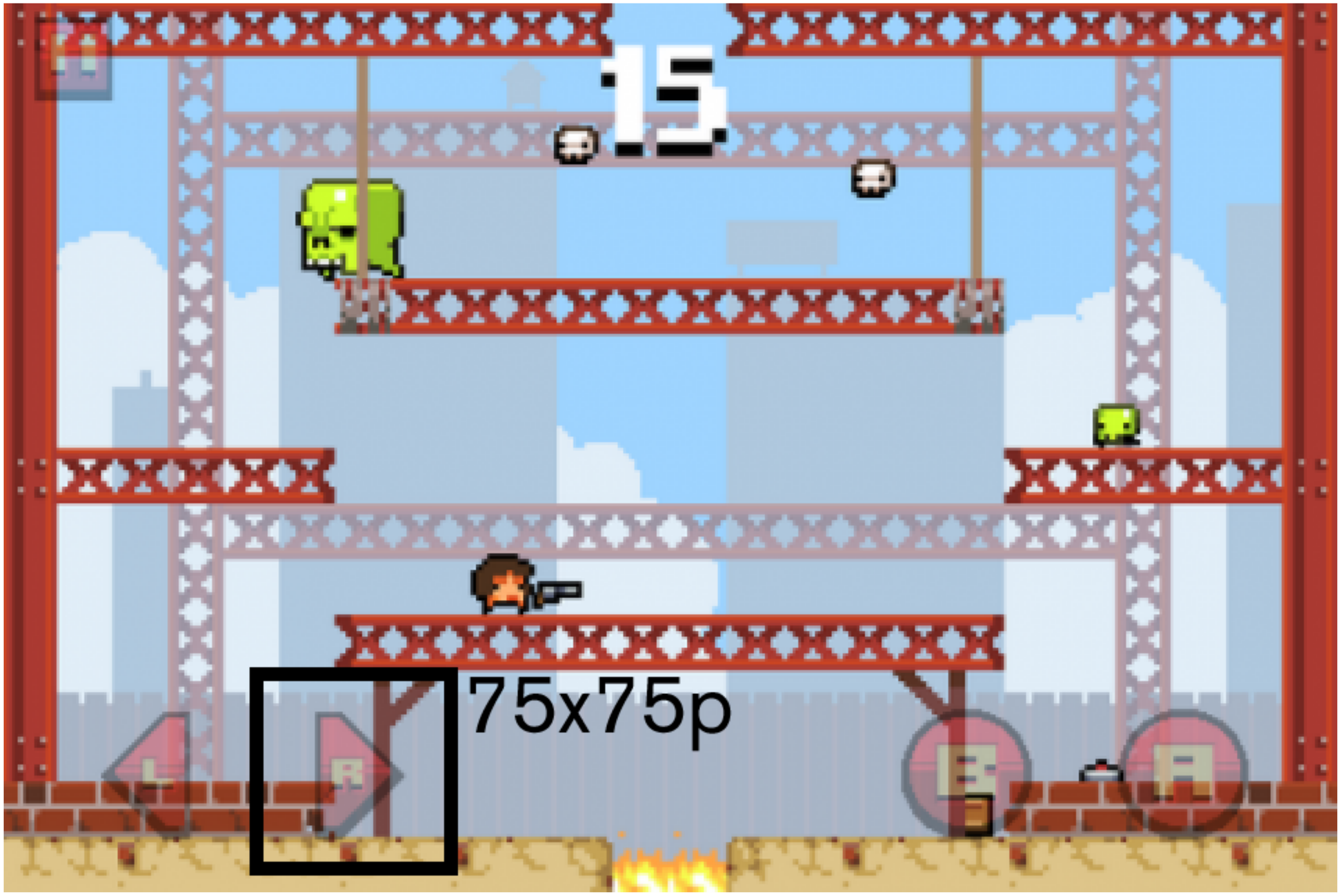
I’d argue that they can’t ever work as well as true touch controls, but they can be pretty good if they take into account all this thinking.

For example, Super Crate Box has pretty good virtual dpads. How did they accomplish this?



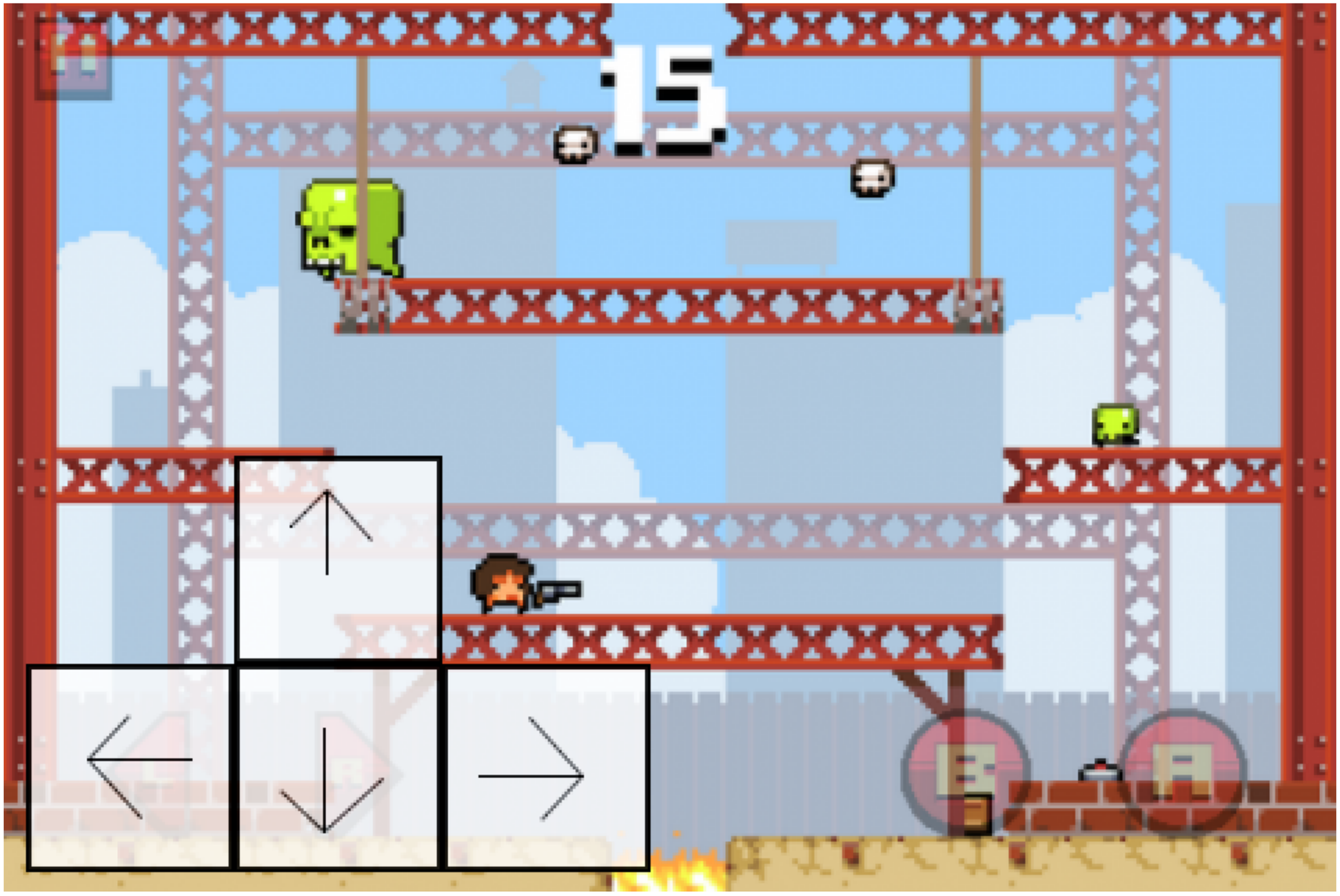
Tuesday, March 6, 12

Keeping in mind apple’s 44 points number. You’d imagine that to be accurately touchable when not looking, buttons would have to be quite a bit bigger than that, and no surprise, super crate box’s hit area is approximately 75x75 points.



Tuesday, March 6, 12

This is a very large button, and it's a very large button that does -one- thing, unlike an analog joystick. Obviously some players will -still- be uncomfortable with this method, but if they're practiced at using vitual pads, I think its safe to say that they'll find super crate box's to be some of the best and very capable of letting them be 100% confident in their decisions. This shouldnt be a surprise, because the entire virtual input system in super crate box is built around increasing the players confidence that they have input what they meant to input. This isn't rocket science. It's actually pretty straightforward, you just need to spend time thinking about it, and crafting your game realistically.



Tuesday, March 6, 12

For example, if we need 75x75 points to be 100% sure of a tap on a virtual button, its easy to see why a virtual joysick, or 4 way dpad just isn't going to work. Not only does it take up a ton of screen real-estate, but any time you make the player move their thumbs over too much space much you risk them losing their spacial understanding and needing to look at their hands. This can even happen with physical controllers.

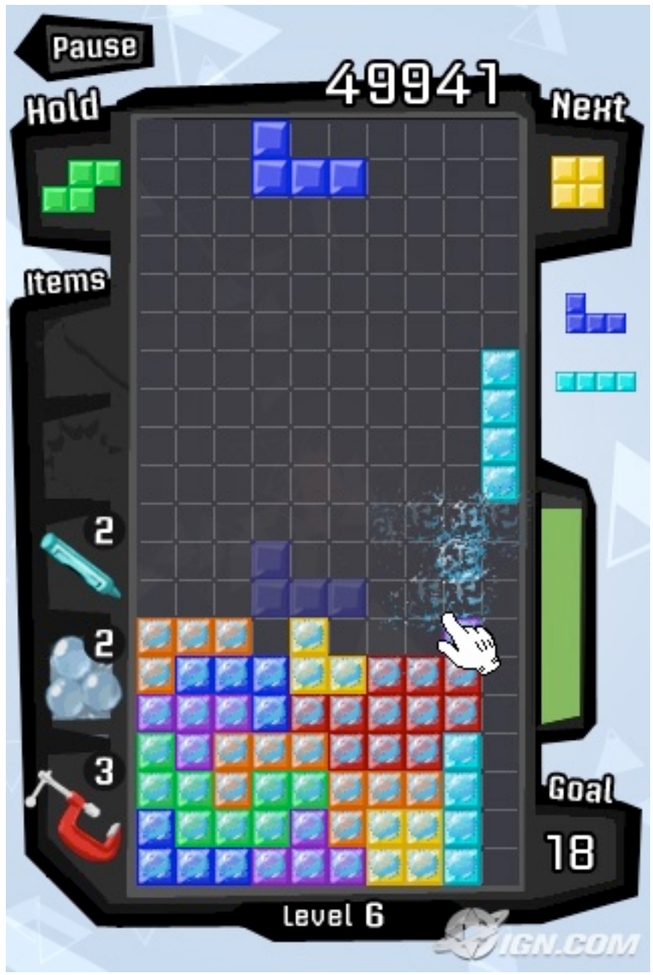
Tuesday, March 6, 12

Theres a good reason why the jaguar controller sucked

A Case Study: Tetris

So lets look at a case study here, Tetris for the iPhone. This is a good reference point because Tetris was originally released in 2009 and had downright terrible controls, both in confidence and comfort. I made a game called Unify based on the idea of what a block dropping game designed for a touch screen would look like, and then Tetris re-released in 2011 with an amazing control overhaul.

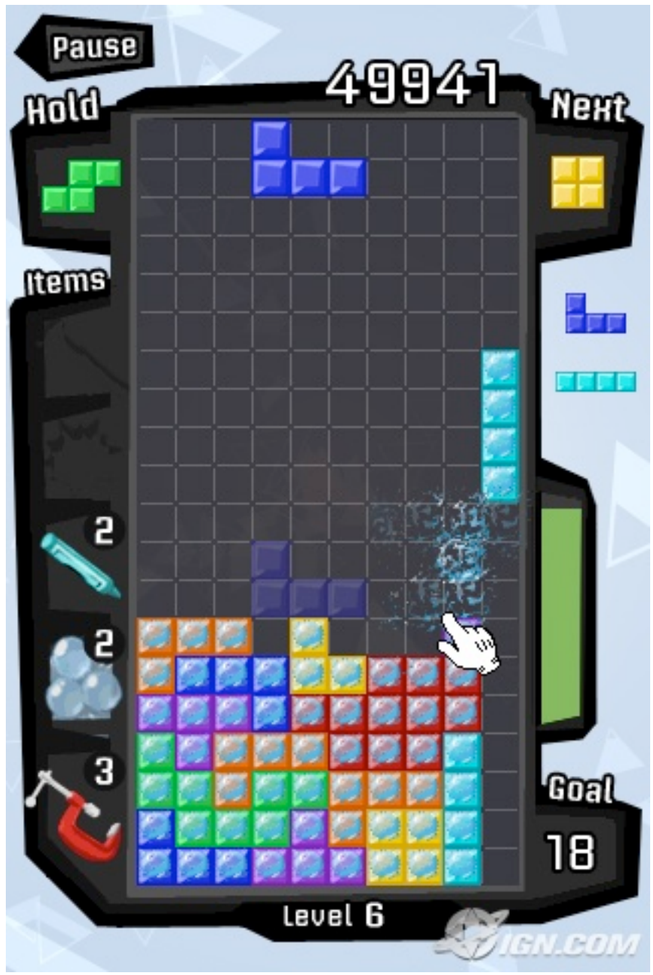
Tetris (2009)



Tuesday, March 6, 12

So lets start with the 2009 Tetris release. Players move the blocks in this game by sliding their finger left and right, swiping down to drop a piece, and tapping left or right to rotate left or right.

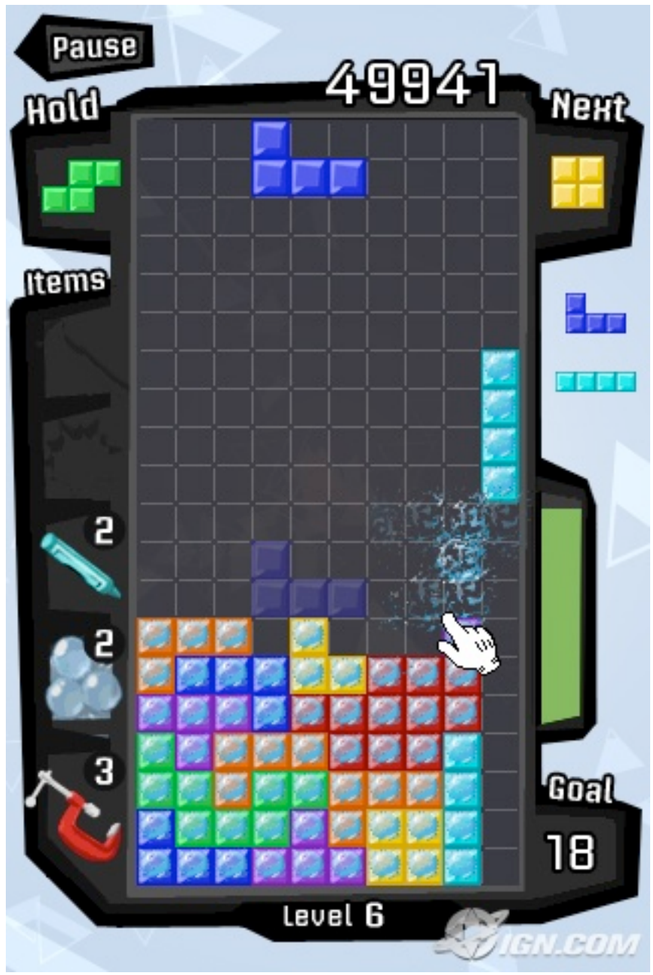
Tetris (2009)



Tuesday, March 6, 12

Lets start with what went well. Tapping on the left side or right side to rotate a piece works great. These are binary operations, essentially huge buttons for the player to tap on, confidence? 100%.

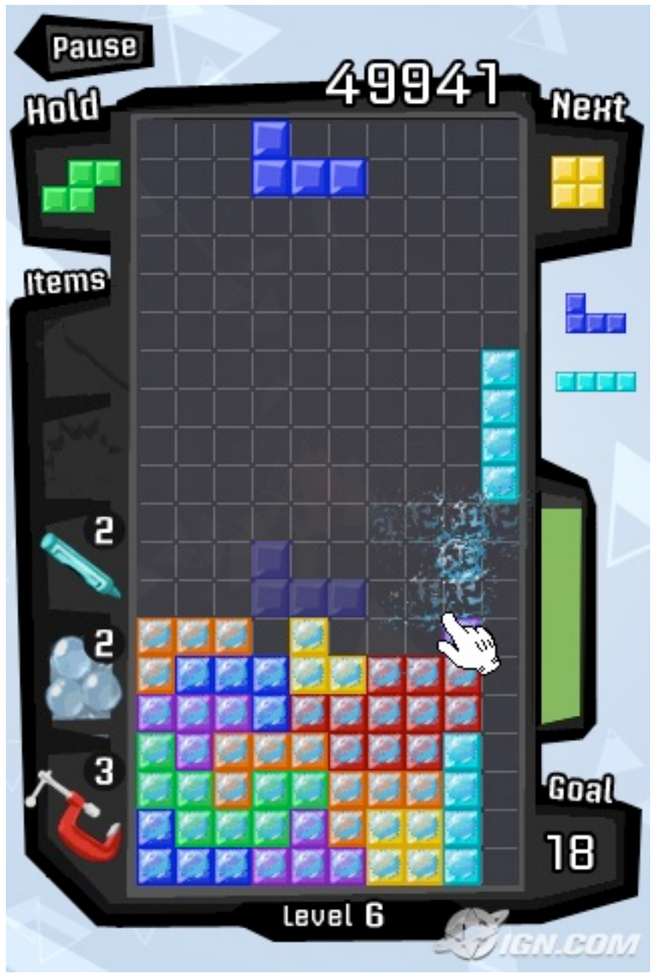
Tetris (2009)



Tuesday, March 6, 12

Unfortunately thats all they got right. Sliding the piece? You might think that this would work great but it turns out the disparity between how big players thumbs are vs. The column size (10p) is rather large. To compensate for this, they made it so players didn't have to move their thumbs very far to move a block. Unfortunately, moving your thumb so much less than 44 points is rather hard to judge from your proprioception, and the game, having been previously built for precise button control-based systems, didnt give any extra time for the player to use their secondary feedback loops. Worse than that, sliding right while holding the phone upright is tiring and uncomfortable. Try it, you can feel the resistance inside your hand.

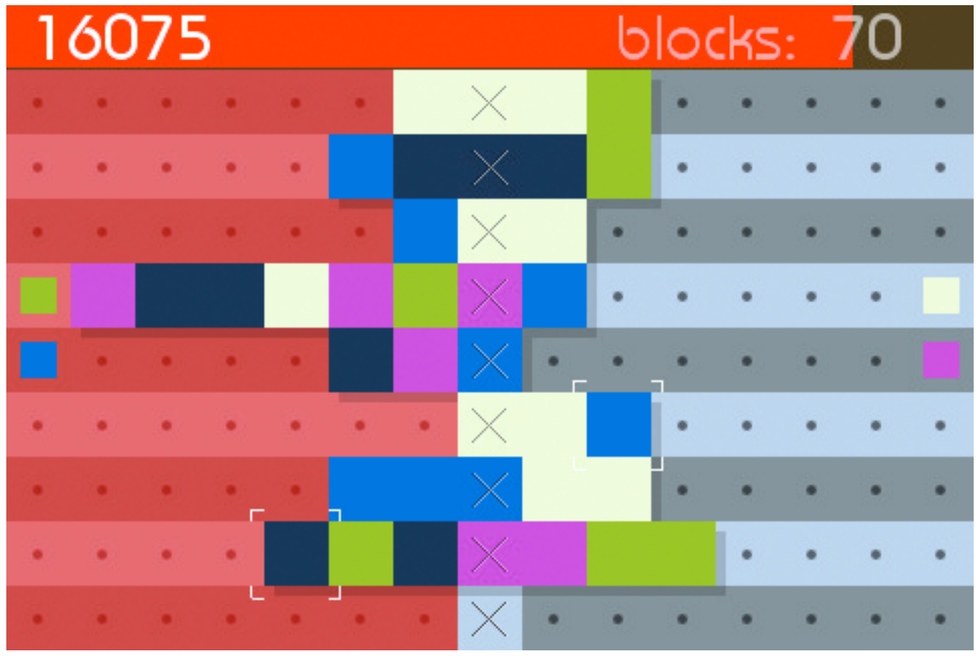
Tetris (2009)



Tuesday, March 6, 12

And by far the greatest mistake here is the swipe down to drop. If you look at Apple's UI elements you'll notice you almost never swipe down while using them (all scrollable menus start at the top, and you swipe up to move them). Theres a good reason for this, swiping down is uncomfortable, and swiping up is pleasurable. In Tetris you drop pieces a lot, and connecting such an important action to an uncomfortable gesture is just bad design.

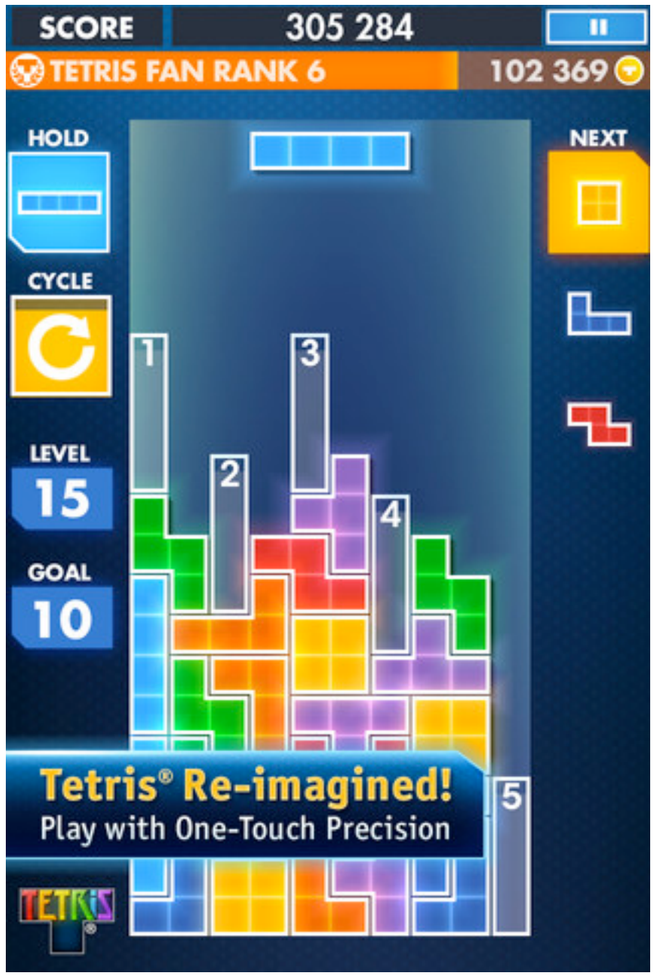
Unify (2009)



Tuesday, March 6, 12

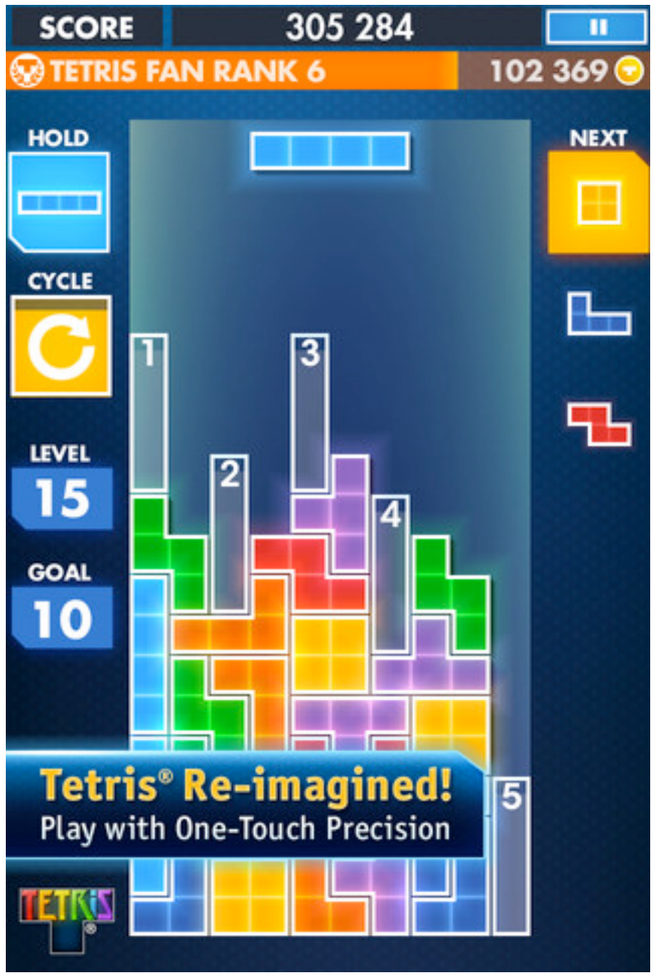
Unify, my take on the genre rotated the iPhone horizontally to allow for comfortable gestures and a higher degree of precision. In Unify players control two blocks at once, and the ‘bottom row’ is in the middle of the screen. Because I only have 8 rows as opposed to 10, rows can be 32p instead of 10p, substantially increasing drag precision for piece movement. Additionally, moving the thumb up and down while holding the phone horizontally feels good, this is just physically how we are built. Even better, pushing your thumbs towards each-other is is actually a pleasurable feeling (especially when this results in blocks exploding). Adding strongly physical re-enforcement sounds for piece movement rounded the game out. These are simple but significant changes in the controls, and they necessitated an entirely differently structured game. This is a common situation. Don’t be afraid to change your game to fit controls.

Tetris (2011)



Now lets look at how Tetris solved it's problem in 2011. I had the benefit of not needing to fit into Tetris branding with my re-imagining (actually, I was required not to :)), but EA had a different situation. They needed to make great controls while retaining the core gameplay of Tetris. And they succeeded brilliantly, by focusing on what was necessary and not being afraid to change the game up a bit.

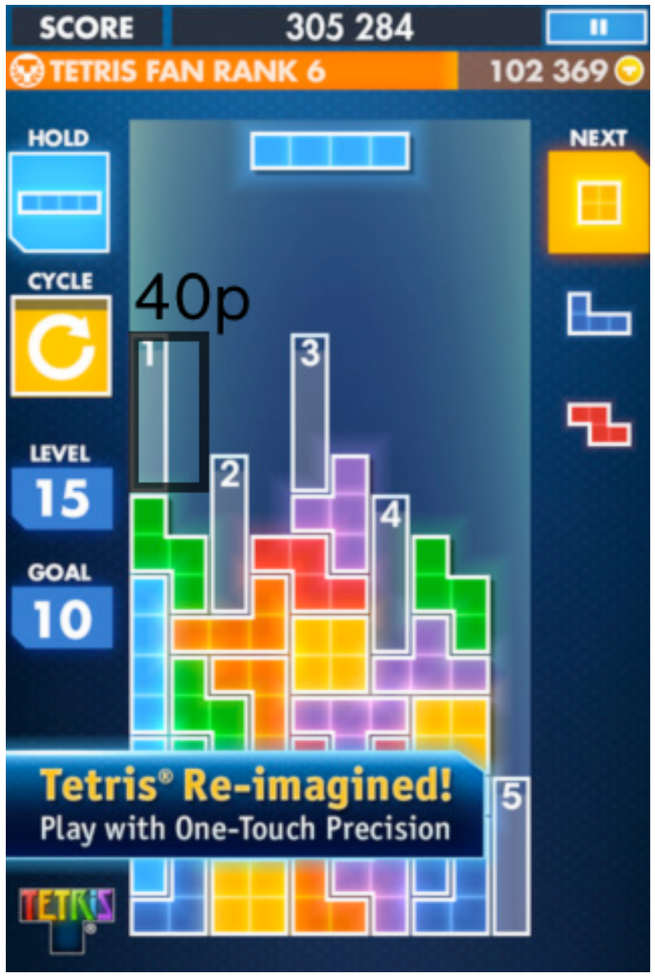
Tetris (2011)



Tuesday, March 6, 12

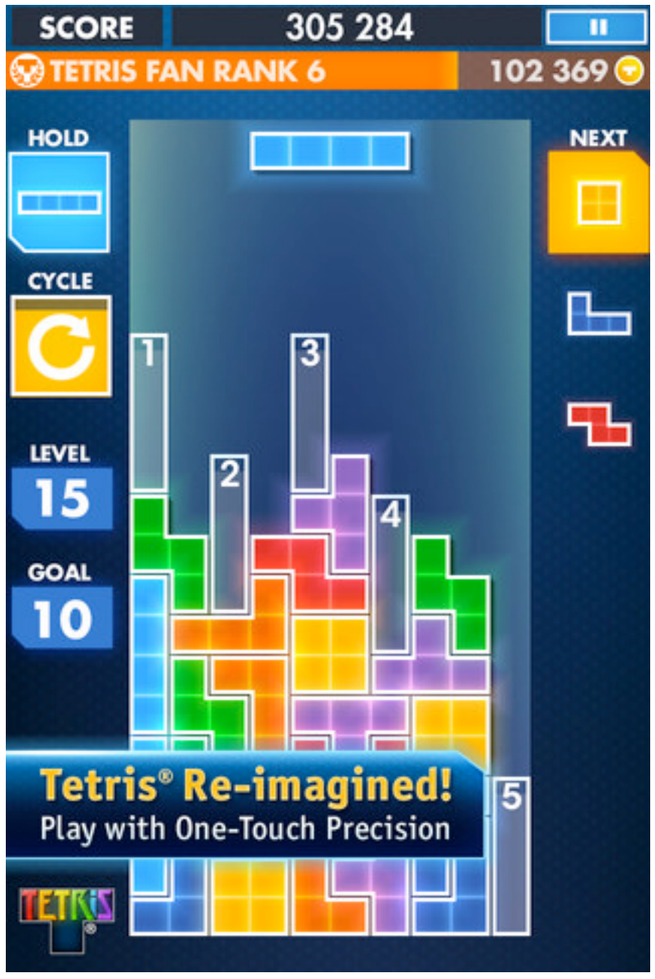
In the new version of Tetris, instead of uncomfortably dragging and dropping pieces, players simply tap on where they want to drop them. Of course, with 10p wide columns, precision would be a problem, so Tetris provides them with 4 outlines of possible drop locations and orientations for the piece. And guess how far away from each other those outlines are?

Tetris (2011)



40 points. A negligible 4 points away from Apple’s given size for 100% user confidence in hitting a button that they are looking at.

Tetris (2011)



Tuesday, March 6, 12

But what if the outline isn't in the right place? Tetris provides a button to tap to for new options. Each time players do, their remaining time to make a decision is reset. Tetris smartly moved the time pressure away from the interaction of the game (where users were having difficulty), and towards a new non-interactive thinking part. Theres a good lesson here. Sometimes your game mechanic will need to be adjusted to fit the constraints of the system. Even though Tetris is arguably a different game on iOS, this new version feels much closer to button-iterations of the game than it's disastrous 2010 version did.

Ok, Enough with the virtual buttons!

Of course this isn't only about virtual buttons. What about games that eschew buttons entirely?

Tuesday, March 6, 12

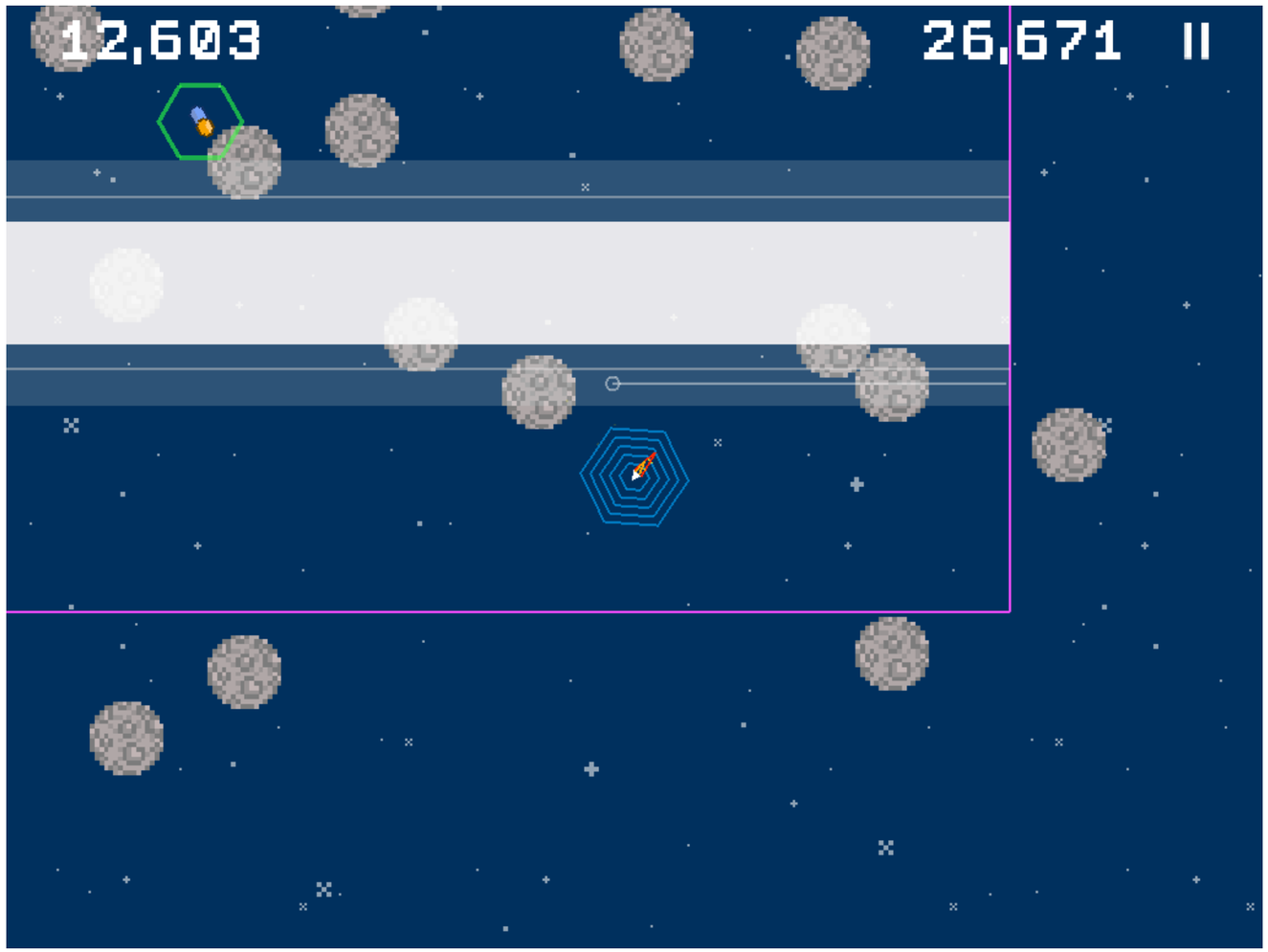
Flight Control was a very early game that did this very well. Players draw a line with their finger and a plane follows it. Extremely simple control. Feel great gestures, and 100% confidence in input. Players never miss tapping a plane, planes always go roughly where the players want. Of course, thats not where the design ends, these good controls needed to be integrated into the design of the game itself. Despite the ease of use, our lines are still limited by our thumb size and our skill in drawing them, and in flight control, how perfect your line is ‘is’ important. Flight Control’s pacing is structured to allow for secondary feedback in the form of checking on your line and re-drawing it if necessary. Importantly, this is a pacing decision to allow for secondary feedback is not one made because of an inadequate control scheme, its one that builds upon the strengths of a great control scheme, and integrates them comfortably into a mechanic. Be careful not to confuse the these rationales.

Fruit Ninja

Tuesday, March 6, 12

Fruit Ninja is a great example of a faster game that still uses analog swipe controls. Gestures feel great on the iPhone, and by making player swipe the entire input of the game they managed to create something that feels great. Paired with appropriately sized fruit (the smallest, that strawberry up top, is 40x40 points), the input system is easy for players to be confident with. Instead of blaming poor abstract controls when they lose, they blame their own reflexes.

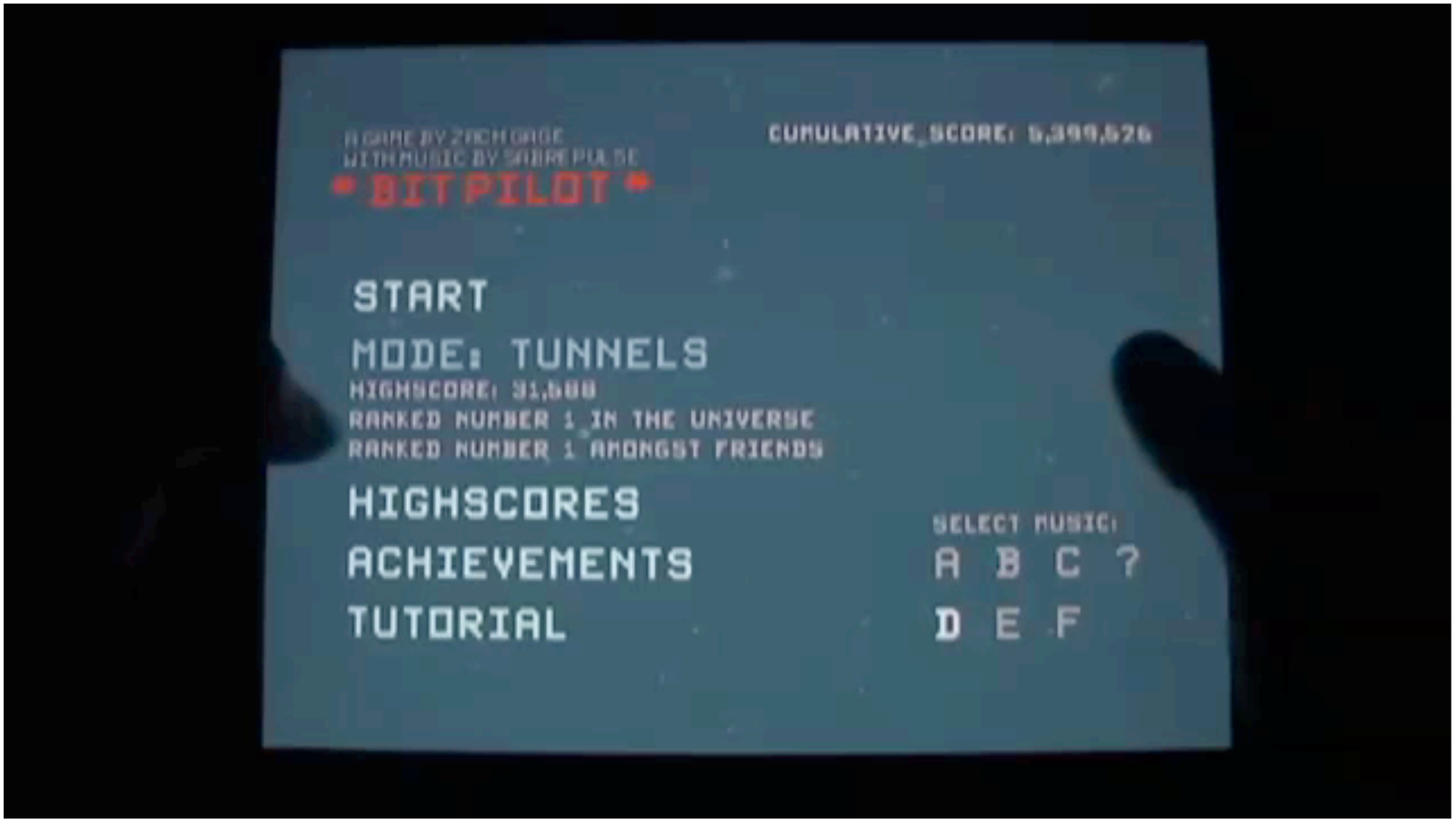
Bit Pilot



Tuesday, March 6, 12

One final example here of a fast game with analog input is another one of my own games, Bit Pilot. Bit Pilot was born out of the desire to make an intense action game on the iPhone. At first glance it might seem like a twin stick asteroid dodging game, but it's actually quite different in implementation.

Bit Pilot



Tuesday, March 6, 12

When players move their thumb, a fraction the force at which they move it is directly applied to their ship. This isn't to say the ship moves one to one with their thumb though. If they swipe right and then left again, the ship moves right slightly and then stops, because they're dealing with forces here, not exact movement. Theres a reason for using forces instead of mapping though, and its that people don't generally think of gestures as exact. When we make a gesture we aren't telling you how we want something, we're roughing it out. This is how gestures work in Bit Pilot. They're roughly what you want to do, and players have to do a few of them to really force their ship to do anything. This is where the trick in bit pilot comes in. In bit pilot players don't control their movement with one joystick, they control it with two. So if they absolutely want their ship to fly right really fast, they swipe right with both of their thumbs at the same time. Both forces together equates to more than enough force to shove their ship to the right. They can even move quickly with one thumb to the right, and adjust with the second thumb if they feel their first gesture was a bit off. Because each of these moves is an individualized gesture, they're easy for the brain to parse and understand, and because player have to make a lot of moves, they get to make a lot of decisions per second, without being stressed out about what their decision will do, or waiting for secondary visual confirmation.

It takes some work to become extremely proficient with this method, but because it allows for such specific accuracy in movement when players get good, they can pull off amazing escapes at breakneck speed while still having the visual processing part of their brains free enough to track fifty to sixty asteroids. This is a control scheme that celebrates touch, and in the end, the control players can have with it is far more accurate then anything they could likely do with a real joystick. And it wasn't any stroke of genius that came up with this, it was just starting from the bottom up. Knowing what kind of game I wanted to make, thinking about what kind of controls are successful on the phone, and then some trial and error, and flexibility.

Lets sum this up.

Okay, I just covered a lot of different things, and likely you wont remember everything exactly, so to sum this up, when you’re designing a game for an unfamiliar interface, remember these three points:

Lets sum this up.

1. Be Flexible

1. Be flexible with your game design. Game genres are as much a product of our interaction possibilities as they are a product of our creativity, its okay, and in fact its normal to let controls influence the direction your game. A genre that works on a controller might not be viable on a touch screen, or any other non-controller interface.

Lets sum this up.

1. Be Flexible
2. Be Serious

2. Spend serious time thinking about how to design simple to intuit controls. And understand what that entails: A skilled player must be able to know what her input on the game environment will do before she can see what it did, every time.

Lets sum this up.

1. Be Flexible
2. Be Serious
3. Integrate Well

3. Once you have good feeling controls, make sure your game design supports their strengths and not their weaknesses.

As I said, this stuff isn't rocket science, all it requires requires is a little bit of time, experimentation, and care. And when you get it right, theres nothing better. Your controls become invisible, freeing players up to concentrate on something far more important, playing the hell out of your game.

and unrelated, these are also pretty good tips for being a nice person.

Thanks

Zach Gage - @helvetica - STFJ.net

tha
nks