



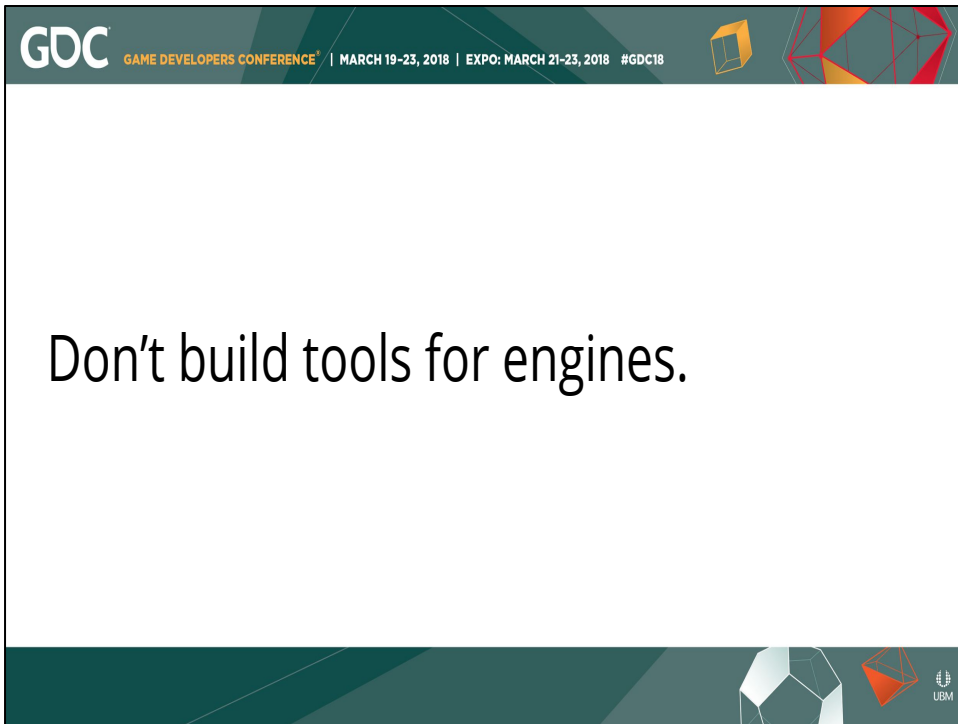
## Take selfie!

Let's get started!

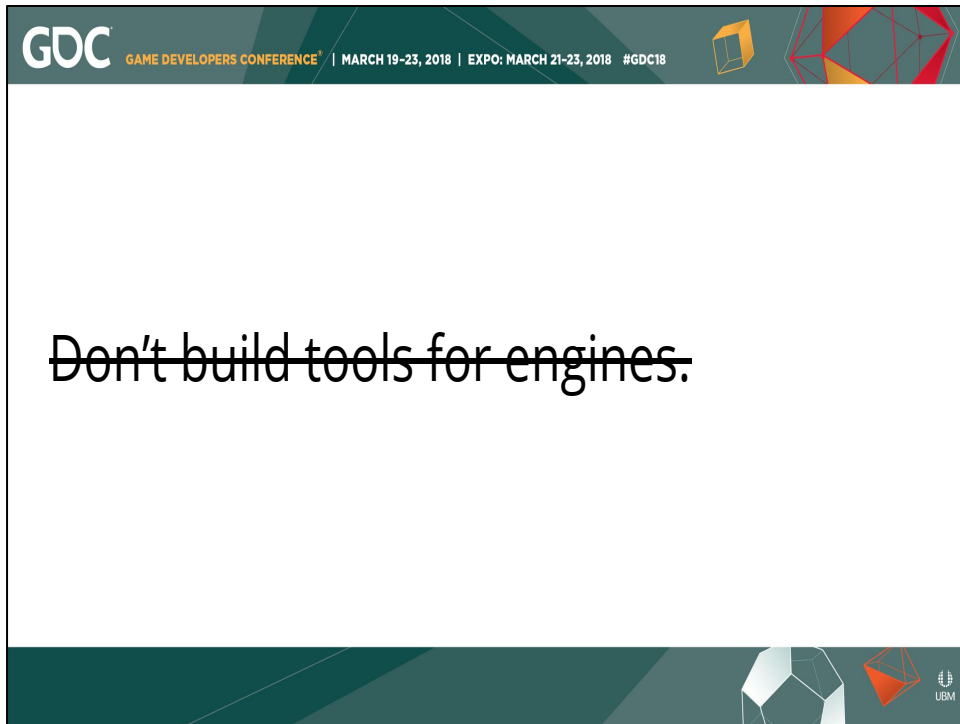
Suppose I want more engineers at my studio to be able to build tools. Maybe these are engineers--even veteran engineers--who haven't done a lot of tool development work in the past. Let's say I want them to be empowered to bring new tools to whatever team they're working on. And I want them to create tools that "fit" into our existing ecosystem.

What advice would I give such an engineer who's interested in starting tool development? What do I wish someone had told me?

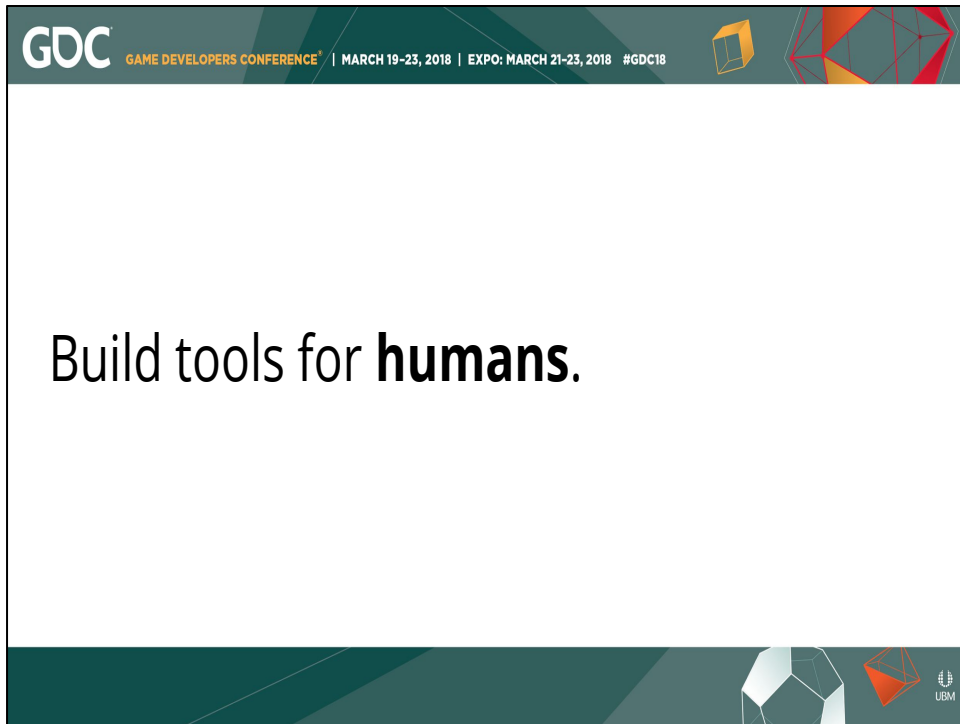
What do I, as a veteran tools engineer, want to impart to them? I might think back over all my experiences and frustrations and trials-and-errors and I might say...



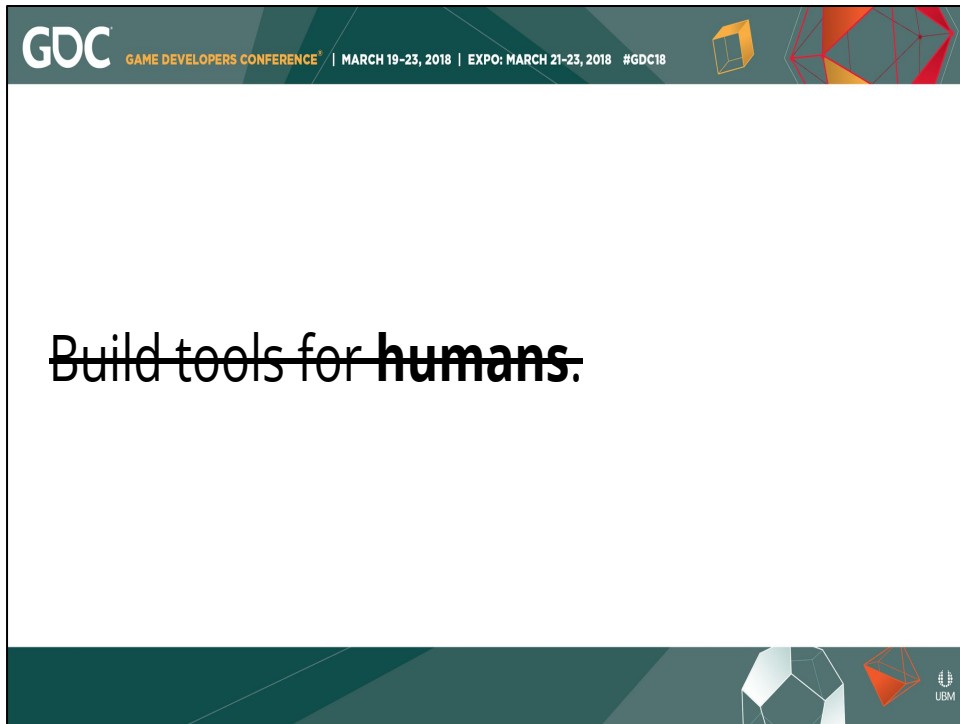
“Don’t build tools for your engine.” Too many of us build tools that have the *engine* as its primary customer. But our engine isn’t our customer. It doesn’t suffer. It doesn’t call for help. It doesn’t have great ideas about how to make tools better.



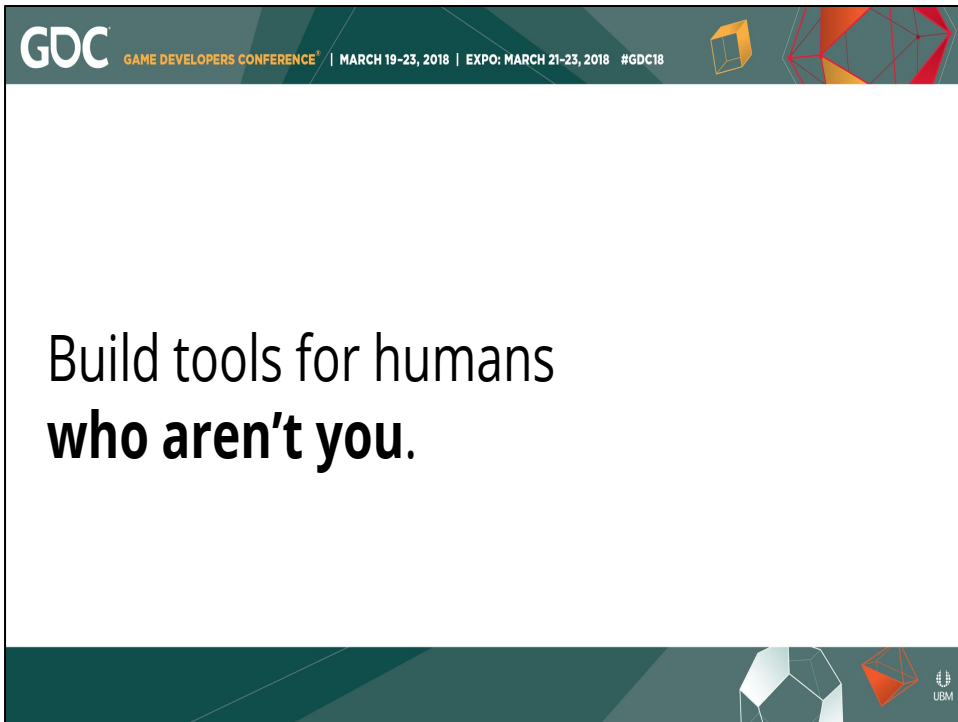
That's nice and pithy, but I think we can do better. It's easy to say "don't do" this or that, but what do you do instead?



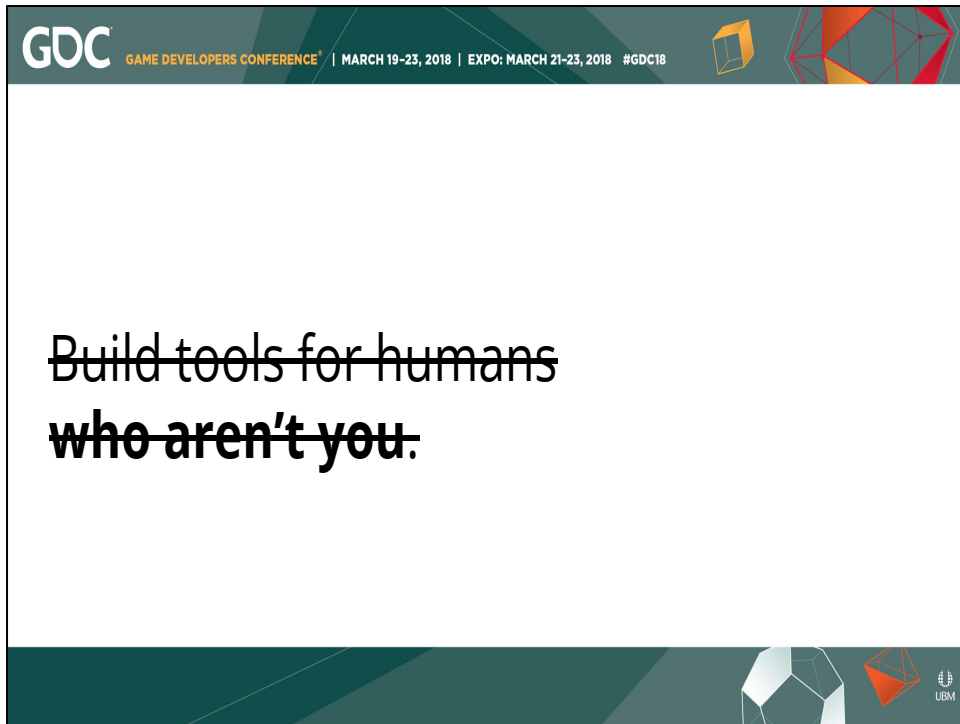
“Build tools for humans.” Yes! Our customers are humans. And humans think about things differently than engines do. Humans also make mistakes.



No, that's not good enough. An engineer might just build tools *for themselves*, something that reflects how *they* think about game content.

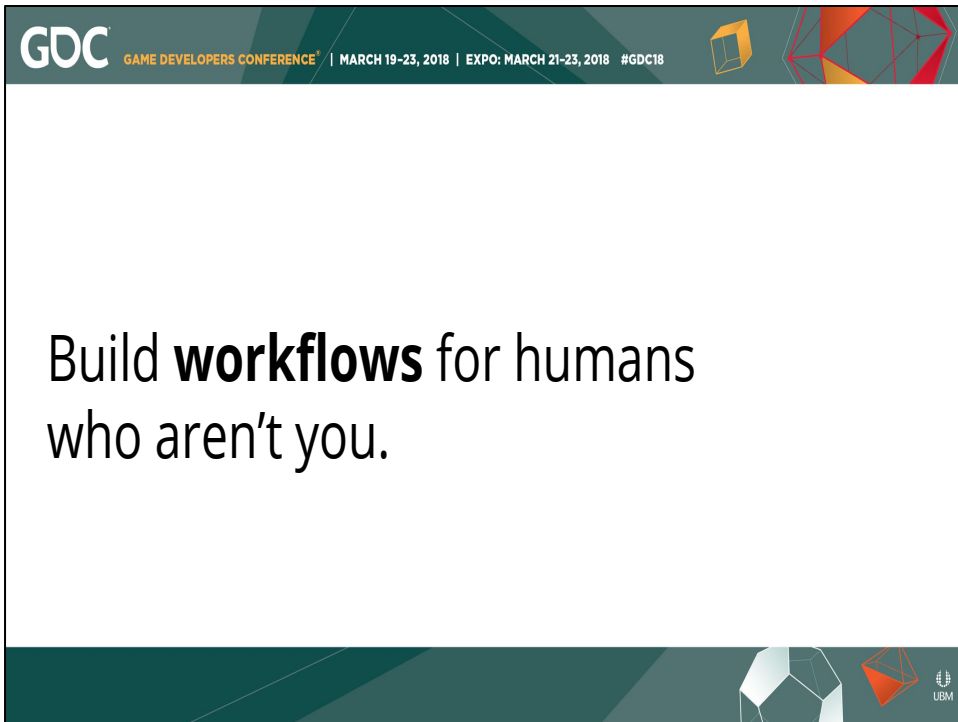


“Build tools for humans... who aren't you.” This is good. Because while “You” are probably a brilliant technical mind, you could be building tools for brilliant *non-technical* people. I like this, though: this is starting to sound like what all the best UX books tell me to do: design with your users in mind!



OK, getting better, but I'm not satisfied with this... Game development has this weird problem: we have a lot of engineers, and they *could* turn out a lot of human-focused tools, true. *But then you just have a pile of tools.* (Anyone here work with a pile of tools? Is it fun?)

So are we building the right *thing*?



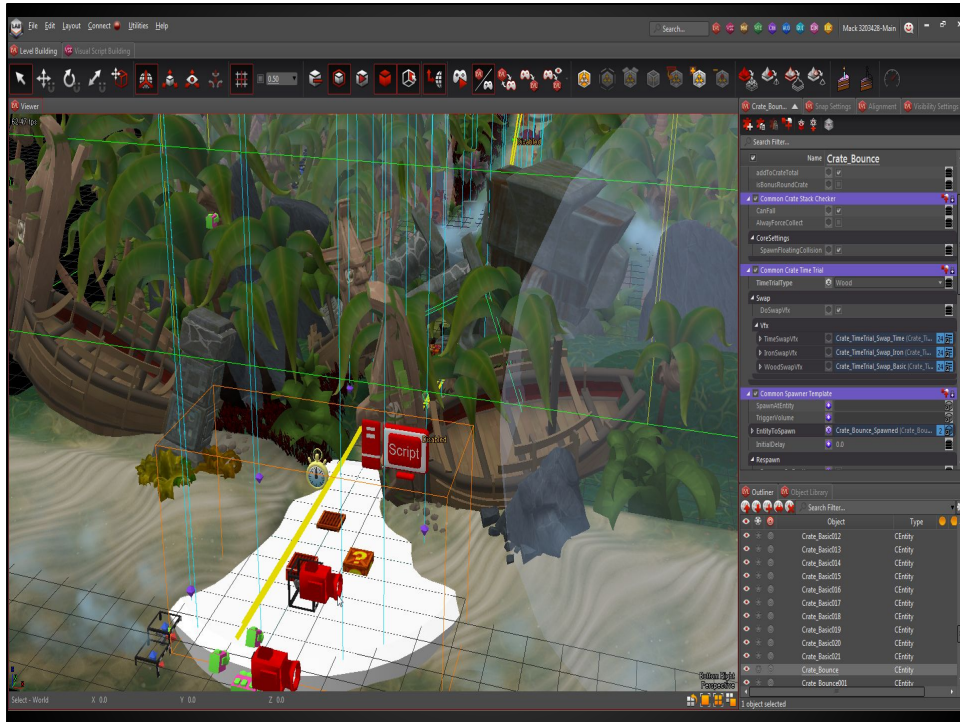
I don't want you to build tools. I want you to build *workflows*, things that comprise all the “steps” content creators take, and all the tools they involve, and how users “flow” through their work. I want users to have a cohesive, coherent experience.

...

This idea--that you ought to build workflows for humans who aren't you--is something I think most engineers would agree with *philosophically*. But even this is over simplistic. But making it practical is *hard*. And making it practical is exactly what we wanted to do, because again, we're interested in enabling other engineers in other disciplines to build great workflows too, and we want them to be successful quickly.

What I'm sharing today grew out of this line of thinking. We wanted to distill learnings from past successes into knowledge that could guide future engineers, even non-tools engineers.



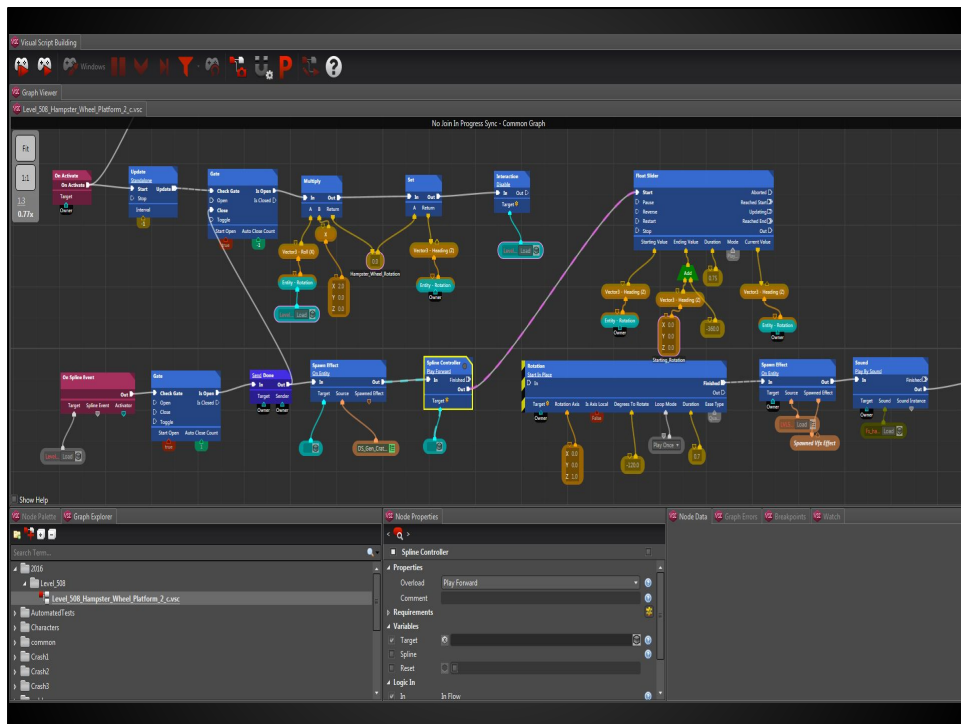


Because we *did* have past successes.

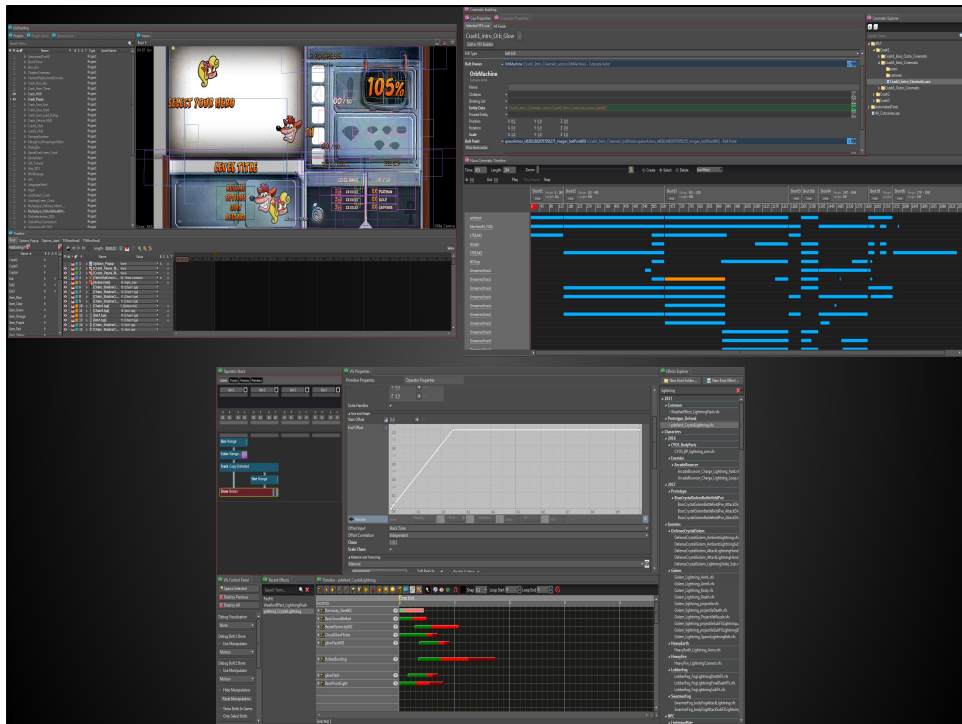
Shortly after I started at VV I worked with other tools engineers to create *Alchemy Laboratory*. Alchemy is VV's proprietary engine tech and Laboratory is its toolset.

All its tools are oriented around content creation workflows, which tend to map to disciplines.

This is World Builder, where we built levels for Skylanders and Crash...

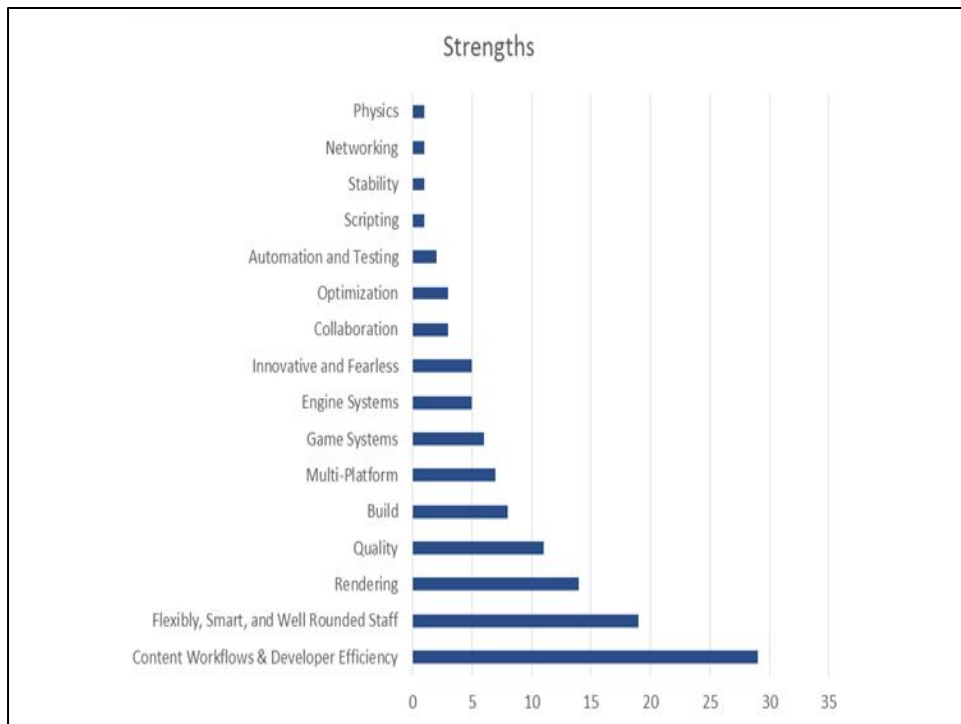


This is Visual Script Builder, where we built visual scripts...



These are some other workflows in the toolset: UI, VFX, and cinematics.

Laboratory was a big investment in workflow development. We came out of it with really good results. I got to work with some crazy smart engineers who had a passion for working with users to produce great results.



The year we brought Lab online, workflows & efficiency were seen as our biggest strengths.



And our users rated our workflows highly compared to others they'd used in the industry.

(We do a technical survey each year to gauge the studio's feelings about our tools and tech. This is from 2015.)

We don't attribute our success to any one guiding principle. Laboratory was the culmination of years of tools engineering expertise. A lot of it came from trial and error and investment in trying new things, and some of it was learned from painful disruptions, false starts, and inefficient deliveries.



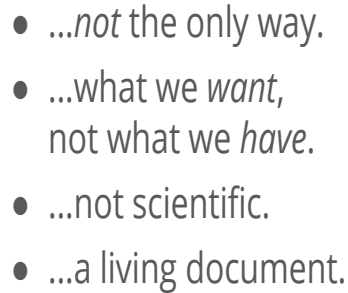
So we wanted to codify some of what we felt had made VV successful at workflow development over the years. We also wanted to provide direction for *future* efforts: declare where we wanted our investments to go. We wanted to make something we could share: with new engineers, veteran engineers from other disciplines, and now good people like you. :)

We created a set of wiki pages and called them our *Workflow Development Guidelines*. The original target audience was engineers who wanted to participate in workflow development, but I think the advice is consumable by anyone working in this space.

The guidelines are technology and project agnostic, and are organized into three categories:

- UI & UX Design
- Engineering / Implementation
- Creative Mindsets

We want engineers to look at these and think, "OK, this is something I can actually do. This is a tradeoff I can make."



- (Some changes as recently as last night!)
- Expect guidelines to change shape as we learn more.



## Workflow Development Guidelines

### UI & UX Design

- Lots of research outside industry.
- This is a “highlight reel”.
- Emphasizes problems we’ve actually encountered.



#### UI & UX Design

- Start with User Interface & User eXperience Design
- **(CLICK)** Software usability is big outside games.
  - Upward trend in UI + UX jobs at organizations
  - Literature is still catching up? Lots of it is geared toward helping people buy stuff online!
  - Wanted something more approachable.
- **(CLICK)** Guidelines here are a “highlight reel”, not the last word.
- **(CLICK)** Wanted to call attention to areas that literature might not emphasize.





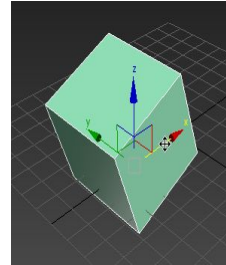
## **NEW SECTION: BOOST WORKFLOWS WITH KEYBOARD SUPPORT**

- Skilled users have high expectations for using keyboards to make their work faster.
- Here're things you can do...



# Boost Flow With Keyboard Support

- Support Precise Input



X: -25.698 Y: -0.638 Z: 0.0



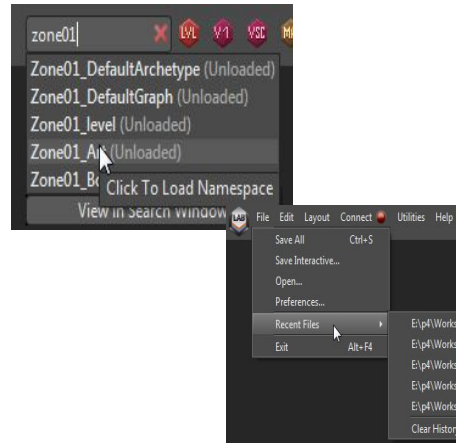
## YOU CAN: SUPPORT PRECISE INPUT

- Designers may know “exacts”: missile count, cinematic length
- Don't force users to figure out how to express it, e.g. with sliders / snapping. Instead, type in! Continue work!
- *Good* to blend precision w/ gross/fine adjustment, e.g. DragSlider.



# Boost Flow With Keyboard Support

- Support Precise Input
- **Accelerate Data Entry by Anticipating Input**



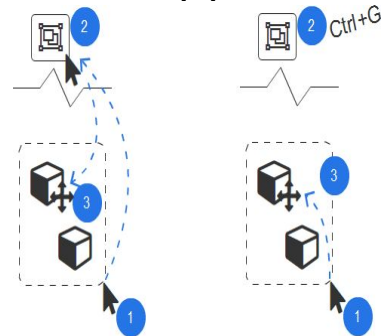
## YOU CAN: ACCELERATE DATA ENTRY BY ANTICIPATING INPUT

- Users should avoid typing what tool knows or can infer.
- Think: IntelliSense.
- Users may know exact names/ids, or just fragments. Support both with *predictive inputs*.
- Or “anticipate” a micro-workflow like File->Open: offer an MRU to accelerate it.



# Boost Flow With Keyboard Support

- Support Precise Input
- Accelerate Data Entry by Anticipating Input
- **Accelerate Tasks with Shortcut Keys**



Shortcut keys can reduce mouse travel for common sequences, like select-group-move.

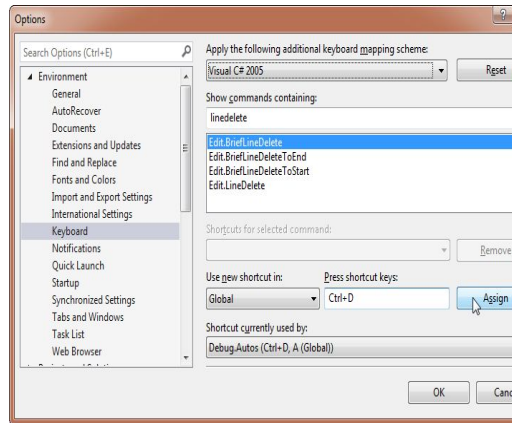
## YOU CAN: ACCELERATE TASKS WITH SHORTCUT KEYS

- Shortcut keys can save users time by reducing mouse movement or streamlining the flow from one activity to the next.
- For example, if a user intends to select two objects, group them, then move the group, then a shortcut key can reduce the amount of mouse movement they need to perform.
- Favor assigning shortcut keys to actions that would otherwise require the user to shift focus away from the next step in their task.



# Boost Flow With Keyboard Support

- Support Precise Input
- Accelerate Data Entry by Anticipating Input
- Accelerate Tasks with Shortcut Keys
- **Beware of Shortcut Key Pitfalls**



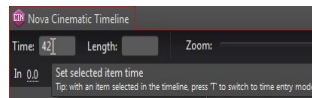
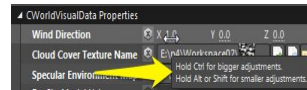
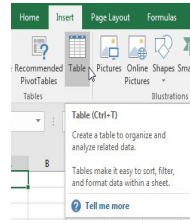
## YOU SHOULD: BEWARE SHORTCUT KEY PITFALLS

- Users prone to typos.
- Be careful of proximity.
- Shortcut keys are scarce.
- Shortcut keys are forever (unless you support custom binding)
- Be judicious about reserving new ones.
- Users internalize and habituate, so disrupt at your peril.



# Boost Flow With Keyboard Support

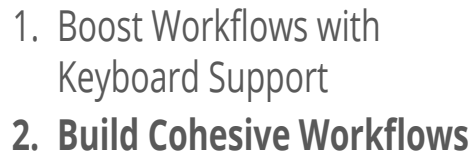
- Support Precise Input
- Accelerate Data Entry by Anticipating Input
- Accelerate Tasks with Shortcut Keys
- Beware of Shortcut Key Pitfalls
- **Make Keyboard Support Discoverable**



## YOU CAN: MAKE KEYBOARD SUPPORT DISCOVERABLE

- Tooltips are great! Consider rich tooltips / infotips like these that *show the shortcut*
- Promote discoverability by deferring to idiomatic shortcuts (Ctrl+S to Save, Ctrl+Shift+S to Save All)

## END OF SECTION

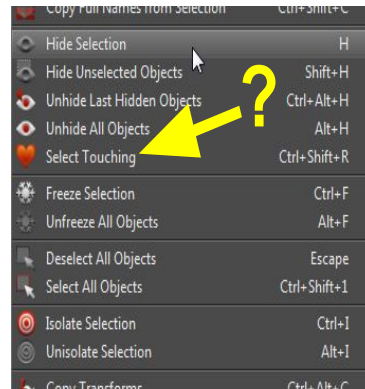


- or, “Features that play together stay together.”
- *Cohesive* feels organized, task-oriented, and integrated with the user's other tasks...
- ...while *incohesive* feels disjointed, scattered, or “incomplete”.
- Here's what you can do...



# Build Cohesive Workflows


- **Promote Discoverability and Findability**



## YOU CAN: PROMOTE DISCOVERABILITY AND FINDABILITY

- **Discoverability** => users can learn what's available.
- **Findability** => users can find what they expect.
- To promote F & D, keep related features close together, "clustered" by task
  - (Here's a Laboratory context menu in our Level Builder workflow. One of these things is not like the other...)
- Keeping things "close together" can mean minimizing,
  - Mouse movement
  - Eye movement
  - Number of clicks
  - Levels of menu depth
- Over time users will build a strong mental map:
  - Will learn where to find what they expect.
  - Will remember what they haven't tried yet.



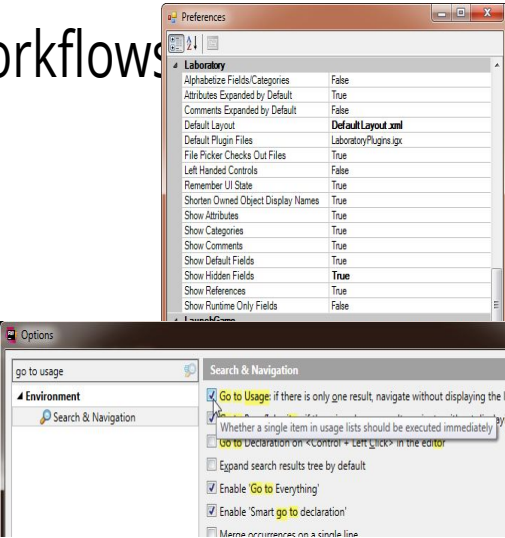


GAME DEVELOPERS CONFERENCE

MARCH 19-23, 2018 | EXPO: MARCH 21-23, 2018 #GDC18

# Build Cohesive Workflows

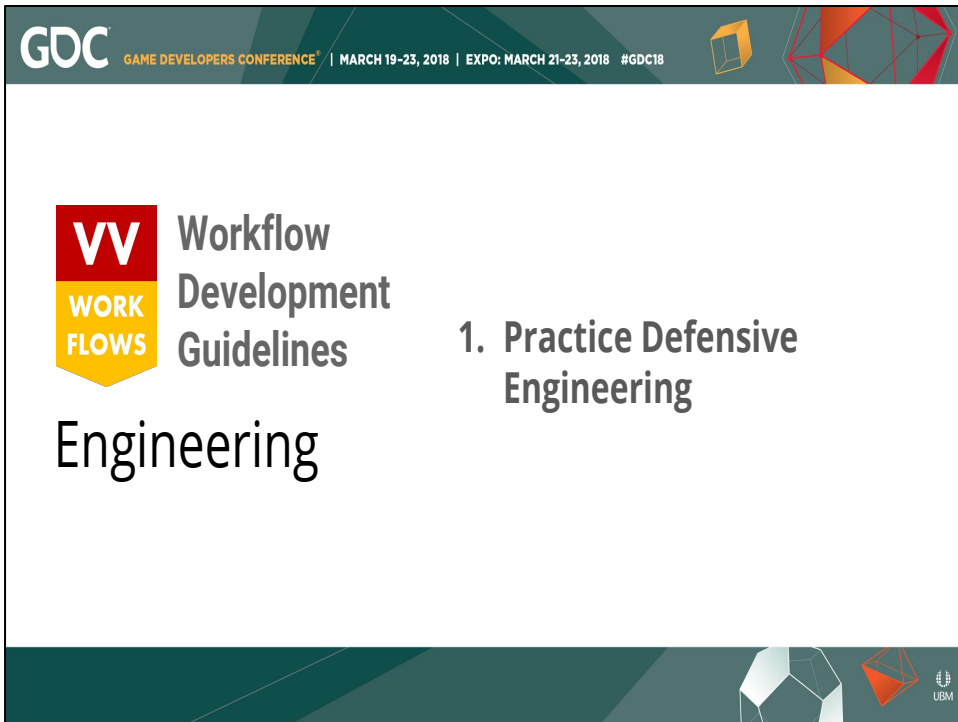
- Promote Discoverability and Findability
- Avoid “Dumping Grounds”



## YOU CAN: AVOID DUMPING GROUNDS

- You see this most with Settings Dialogs, and in rushed UI designs where widgets are thrown into a pile because it wasn't clear where to put them.
- Dumping grounds indicate poor cohesion; maybe you're not clustering things appropriately?
- You're at risk of users doing lots of parsing & scanning.
- The bigger they get, the harder they are to internalize.
- Pay attention to grouping, searchability when exposing knobs and dials.
  - Good trend: “searchable” settings, e.g. ReSharper, iOS





## **NEW CATEGORY: ENGINEERING**

- General guidance about implementation.
- First, "PRACTICE DEFENSIVE ENGINEERING"
- Here're some ways...



# Practice Defensive Engineering

- Consider Potential for Abuse

**Easy > "Correct"**  
**Easy > "Supported"**



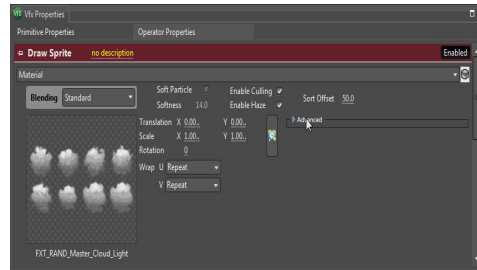
## YOU CAN: CONSIDER POTENTIAL FOR ABUSE

- Users are always looking for paths of least resistance. Even if it's tedious.
  - "It's amazing what content creators will put up with."
  - Scott Meyer's #1 advice: "Make Interfaces Easy to Use Correctly and **Hard to Use Incorrectly**"
- If it's easier to do something the "wrong way" and still get the same result, you should expect that users will converge on that way.
- Users may find "shortcuts" that circumvent processes designed to validate and sanitize data.
  - a. Users would modify IGX data directly, maybe because of a merge. Lab wasn't always great at catching it..
  - b. If flow is bad, they desire for shortcuts goes up!



# Practice Defensive Engineering

- Consider Potential for Abuse
- **Don't "Expose Everything"**



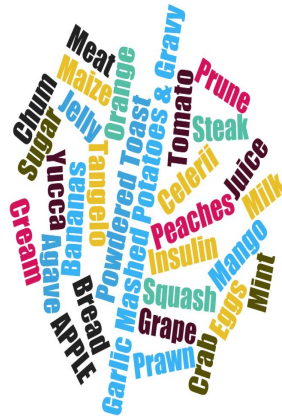
## YOU CAN AVOID "EXPOSING EVERYTHING"

- Similar to "avoid dumping grounds", but this is about focusing and reducing demands on users' attention.
- There are more knobs and dials in an engine than users know what to do with.
- If confused, they might "probe" the UI into a confusing state and need help getting out.
- Do your best to expose only the subset a user needs to do their job.
- Unusual use cases can be addressed with "Advanced" categories.



# Practice Defensive Engineering

- Consider Potential for Abuse
- Don't "Expose Everything"
- **Don't Rely on Tribal Knowledge**



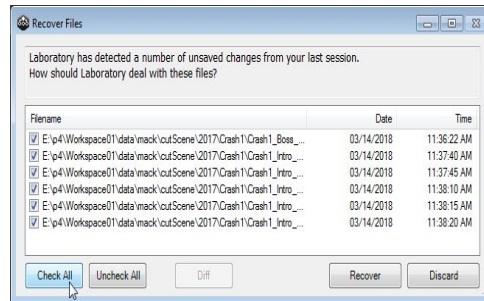
## YOU CAN: AVOID RELYING ON TRIBAL KNOWLEDGE

- Fallacy: "Most users will know the 'right' way to do something."
  - Hurts confidence.
  - "What do I do next?"
  - "Is this going to work?"
  - "How do I make what I want?"
- Especially for programmers, tribal knowledge sometimes includes what you *shouldn't* do. This can be especially dangerous, since new efforts might be built on sinkholes.
- Was hard to find a visual aid for this slide, but I thought this was fun. This word cloud is all the names we ever gave to tools in the Alchemy ecosystem before we moved to Laboratory.
  - (Can anyone tell me which was the VFX workflow?)



# Practice Defensive Engineering

- Consider Potential for Abuse
- Don't "Expose Everything"
- Don't Rely on Tribal Knowledge
- **Tolerate Surprises**



## YOU CAN: TOLERATE SURPRISES

- Do what you can to tolerate the kinds of surprises that can cause people to lose work.
- Laboratory has fault tolerance:
  - Autosave & recovery (power outages)
  - Process isolation



- Next up: Consider Content Models
- This is “Put yourself in the user’s shoes” guidance:
  - Understand what content concepts they expect to work with





*The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.*

10 Usability Heuristics for User Interface Design

<https://www.nngroup.com/articles/ten-usability-heuristics/>

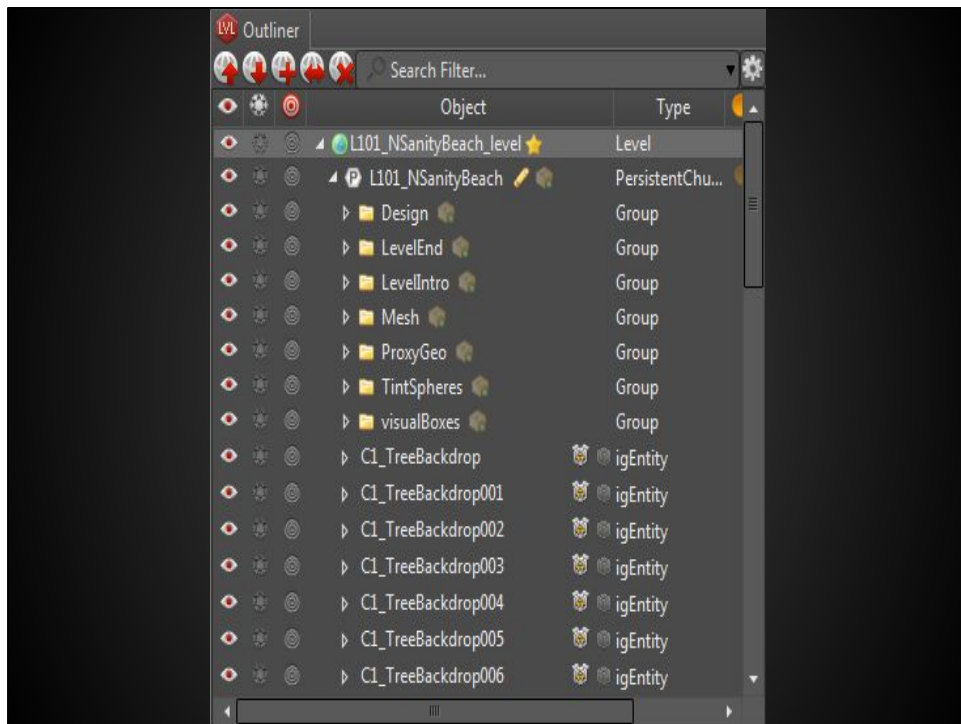
## **“Content Model”**

the concepts and relationships with which users express ideas and build content in their workflow.



### **“CONTENT MODEL” DEFINITION**

- Jakob Nielsen is a researcher with NNG, and for years has been doing usability work concentrated on making the Internet easier to use.
- (READ QUOTE)
- I think this applies to our workflows, which should speak the language used by a content creator.
- But too often, I think, workflows speak the language of the system that will consume the data, e.g. the game engine.
- **(CLICK)** So let's let CONTENT MODEL mean...
- Basically, it's the metaphor in which content creators work.



### EXAMPLE: LVL BUILDER CONTENT MODEL

For example, Laboratory's Level Builder workflow exhibits a Level/Layer/Group/Object content model for organizing entities in a map.

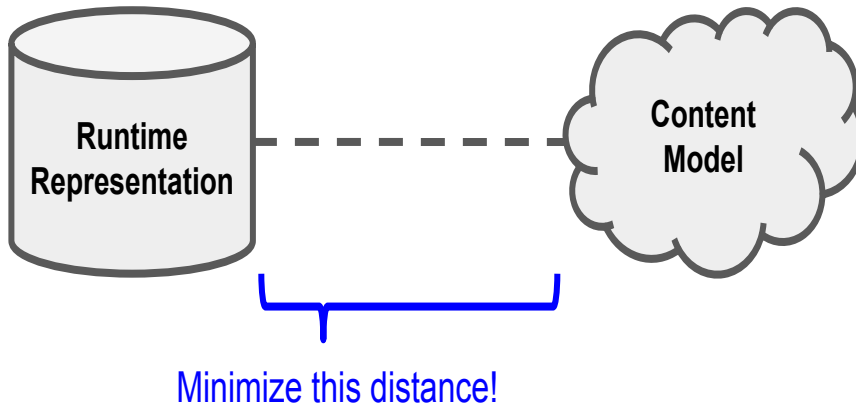


## EXAMPLE: CIN BUILDER

- Good tools and workflows will abstract away engineering complexity and fill data gaps so users have less mental mapping to do..
- For example, Cinematic Builder UI expresses VFX cue as...
- Underneath, a VFX cue is modeled as an array of “spawn” and “kill” keyframes that aren’t inherently related and don’t map to this metaphor very well.
- Predecessor workflow modeled this more directly, and was confusing to users.
- This workflow shows the cue as a contiguous event in time.




# Consider Content Models



## BEST PRACTICES FOR CONTENT MODELS

- Good workflows that keep users close to their content will allow users to concentrate less on our runtime's demands and more on the content they're building.
- There are (at least) two important representations in a workflow: the runtime representation and the user's content model. If you, as an engineer, can minimize the distance between those representations, you will reduce complexity.

**GDC** GAME DEVELOPERS CONFERENCE<sup>®</sup> | MARCH 19-23, 2018 | EXPO: MARCH 21-23, 2018 #GDC18



**Workflow  
Development  
Guidelines**

- “Touchy feely” ROI
- Reinforce easy-to-forget wins
- Be *aspirational!*

**Creative  
Mindsets**



## NEW CATEGORY: CREATIVE MINDSETS

- I call this category, “**Creative Mindsets.**”
- My goal with this category:
  - **(CLICK)** Capture benefits that are hard to quantify but have observable impact on user satisfaction. Call it “Touchy Feely” ROI.
  - **(CLICK)** Reinforce “common sense” that’s easily forgotten or overlooked while in-the-trenches.
  - **(CLICK)** Inspire! Invite engineers to apply their unique expertise to the creative process.



## NEW SECTION: PROMOTE USER CONFIDENCE

- We'll start with **Promoting User Confidence**.
- When a user is **confident**, they will try cool things without fear.
- When they're **not confident**, they may *avoid* workflows altogether!
- *more...*



# Users are confident when...

- ...tools do what they ask.
- ...they can change their minds quickly.
- ...they aren't afraid of breaking other people.

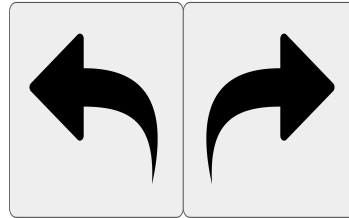


- We've observed (and have been told) that users work confidently when:
  - a. tools are doing what they ask,
  - b. they can change their minds quickly,
  - c. they aren't afraid of breaking other people.
- Here's what you can do...



# Promote User Confidence

- Undo/Redo Must Work



## YOU CAN: MAKE SURE UNDO/REDO WORK

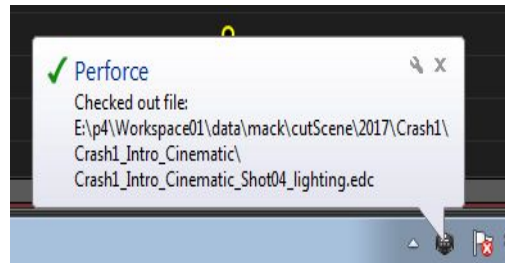
- Critical for **confidence to try new things**
- “What-if thinking” IS WHAT WE WANT!
- Also improves iteration time by reducing rework.
- Reduces the cost of making mistakes.
- Importance impacted architecture: Lab offered undo/redo as a service to all plugins thanks to a shared data model.





# Promote User Confidence

- Undo/Redo Must Work
- **Support Source Control**



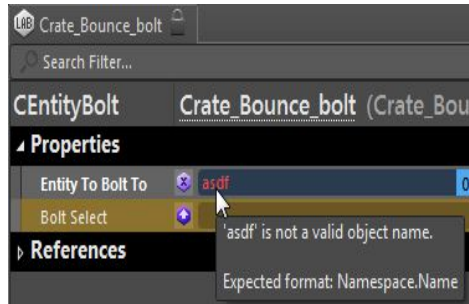
## YOU CAN: SUPPORT SOURCE CONTROL

- Save users a lot of confusion mucking around in Perforce.
- State of your data + state of depot = *large mental load*. Hard to do manually, esp. with complex dependencies.
- This introduces doubt, which erodes confidence.
- Another architectural impact: like Undo, Lab offered Perforce integration as a service.



# Promote User Confidence

- Undo/Redo Must Work
- Support Source Control
- **Do Rigorous Data Validation**



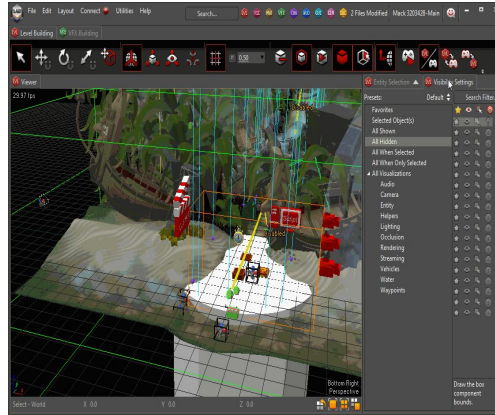
## YOU CAN: DO RIGOROUS DATA VALIDATION

- Don't allow users to introduce bad data downstream.
  - e.g. exporting *expensive assets*, adding a *bad reference*
- Users might not know what "correct/safe" is. So expect odd input.
- If something's wrong, tell what's wrong,
- *and how to fix it.*



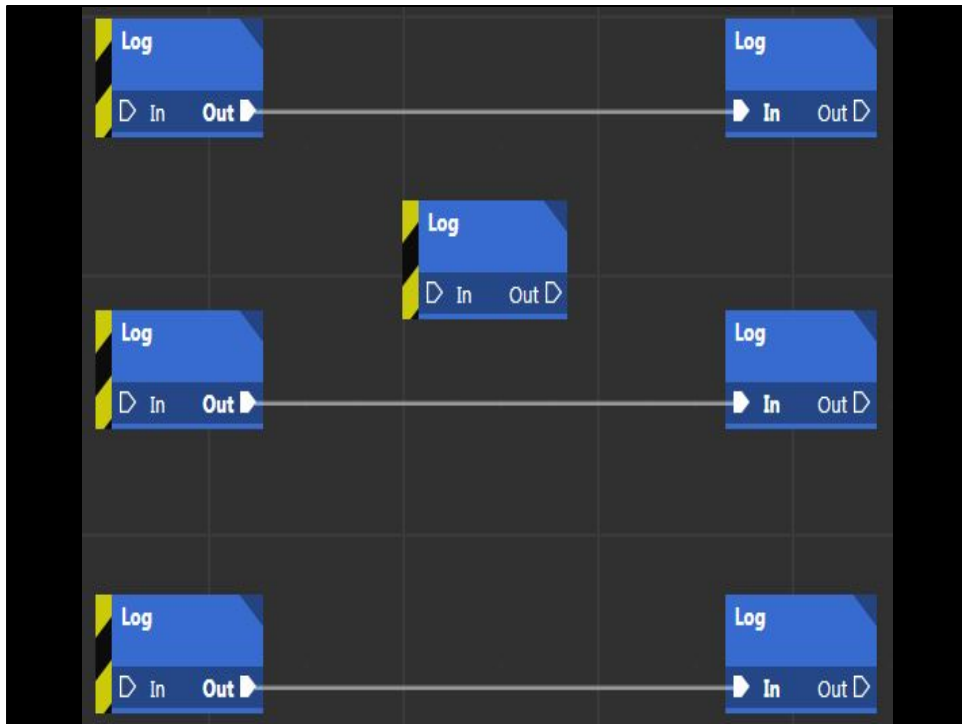
# Promote User Confidence

- Undo/Redo Must Work
- Support Source Control
- Do Rigorous Data Validation
- **Consider Previewing Changes**

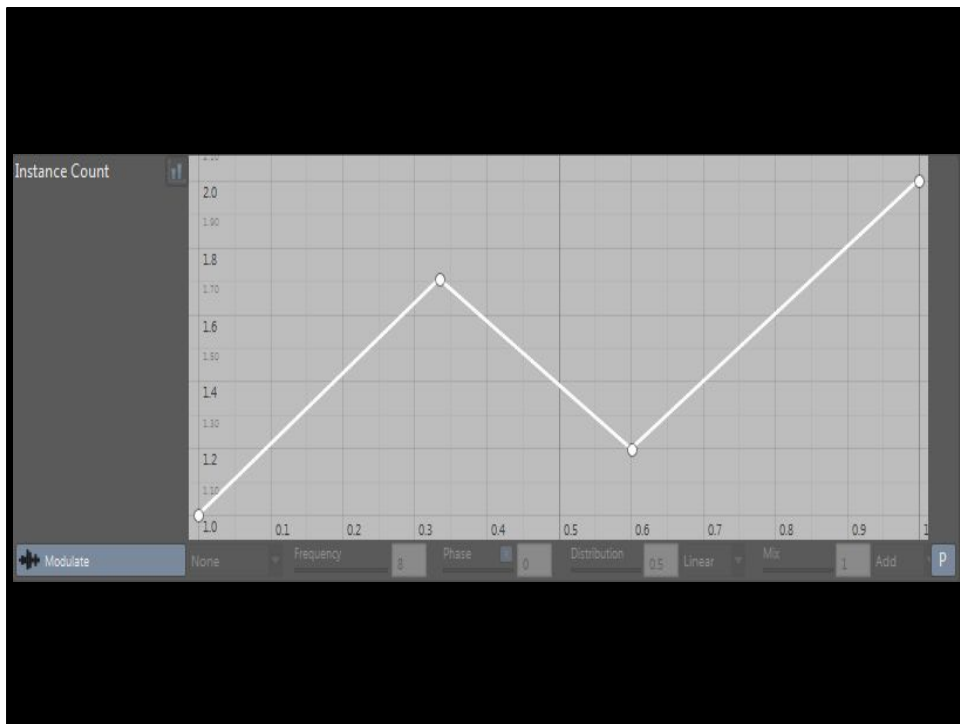


## YOU CAN: PREVIEWING CHANGES

- If an operation makes big changes, help user see “what will happen”.
- Give an opportunity to change their mind for no additional cost.
- You can do this *before or after* a change has begun.
  - *more...*



For example, Visual Script shows user what the result of a node splicing operation will be after they've begun.

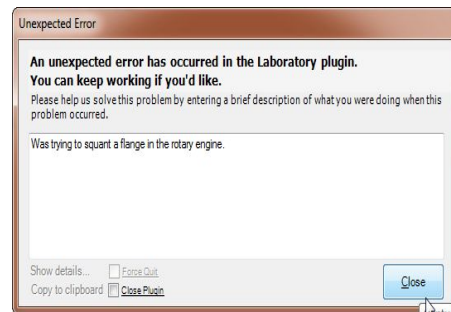


Lab's newest curve editor has a nifty feature that previews results before a click ever occurs.



# Promote User Confidence

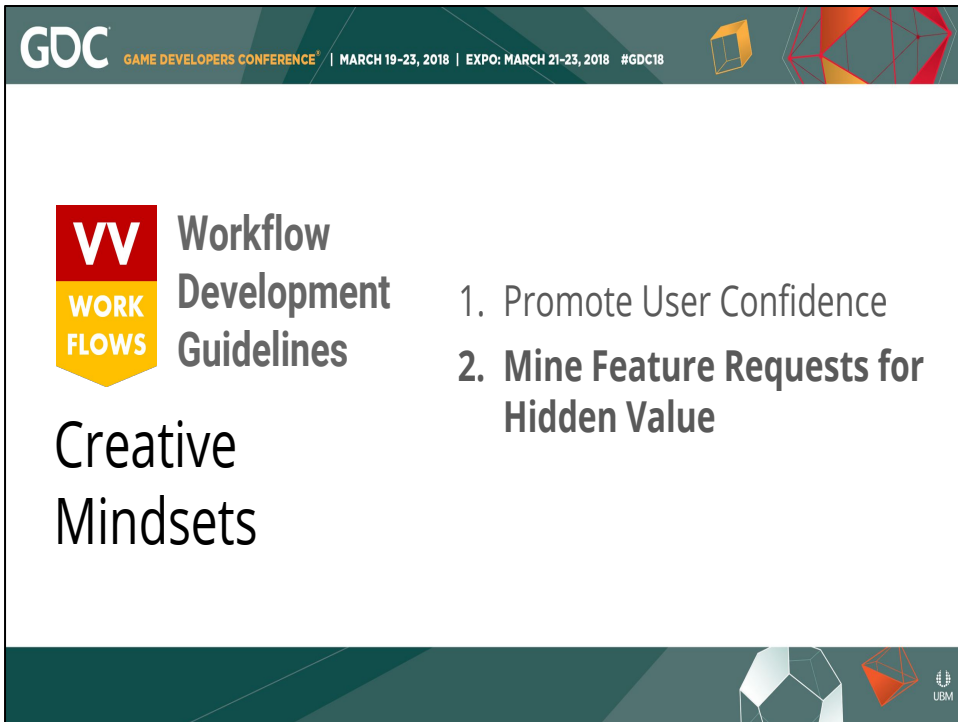
- Undo/Redo Must Work
- Support Source Control
- Do Rigorous Data Validation
- Consider Previewing Changes
- **When Errors Occur, Show the Path to Safety**



## YOU CAN: SHOW PATH TO SAFETY WHEN ERRORS OCCUR

- Anything worse than not knowing how to fix an error?
  - Shakes confidence: “don’t know what I’m doing!!!”
- So, make errors helpful. When they occur,
  - Recover gracefully, guide user to familiar state to continue from.
  - If corrective action required, be clear about next steps, e.g. restart, reload.
  - Consider engaging users to give feedback on error messages.

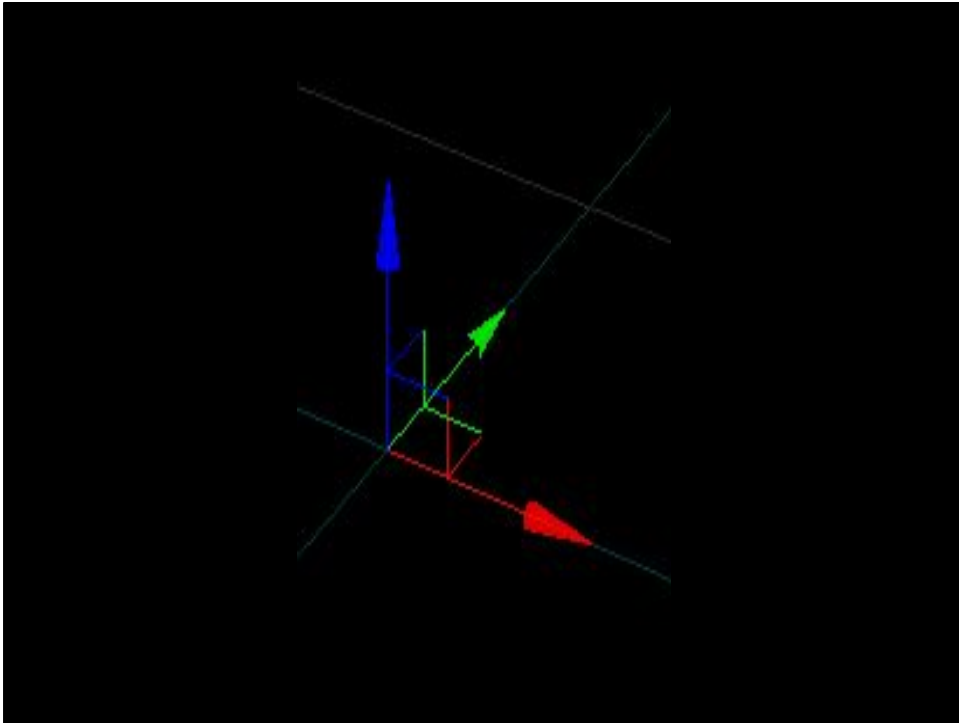
## END OF SECTION



## **NEW SECTION: MINE FEATURE REQUESTS FOR HIDDEN VALUE**

(This is a guideline that seems to resonate with the most people, engineers and content creators alike.)

- Beware of treating user requests as specifications. Risks:
  - incohesive solutions, “Franken-app”
  - “band-aid” solutions that leave root causes unaddressed, inviting subsequent band-aids for the same problem
  - inheriting bad habits from other workflows
- Learn to recognize opportunities to help users:
  - in ways that might not necessarily match stated requests,
  - in ways they don’t expect or know to ask for
- “I want it to behave just like After Effects / Excel” may indicate communication gap; you’re getting only one part of the picture: dig deeper.



## INTERVIEW QUESTIONS

- We assess tools candidates partly on their intuition for this.
- Favorite interview question: a user asks for their transform widget to be bright yellow.
- Some candidates immediately describe changing color values in code.
  - Then I'll say, another user wants it thicker.
    - Then, bigger.
      - Now you have a big, thick, yellow gizmo.
- Real problem: *it's hard to see*.





# Mine Requests for Hidden Value

1. Why is the user concerned with this at all?
2. Could others benefit if... ?
3. What player experience are they going for?
4. Any risk of unintended consequences?



## MORE MINING

- Talk to users! Discover additional value!
- Try these questions...
  - **(CLICK) Back up a step:** Why is user concerned with this in the first place?
  - **(CLICK) Could we look at this differently?** Would a more general solution bring more value?
  - **(CLICK) What player experience are you targeting?** Is there a knowledge gap, and another tool needs to be employed?
  - **(CLICK) Any potential bad habits?** E.g. would a proposed default increase likelihood of blowing your budgets?

## END OF SECTION



GAME DEVELOPERS CONFERENCE

MARCH 19-23, 2018 | EXPO: MARCH 21-23, 2018 #GDC18



Workflow

Development

Guidelines

Creative

Mindsets

- Promote User Confidence
- Mine Feature Requests for Hidden Value
- Keep Users Close to Their Content**

## NEW SECTION: KEEP USERS CLOSE TO THEIR CONTENT

- We want creatives to see, hear, and feel what the player will see, hear, and feel.
- SUCCESS is making exploring a creative space fast, easy, and rewarding.
- INCREASE time spent: seeing, hearing, and feeling the content
- DECREASE time spent thinking about how to operate tools and navigate pipelines.
- This can be done by helping creative people notice little details, or by helping them appreciate content in-context; by enabling them to broadly explore lots of ideas, or deeply explore one or two of the best ideas.
  - Tools can innovate here.

## UGC: BLISTERINGLY FAST!

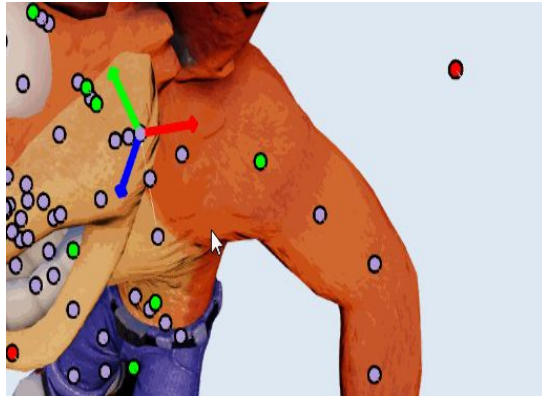
- Tools engineers can find lots of inspiration in UGC tools.
- Their goal: help people express themselves--cohesive content with reasonable effort.
  - Skilled users can create content at *blisteringly fast speeds*.
- **Are these workflows participating in the creative process itself?**
  - Help player see possibilities
  - Coach player through trying new things
- Compelling!

How do you keep users close to their content?



# Keep Users Close to Their Content

- Support Direct Manipulation



## YOU CAN: SUPPORT DIRECT MANIPULATION

- Direct manipulation means make natural-feeling changes directly to content instead of some intermediate representation.
  - E.g. dragging manipulators, picking geo in a scene instead of from a list
- We embraced this in Crash Bandicoot by enhancing an older list-based bolt picker with a gizmo that let users choose and adjust VFX spawn points more directly.
- *more...*

- In addition to allowing the user to pick an object from a list, allow them to click the object in the scene.
- In addition to having a button to assign a material to an object, allow the user to drag the material onto the object.
- Instead of presenting ranged values as a single text box, use sliders (or slide-like gesture, ala numeric draggable spinners) to afford the user fine control over adjustments.
- Transform widgets with built-in constraints are great example: click the part of the manipulator that maps to the constraint you want



# Keep Users Close to Their Content

- Support Direct Manipulation
- **Don't Obscure Content, Augment It**



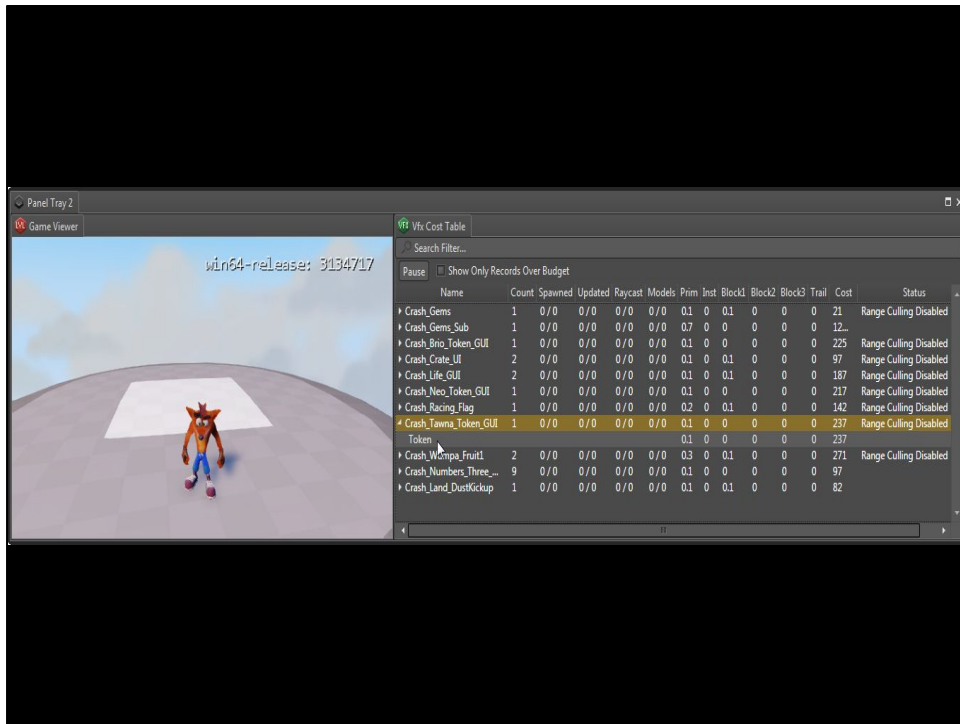
## YOU SHOULD AUGMENT CONTENT, NOT OBSCURE IT

- If you want to keep users close...
- Don't come between them and the content they're trying to create.
- A simple example, a trend we see more and more: instead of shading a selected object--which obscures its appearance--just outline it.
- *more...*



## DEBUG SPEW!

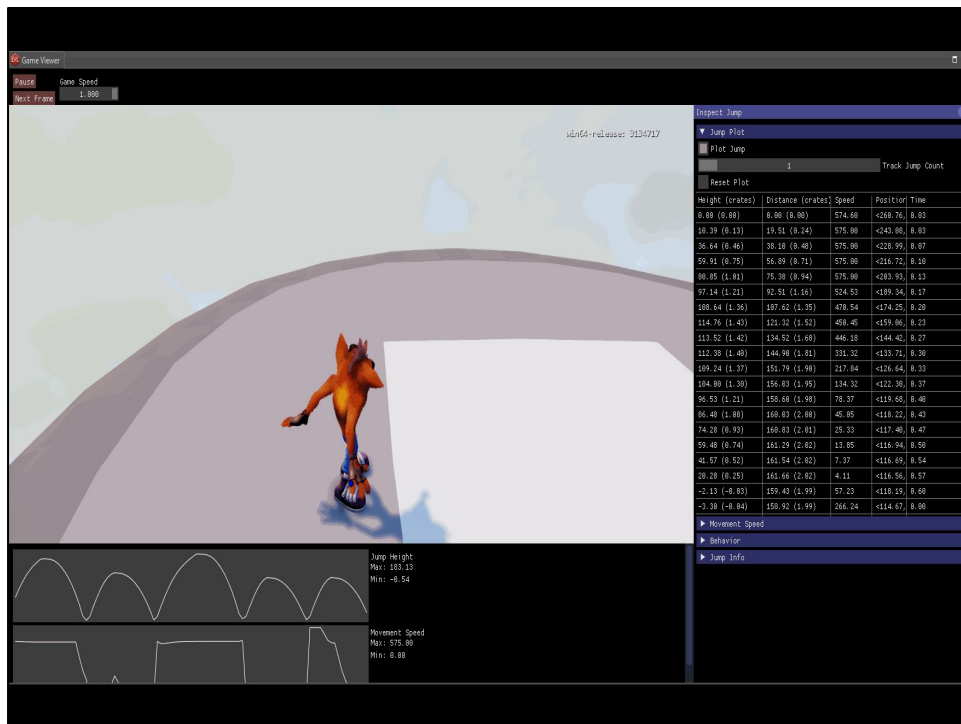
- E.g. don't display tables of real-time data on top of the game window.
- They tend to be very hard to read.
- We did this for a long time and began moving away from it.
- Instead, IGM dedicated screen real-estate to presenting data visualization.
- ImGui



## EVEN BETTER...

- But, Lab was a more powerful visualizer, with richer UI.
- So went the other direction: brought the game into Lab.
- Users could arrange it in their layout like any other panel.





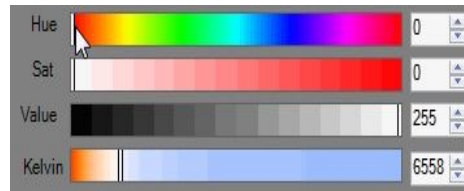
## VISUALIZATION!

- “Augmenting” might mean visualizing information that’s already there.
- From IGM: visualize Crash jumps for fine-tuning.



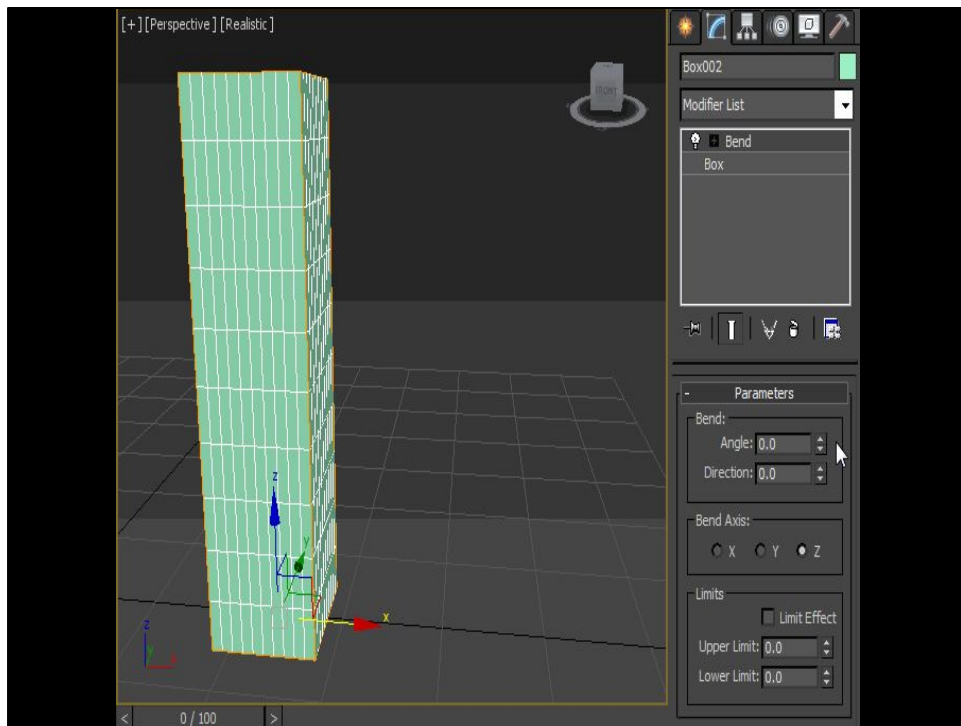
# Keep Users Close to Their Content

- Support Direct Manipulation
- Don't Obscure Content, Augment It
- **Promote "Mixing Board Thinking"**



## YOU CAN: PROMOTE MIXING-BOARD THINKING

- Here "mixing board" means controls that **blends and constrains many inputs** into one or more results.
- This approach:
  - a. relieves users of having to maintain mental models of complex data structures and constraints, and
  - b. invites them to "play" with the result and try combinations safely within parameters that you enforce behind-the-scenes.
- You see this all the time with character creators: tools that help players be creative and produce good-looking results.
  - a. Skyrim does it by exposing lots of pickers and sliders.
  - b. While Black Desert leans into direct manipulation.
  - c. Both are "mixing boards": facilitate variation, apply constraints, abstract away complexity
- Best of all? Workflows that offer "knobs and dials" to control influence, weighting, intensity, etc. can promote [emergence](#) during the creative process.
  - a. Help creatives "stumble upon" great work.



## MODIFIER STACKS AS MIXING BOARDS

- Some workflows support modifier stacks, like Max.
  - I think: “modular mixing boards”
  - Results of one board feed into another
- You might think of animation blending workflows as mixing boards that are themselves mixed - dialed up and down



# Keep Users Close to Their Content

- Support Direct Manipulation
- Don't Obscure Content, Augment It
- Promote "Mixing Board Thinking"
- **Promote "Copy, Transform, Combine"**

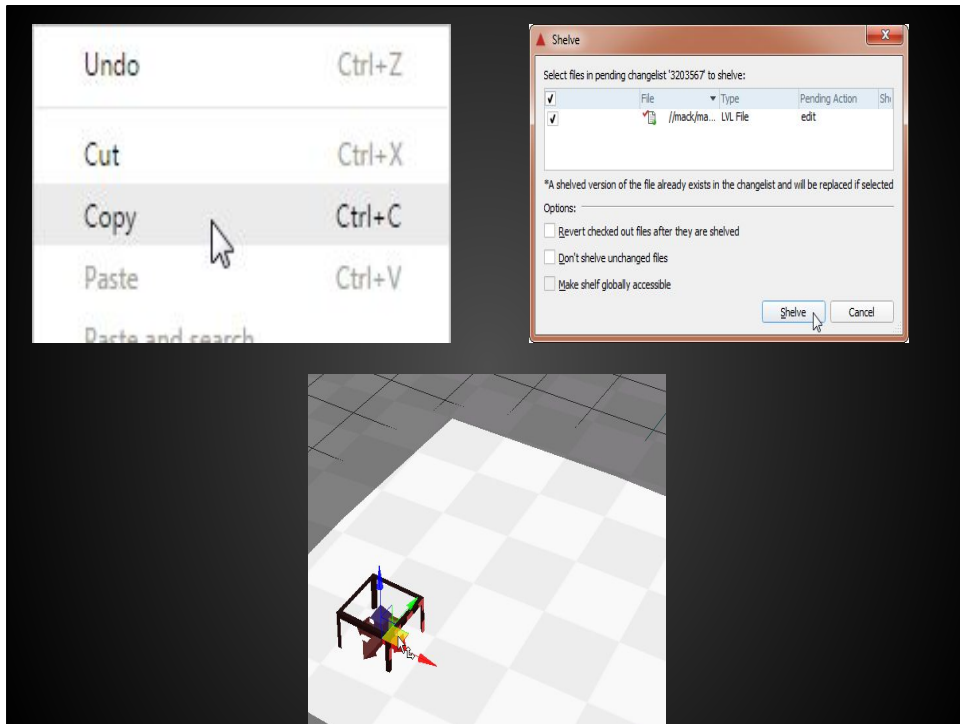
*"Everything is a Remix"*  
by Kirby Ferguson

<https://www.everythingisaremix.info/watch-the-series/>



## YOU CAN PROMOTE COPY/TRANSFORM/COMBINE THINKING

- This is the most abstract guideline?
- Very concise expression of a powerful idea behind the best tools.
  - I found it very inspirational and I'm still trying to articulate it in guideline form.
- Here we go...
  - I was exposed to this in a video called "Everything is a Remix" by Kirby Ferguson
  - These three steps are important to any creative endeavor, but it's obvious to me that this applies to content creation tools in important ways.
  - You can help users explore creative spaces--*and stumble on new ideas*--by building workflows that excel at these operations.



**Copying** is foundational to content creation.

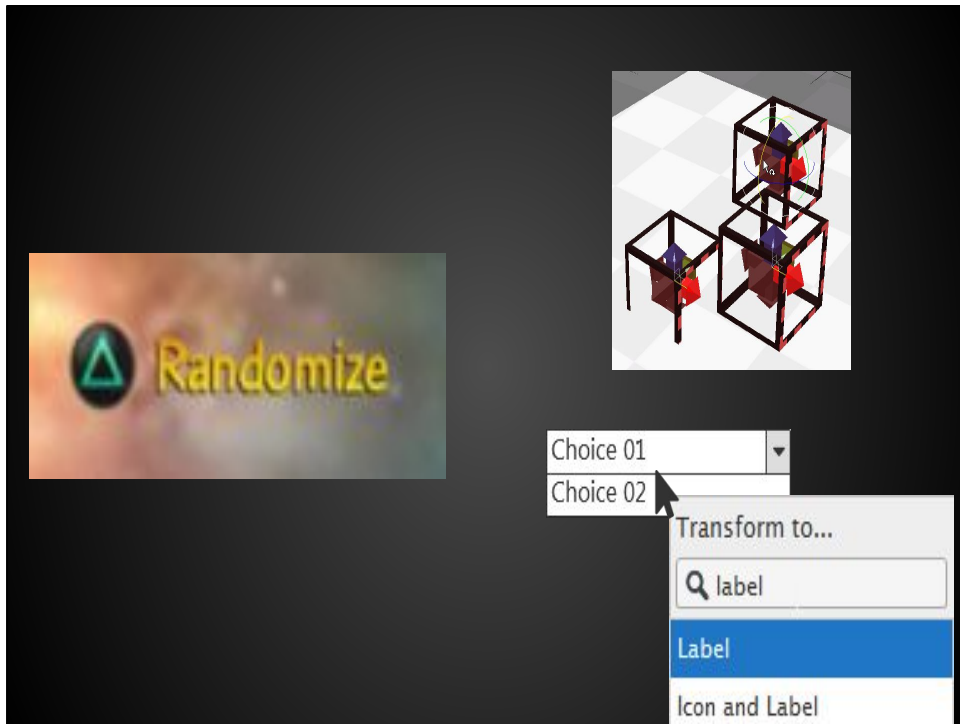
- It skips time-consuming boilerplate work.
- It's a branch in a thought process.

It lets users:

- Rapidly create sets and repetitions that can vary independently
- Create ad hoc "checkpoints", like shelving a copy of your changes to return to later.

What you can do:

- Support copy/paste, for values, selected objects, etc.
- Support "prefabs" / "instancing" so copies can inherit subsequent transformations
- Support robust cloning in your data models (e.g. Alchemy's cloning & ownership)
- Support *fast* copying
- Help infer metadata & bookkeeping, like new object names

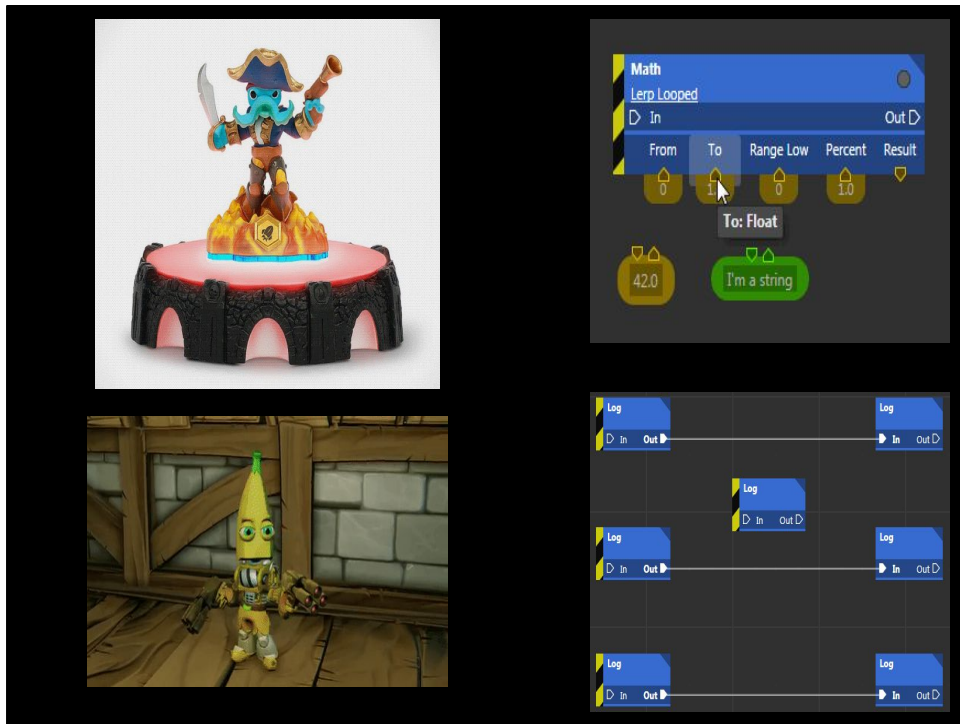


**Transformation** is the next step in creating *variation*, the vehicle with which we explore creative spaces.

- Typically follows a copy
- Sometimes users have a clear end-result in mind, so transformations should be straightforward.
- Sometimes they don't. Transformations might be done experimentally, i.e. without clear intent, as part of a "let's try this" idea. Tools can help infer things to make that possible.

What you can do:

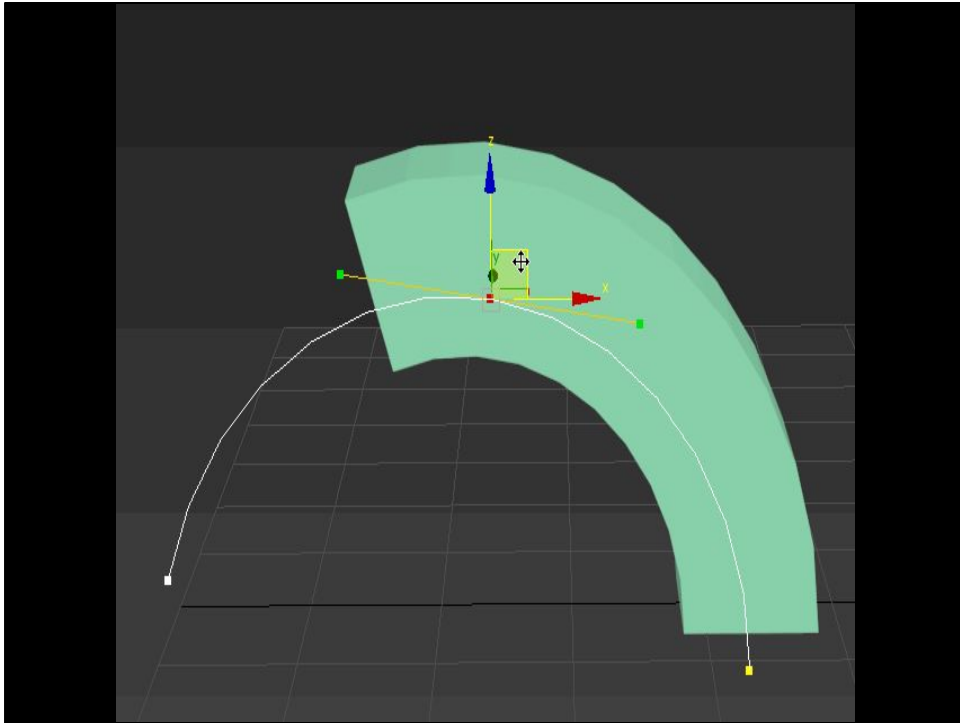
- Support random variation
- Support mutations of *sets*
- Get users to success faster by guiding their transformations, e.g. with inferred snapping
- Help users change content between compatible representations



**Combining** concepts contributes to uniqueness and innovation.

Things you can do:

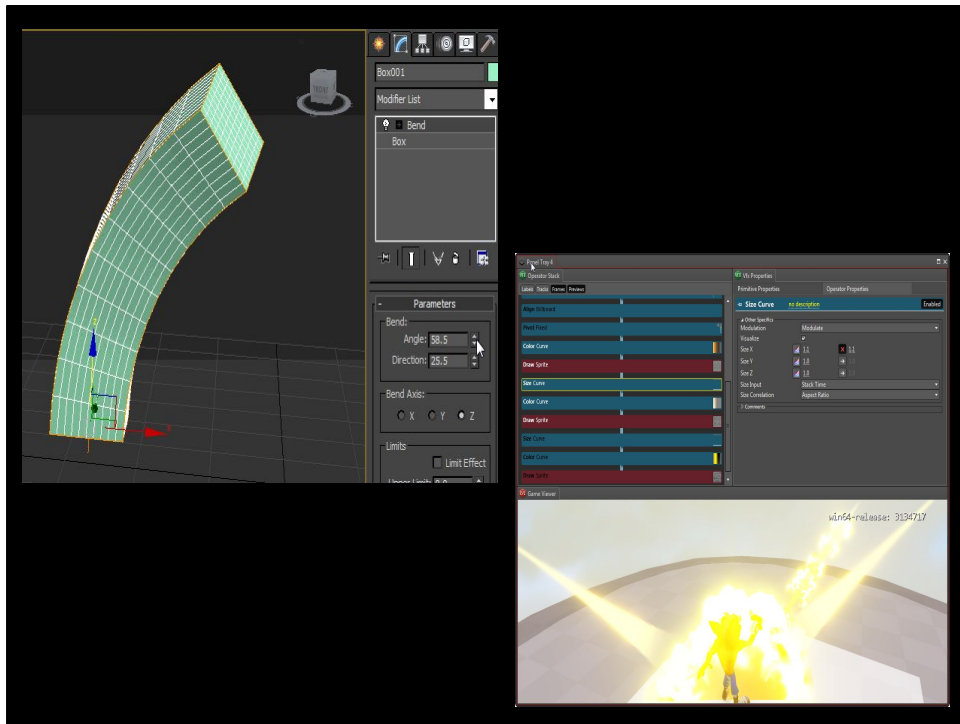
- **show how** things can be combined (and how they can't)
- **accelerate combination** with implicit edits (e.g. [Visual Script's node splicing](#))



### INTERPRET COMBINATIONS IN MEANINGFUL WAYS

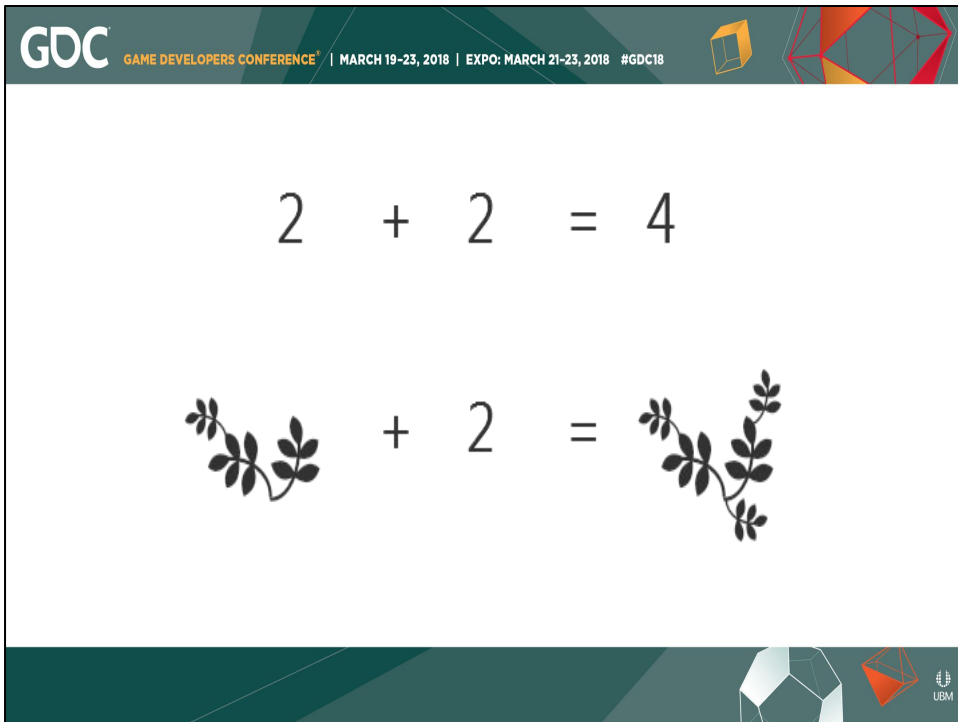
- Transformation can occur explicitly (e.g. moving a vertex),
- but **combination lets it happen implicitly** (e.g. procedural deformation).
- Tools can **interpret combinations in meaningful ways**.
  - by selectively use only parts of the combinants





## MORE MODIFIER STACKS

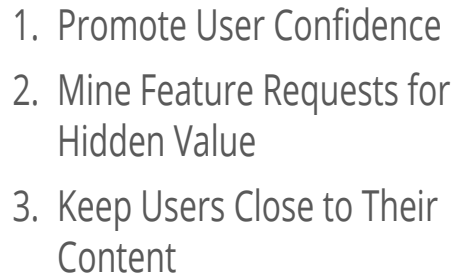
- I think modifier stacks are a great manifestation of these principles.
- They yield powerful idea: “combine results of transformations”.
  - I remember MAX modifier stack being profound
  - Lab VFX Operator stack inspired by this idea
- When intermediate transformations are parametric, **content becomes programmable**.
- **Variation becomes injectable**.



## NOT NEW, BUT HARD TO GROK

- Not new insights about C/T/C!
- But people “not in the trenches” “grok”/internalize in terms of workflows.
- That’s the gap the guidelines want to fix.
- For techno-creativity like game development, the “magic” here--the conceptual leap I want to shout from rooftops--is interpreting creative content combinations in meaningful ways.
  - **(CLICK)** Going from: **2 + 2 = 4**
  - **(CLICK)** To: **“Tree+2=More Tree”**
  - I suspect there’s untapped potential here for innovation: iterating on player experiences, making content creation a more meaningful, valuable exercise.
- I think we engineers can contribute a lot here,
  - Engineers already understand this in several forms: polymorphism, operator overloading.
  - We’re good at “making the incompatible “compatible””--by rethinking types and complex models
  - We’re good at “content algebras”
- To keep users close to their content, we should give them less reason to be distracted from it.
  - By making “creation” more seamless, making combination effortless, they can stay in flow.

**END OF “THINKING ABOUT THE USER EXPERIENCE” CATEGORY**



- This part was abstract and long, so let's recap.
- By:
  - Promoting User Confidence
  - Mine Feature Requests for Hidden Value
  - Keep Users Close to Their Content
- ...the guidelines aim to help readers “put themselves in users’ shoes”.
- This is one of the key philosophies behind Alchemy’s development,
- But going forward, past Alchemy, these and the other guidelines aim to inspire the way VV approaches workflows in general.

GDC

GAME DEVELOPERS CONFERENCE<sup>®</sup> | MARCH 19-23, 2018 | EXPO: MARCH 21-23, 2018 #GDC18



VICARIOUS<sup>™</sup>  
VISIONS

[vvisions.com](http://vvisions.com)

We're hiring!

Questions?

[jestewart@vvisions.com](mailto:jestewart@vvisions.com)

@object01 #GDC18



UBAM

That's my talk!

This is VV at a party! (I'm the one doing **jazz hands**.)