IT STINKS AND I DON'T LIKE IT

BUILDING A BETTER ENGINE EXPERIENCE AT INSOMNIAC GAMES

Hey everyone, I'm sure you've heard it a million times already, but make sure that you silence your cell phones, and make sure to fill out those surveys!

**NOTE: Any "•" indicates advancing a bullet point. If you would like to download the full powerpoint/keynote of this presentation, please visit the following site where you can download it and view it as it was intended to be viewed:

bit.ly/insom_rnd

Thanks,
Sean Ahern

Thank You!

First off - thank you all for being here, I hope that we're going to have an awesome hour together

Sean Ahern

Information Architect - Insomniac Games

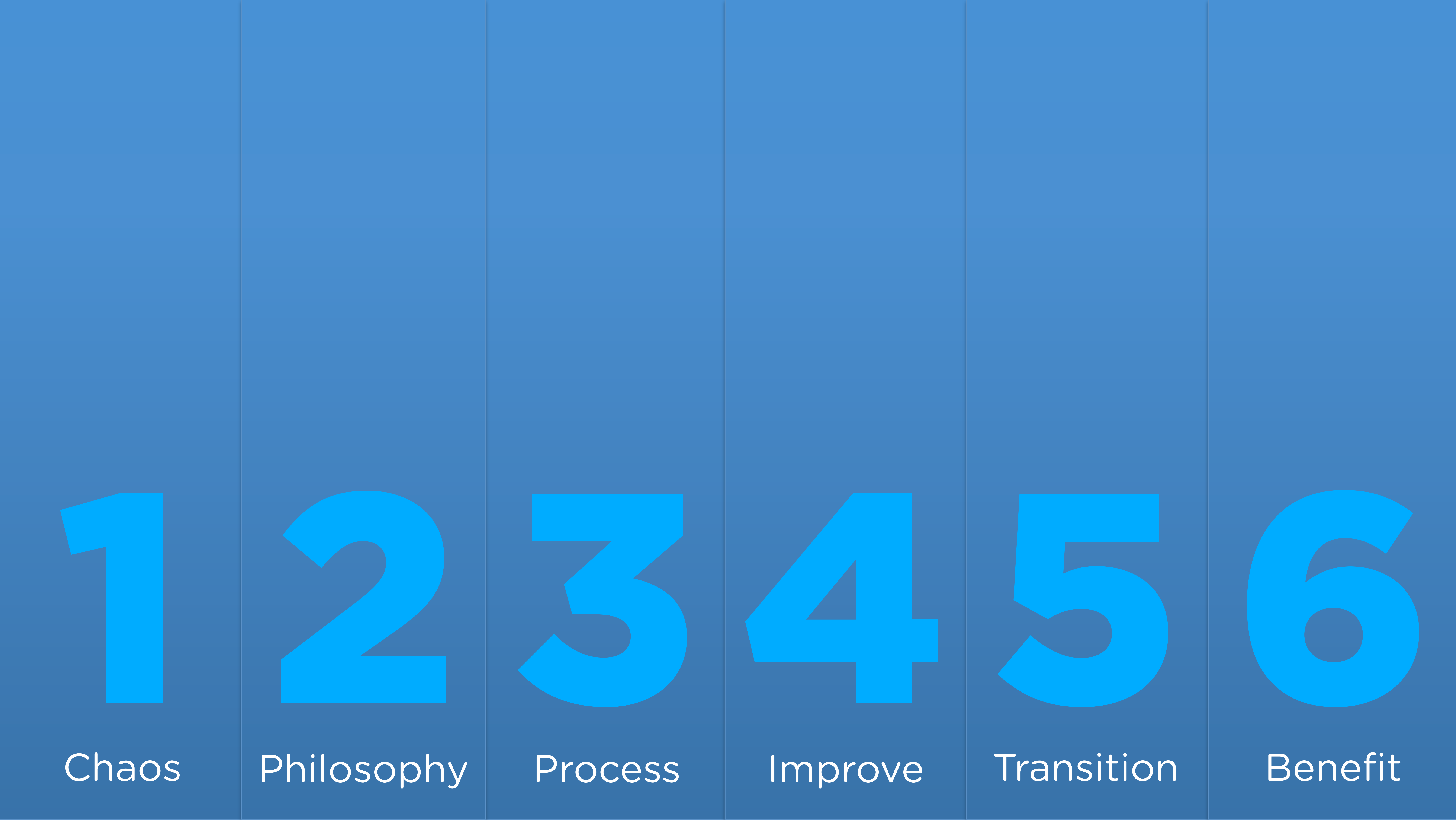
Motion Graphics - Pharmaceutical Industry

So hey everyone, my name is Sean Ahern

- A lot of people think that I look like Kevin Bacon

- At Insomniac Games I’m an Information architect, and I’ve been doing this for about the past 2 years now.
- What is an Information Architect? In the simplest of terms, I make our tools look pretty and work as well as they can.

- Prior to Insomniac I worked for 3 1/2 years doing Motion Graphics work for the Pharmaceutical Industry



1 Chaos

2 Philosophy

3 Process

4 Improve

5 Transition

6 Benefit

So today we’re going to be talking about 6 things:

- We’re going to start out with the overall State of chaos that existed with our tools in beginning

- Then we’re going to go into Insomniac as well as my own user experience and usability philosophies

- After this we’ll talk about our process for tools development

- Next we’ll touch on how we improve tools User Experience once they’re released and into the wild

- Following this, we’ll talk about our transition from a traditional tool set to a web-based tool set

- Finally, we’ll finish up with how we’ve benefitted from focusing so heavily on the user experience and usability, as well as take any questions that you guys may have

In the beginning…there was chaos

Programmers making tools for programmers

Tools didn’t make sense

Icons didn’t make sense

Pun based nomenclature

Visual clutter

Downtime galore

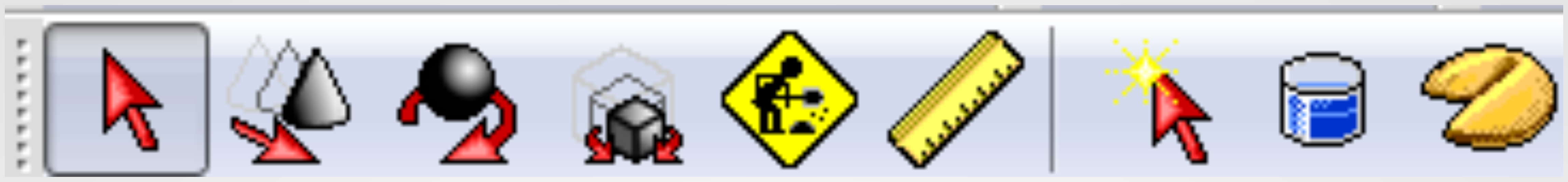
- When I first started, there was a feeling in other departments that our tools were suffering because it was almost as if it was just “programmers making tools for programmers”.

This was primarily due to 5 key issues:

- Tools didn’t make sense
- Icons didn’t make sense
- Pun nomenclature
- Visual clutter
- Downtime galore

Construction Zone

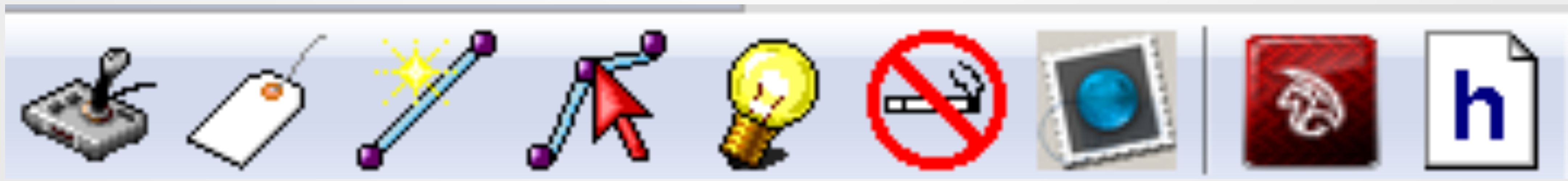
Fortune?



Select Point

Tag Selected

Select/ Create curve point



Create Curve

... Uhhhhhh?

So in order to help illustrate this, I want to show an example from our old tools. This was the main toolbar from our old tool set

Now there are some obvious ones up here such as select, move, rotate, scale

There are also some others that are a bit more unclear:
- So first...maybe this is a construction zone?
- Next I guess that this one is selecting a point
- A...fortune?
- I suppose this one is tagging whatever we have selected
- Creating a Curve
- Selecting/ Creating Curve /Point
- Annnnd...uhhhhhhhhh?

WTF?!

So with this one...were they reminding us not to smoke and be healthier?

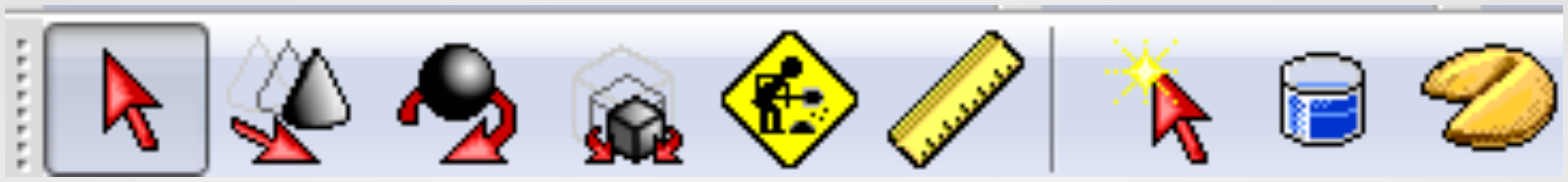
- What does this thing even do?

Poll audience

Alright, well personally I guess it’s a button that makes something that gets rid of smoke effects or creates a volume that doesn’t allow smoke effects? I don’t know, that’s my best guess

Construction Zone

Fortune?



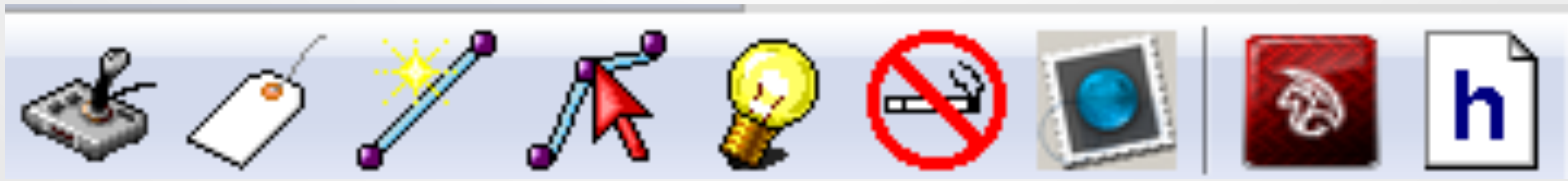
Select Point

Select/ Create curve point

Tag Selected

Make Stamp

Create Header



Create Curve

No smoke effects?

Open Maya

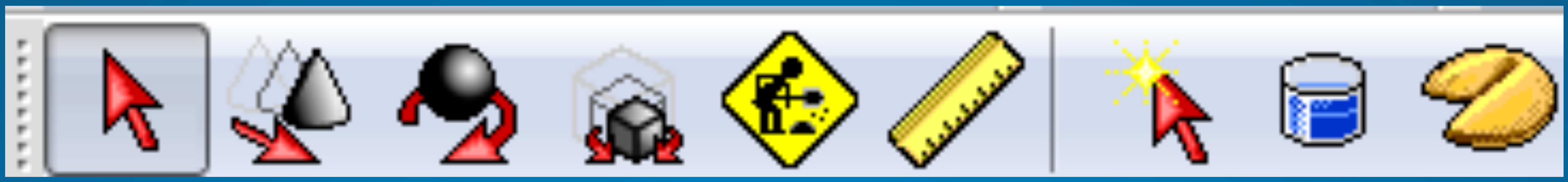
So I guess we’re going with no smoke effects.

Others:

- Then we’ve got making a…stamp?
- Opening Maya with what’s selected
- Finally we have Creating a Header

Duplicate Selected Numerous Times

Place Clue



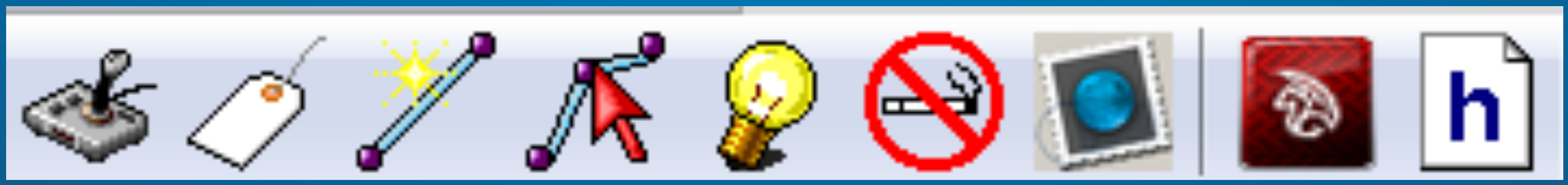
Place Entity

Place Locator Object

Edit Selected Curve

Create Post Vol

Flush Symbol Defs



Create Curve

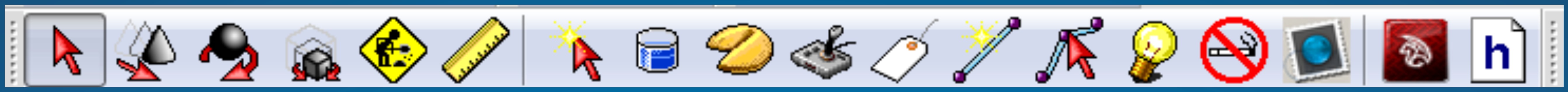
Light Objects in Scene

Open Maya With Selected

Now what it really was, was something more along the lines of this:
Duplicate selected numerous times, place an entity, place a clue (whatever that is), Place a locator object (not sure what that is), create a curve (yess! got one right), edit the selected curve.

•Now for our No smoking Sign... survey SAYS!...Light...objects in scene...And the thing about it is, I had to actually look it up in the tools to find this out. Anyway...continuing along.

•Create a post volume, Open maya with selected, and finally Flush symbol definitions.



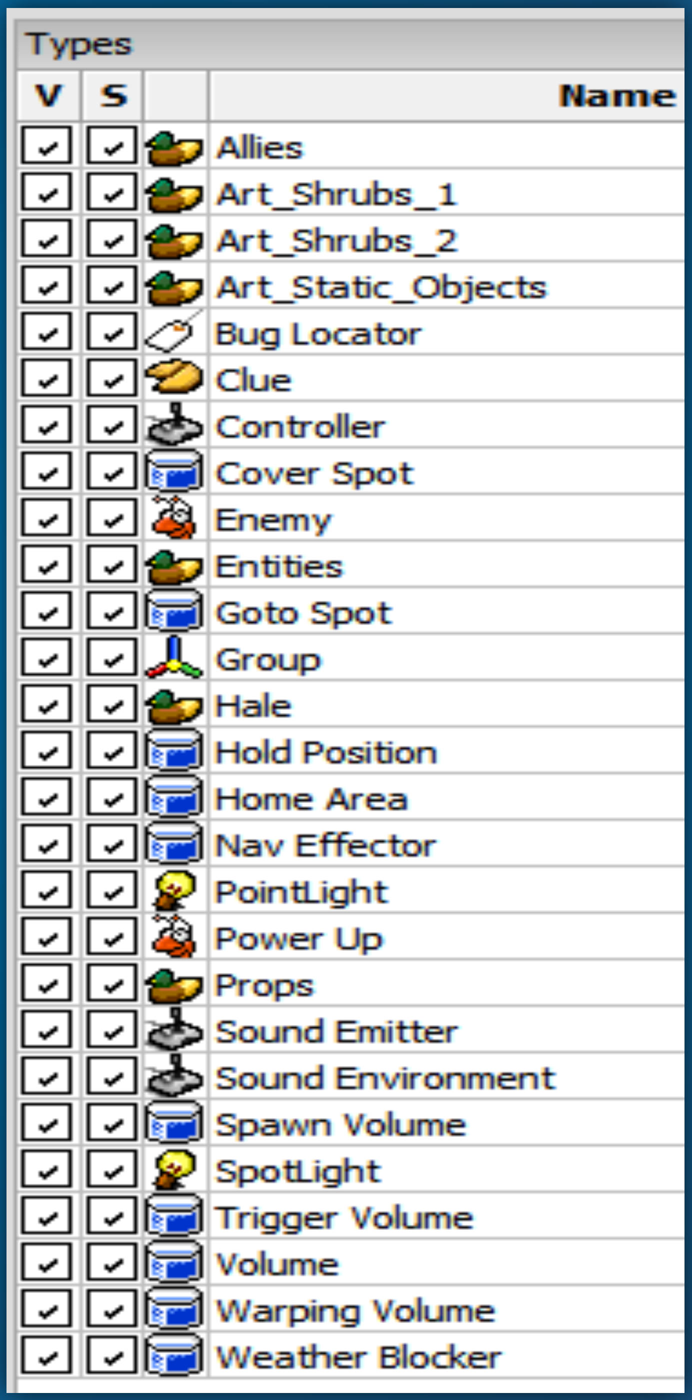
Here’s what I thought:	Here’s what they are:
Select	Select
Translate	Translate
Rotate	Rotate
Scale	Scale
Construction Zone	Duplicate selected numerous times
Measure	Measure
Select Point/Select new	Place entity objects
Place volumes	Place volumes
Fortune	Place clue objects
Place controller	Place controller
Tag	Place locator objects
New Curve	New Curve
Select Curve Point	Edit selected curve
Make or select lights	Place lights
No Smoke? No Smoking zones	Light objects in the scene
Make stamp	Place post processing volume
Open maya	Open maya with selected objects
Open headers	Flush symbol defs to re-parse headers

So the thing is, I’ve been using a ton of different engines and tool sets since I started in the 3rd grade with DoomIT. I’ve used Source, QRadiant, UDK, etc…really a *ton* of different tools throughout the years.

Now even with using all of these different software packages throughout the years, I still only got 1/2 of them right

Something doesn’t add up here

Single icons applied to *multiple* types



Types

V	S		Name
☑	☑		Allies
☑	☑		Art_Shrubs_1
☑	☑		Art_Shrubs_2
☑	☑		Art_Static_Objects
☑	☑		Bug Locator
☑	☑		Clue
☑	☑		Controller
☑	☑		Cover Spot
☑	☑		Enemy
☑	☑		Entities
☑	☑		Goto Spot
☑	☑		Group
☑	☑		Hale
☑	☑		Hold Position
☑	☑		Home Area
☑	☑		Nav Effector
☑	☑		PointLight
☑	☑		Power Up
☑	☑		Props
☑	☑		Sound Emitter
☑	☑		Sound Environment
☑	☑		Spawn Volume
☑	☑		SpotLight
☑	☑		Trigger Volume
☑	☑		Volume
☑	☑		Warping Volume
☑	☑		Weather Blocker

So I also mentioned that we had a problem with our Icons not making sense. One of the big issues that we had was that we were using single icons for multiple different types of assets.

A number of these just don't make sense.

We have a duck for an ally, art_shrubs, whatever a "Hale" is, Props, Entities, etc. We have an ant being used for enemies and power ups.

I mean, my first question is: why are we even using ducks and ants to begin with? Are we making a zoo game or something else?

Why aren't we using appropriate icons for each thing?

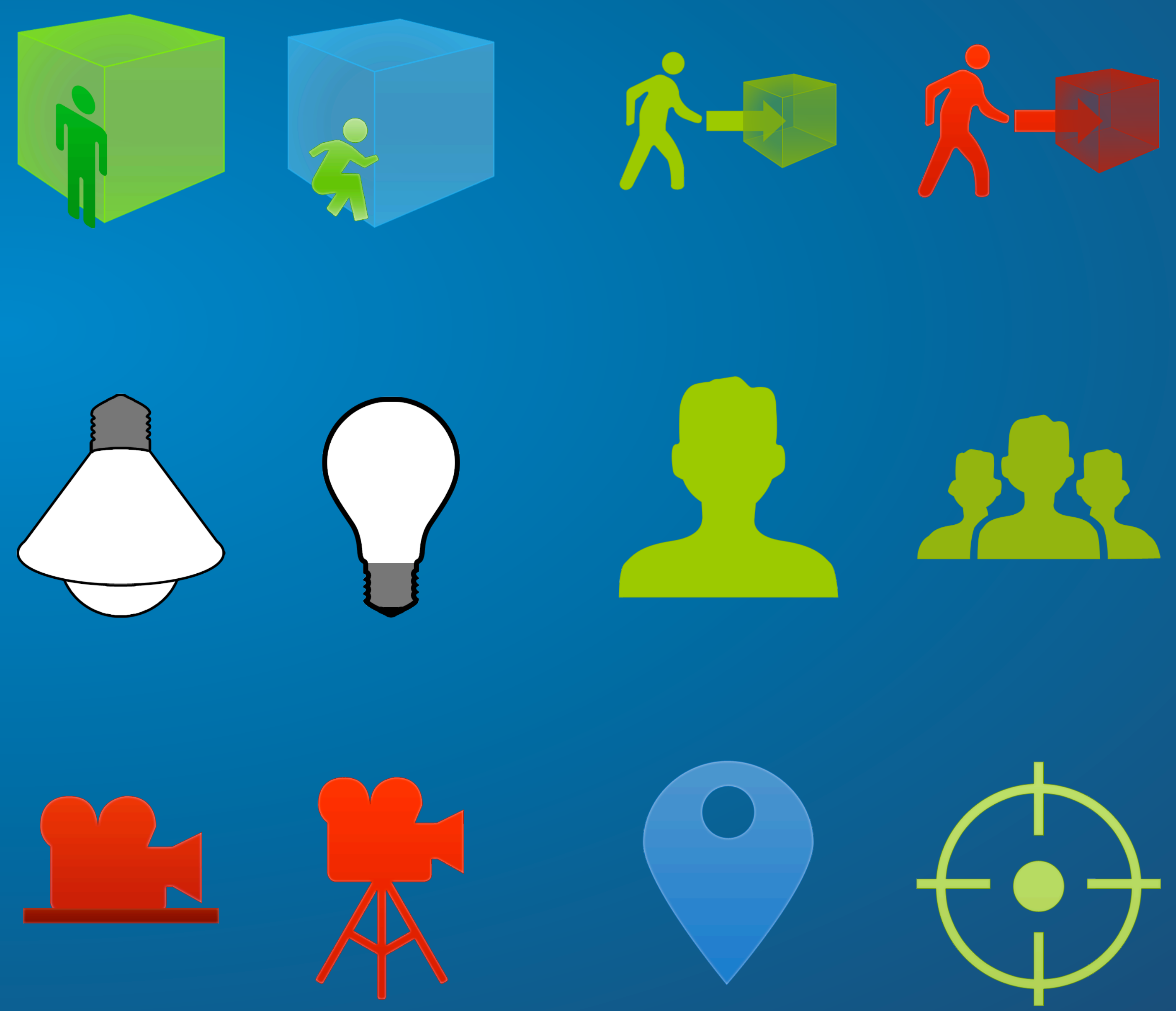
Icons Making Sense

Color coded based on categorization

Different icons for different types

Icon visually represents intended functionality

If at all possible use known images



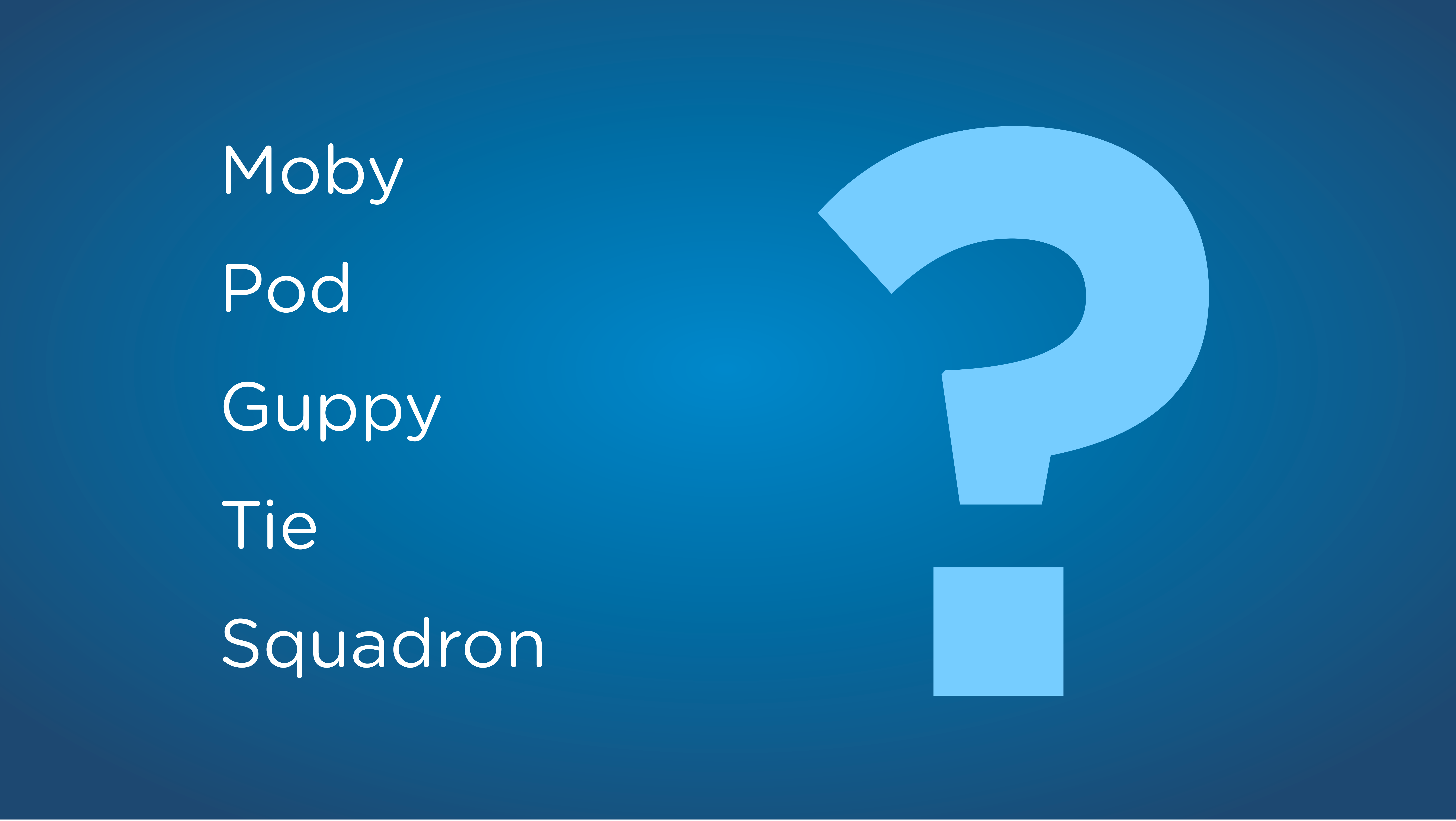
So what we did was completely revise the icons so that they do make sense:

- The first thing we did was make sure that things were color coded based on their categorization:
green: friendly, red: enemy

- Next, we have different Icons being used for different types of things. Instead of just having one lightbulb icon for all lights, we have a different icon for a spot light and a point light

- We also have the icon visually represent functionality - as you can see here, we have a single and a team spawn point, and it's green as well so that you can tell that its friendly.

- Finally, if we can, we want to make sure that we're using known images so that we can convey meaning quickly and easily:
Here we have a stationary camera, we're using a crosshair for what we're calling a firepoint, etc



Moby

Pod

Guppy

Tie

Squadron

So like I mentioned, we greatly suffered from a ton of pun based nomenclature - especially confusing for someone new to insomniac

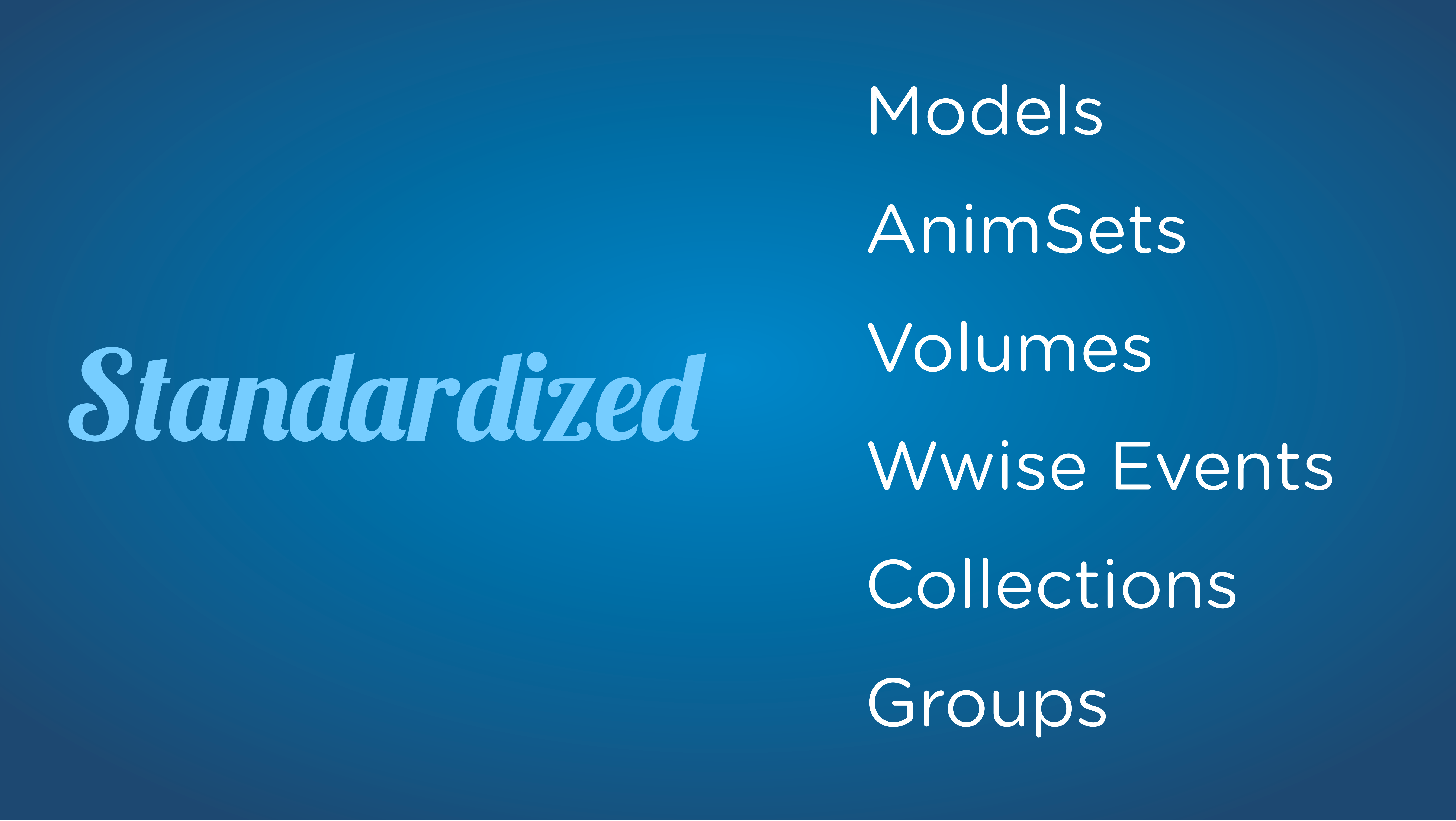
When I started at insomniac, I had no idea what anyone was talking about, and it took me a good seven to eight months to really understand what people were talking about.

So we’ve got the following:

Moby – Movable object, anything that can be animated
Pod – A Group of Mobys
Guppy – A Smaller Moby

Then we’ve also got
Tie – Static Mesh/Model in a level
Squadron (or a group of tie fighters for all you star wars fans) - a group of ties

- Especially confusing for someone new to Insomniac



Standardized

Models

AnimSets

Volumes

Wwise Events

Collections

Groups

So we went and standardized all our asset types. No longer do we have Moby, or Tie, or anything else. Now we have things like:

- Models, AnimSets, Volumes, etc

We wanted to make sure that we were using industry standard terminology to make sure that regardless of whether you were new to the company, transferring departments, whatever. We wanted to make sure that people knew what everyone was talking about without having to have any explanations required.

Asset

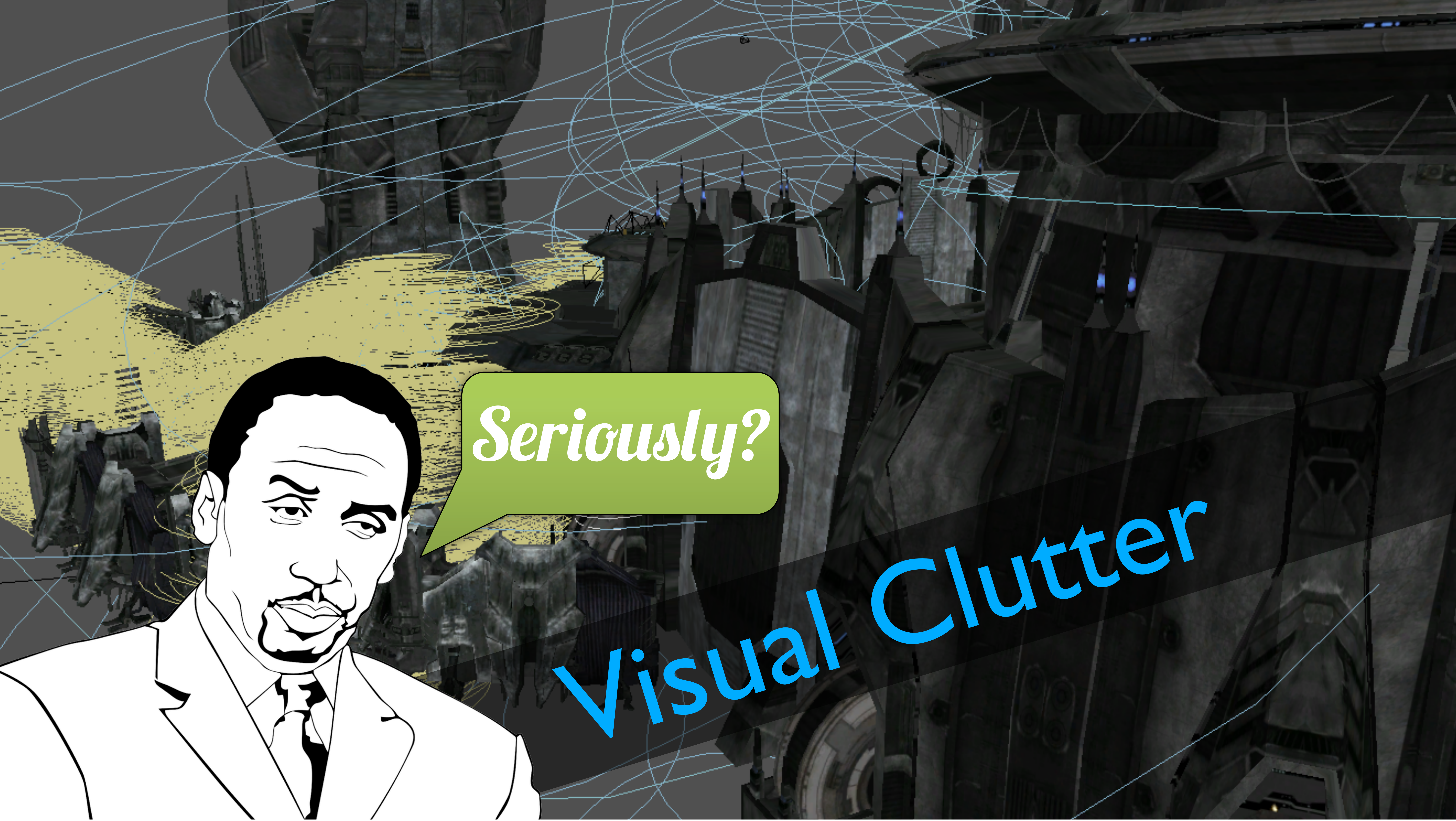
Instance

We also further simplified everything: everything is divided into what we’re calling Assets and Instances

- Assets are our base/source file

- Instances are copies of the asset that have been placed into a level

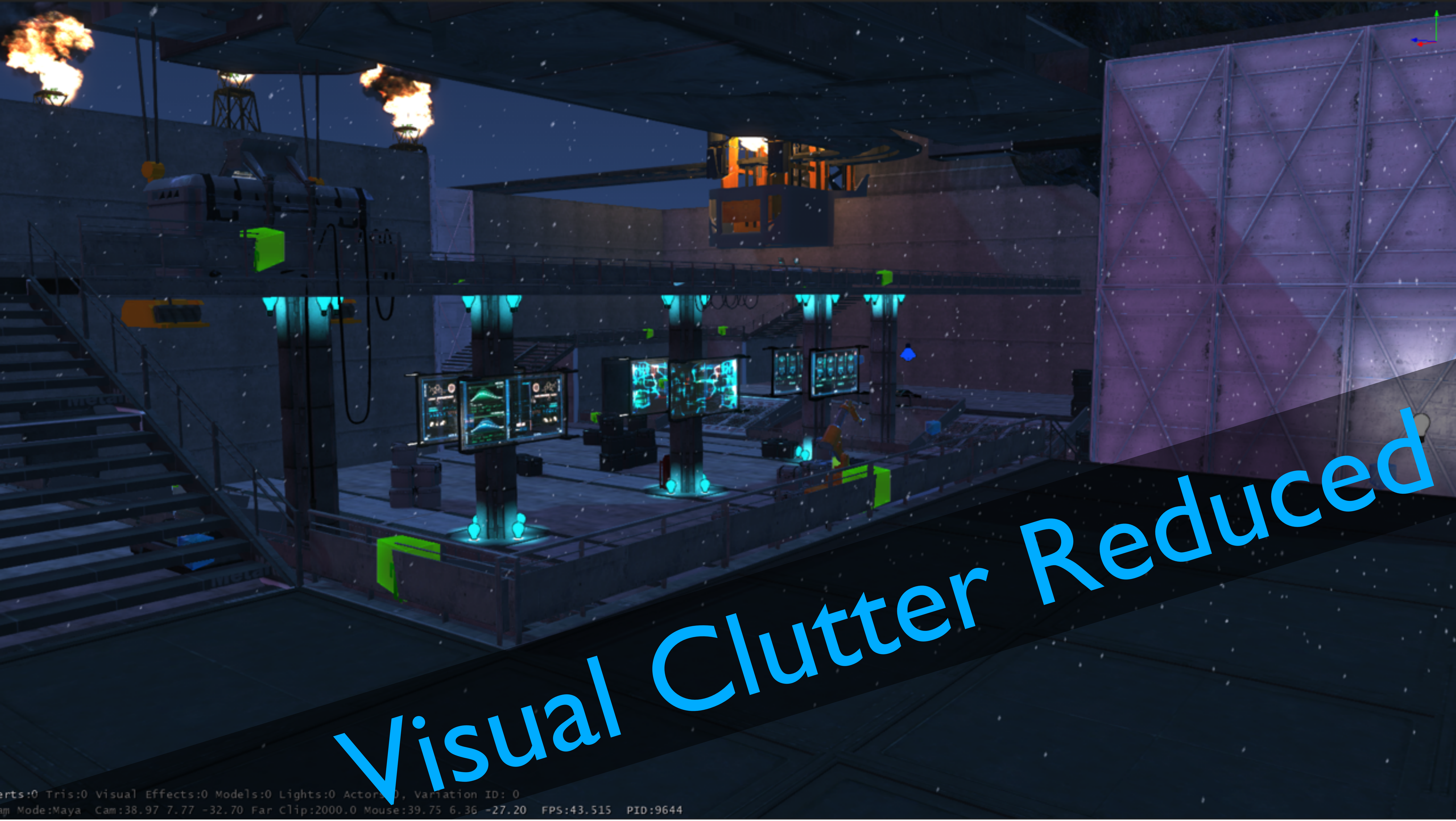
Assets and Instances can be further classified by what type of thing they are: Model Instance/Asset, VFX Asset/Instance, etc



Another big problem that we suffered from was Visual Clutter in our level editor

- What are we even looking at?

This is just an example from Resistance 3 of a tiny section of one of our levels. The first time that I loaded up the tools at Insomniac to check out how their levels worked, it took me a good **hour** and asking someone else for help before I could even work with anything in the level



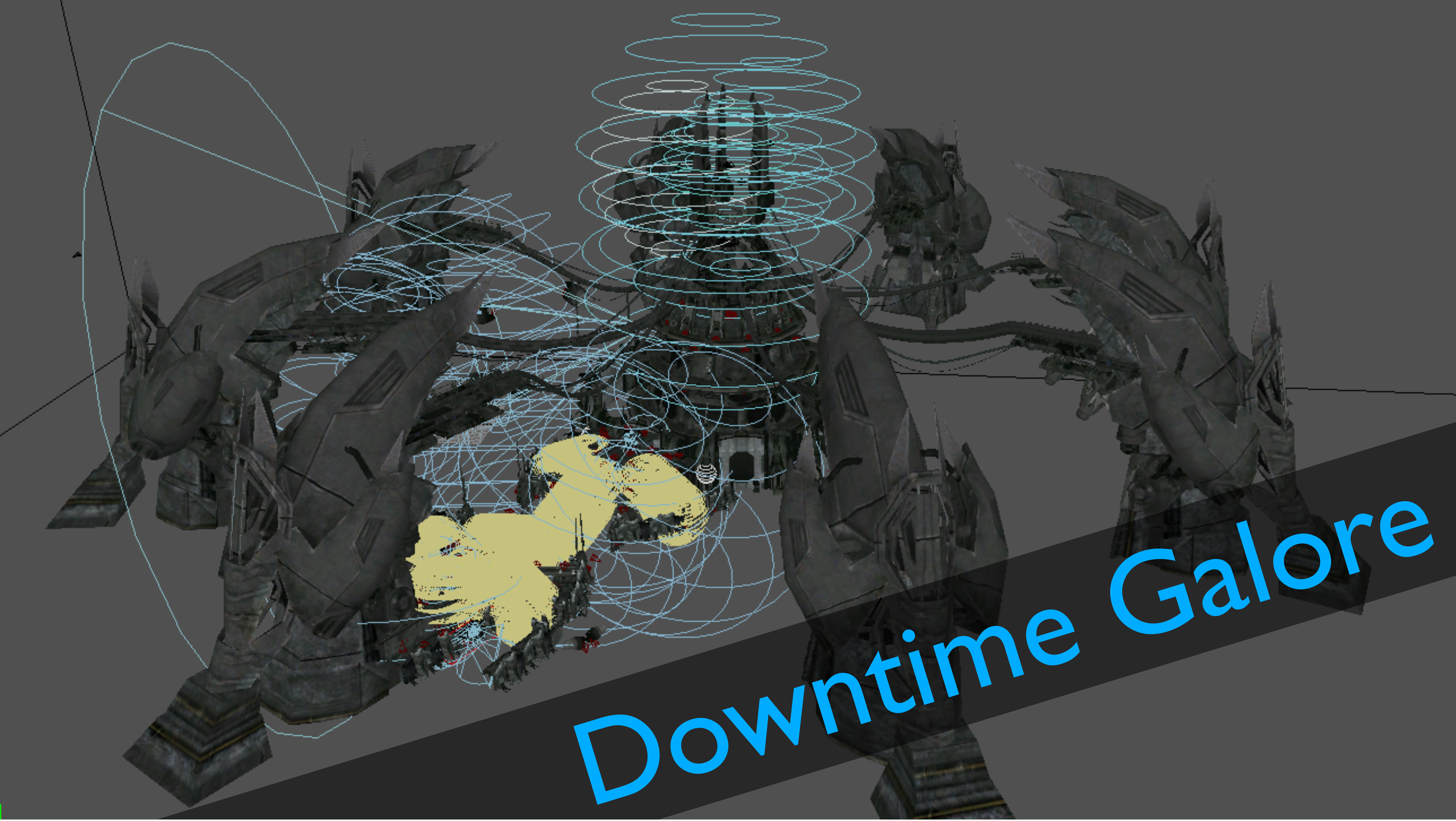
Visual Clutter Reduced

```
erts:0 Tris:0 Visual Effects:0 Models:0 Lights:0 Actors:0, Variation ID: 0
am Mode:Maya  Cam:38.97 7.77 -32.70 Far Clip:2000.0 Mouse:39.75 6.36 -27.20  FPS:43.515  PID:9644
```

With all of the changes that we’ve done, everything is now **much** easier to visually parse

I can actually see what I’m working with immediately without having to turn off a bunch of wireframes and debug spheres

Now granted, I can turn on the debug info/wireframes if I want to but they’re off by default



Downtime Galore

One of the problems we had with our old tools was a ton of inefficiency in how the tools worked - leading to a TON of downtime.

This particular sub-section of a level in Resistance 3 took 5-7 minutes to load up. It’s not even that immense of a level either, there’s less than 500 entities contained within the level. When loading up a level, basically had to leave your desk as doing anything else could freeze the program.

Load A Level

5 - 7

Minutes

Changes In Visibility

3 - 4

Minutes

Load Any Changes

5-10

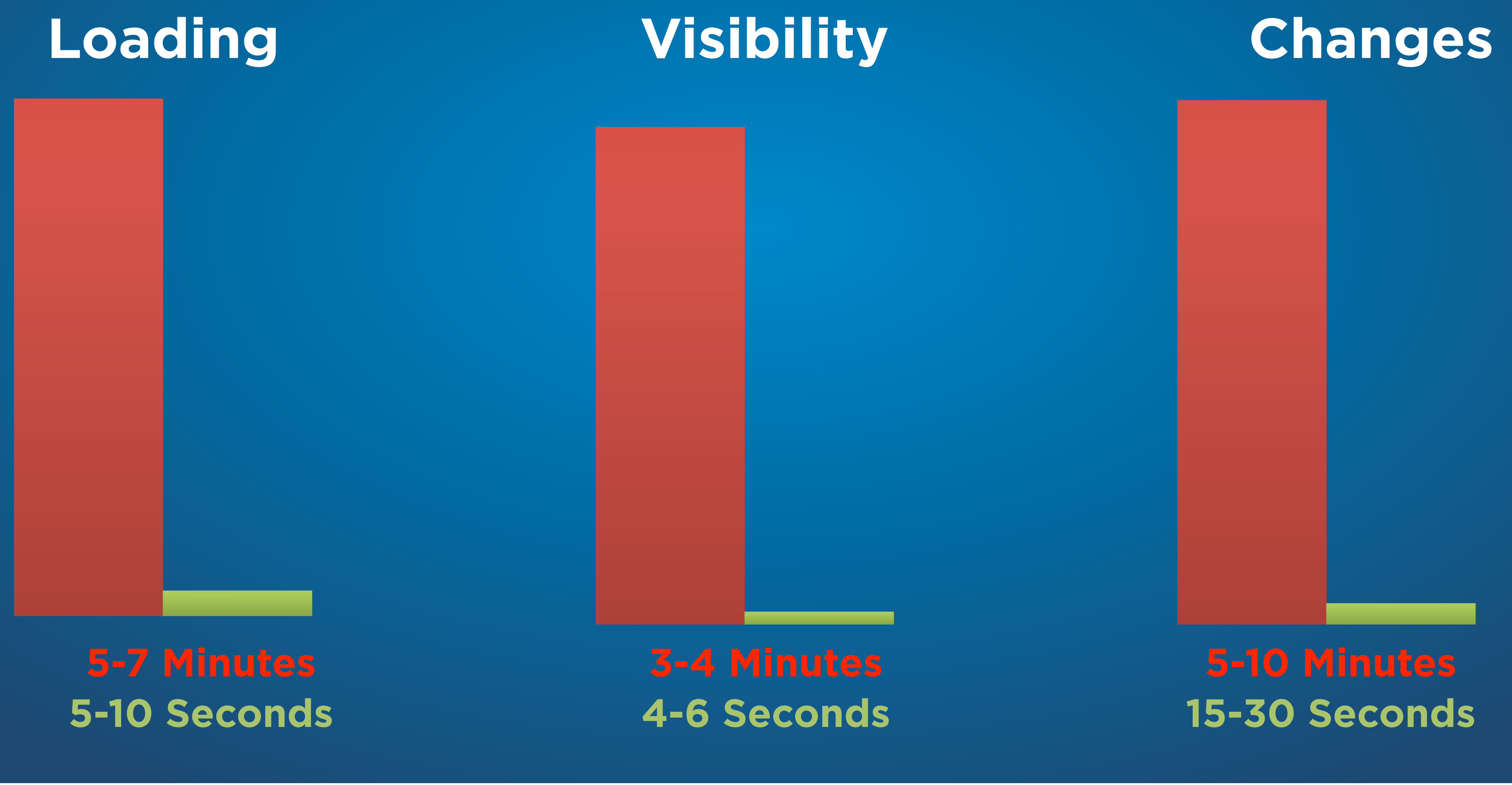
Minutes

- So on average, it took 5-7 minutes to load up a level
- If we then wanted to turn visibility of a section of a level on or off, that usually took about 3-4 minutes
- Then finally, if we wanted to make a change and then wanted to see it in the game, that took about 5-10 minutes

One thing to keep in mind was, the change may not have worked how you wanted, so yo would have to redo whole thing and wait another 5-10 minutes again.

The reason why was primarily due to how our old tools were structured. We used a packed build system. Everything that could possibly be loaded was packed into single giant Wad file. Game loads everything from single wad - no partial builds, so that it would always have to rebuild from scratch each time.

Downtime…No-More



We then changed our asset building system from Packed Build to more of a Loose Loading system. This allows each asset to be built individually and loaded into the game separately. Because of this, we have much less downtime compared to how things used to be.

Everything that used to take minutes before is now only taking seconds

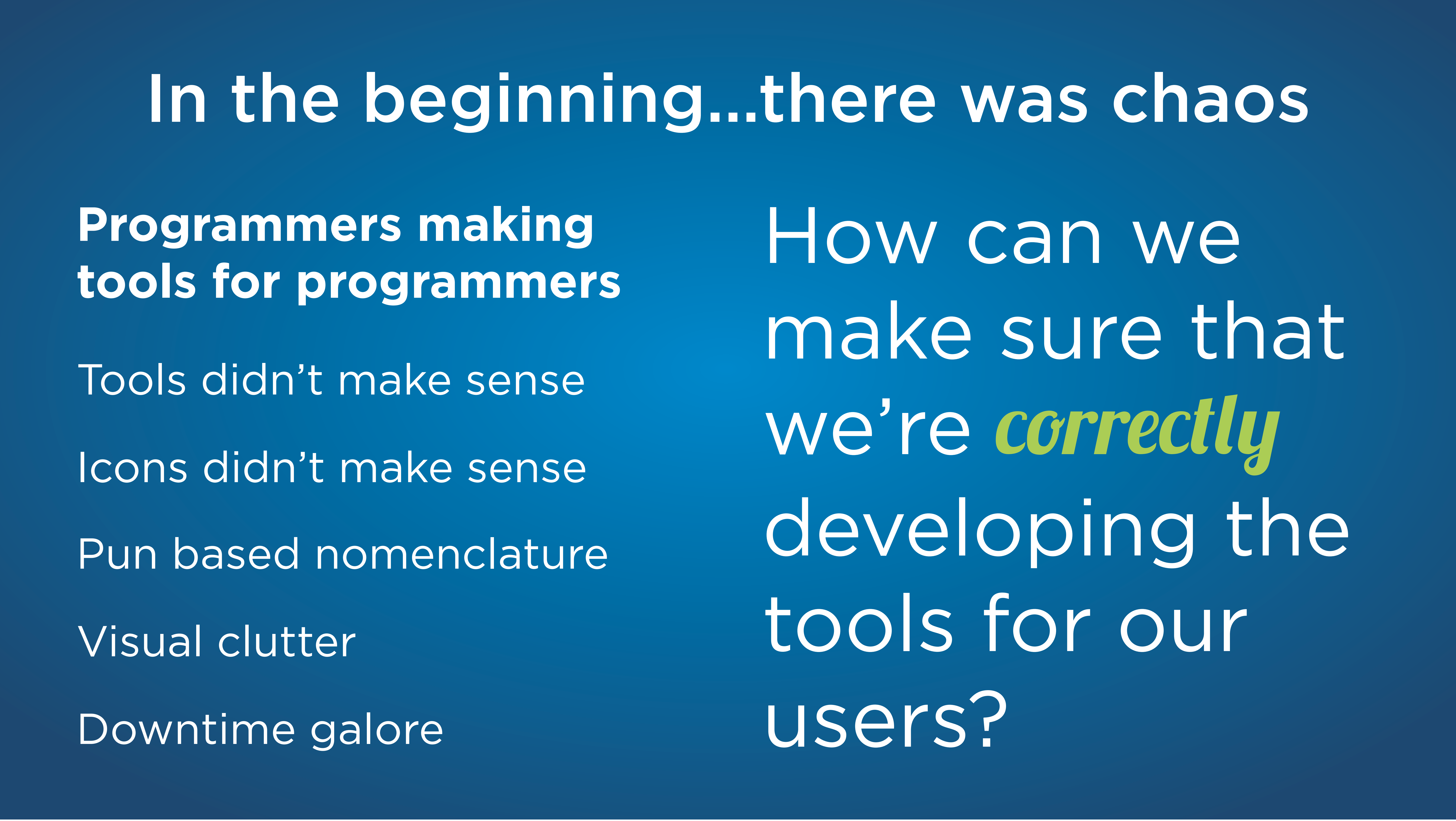
- Levels: 5-7 mins to 5-10 secs
- Vis on/off: 3-4 Mins to 4-6 secs
- Changes: 5-10 mins to 15-30 secs *if that*

Spyro the Dragon
Spyro 2 Ripto's Rage!
Spyro Year of the Dragon
Ratchet & Clank
Ratchet & Clank Going Commando
Ratchet & Clank Up Your Arsenal
Ratchet Deadlocked
Ratchet & Clank Future Tools of Destruction
Ratchet & Clank Future A Crack in Time
Ratchet & Clank Future Quest for Booty
Ratchet & Clank All 4 One
Resistance Fall of Man
Resistance 2
Resistance 3

The one thing that I want to stress is this: The tools weren’t always like this, it’s just how things were when I started at Insomniac. We didn’t decide to make property panels really cluttered right off the bat, it was due to a number of additions over the course of a number of games

- If you’re at all familiar with Insomniac, you know that we’ve made a number of very successful games with the old tools

The problem was that previously we didn’t have an artist on the team to design the UI, check consistency, etc
The tools were built for runtime speed/efficiency rather than being able to look pretty



In the beginning…there was chaos

Programmers making tools for programmers

Tools didn’t make sense

Icons didn’t make sense

Pun based nomenclature

Visual clutter

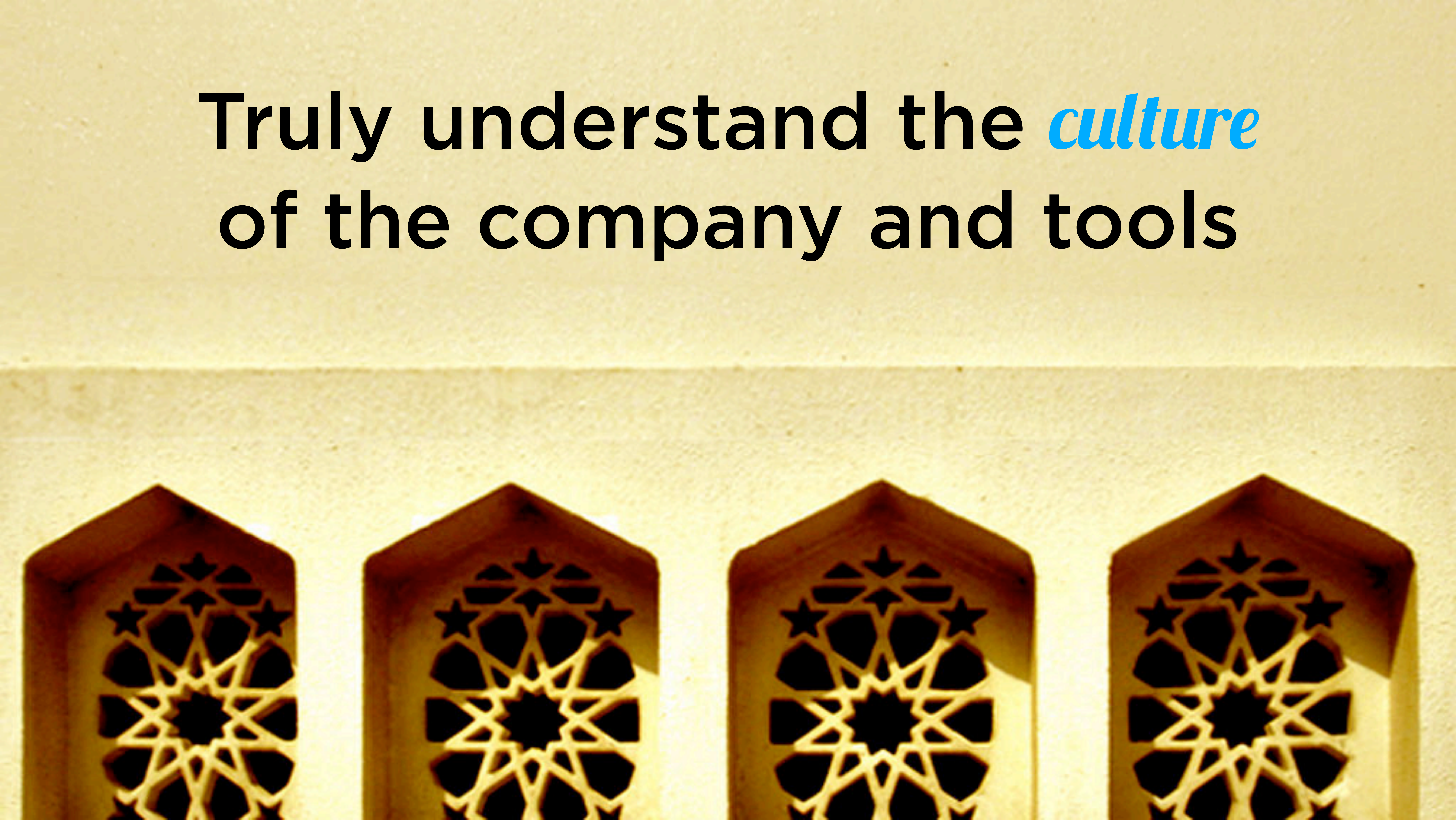
Downtime galore

How can we make sure that we’re *correctly* developing the tools for our users?

So with this in mind, I can’t fault tools programmers because of all of this

- We just need to keep this in mind in the future so that we can make the right decisions going forward **&** minimize any and all feature creep

Truly understand the *culture* of the company and tools



Have to truly understand the “culture” of the tools and how people work with them

Now what I mean by “culture” is this - Companies often have different ways of working, different workflows, and even different personalities. It’s important to understand how a company works, as well as the different departments work in order to get the best results.

You also need to be very cognizant of workflows and habits with different groups of developers. It’s all well and good to make something that **you** think is awesome (and it very well may be **really** awesome), but if it isn’t awesome for the people that are using the tools, it’s basically worthless.

Because of this you have to have a deep-seeded understanding of not only what was great with the old tools, but you also want to be aware of the primary hindrances to development and the things that could be improved upon

Everything is a usability problem

Easier

Better

More Efficient

Insomniac's has a number of usability and user experience philosophies, but we do have a core user experience philosophy:

- Everything, regardless of what it is, is a usability problem. It doesn't matter if it's an email, this presentation, a new level editor, every single thing that we do is a usability problem to be solved.

- We want to make sure that we design any new systems/tools to make both the lives and work of everyone easier, better, and more efficient

This all very much falls in line with my own usability goals and philosophies of streamlining the process with more efficient tools

More time for *more awesome*

What I want to do is have more time for more awesome. I want to take all of the time that you spend fiddling with the tools/details of game development and put it towards making awesome content

Jim Brown who's the Lead Level Designer over at Epic gave a presentation at the 2010 GDC, and one of the main points that he gave was this: Say you have a company of 200 people, and you do one thing that saves everyone **just** five minutes a day. Well when you add that up, you end up saving two months of time. As he mentions, what can your team do with two extra months that you didn't have before?

Make the tools so ***straightforward*** that ***anyone*** can use them

I really want to have the tools be so intuitive that anyone – regardless of their specialization, can go in and make something cool. I want a level designer to be able to *roughly* get their idea across with lighting/gameplay events, so that the lighting guy can come in and say “Oh, ok, I see what you were trying to get across here. Now let me go in and do my thing to make this look awesome”. I want to have your wife, friend, or even your kid come in, rough out a basic level with as little time and effort spent having to explain the tools to them ahead of time. This leaves room for the experts to do amazing work once they see what the designer or artist was trying to do

Get in and play, while leaving room for power users to do their thing

Really though, I want to allow you to just get in and play while still leaving room for the power users to do their thing

Let’s take cars for example. You don’t have to be a mechanic to drive from your home to work – if you did, no one would be driving around right now. For the most part, you just want to get in, put your keys in the ignition, and go. If you were more mechanically inclined and wanted to work on the car, then you should be able to do that too, without having it be a requirement.

I want to allow you to be able to just pick up and play with the tools while still leaving room for the experts and power users to apply their craft to making amazing content.

So how do we accomplish ALL of this?

With great *power* comes great *responsibility*

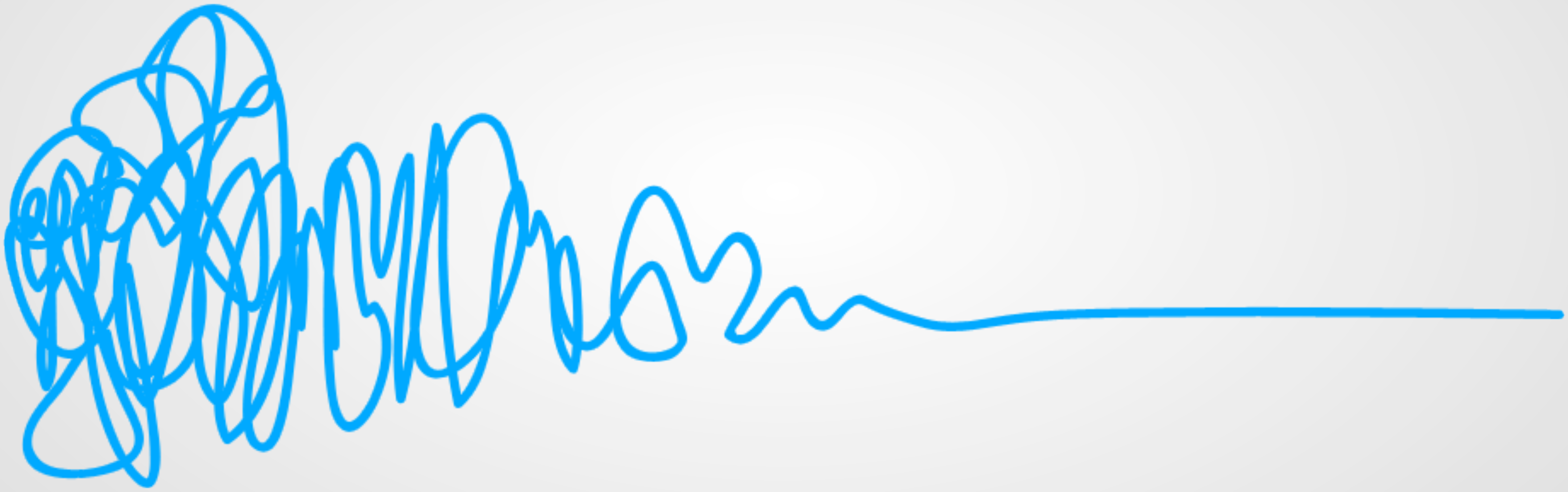
So one thing to keep in mind - with all the changes we're making/tools designing, must understand impact of what you change and also know why you're changing it

The reason why is because the changes that are made can impact:
- Games
- Jobs and Careers
- Even Lives

So when I first started working at Insomniac, I would walk around and see people doing all of these awesome things. I'd see people making crazy physics systems, camera systems, doing awesome animations…and I'd think "man, these guys are doing all of these awesome things and all I'm doing is making some icons and color palettes". But when I actually took a step back and thought about it…I realized that what I was doing affected everyone that worked on any of our games, and that's a pretty big responsibility.

So again, it's extremely important to remember and stay aware of these responsibilities

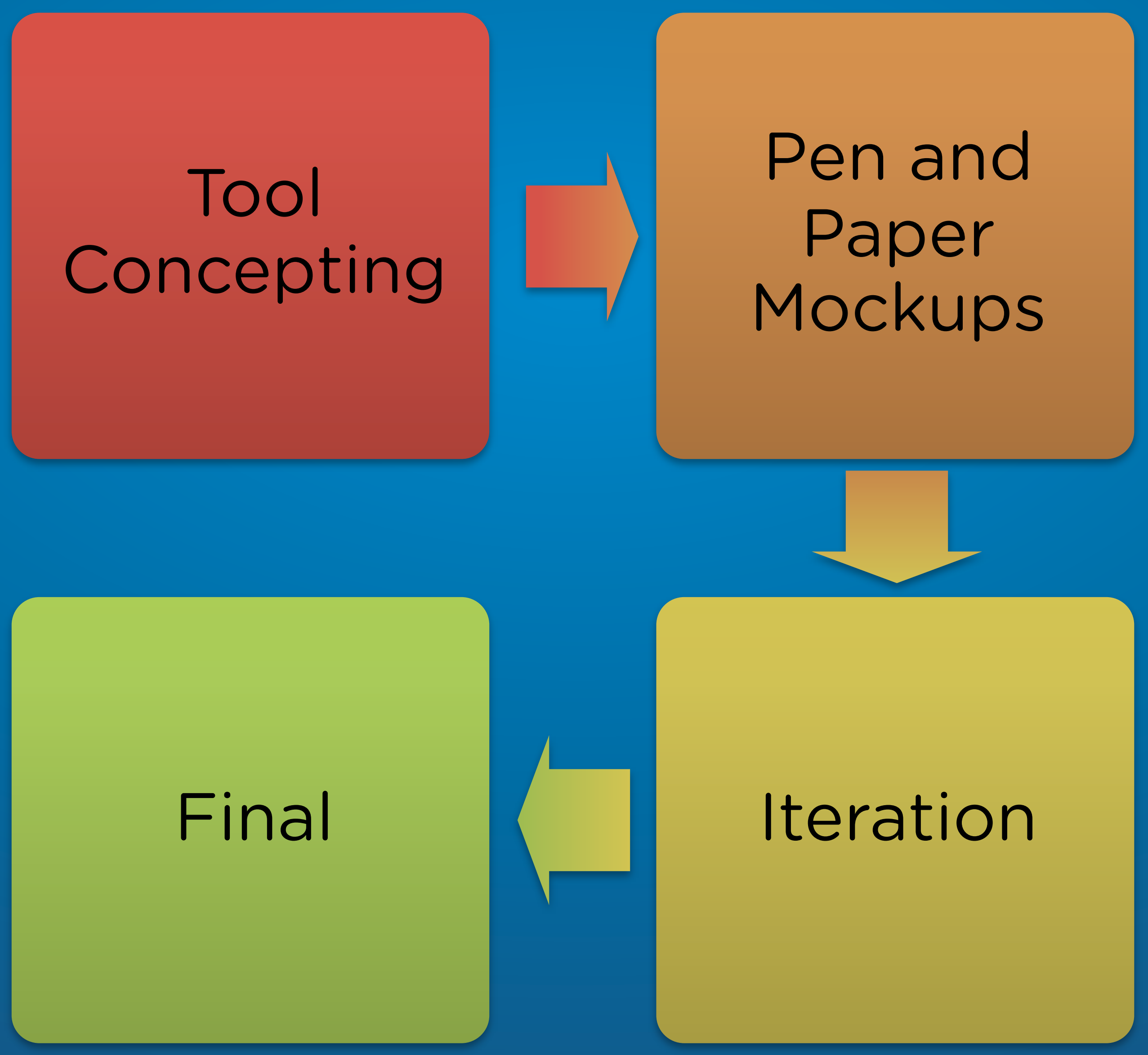
The UX Process



So the process that we have for designing our tools and pretty much the usability process as a whole looks a lot like this:

We start out with a whole lot of really broad ideas of things that could work or would be excellent additions to the tools. Then over the course of this process, we end up streamlining - getting rid of the excess and bad ideas, and eventually end up with a very focused and polished final product

Our Process



Tool Concepting

Pen and Paper Mockups

Iteration

Final

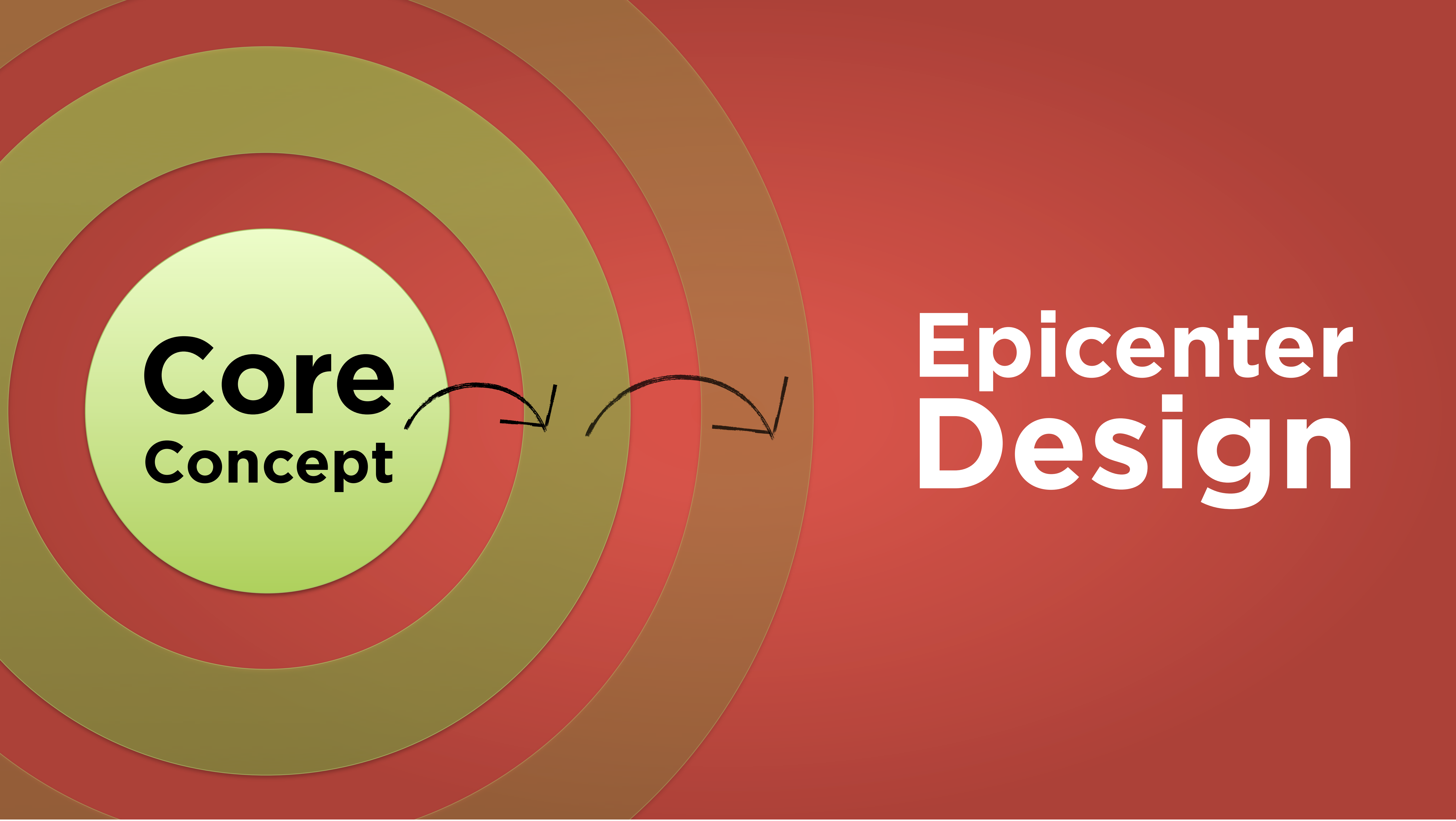
For our tools design, we use a four step process which is actually pretty similar to game design overall:

- You start with your concepting and pre-production phase
- We then move on to pen and paper mockups in order to get a good idea of layout, look, aesthetic, flow, etc
- Following this we have a TON of iteration - something I'm sure you all are **very** familiar with
- Once we've done all of our iterations and feel like we're in a great spot, we reach the "final" version

I say "final" with quotes because as most of you know - it may seem great, and once it's released into the wild we realize that we *weren't reallllllly done* like we thought we were

Concepting

First we start out with Concepting the tool



Epicenter Design

Core Concept

At Insomniac Games, we employ what’s called “Epicenter Design” - which is a term I picked up from the guys over at 37 Signals.

- What epicenter design really means is, you start from the core concepts and mechanics of a system

- So then once those are nailed down you build outward

This concept is really explained and solidified with a quote from John Gruber…

> “Figure out the ***absolute least*** you need to do to implement the idea, do just that, and then ***polish the hell out of the experience***”
>
> – ***John Gruber***

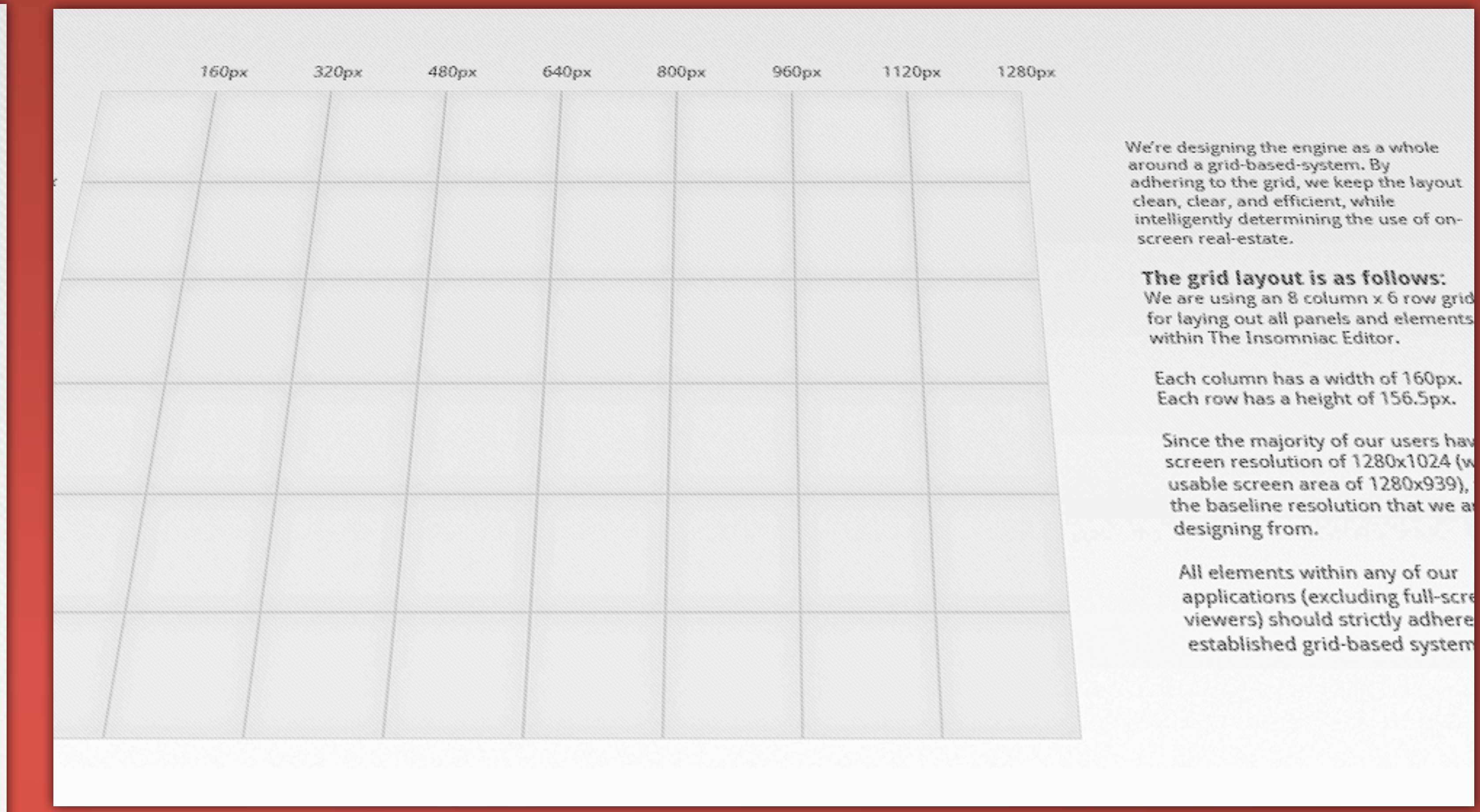
You really want to figure out what the **purpose** of the tool is so that you can limit the scope to only what *really* needs to be there.

Deliver a Singular and Integrated Experience

One of the main principles that we adhered to from the beginning is that we design and deliver a singular and integrated experience.

We want to make sure that the look, feel, and workflow for **every** part of the tools is as integrated and seamless as possible

Develop and *Adhere* to a Style-Guide



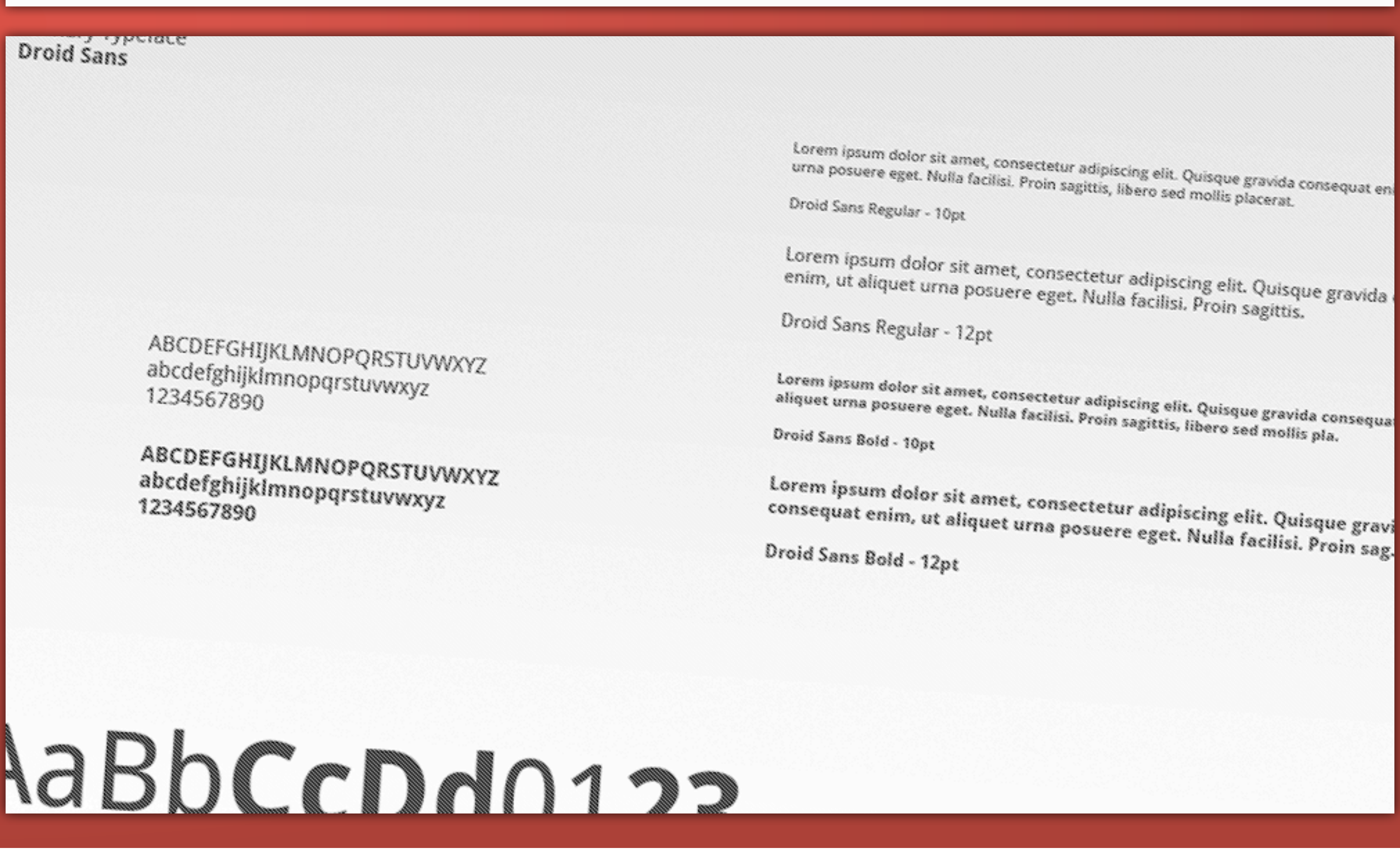
We're designing the engine as a whole around a grid-based-system. By adhering to the grid, we keep the layout clean, clear, and efficient, while intelligently determining the use of on-screen real-estate.

The grid layout is as follows:
We are using an 8 column x 6 row grid for laying out all panels and elements within The Insomniac Editor.

Each column has a width of 160px.
Each row has a height of 156.5px.

Since the majority of our users hav screen resolution of 1280x1024 (w usable screen area of 1280x939), the baseline resolution that we a designing from.

All elements within any of our applications (excluding full-scre viewers) should strictly adhere established grid-based system



Droid Sans

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque gravida consequat en urna posuere eget. Nulla facilisi. Proin sagittis, libero sed mollis placerat.

Droid Sans Regular - 10pt

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque gravida enim, ut aliquet urna posuere eget. Nulla facilisi. Proin sagittis.

Droid Sans Regular - 12pt

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque gravida consequa aliquet urna posuere eget. Nulla facilisi. Proin sagittis, libero sed mollis pla.

Droid Sans Bold - 10pt

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque gravi consequat enim, ut aliquet urna posuere eget. Nulla facilisi. Proin sag.

Droid Sans Bold - 12pt

In order to do this, we found that developing a style guide allowed us to develop conventions at the outset that dictated visual requirements, sizing and placement of elements, as well as develop expectant user workflows

By doing this we ensure a consistent look and feel throughout the entire tools

It also allows the user to learn expectant behavior and functionality when they are presented with consistent look, feel, and action with different portions of the tools

Design is a *conversation*, not a monologue

So the thing to keep in mind is that the design process is a conversation, not a monologue - It's not a one-man show.

You want to make sure that through this whole concepting phase, that you're involving all of the relevant parties in discussions, and not just placating them. You want to make sure that you're involving all of the necessary stakeholders - ie. those who will be making the most use of the tool, as well as the department leads

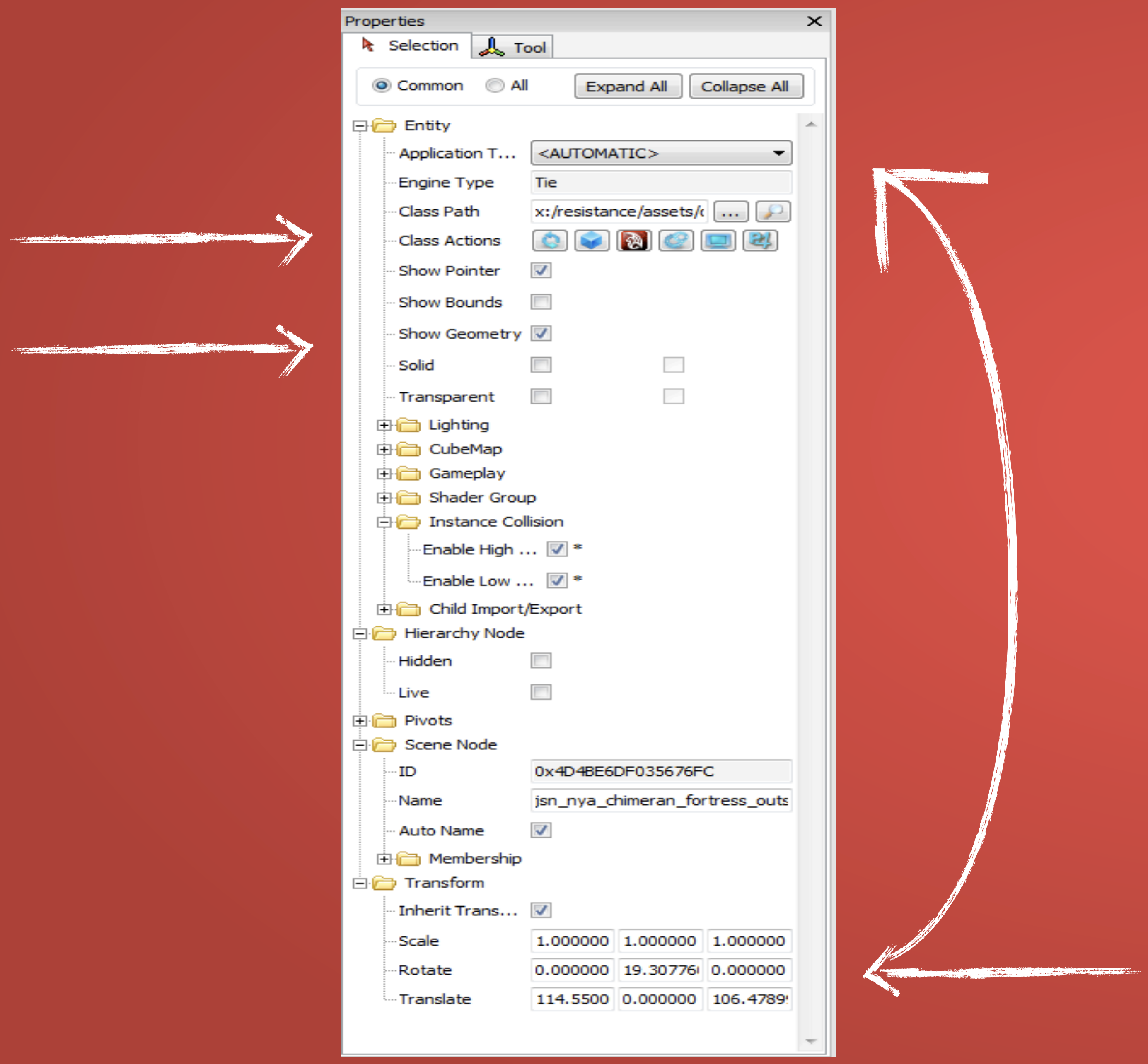
Ask the *Right* Questions

During these discussions, you need to make sure that you're asking the right questions:

What things were awesome in the old tools? What things didn't work as well as they could have in our old tools? How can we get this information and how can we go about fixing them as well?

What is the most important information that the user will need with any tool? How can we make sure that this is displayed prominently and is the most easily accessible?

Old Properties Panel



Properties

Selection | Tool

● Common ○ All

⊟ Entity
- Application T... <AUTOMATIC>
- Engine Type: Tie
- Class Path: x:/resistance/assets/(
- Class Actions
- Show Pointer ☑
- Show Bounds □
- Show Geometry ☑
- Solid □ □
- Transparent □ □
- ⊞ Lighting
- ⊞ CubeMap
- ⊞ Gameplay
- ⊞ Shader Group
- ⊟ Instance Collision
  - Enable High ... ☑ *
  - Enable Low ... ☑ *
- ⊞ Child Import/Export

⊟ Hierarchy Node
- Hidden □
- Live □

⊞ Pivots

⊟ Scene Node
- ID: 0x4D4BE6DF035676FC
- Name: jsn_nya_chimeran_fortress_outs
- Auto Name ☑
- ⊞ Membership

⊟ Transform
- Inherit Trans... ☑
- Scale: 1.000000 | 1.000000 | 1.000000
- Rotate: 0.000000 | 19.30776(| 0.000000
- Translate: 114.5500 | 0.000000 | 106.4789!

Transform information was used most often

Overwhelming the user with unnecessary information

So an example of how we approached this was our old Properties Panel:

•After ux testing and shadowing (more of that in a bit), we found that the most used info was the transform section

•So if this was used the most, why is it at the bottom? Why don't we have it at the top? Why are we making them scroll all the way to the bottom every time they select something new to get to this information?

•We also found that we were overwhelming our users with a ton of unnecessary information.

•These buttons at the top, why aren't they a right click?

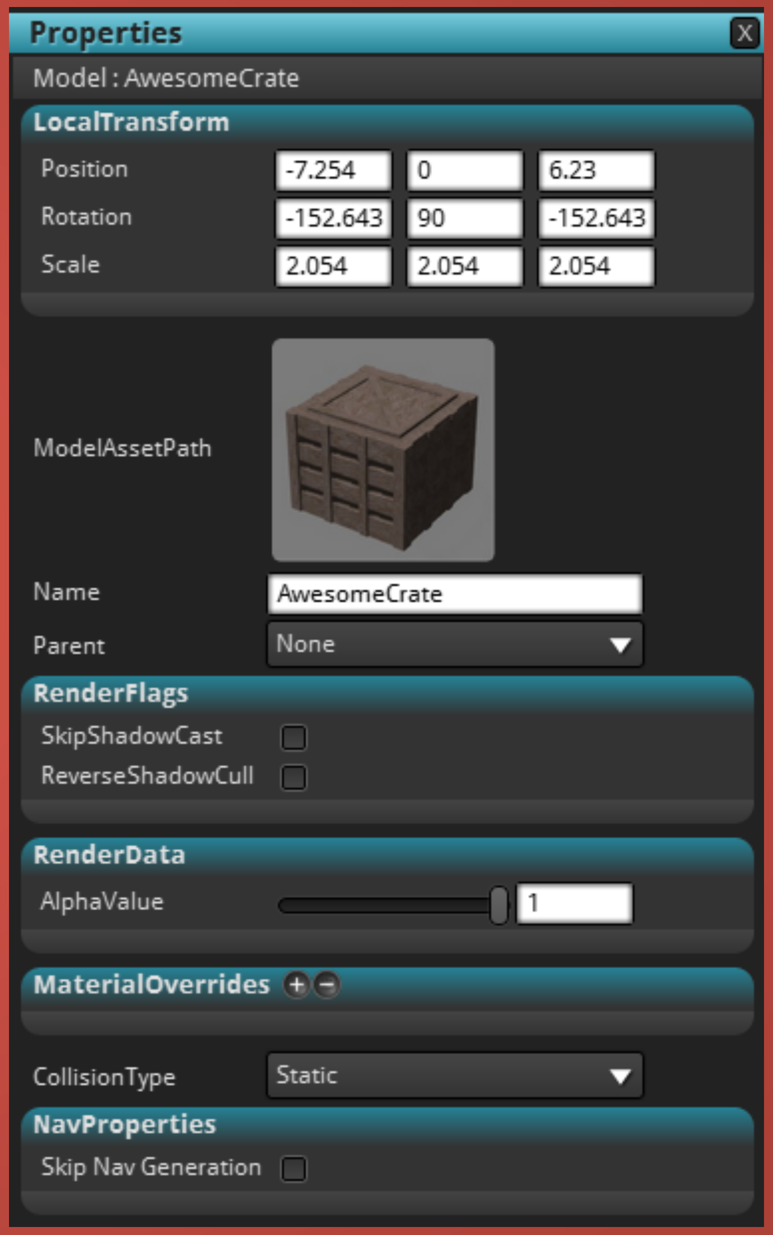
•The show geometry and bounds information, how come these aren't global settings?

Revised Properties Panel

Information prioritization

Only show what needs to be shown

Only display what the user can work with



Properties

Model : AwesomeCrate

LocalTransform

Position -7.254 0 6.23

Rotation -152.643 90 -152.643

Scale 2.054 2.054 2.054

ModelAssetPath

Name AwesomeCrate

Parent None

RenderFlags

SkipShadowCast □

ReverseShadowCull □

RenderData

AlphaValue 1

MaterialOverrides

CollisionType Static

NavProperties

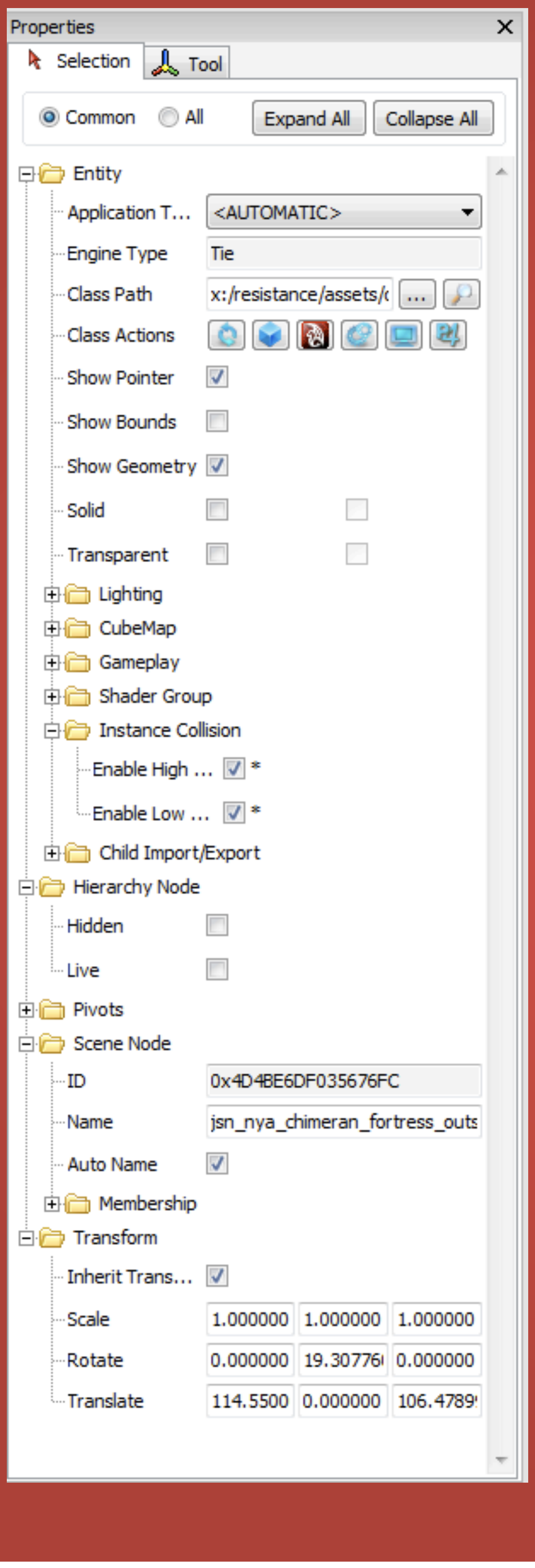
Skip Nav Generation □

By asking all these questions, we were able to fully revise the properties panel:

- All of the information was completely rearranged due to what got used the most often

- We're only showing what needs to be shown: what user can work with. A lot the extra things were put into right click menus or global preferences.

- We also made sure that all of the un-necessary information was hidden. The reason for this is when you're faced with too many decisions it ends up being paralyzing. You can't really make a "right" decision because there are too many options to choose from without a clear "right" answer.

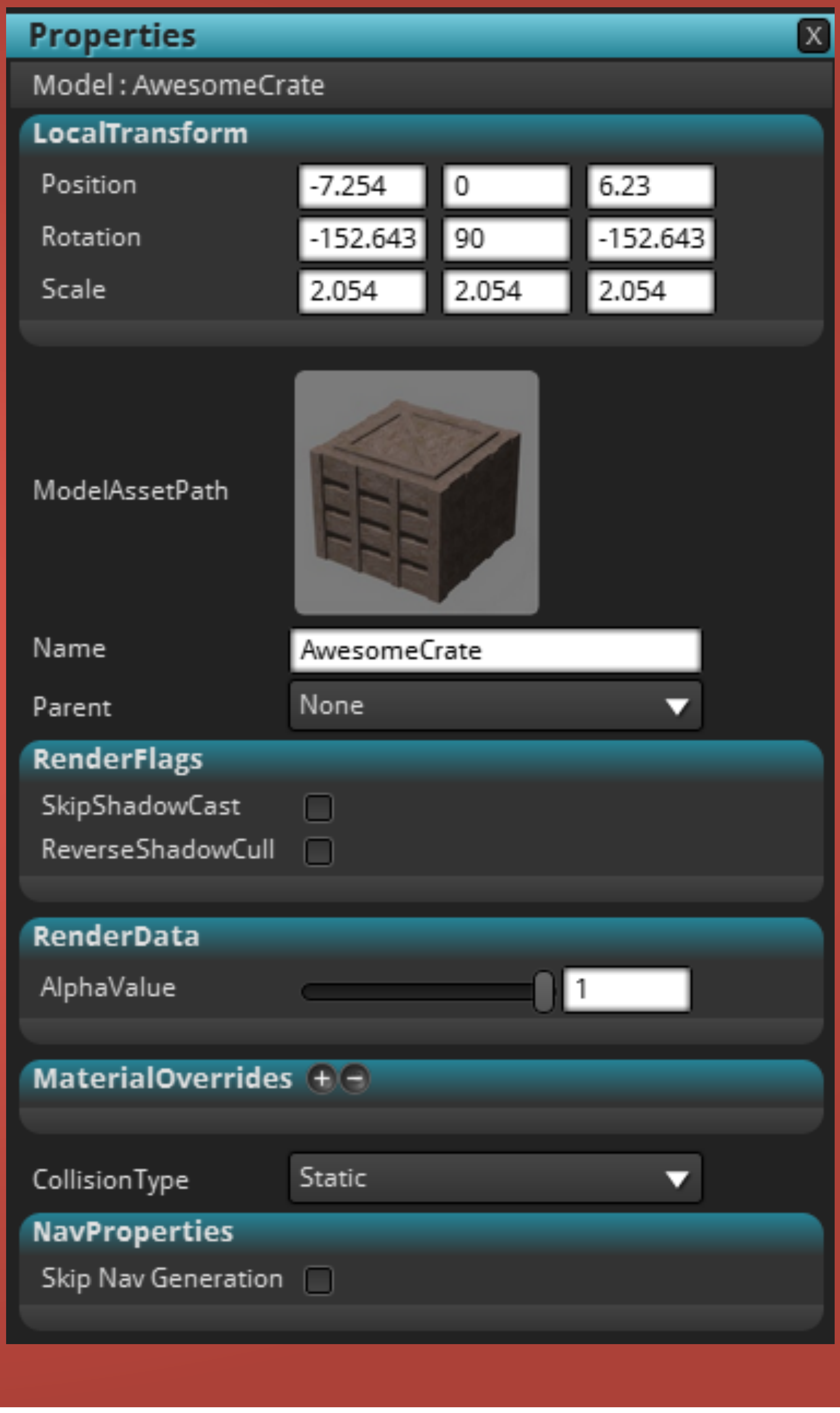


Properties

Selection | Tool

● Common ○ All [Expand All] [Collapse All]

- Entity
  - Application T… <AUTOMATIC>
  - Engine Type: Tie
  - Class Path: x:/resistance/assets/
  - Class Actions
  - Show Pointer ☑
  - Show Bounds □
  - Show Geometry ☑
  - Solid □ □
  - Transparent □ □
  - Lighting
  - CubeMap
  - Gameplay
  - Shader Group
  - Instance Collision
    - Enable High … ☑ *
    - Enable Low … ☑ *
  - Child Import/Export
- Hierarchy Node
  - Hidden □
  - Live □
- Pivots
- Scene Node
  - ID: 0x4D4BE6DF035676FC
  - Name: jsn_nya_chimeran_fortress_outs
  - Auto Name ☑
  - Membership
- Transform
  - Inherit Trans… ☑
  - Scale: 1.000000 1.000000 1.000000
  - Rotate: 0.000000 19.30776 0.000000
  - Translate: 114.5500 0.000000 106.4789



Properties

Model : AwesomeCrate

LocalTransform

Position	-7.254	0	6.23
Rotation	-152.643	90	-152.643
Scale	2.054	2.054	2.054

ModelAssetPath

Name: AwesomeCrate

Parent: None

RenderFlags

SkipShadowCast □

ReverseShadowCull □

RenderData

AlphaValue: 1

MaterialOverrides

CollisionType: Static

NavProperties

Skip Nav Generation □

Look at not only our old tools, but at what everyone else in the industry is using

So we’re not only looking at our old tools, we also want to look at what everyone else in the industry is using as well.

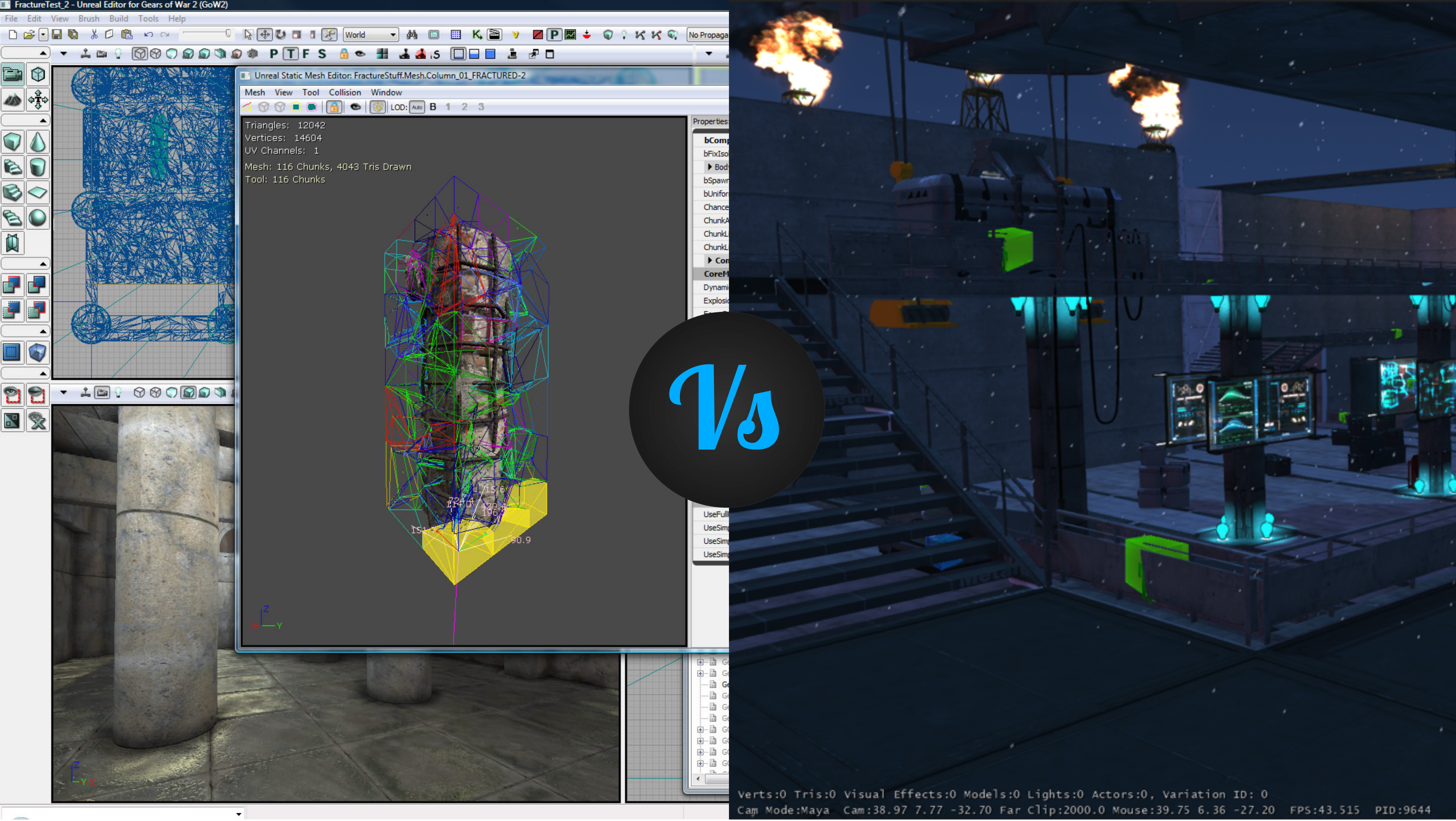
What improvements, choices, and user experience decisions can we find in these programs? Additionally, how can we improve upon these to make our tools a clear and obvious choice for our users?

It’s incredibly important to not just copy these choices, we also need to know why the decisions were made in the first place

So I play a game called StarCraft 2, now, show of hands - how many of you also play StarCraft 2? Ok well, for those that haven't played it, it's really awesome - full of strategy, everything you could want in a good game. Now I don't know about you, but personally if I want to get better, what do I do?

•Well, I watch the pros play. Now, if I watch someone like Nestea or JulyZerg play, copy their strategy, and lose…most likely I'm going to just think "well I lost because I'm not as good as them". Really what it **probably** was, was that I was just copying their strategy without knowing **why** they did what they did, or why they made the decisions that they did.

See, by knowing the why, you can then make better educated decisions in the future.


FractureTest_2 - Unreal Editor for Gears of War 2 (GoW2)
File Edit View Brush Build Tools Help
World
No Propaga
P T F S
Unreal Static Mesh Editor: FractureStuff.Mesh.Column_01_FRACTURED-2
Mesh View Tool Collision Window
LOD: Auto B 1 2 3
Properties
Vs

```
Triangles:  12042
Vertices:  14604
UV Channels:  1
Mesh: 116 Chunks, 4043 Tris Drawn
Tool: 116 Chunks
```

```
Verts:0 Tris:0 Visual Effects:0 Models:0 Lights:0 Actors:0, Variation ID: 0
Cam Mode:Maya  Cam:38.97 7.77 -32.70 Far Clip:2000.0 Mouse:39.75 6.36 -27.20  FPS:43.515  PID:9644
```

So, how do our tools stack up with what else is out there?

We check this through efficiency testing. We’ll do a set of tasks on one set of tools (say UDK), then replicate the same task on our tools. By doing this we’re able to see how long it takes across the board. If it takes longer for our tools, how can we improve this to be the fastest and most efficient possible (even more-so than what else is out there)?

Also looking at non VG software

So we’re also looking at a ton of non-video game software and engines

•Photoshop, After Effects. We’re even looking at music software like traktor, ableton, logic, etc

So how many of you out there - show of hands - own an Apple device?
•Alright, so you all probably know that apple makes beautifully designed and easy to use hardware and software

We’re making sure that we’re checking the workflow, user experience decisions, UI, overall design, visuals, etc

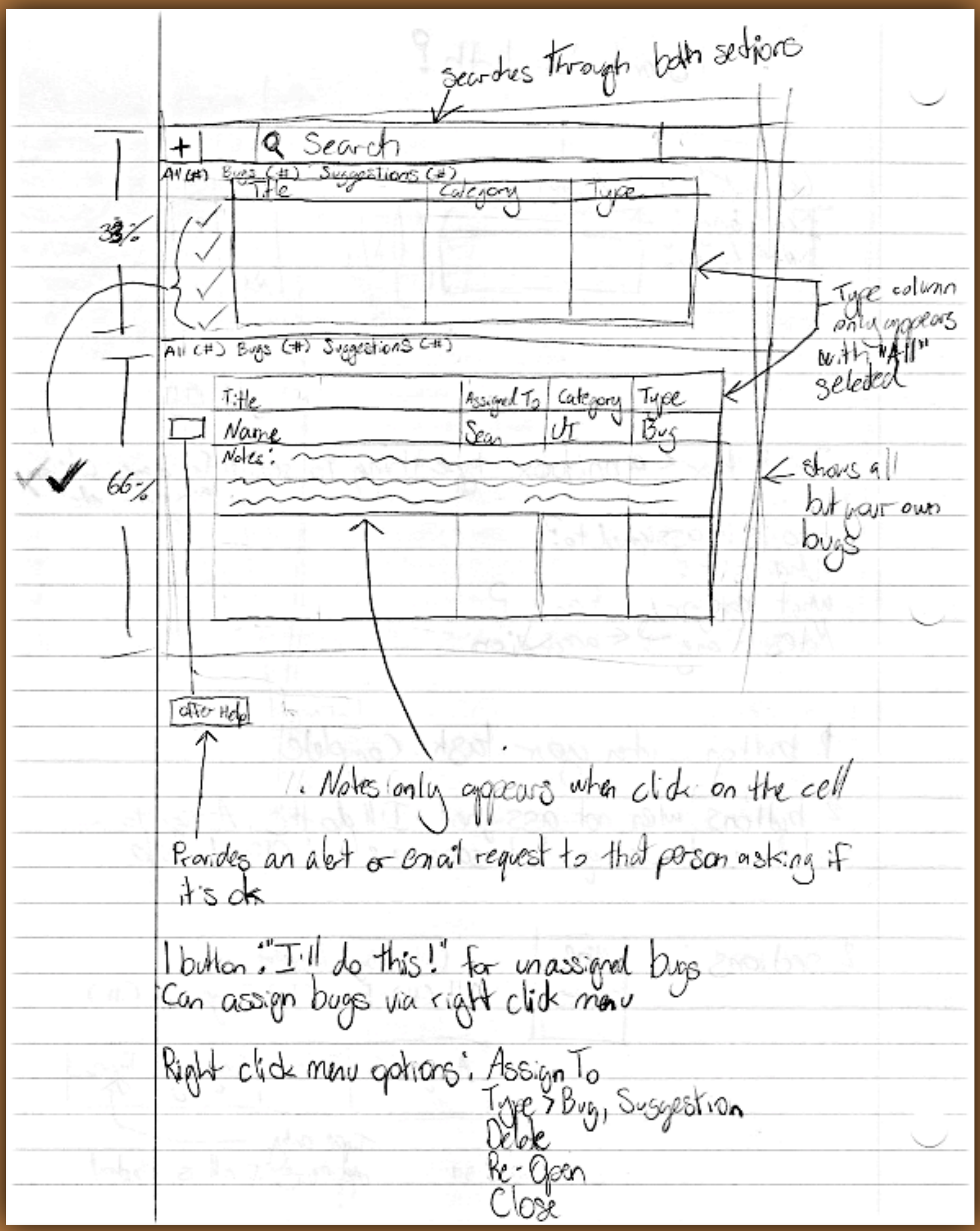
Mockups

Once we finish all of the pre-production and concepting of the tool, we get to work on some mockups

“Pen and paper” mockups

Fastest Iteration

Minimal investment


searches through both sections
+
Search
All (#) Bugs (#) Suggestions (#)
Title
Category
Type
33%
Type column only appears with "All" selected
All (#) Bugs (#) Suggestions (#)
Title
Assigned To
Category
Type
Name
Sean
UI
Bug
Notes:
66%
Shows all but your own bugs
Offer Help

Notes only appears when click on the cell

Provides an alert or email request to that person asking if it's ok

1 button: "I'll do this!" for unassigned bugs
Can assign bugs via right click menu

Right click menu options: Assign To
Type > Bug, Suggestion
Delete
Re-Open
Close

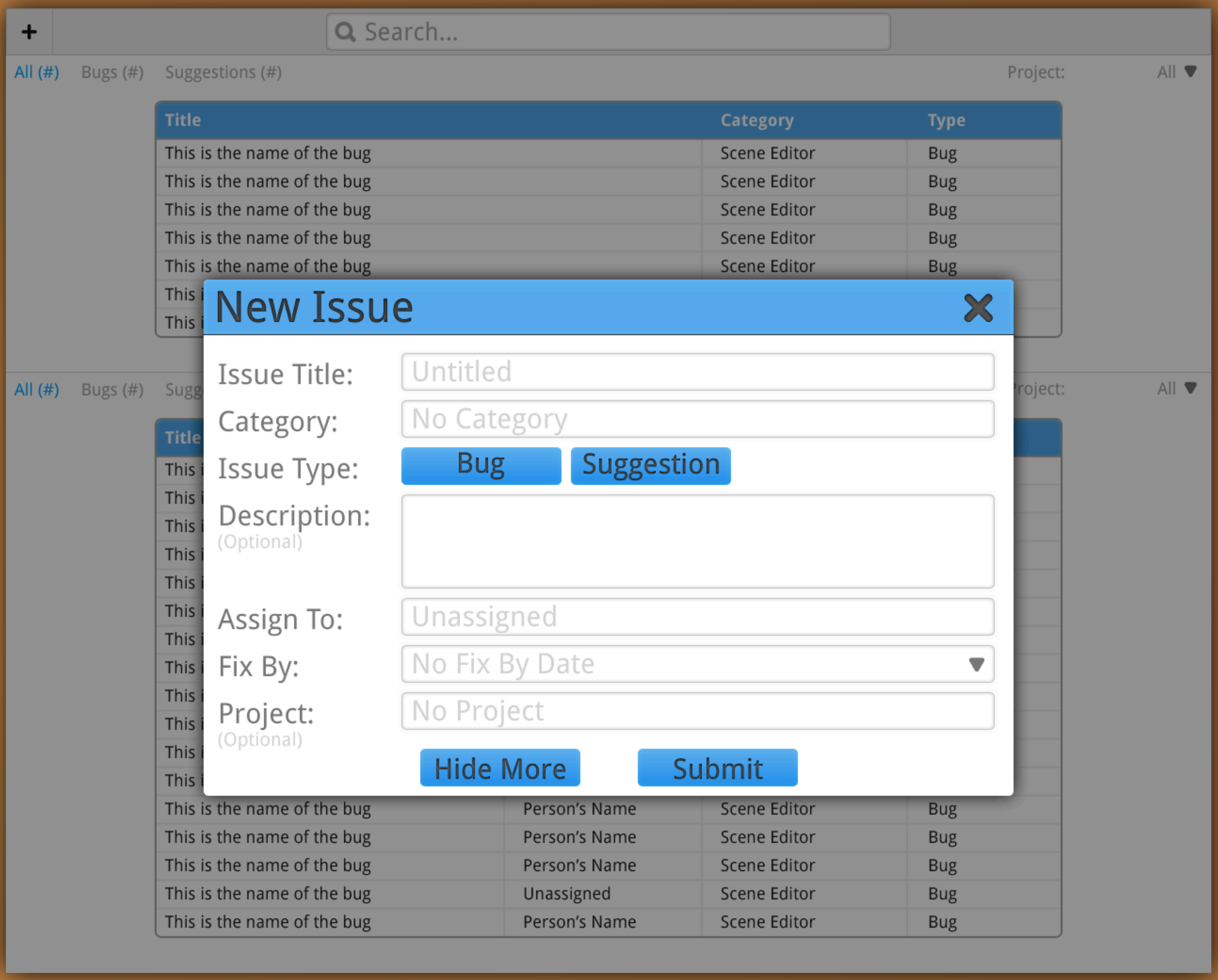
- So what we do is start with pen and paper mockups

- We found that this was the fastest way to iterate the look, the layout, any workflows, etc

- The thing is that you want the absolute minimal investment at this point - easy to get rid of if not appropriate. At this point you’re making so many changes that it really doesn’t make sense to invest a ton of time and energy yet

Photoshop > Code

Tools developed much faster



Title	Category	Type
This is the name of the bug	Scene Editor	Bug
This is the name of the bug	Scene Editor	Bug
This is the name of the bug	Scene Editor	Bug
This is the name of the bug	Scene Editor	Bug
This is the name of the bug	Scene Editor	Bug
This		
This		

New Issue

Issue Title: Untitled

Category: No Category

Issue Type: Bug Suggestion

Description: (Optional)

Assign To: Unassigned

Fix By: No Fix By Date ▼

Project: (Optional) No Project

Title			
This			
This			
This			
This			
This			
This			
This			
This			
This			
This			
This			
This			
This is the name of the bug	Person's Name	Scene Editor	Bug
This is the name of the bug	Person's Name	Scene Editor	Bug
This is the name of the bug	Person's Name	Scene Editor	Bug
This is the name of the bug	Unassigned	Scene Editor	Bug
This is the name of the bug	Person's Name	Scene Editor	Bug

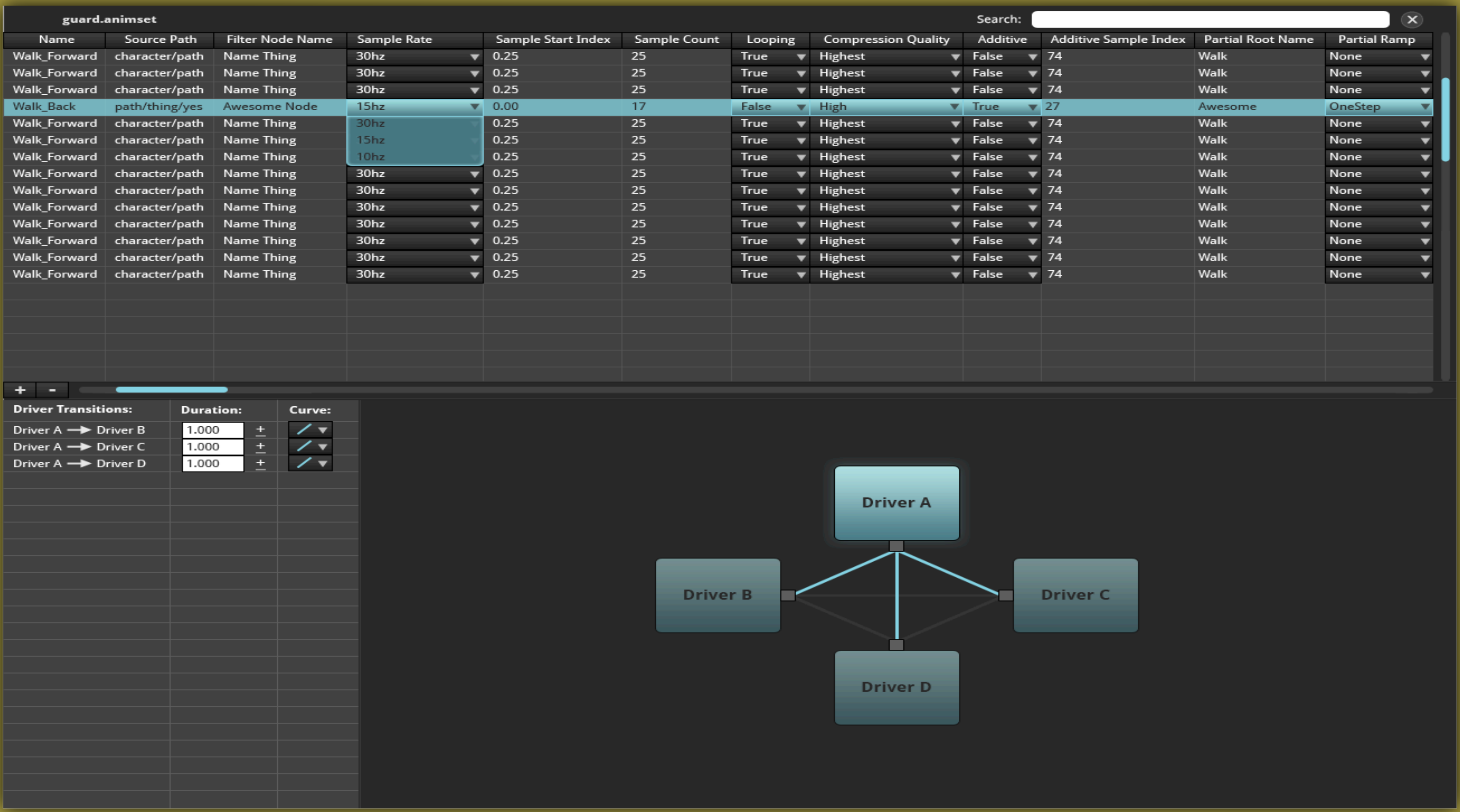
Once pen and paper wireframes ok'd, they're brought into Photoshop or some other wireframe/ prototyping software in order to give everyone a real concrete idea of what the tool will end up looking like.

- Absolutely no code written until the mockups are ok'd by all relevant parties.
Until the ok is given, not a single line of code is made. We've found that it's **much** faster to change a photoshop mockup than it is to rework code

- Because of this, we've found that the tools have been developed much faster

Iteration

Ok, so now that the original mockups have been given the green light, let’s start iterating



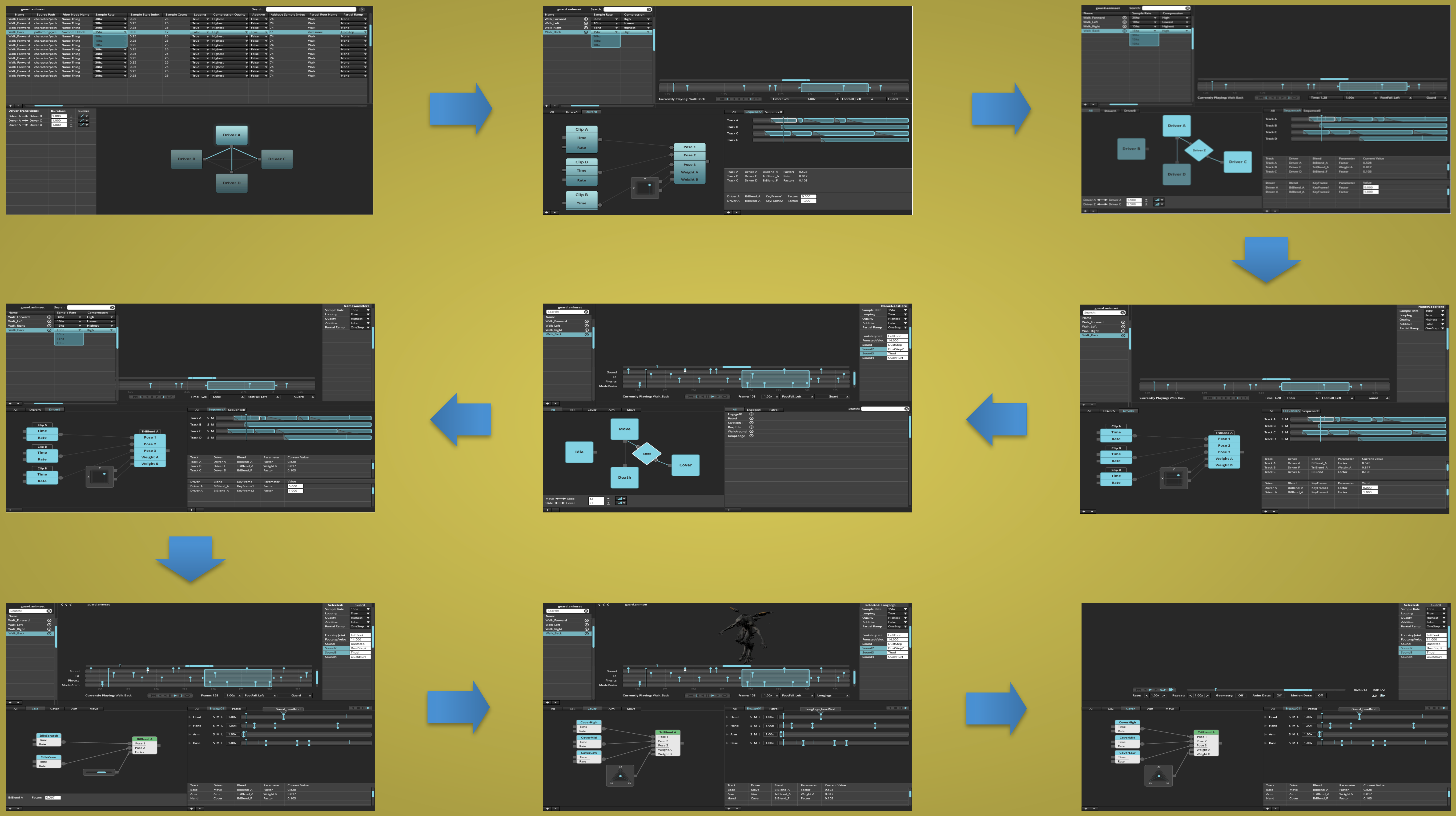
guard.animset

Search:

Name	Source Path	Filter Node Name	Sample Rate	Sample Start Index	Sample Count	Looping	Compression Quality	Additive	Additive Sample Index	Partial Root Name	Partial Ramp
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Back	path/thing/yes	Awesome Node	15hz	0.00	17	False	High	True	27	Awesome	OneStep
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	15hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	10hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None
Walk_Forward	character/path	Name Thing	30hz	0.25	25	True	Highest	False	74	Walk	None

Driver Transitions:	Duration:		Curve:
Driver A → Driver B	1.000	±	╱
Driver A → Driver C	1.000	±	╱
Driver A → Driver D	1.000	±	╱

So this is the first photoshop mockup that we did of our Animset Editor



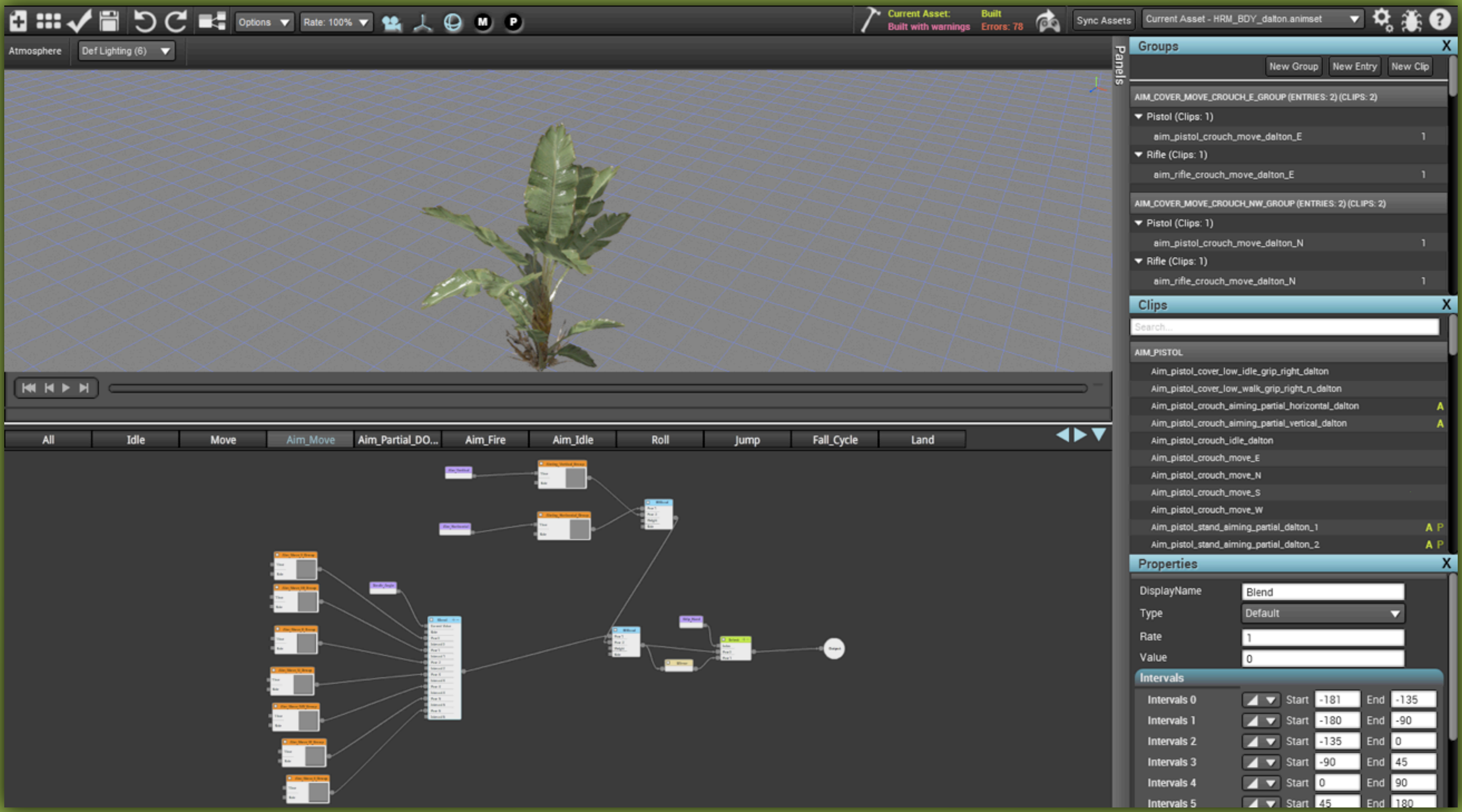
And then here are all of our numerous iterations of the Animset Editor

Again, we’re still doing all of the iterations in Photoshop, because again, it’s faster in Photoshop than it is in Code

“Final”

After all fixes and revisions are done in photoshop, we reach the “final” - live version

- Now I use the airquotes because by final, I really mean that it’s the first version that are actually in use in production



AIM_COVER_MOVE_CROUCH_E_GROUP (ENTRIES: 2) (CLIPS: 2)

- Pistol (Clips: 1)
  - aim_pistol_crouch_move_dalton_E 1
- Rifle (Clips: 1)
  - aim_rifle_crouch_move_dalton_E 1

AIM_COVER_MOVE_CROUCH_NW_GROUP (ENTRIES: 2) (CLIPS: 2)

- Pistol (Clips: 1)
  - aim_pistol_crouch_move_dalton_N 1
- Rifle (Clips: 1)
  - aim_rifle_crouch_move_dalton_N 1

AIM_PISTOL

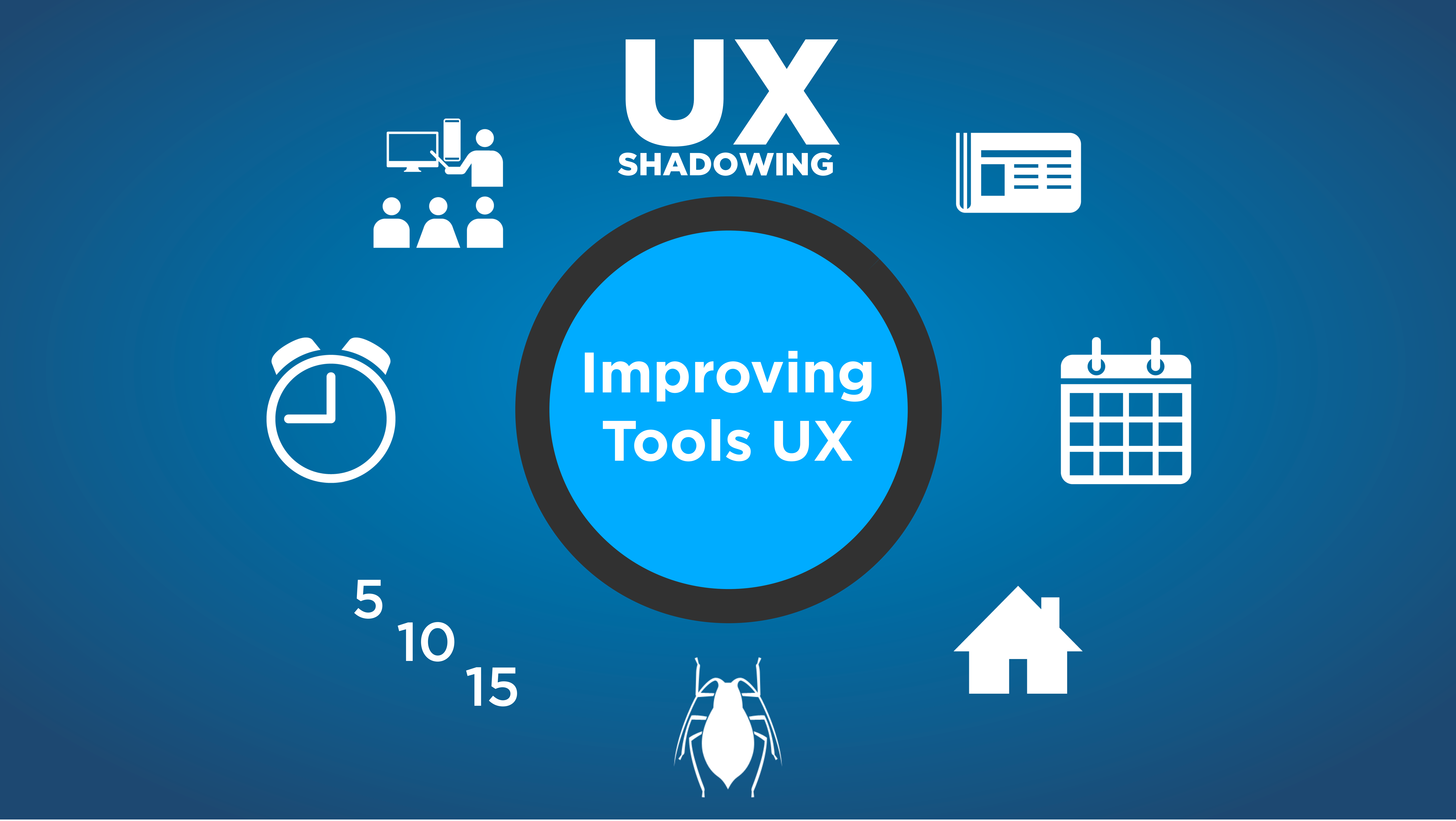
- Aim_pistol_cover_low_idle_grip_right_dalton
- Aim_pistol_cover_low_walk_grip_right_n_dalton
- Aim_pistol_crouch_aiming_partial_horizontal_dalton A
- Aim_pistol_crouch_aiming_partial_vertical_dalton A
- Aim_pistol_crouch_idle_dalton
- Aim_pistol_crouch_move_E
- Aim_pistol_crouch_move_N
- Aim_pistol_crouch_move_S
- Aim_pistol_crouch_move_W
- Aim_pistol_stand_aiming_partial_dalton_1 A P
- Aim_pistol_stand_aiming_partial_dalton_2 A P

DisplayName: Blend
Type: Default
Rate: 1
Value: 0

Intervals

Intervals 0	Start	-181	End	-135
Intervals 1	Start	-180	End	-90
Intervals 2	Start	-135	End	0
Intervals 3	Start	-90	End	45
Intervals 4	Start	0	End	90
Intervals 5	Start	45	End	180

This is the first, live version in our tools



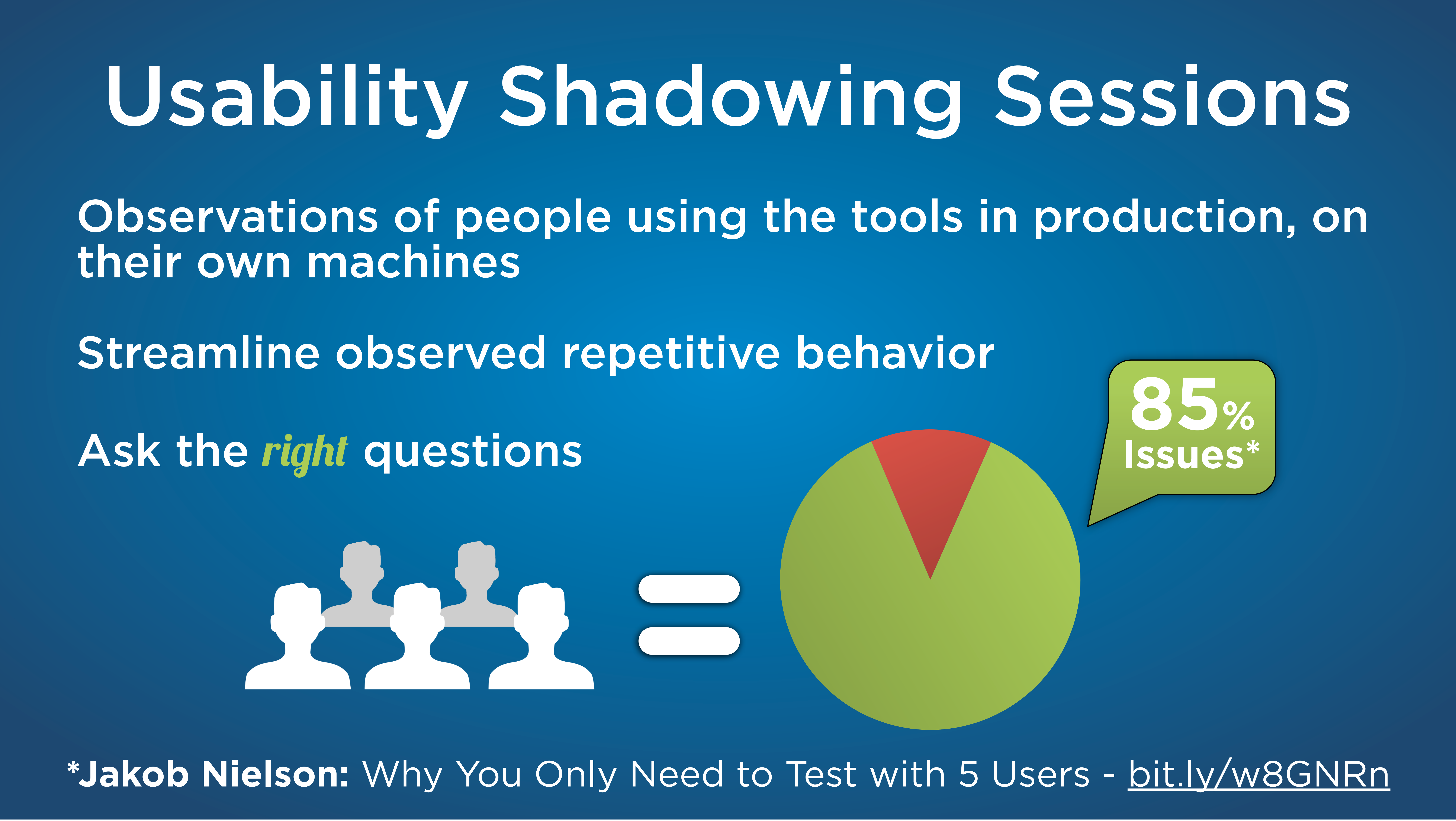
UX SHADOWING

Improving Tools UX

5 10 15

So once the tools are out and in the wild, how do we keep improving the user experience? Well we do this with about eight different things:

- Usability Shadowing Sessions
- Impromptu usability tests
- Our usability week
- What we call the Core Team Open House
- Our Bugs and Suggestions Form
- Top Critical Issues Lists from leads
- Top Time Wasters
- Tools education



Usability Shadowing Sessions

Observations of people using the tools in production, on their own machines

Streamline observed repetitive behavior

Ask the *right* questions

*Jakob Nielson: Why You Only Need to Test with 5 Users - bit.ly/w8GNRn

Seeing the tools in use makes a world of difference. While we *THINK* we know the best things for people, the tools, and the needs of our users, most of us aren't level designers, animators, designers, etc. A lot of the time there are key bits of information - whether it's a workflow issue, or behaviors or habits, we don't always get it right. That's where usability shadowing comes into play.

•One of the more important things that we do is Observing people using the tools in production, on their own machines This allows the individuals to feel more familiar with their surroundings and will work more closely to how they normally do when compared to working in a "test" setting

•We can make sure that we're able to streamline a lot of the repetitive behavior that we notice throughout our shadowing session, and again…

•We want to make sure that we're asking the RIGHT questions while we're observing everyone working. This way we can discover the core issues and short-comings of the tools at hand

•The thing about all of this is that it can be done REALLY cheaply - without the need for complex testing equipment, eye tracking software, etc. By shadowing even five people, you can usually obtain roughly 85% of all problem issues. This isn't just from me, this is from Jakob Nielson, widely considered as THE usability expert.

“The Miyamoto Kidnappings”

So then we move on to our impromptu usability tests.

If any of you were at GDC 2009’s Keynote presentation by Iwata, you would have heard about what he called “The Miyamoto Kidnappings”

Whenever Miyamoto was working on a new game or idea or project, he would randomly “kidnap” a Nintendo employee that had nothing to do with production, put them in a room with developers, handed a controller, and the only instructions that they were given was to play. See Miyamoto had the belief that since you weren’t going to have the developers in the room with you while you played, that it only made sense to watch and observe the players. He wanted to see how they succeeded, failed, where they were happy and sad, and fix as many of the issues as possible before the game was released.

We do something very similar at Insomniac, except we actually ask questions while we’re observing so that we can really **know** the issue rather than guessing at the cause.

“Usability Week”

Was one day, now a week

2012

🟩 J	F	M
A	🟩 M	J
🟩 J	A	🟩 S
O	N	🟩 D

One Dept Per Day

The next thing that we have is what we call our Usability week

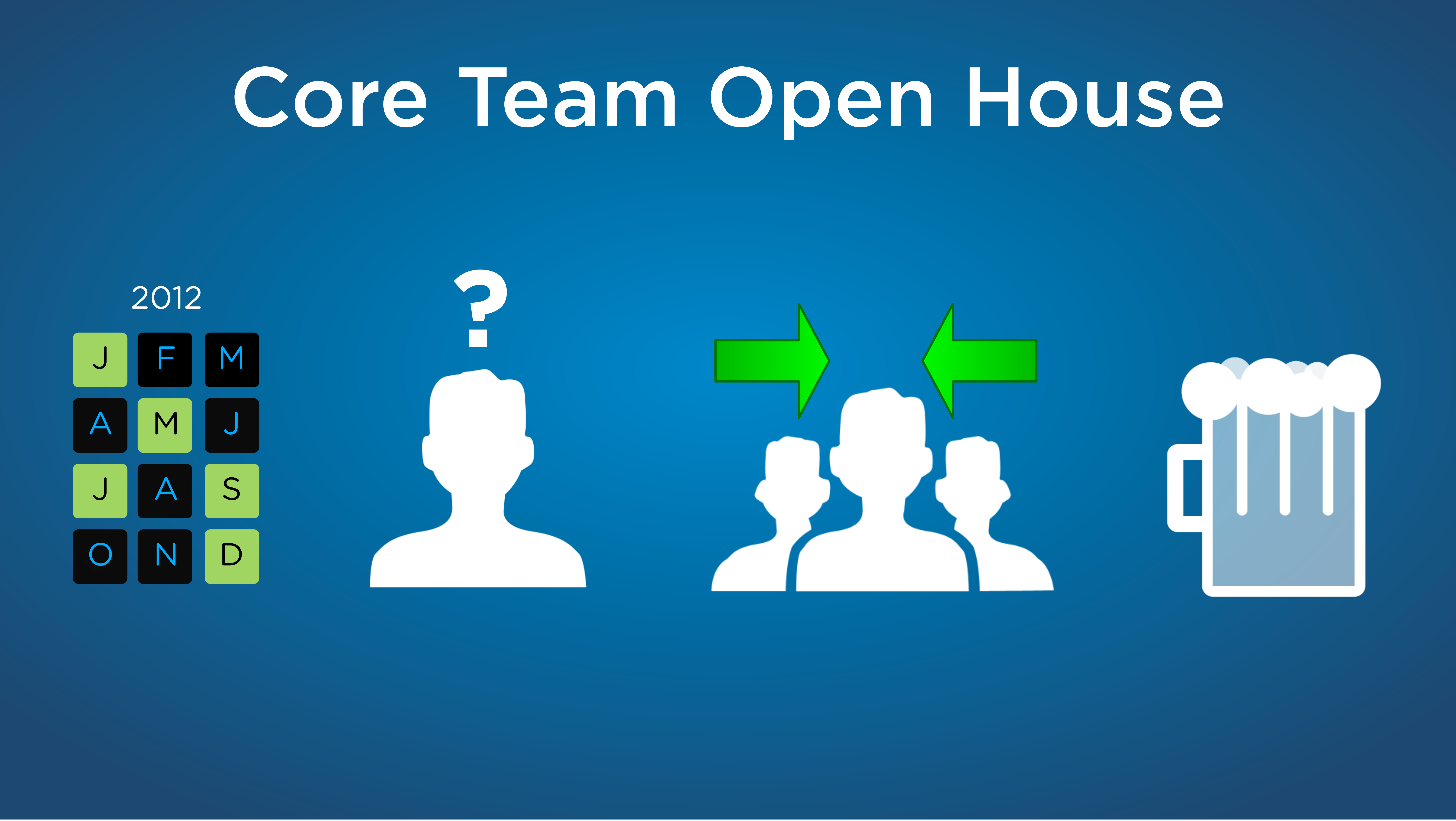
- We used to have just a usability day, however we found that we weren’t getting nearly enough useful and actionable information, so we extended it to a week

- This week occurs roughly every two to three months.

- Bring various people from different departments in to use the tools – each dept gets one day of UX testing

- We’re also able to get actionable feedback from multiple people from different departments, which allows us to make sure that we’re not basing all decisions on what could be a biased opinion of one or two people. We also see any trends in workflow habits or behavior that are happening that we may not have noticed before.

It’s tremendously important to make sure that we receive feedback from multiple people to get the 90% case



Core Team Open House

2012

J	F	M
A	M	J
J	A	S
O	N	D

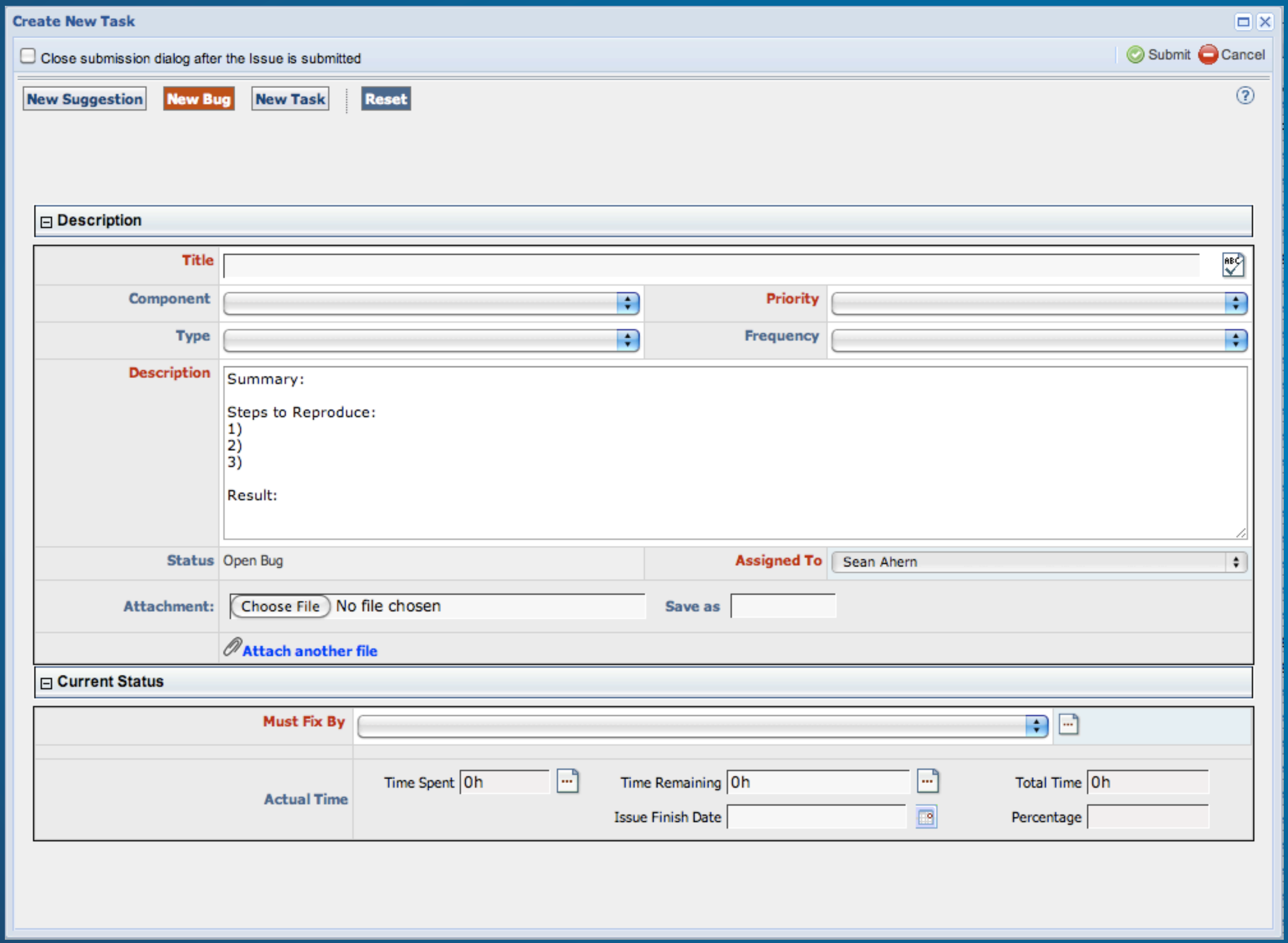
We also have what is called our Core Team Open House

- This is open to the entire company every two to three months or so.

- We think that this is an excellent chance to not only use the tools in various stages of development, but people can ask questions as well to those who are actually developing the tools.

- We've found that it has **greatly** helped to close gap between programmers and all of the other departments

- The most important part of all though, we've got free beer + drinks! Who doesn't love free beer and drinks right?



☐ Close submission dialog after the Issue is submitted

⊟ Description

Title:

Component: | Priority:

Type: | Frequency:

Description:

```
Summary:

Steps to Reproduce:
1)
2)
3)

Result:
```

Status: Open Bug | Assigned To: Sean Ahern

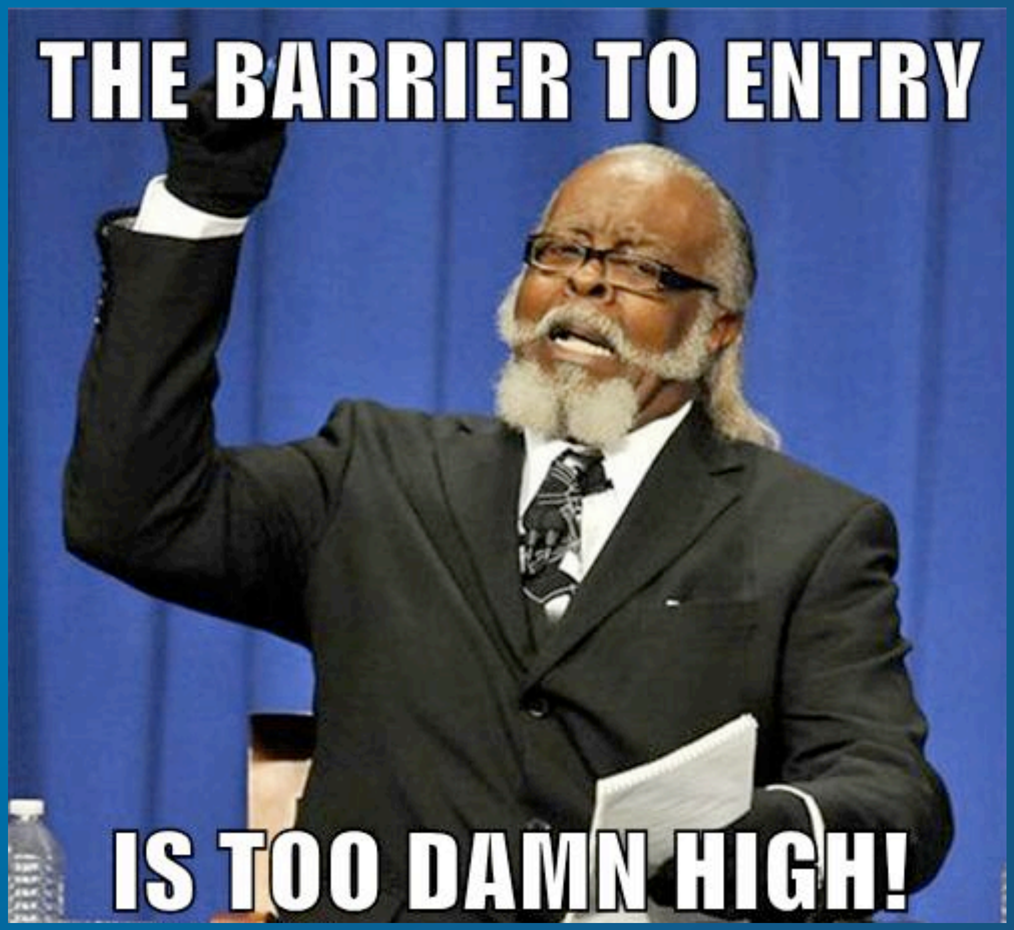
Attachment: No file chosen | Save as:

⊟ Current Status

Must Fix By:

Actual Time: Time Spent 0h | Time Remaining 0h | Total Time 0h

Issue Finish Date: | Percentage:

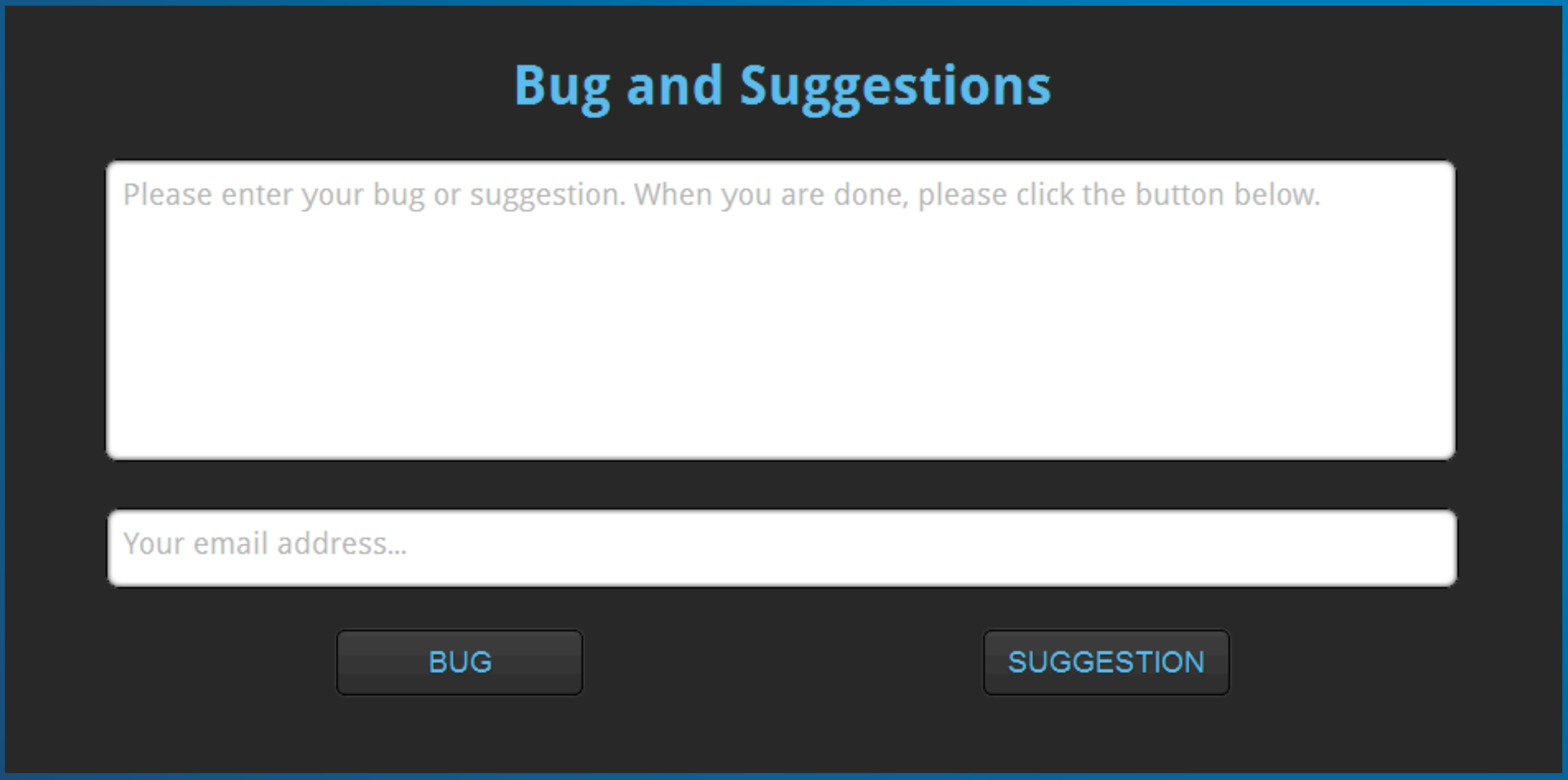


- Early on in the tools development, we felt that the barrier to entry to get feedback and bugs was too damn high!

- Now if any of you have ever used Devtrack, you’ll realize that it’s totally not an optimal solution for what we were looking for.

At the early stages of tools development, it’s important to get as much feedback as possible

Bugs/Suggestions Form


Bug and Suggestions
Please enter your bug or suggestion. When you are done, please click the button below.
Your email address...
BUG
SUGGESTION

Fastest and Easiest Feedback

What we did though was create our bugs and suggestions form.

- This is built directly into the tools - you click on a button in the top corner, and you get this dialog that appears. You put your quick bug or suggestion right in there, add your email address, and you're off. This feedback is then compiled and sent out to our development team each day.

- We found that this is absolutely the fastest and easiest way to get feedback to us.

Overall we've found that this has been a tremendous success.

Issues and Time Wasters

5

10

15

Top Time Wasters:

Optional

More efficient workers = Better and happier workers

- We also have our list of top critical issues which is compiled by our department leads. We put this on an internal web-page for all to see. It allows us to see what’s important prior to deadlines, demos, etc, as it helps to focus and direct our efforts.

- Another thing we have is our top time wasters which we do every three weeks or so. These are the top things that are taking the most time for each person

- We’ve made these optional, but have found them to be quite beneficial, because…

- More efficient workers are better and happier workers

Tools
Education

Finally we have a whole suite of education for our tools. Due to how radically different a number of our systems worked from what everyone else was used to, we needed to make sure that there was as little ramp up time/learning curve as possible.

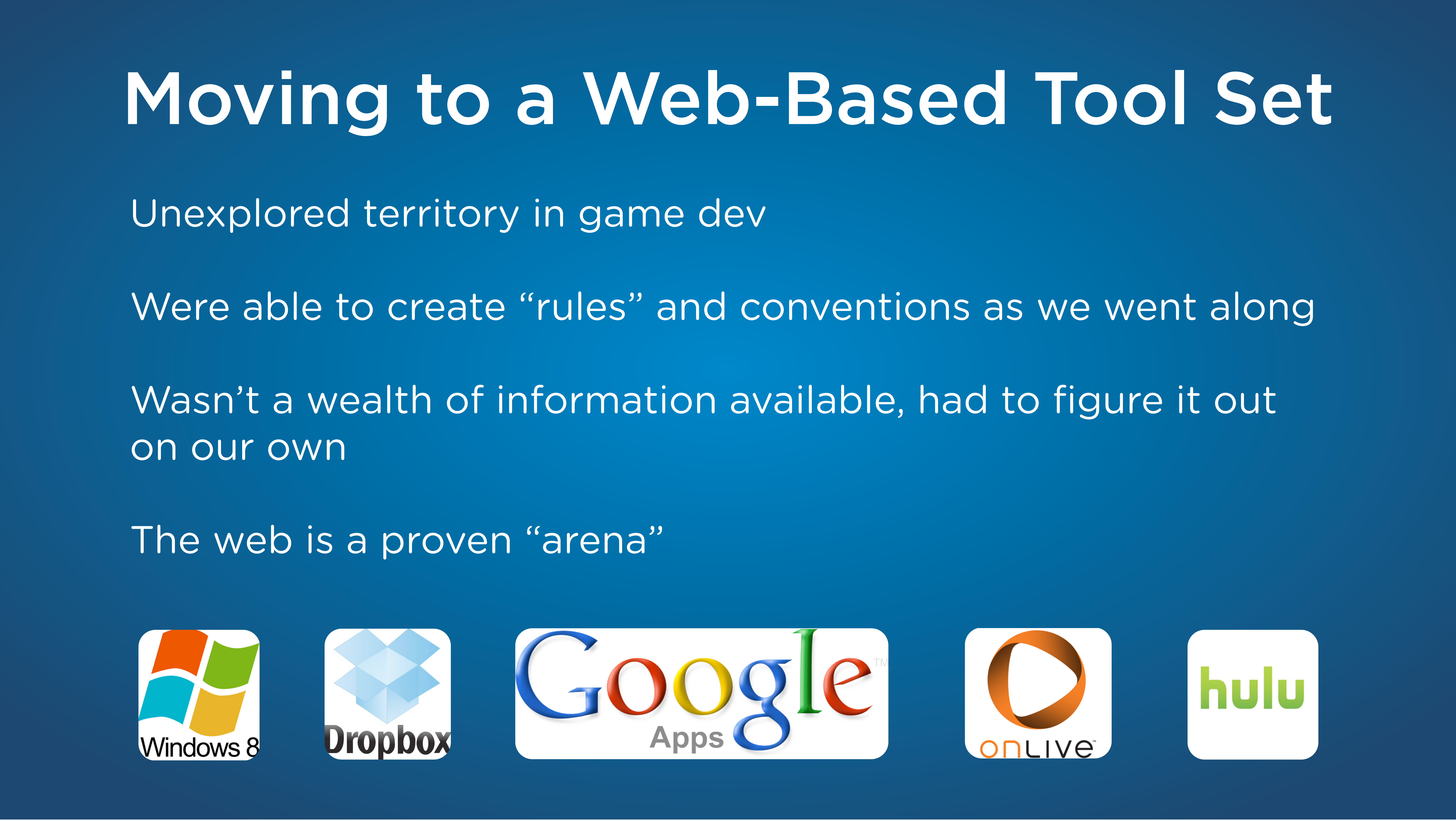
- Videos: Felt that it's much easier to *show* someone via demonstration rather than have them read through a text tutorial on a wiki.

- Help System: So one of the problems that we face is that our tools are very "hot-key" based, so it's incredibly important that everyone has easy access to not only the hot-keys and corresponding functionality, but also an explanation of what each feature does.

- Intro: Presented the tools to each department over 1-2 hours once the tools rolled out. Answered questions and provided guidance on features they would be using the most. All intros were recorded in video form and available via our video tutorials.

- We also brought people on to the tools early: Brought 1-2 people from each department on to the tools early so they could act as a liaison between tools devs and rest of the team. Were given 1-2 month head start on every one else. They also helped provide invaluable feedback as to tools functionality and how it **should** work.

- Visible presence: Made sure that we walked around at least once per day to help answer any questions or help anyone who had issues with the tools during the first few weeks that the tools were in production.



Moving to a Web-Based Tool Set

Unexplored territory in game dev

Were able to create “rules” and conventions as we went along

Wasn’t a wealth of information available, had to figure it out on our own

The web is a proven “arena”

Windows 8
Dropbox
Google Apps
OnLive
hulu

Now I’m sure that many of you have already heard by now that we’ve moved from a traditional tool set to a web-based toolset.

•When we first started, and even still today, it’s very much unexplored territory in game development

•The nice thing about this is that we were able to create rules and conventions as went along

•Unfortunately though, there wasn’t ***really*** a lot a lot of existing information out there, so we really had to figure out a lot of this on our own

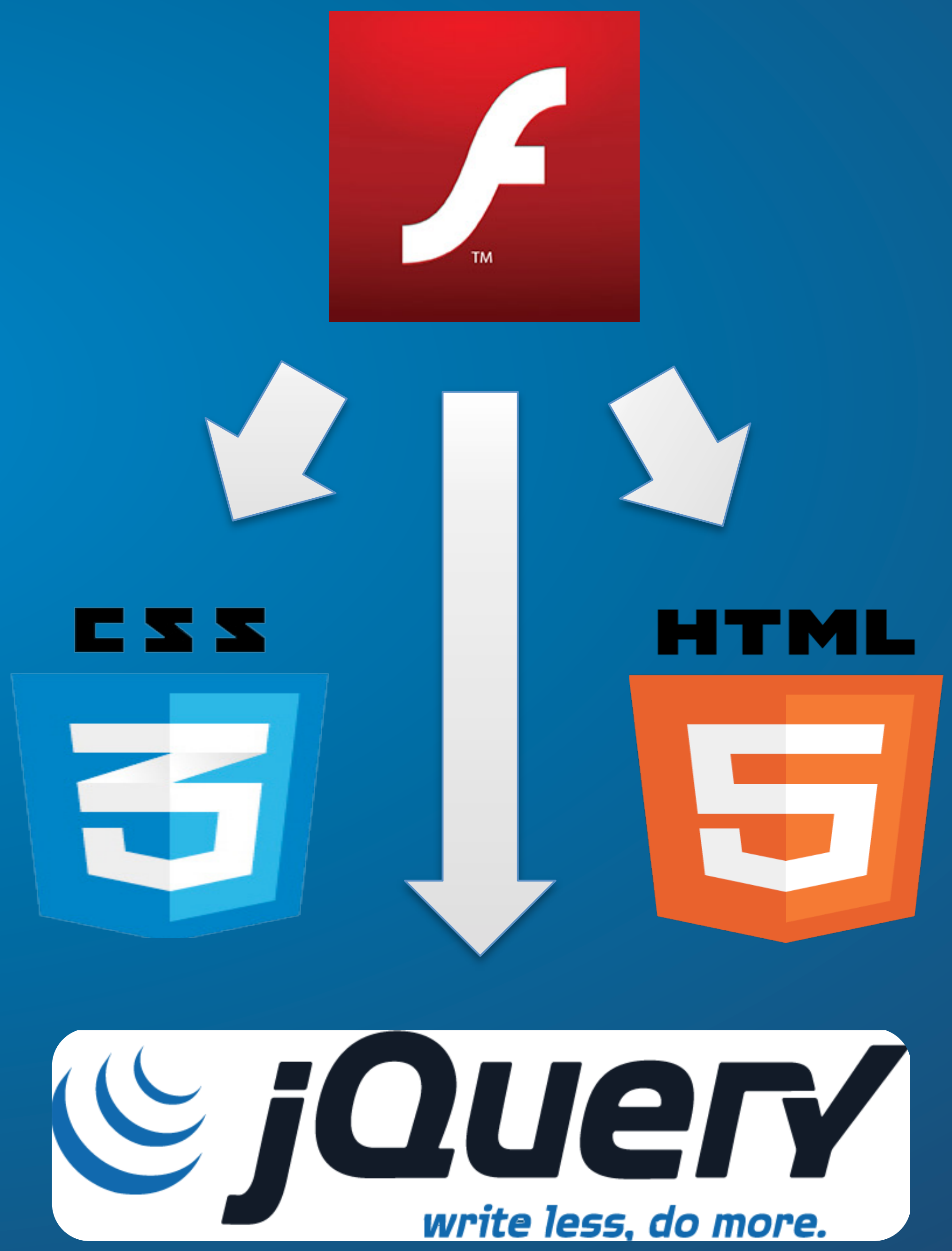
•We noticed though that over the course of development of our tools, the web has become a proven arena. With things like google docs, Windows 8’s metro apps, OnLive, Hulu, etc, everyone is making web-apps now. So because of this we feel like we’re heading very much in the right direction.

HTML5 vs Flash?

Flash great when we started
Proved problematic in the end

Flash to HTML5
- Huge performance gains
- No compiler or proprietary software

Non-programmers can now develop and design systems for the tool set



- When we started out, flash was great. There were a **ton** of pre-existing libraries because flash has a lot of strengths for making web-based applications.

- Unfortunately as we went on we realized that flash was proving to be quite problematic. The main issues stemmed from us being stuck using ActionScript 2 due to Scaleform Limitations at the time. We also had the problem of a lack of ActionScript knowledge on the team, as well as ActionScript 2 being both memory intensive, archaic, and incredibly hard to debug

- So we moved from Flash to JQuery, HTML5, Canvas, Css3, etc and have noticed improvements across the board!

- Right away we noticed huge performance gains across the board

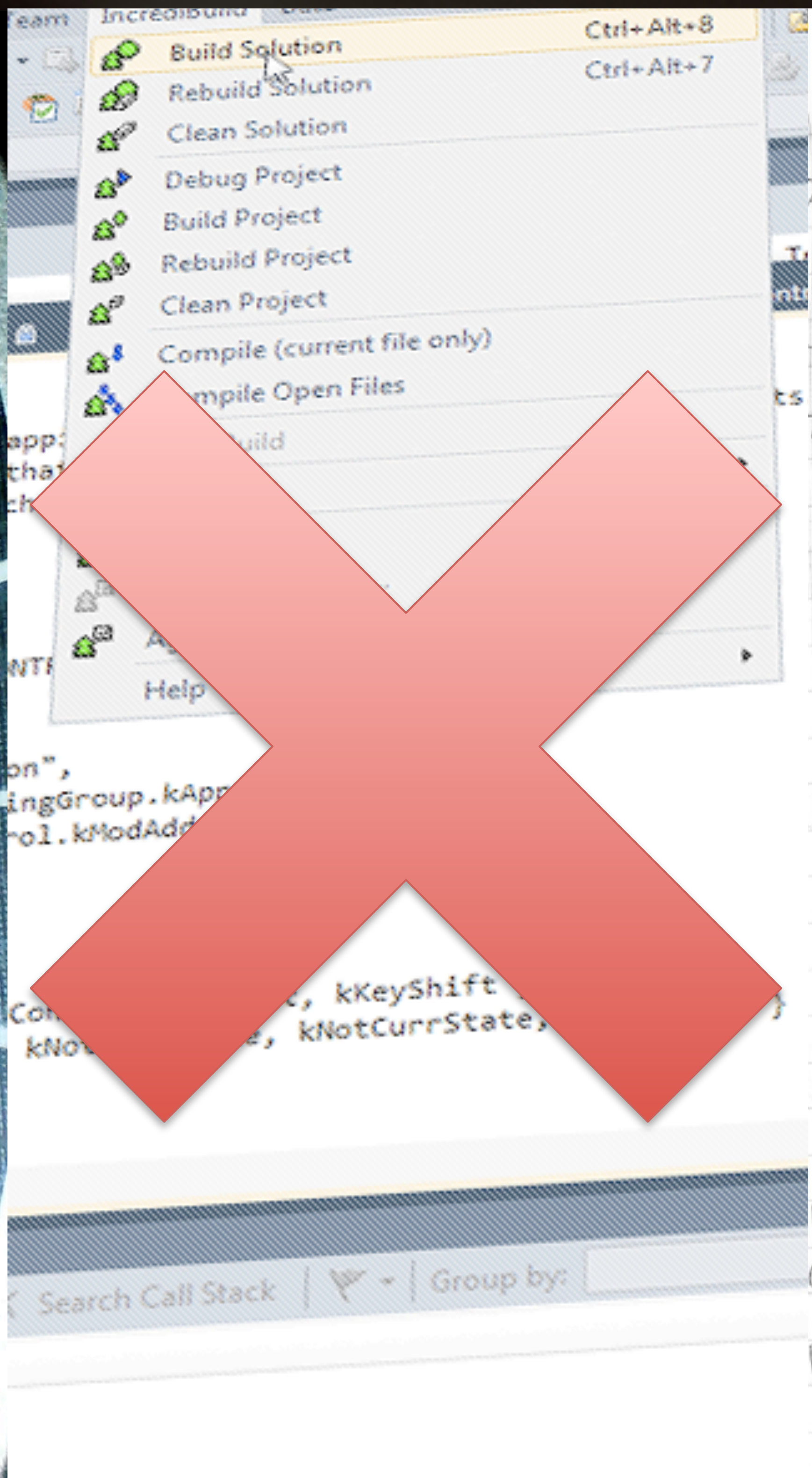
- There's also no longer a need for any special compilers or proprietary software needed - you can open up Notepad or Notepad++ or anything else and start making tools for our toolset.

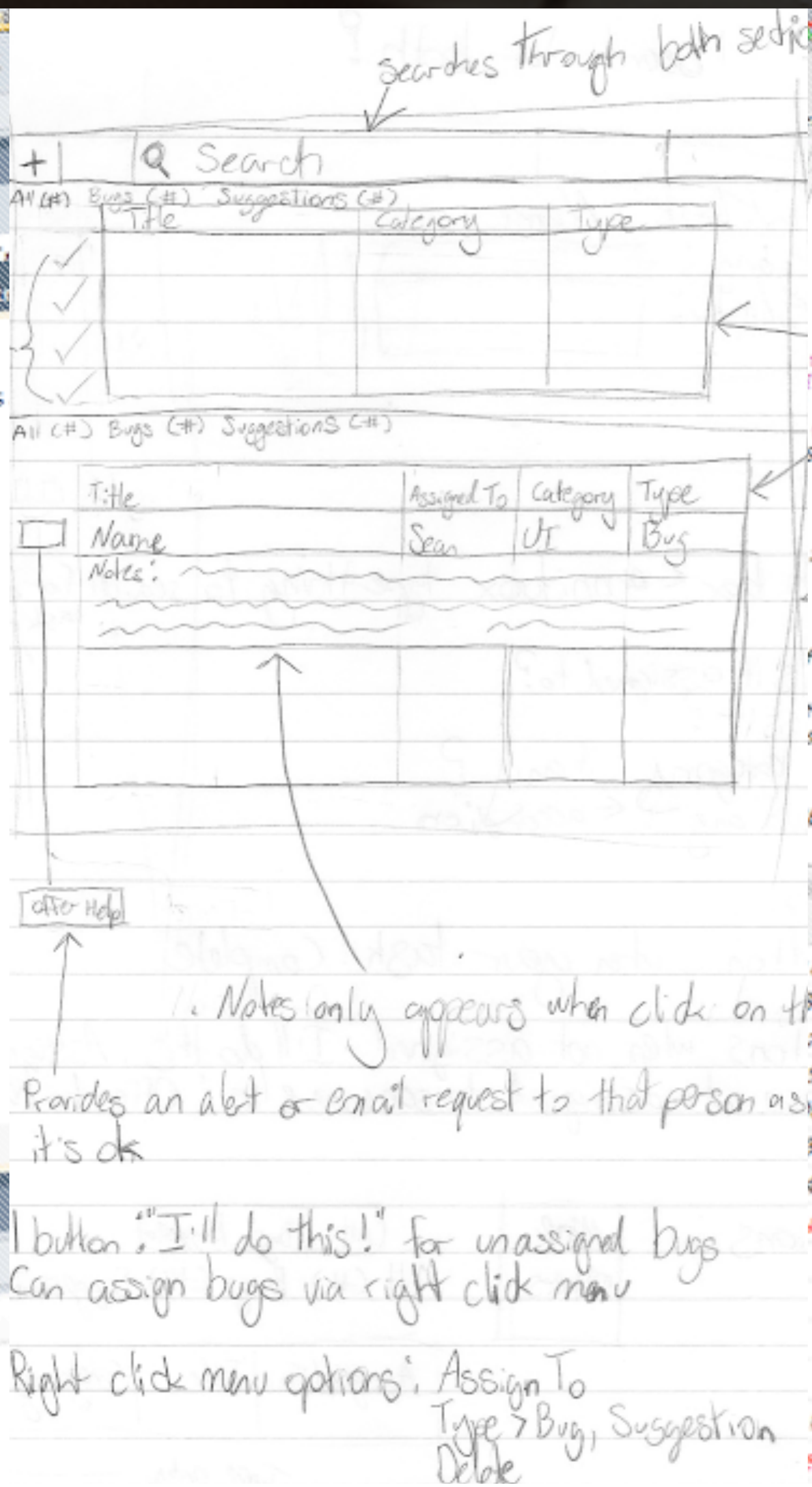
- Non-programmers can now also now make tools and additions for our toolset. As long as know basic web development tech (jquery, html, etc), you can design for our tools. This **drastically** lowers the barrier to entry for making changes or modifications for our tools

Easier, Better, Faster, Stronger

Now with all of the changes that we've made across our toolset, we've found that our tools have become easier, better, faster, and much stronger as a result.

Easier, Better, Faster, Stronger



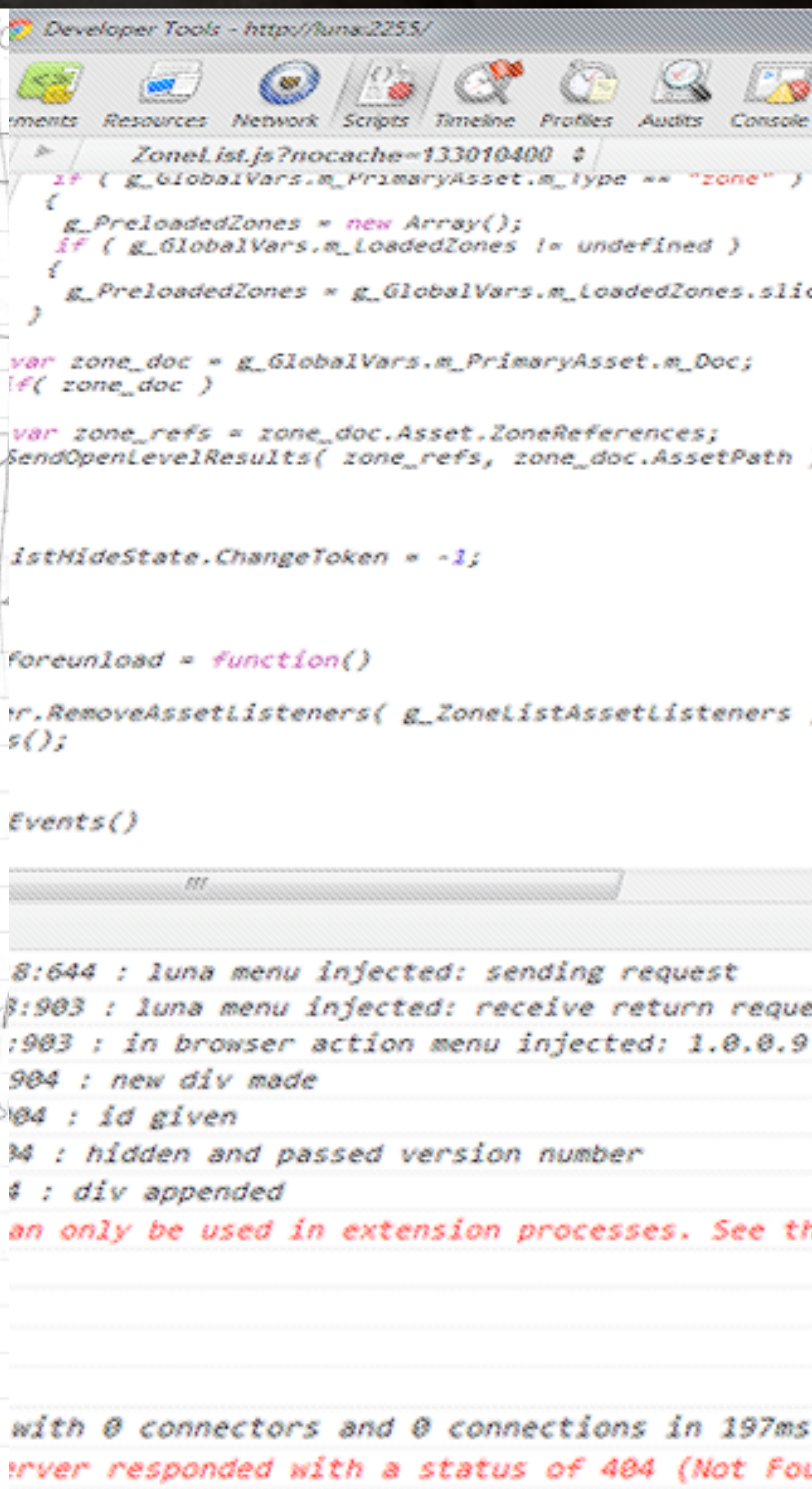

Searches through both sectio
+
Search
All (#) Bugs (#) Suggestions (#)
Title
Category
Type
All (#) Bugs (#) Suggestions (#)
Title
Assigned To
Category
Type
Name
Sean
UI
Bug
Notes:
Offer Help

. Notes only appears when click on th

Provides an alert or email request to that person as it's ok

1 button: "I'll do this!" for unassigned bugs
Can assign bugs via right click menu

Right click menu options: Assign To
Type > Bug, Suggestion
Delete


Developer Tools - http://luna:2255/
ments
Resources
Network
Scripts
Timeline
Profiles
Audits
Console
ZoneList.js?nocache=133010400

```
  if ( g_GlobalVars.m_PrimaryAsset.m_Type == "zone" )
  {
   g_PreloadedZones = new Array();
   if ( g_GlobalVars.m_LoadedZones != undefined )
   {
    g_PreloadedZones = g_GlobalVars.m_LoadedZones.slic
  }

var zone_doc = g_GlobalVars.m_PrimaryAsset.m_Doc;
if( zone_doc )

var zone_refs = zone_doc.Asset.ZoneReferences;
SendOpenLevelResults( zone_refs, zone_doc.AssetPath


istHideState.ChangeToken = -1;


foreunload = function()

r.RemoveAssetListeners( g_ZoneListAssetListeners
s();

Events()
```

```
8:644 : luna menu injected: sending request
3:903 : luna menu injected: receive return reque
:903 : in browser action menu injected: 1.0.0.9
904 : new div made
04 : id given
04 : hidden and passed version number
4 : div appended
an only be used in extension processes. See th



with 0 connectors and 0 connections in 197ms
erver responded with a status of 404 (Not Fou
```

- With all of the processes that we've implemented, UI iteration is now extremely fast.

- With the exception of flash, there's no longer any building or compiling - you make your changes and reload the web-page.

- We can now rapidly prototype tools. We can have one person coding the back-end with another person coding the visuals and aesthetics at the same time with css.

- We can also use Chrome's Developer Tools to make live changes. This lets us receive actionable feedback **immediately** which is **really** awesome - let me tell you.

How We've Benefitted

Tools actually *make sense*

Usability shadowing sessions

A *faster* and more *efficient* tool set

So how have we actually benefitted from all of this focus on usability?

- Our tools actually make sense now. We've completely redesigned our icons, workflows, and everything else as well. With our web-based tool set, we can make live tool changes to really make sure that we're right on with our designs.

- With our Usability Shadowing, we've been able to drasically reduce the amount of visual clutter to only display the necessary amount of information. We're picking up a ton of issues that we weren't aware of before, and because of this, we've been able to really streamline our experience with the things that we've observed. We're making sure that we're finding as many issues early and often so that we can fix them before they hit production.

- We also have a tool set and engine that is faster and much more efficient than it was before. We've reduced the downtime, made sure that our levels load faster, and we can see all of these changes in the game much faster than ever before.

In Closing...

Better, Faster, More Efficient

Easy To Use

Want To Use The Tools

So in closing...

We had an entire tool set switch in the middle of production, and in less than a month people are working

- Better, Faster, and much more Efficiently than ever before
- We also found that it’s been easier for new hires to jump on to the tools with as minimal ramp-up time as possible
- And most importantly people actually want to use the tools

“Perfection is achieved, not when there is ***nothing more to add***, but when there is ***nothing left to take away***”

- The Little Prince

So before we wrap up, I want to stress one thing:
What I’ve said today are not laws, but instead are guiding principles that we uphold as something to constantly strive towards with everything that we do moving forward.

Thank You!

You’ve been *awesome*

So hopefully I’ve given you all a few ideas on how to better use usability and user experience in your own studios. Now I want to thank you all for coming, you’ve been totally awesome.

We’re Hiring!

Come see us at

Booth #1314

Sean Ahern

sahern@insomniacgames.com

bit.ly/insom_rnd

Now we’re hiring! So come check us out.

Once again, my name is Sean Ahern, you can reach me at the email address on screen, and grab this presentation at the address above.

If there are any questions I’ll take them now.