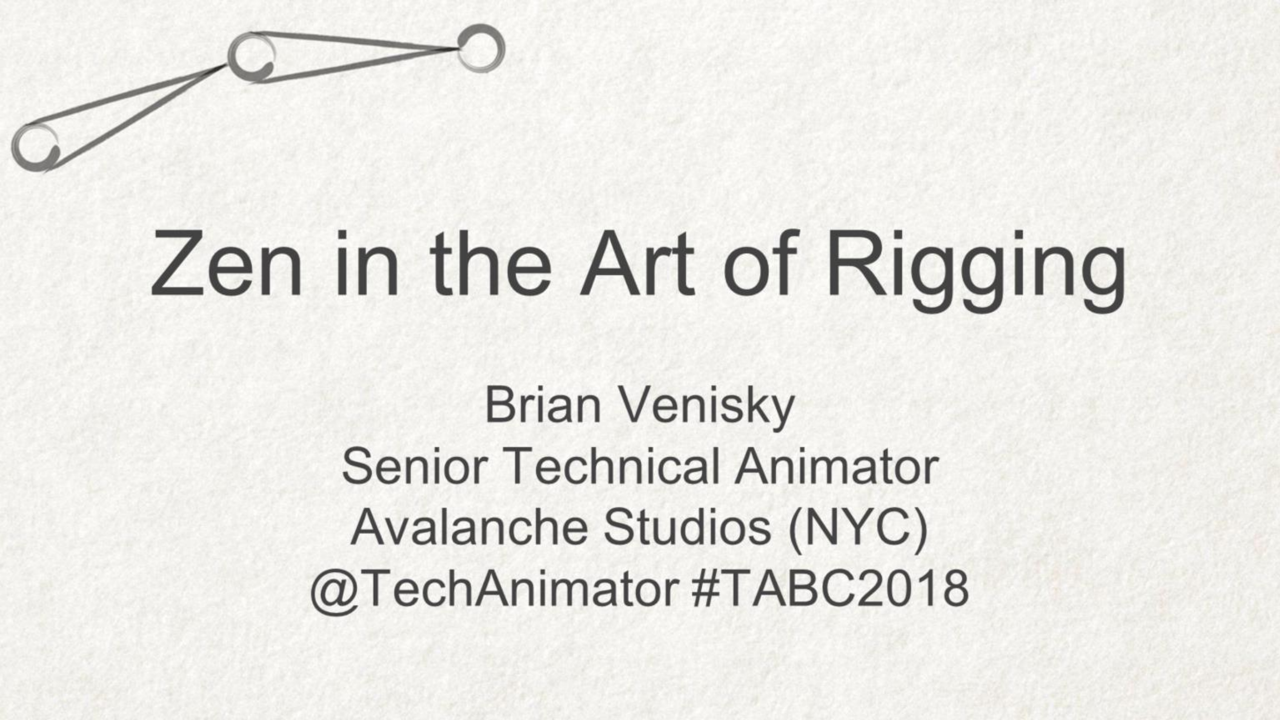
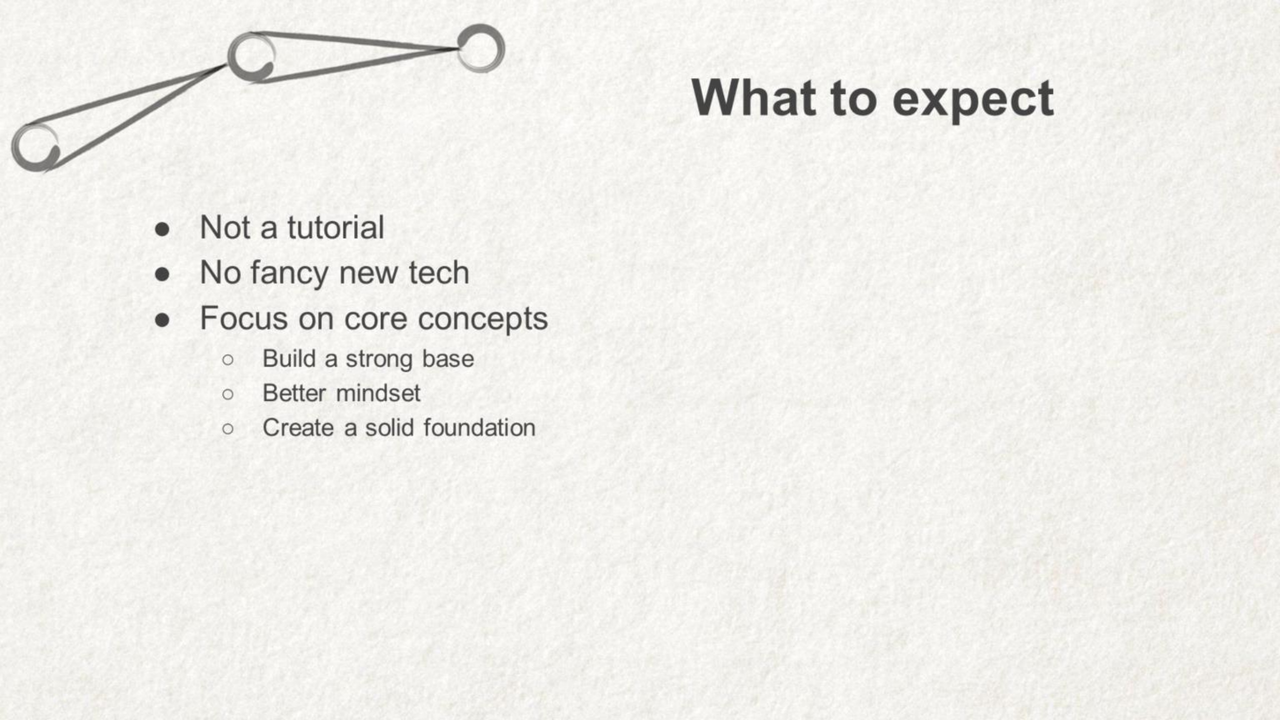


Zen in the Art of Rigging

Brian Venisky
Senior Technical Animator
Avalanche Studios (NYC)
@TechAnimator #TABC2018

- Hello! I am Brian Venisky, a Senior Tech Animator at Avalanche Studios in NYC and I'm here to kick off the Rigging Power Hour with Rich and myself
- A note before we get started since not all of you were probably around at the start of the bootcamp
    - The slides for this presentation will be put up online, so don't worry if you're taking notes off the slides and you miss something



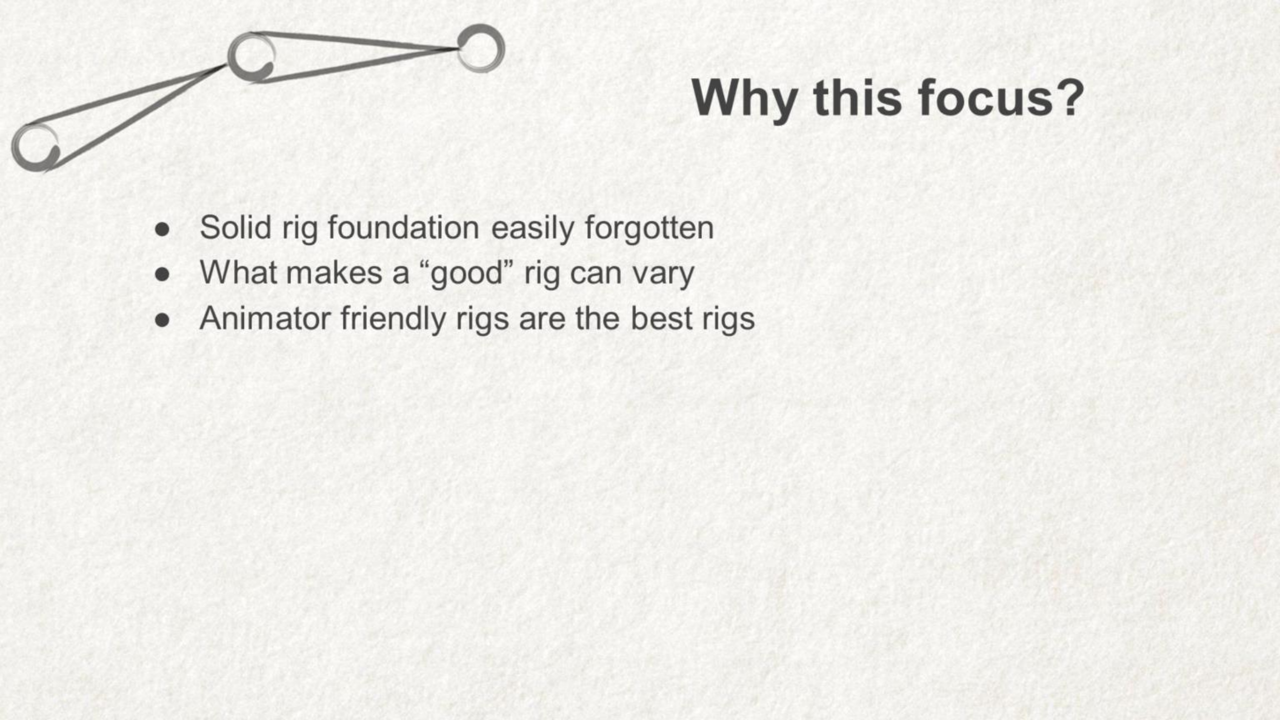
What to expect

- Not a tutorial
- No fancy new tech
- Focus on core concepts
  - Build a strong base
  - Better mindset
  - Create a solid foundation

WHAT TO EXPECT

- Before I get started I want to go over what to expect from this talk because
- It’s not meant to be a tutorial or teach you how to rig
- And I won’t be showing off any sort of revolutionary new tech and from experience I know that conference goers tend to expect the latest advances.
- I actually want to do the opposite today
- My goal is to focus on the core concepts that help build a strong base for your rigs and have you walk away today with a better mindset when you think about building your rigs and creating tools within a game development environment
- If you’re newer to rigging, hopefully you’ll learn a lot
- And if you’re seasoned veteran, then hopefully this will all serve as a refresher
- With the chance you’ll pick up something new you

haven’t thought of before

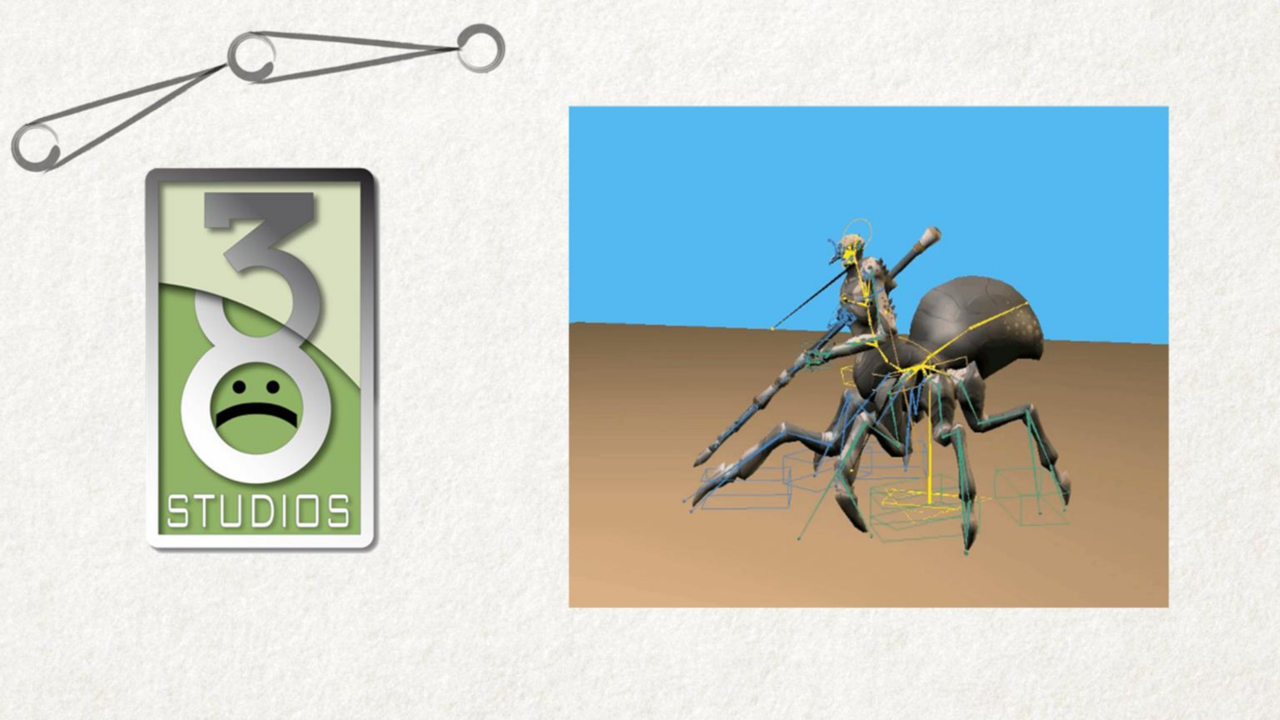


Why this focus?

- Solid rig foundation easily forgotten
- What makes a “good” rig can vary
- Animator friendly rigs are the best rigs

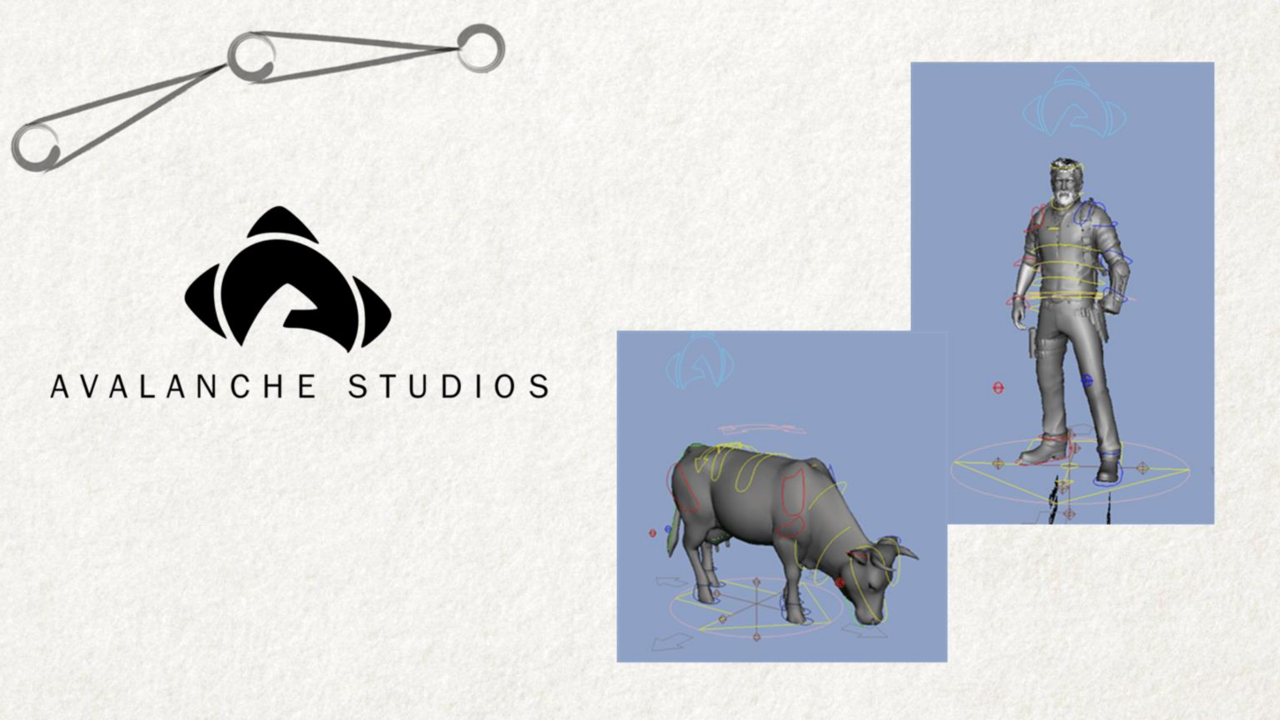
WHY THIS TOPIC?

- I find that the thought process of building rigs off of a good foundation is easily lost and forgotten
    - So I want to take it back to some basics
- As there’s really no exact science for what makes a “good” rig
- It varies from person to person and team to team depending on project needs
- You can however, make sure that a rig is is considered a good one by making it as animator friendly as possible
- And while there are many good learning resources out there for rigging
    - There’s no tutorial or how-to that is going to verify these variables for you as they are specific to you and your project



ABOUT ME

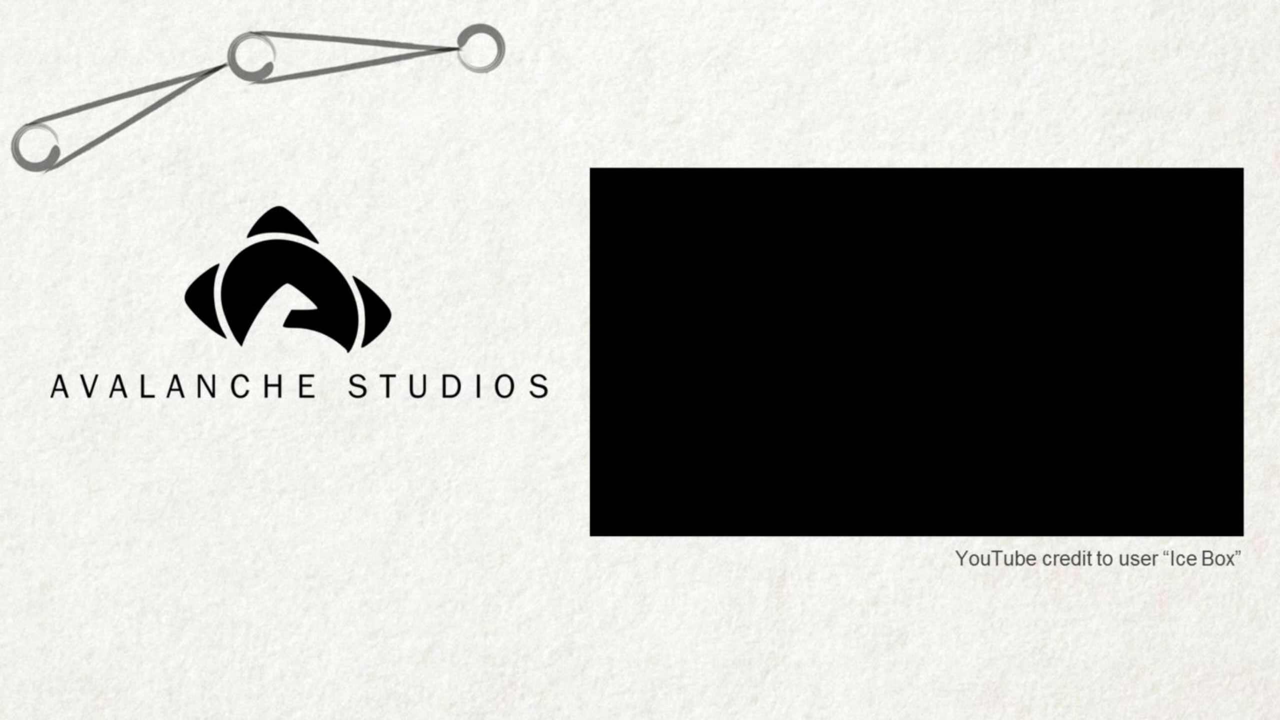
- I’ve been a game developer for 8 years now
- And along the way I’ve rigged and skinned many different types of creatures and characters
- Ranging from mythical MMO creatures from my 2 years at 38 studios



AVALANCHE STUDIOS

ABOUT ME

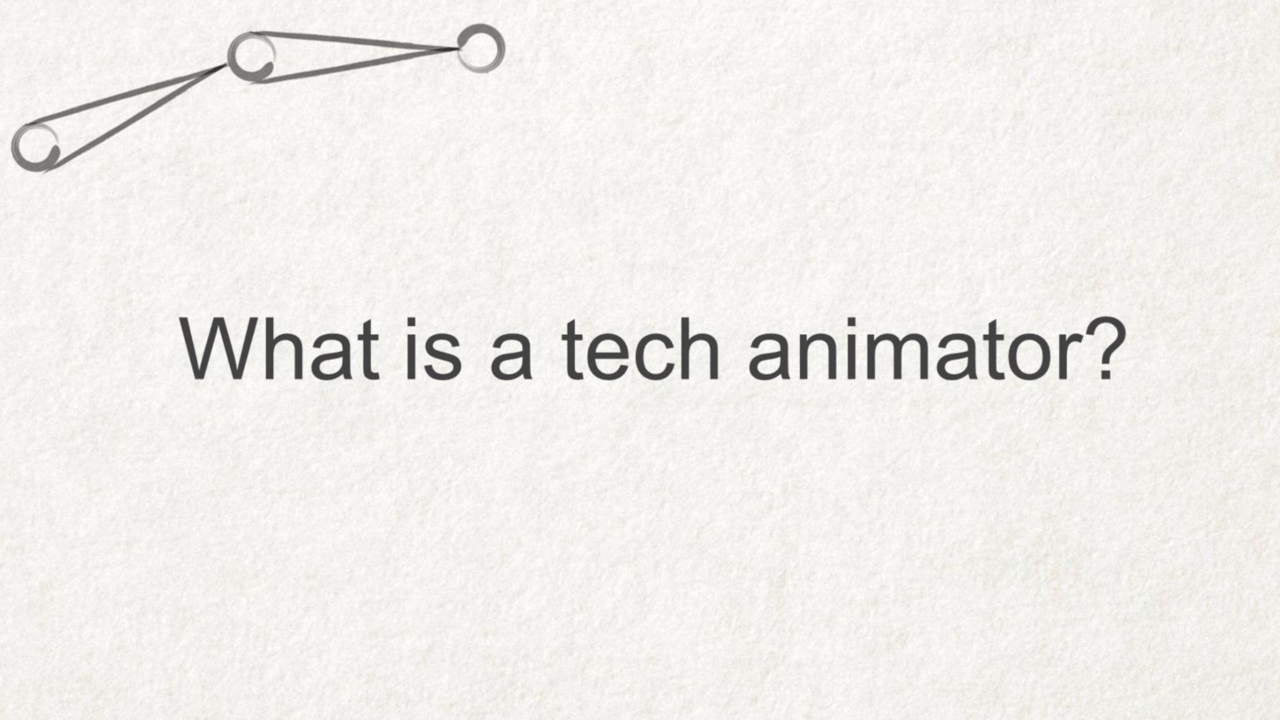
- To your typical bipeds and quadrupeds that I’ve been used to for almost 6 years now at Avalanche in New York
- You might be thinking “Going from fantasy creatures.... to cows... isn’t quite glamorous… but let me tell you…



YouTube credit to user “Ice Box”

ABOUT ME

- (Let video play….)
- ...my hard work certainly doesn't go unappreciated
- Besides making rigs for virtual cows that end up becoming rockets...
- I’ve also had to focus on a lot things besides straight up rigging...which leads me to a very important question to kick things off...



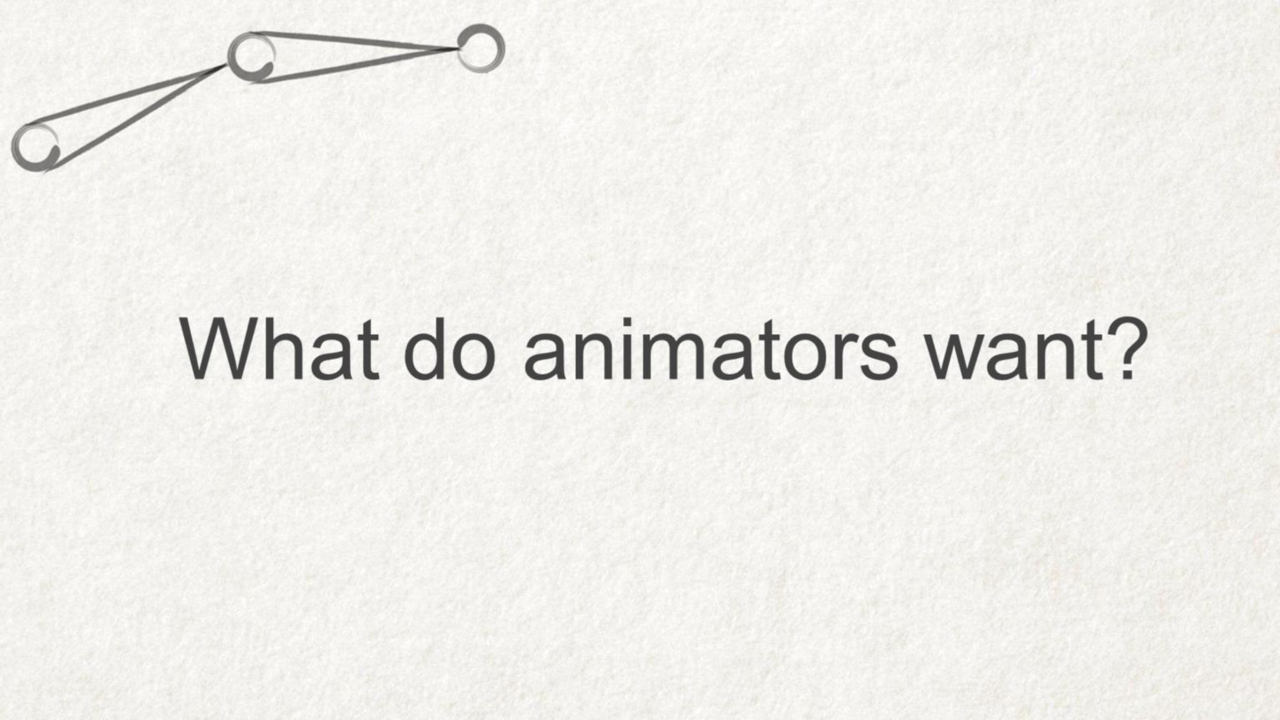
What is a tech animator?

WHAT IS A TECH ANIMATOR

- A question that surely needs years upon years of curated roundtable discussion
- If you aren’t a tech animator...you might be saying right now “hey pal this is the tech ART bootcamp”
- But tech animators ARE tech artists my friend.
- Sure we make rigs and from time to time we’re known to create an animation or 2
- But we’re not just building character and face rigs
- We’re writing tools for animators
- Working in engine on state machines
- And adding code to our game editors
- Among many other tasks within the “tech art” realm
- Depending on your studio, you may just be labeled a “tech artist” anyway
- No matter what, your job each and every day is to support the animation team in whatever way you can

and be a problem solver. Doesn’t matter if your rigging, writing, or donning that mocap suit for a day.

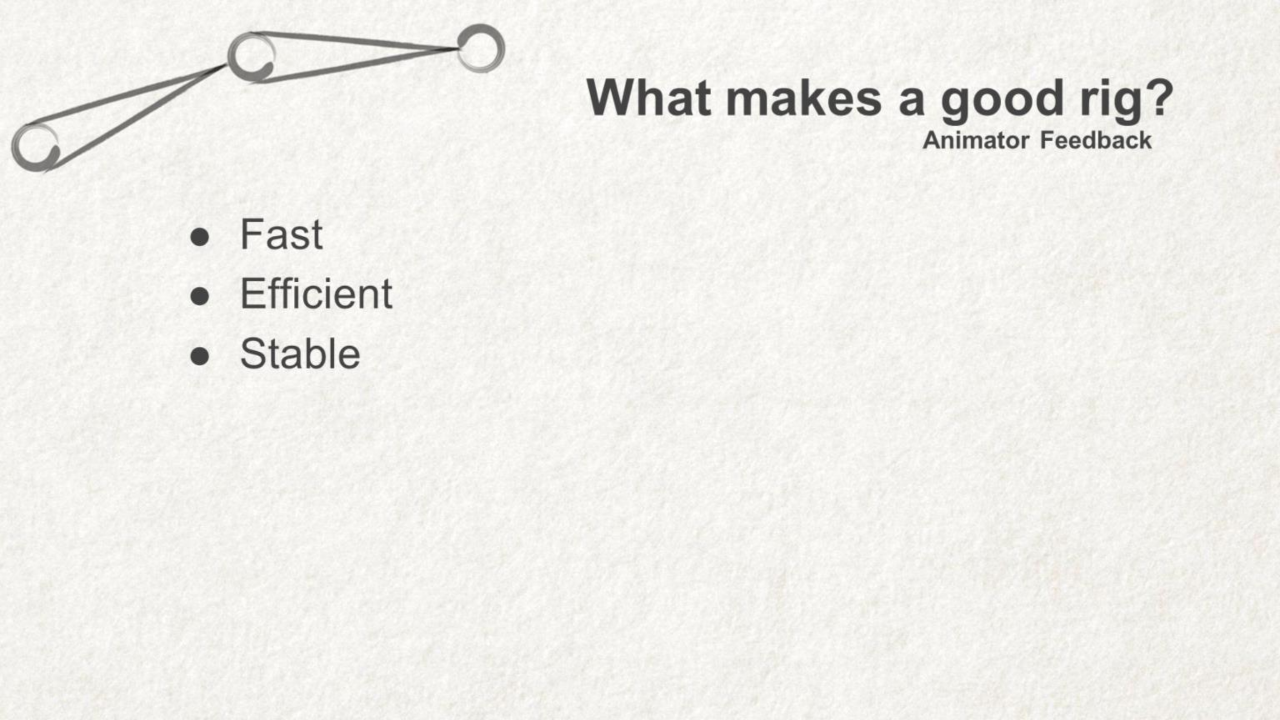
- Be at peace with the fact that no one really knows WHAT to call you



What do animators want?

WHAT DO ANIMATORS WANT

- Since we’re in the habit of supporting animators as tech animators...we should try and understand what THEY want
- You are building rigs for them...and they are very much your clients. Your work is very dependant on their needs and wants, and it is your job to deliver.
- In order to find some more of the “global” needs
    - I asked a group of animators of varied skill levels some questions to find out what they prefer and their thoughts on rigs
    - And I’ll be referring to some of these questions and results throughout this talk



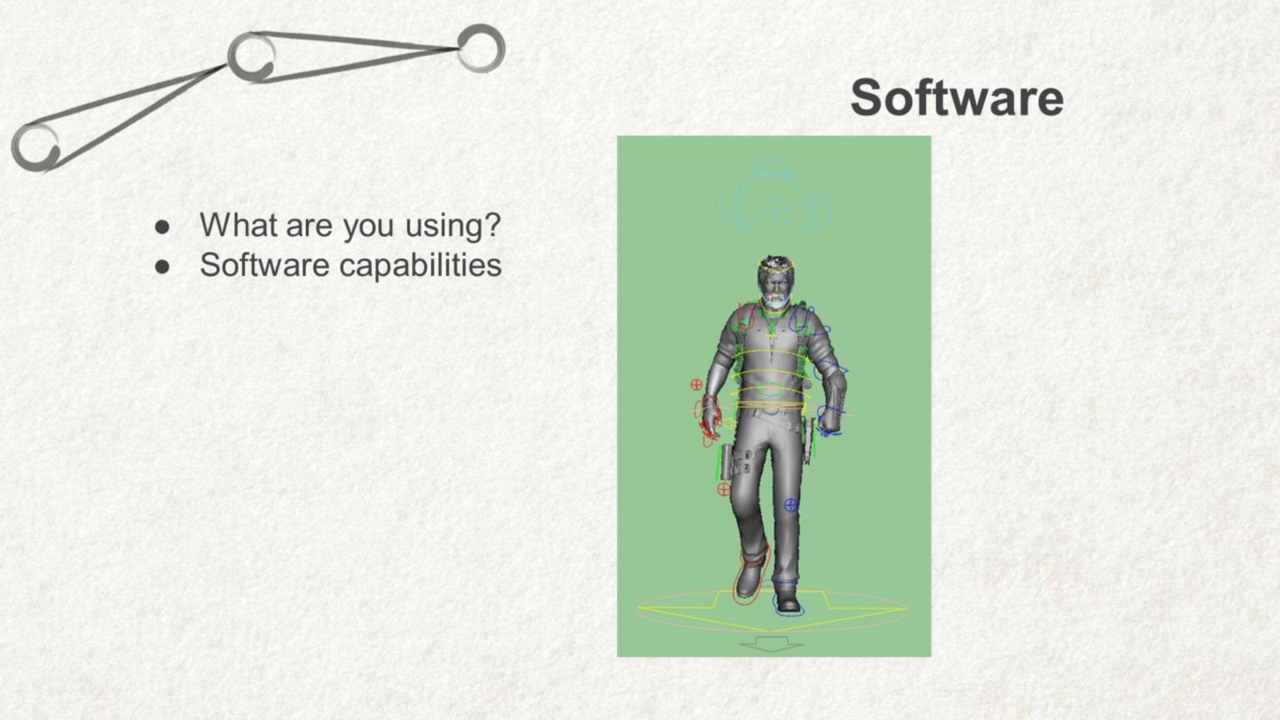
What makes a good rig?

Animator Feedback

- Fast
- Efficient
- Stable

WHAT MAKES A RIG GOOD

- So let’s kick it off with one!
- What makes a rig, good
- Every single animator had some form of one of these.
  - They want the ability to work quickly
  - In an efficient manner
  - And not have to worry about unexpected rig problems or breaks
- Animators seem to really care about the usability and performance of their rigs over anything

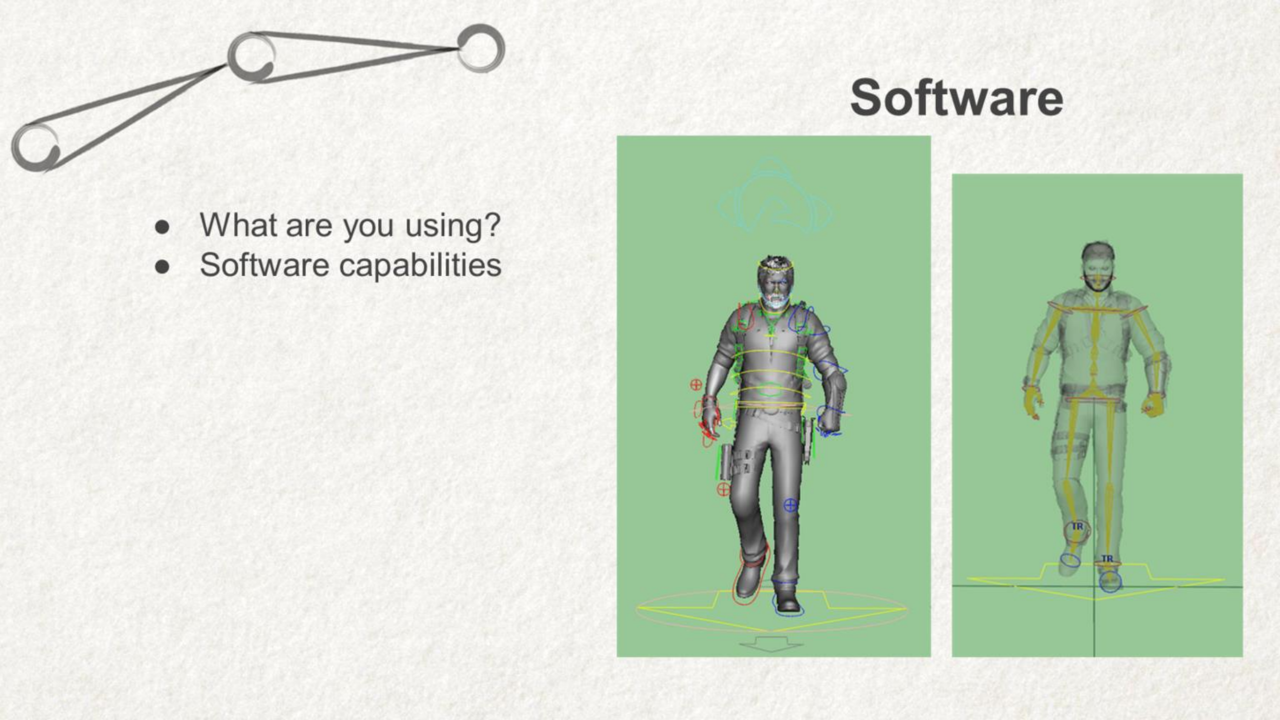


Software

- What are you using?
- Software capabilities

KNOW YOUR SOFTWARE

- Just a heads up, I’m presenting with Maya in mind, but most concepts tend to be the same across all 3D packages, so just be aware of the software you use and what it can and can’t do out of the box
- You should know what included features are available to you and what tools already exist that you can get online. You can start to plan out what custom and in house tools you’ll need to make yourself based off of all this.
- For example at Avalanche we use both Maya and MotionBuilder
    - And animators need the flexibility to work in either…
    - Meaning not only did we need this maya rig you see...
    - NEXT SLIDE

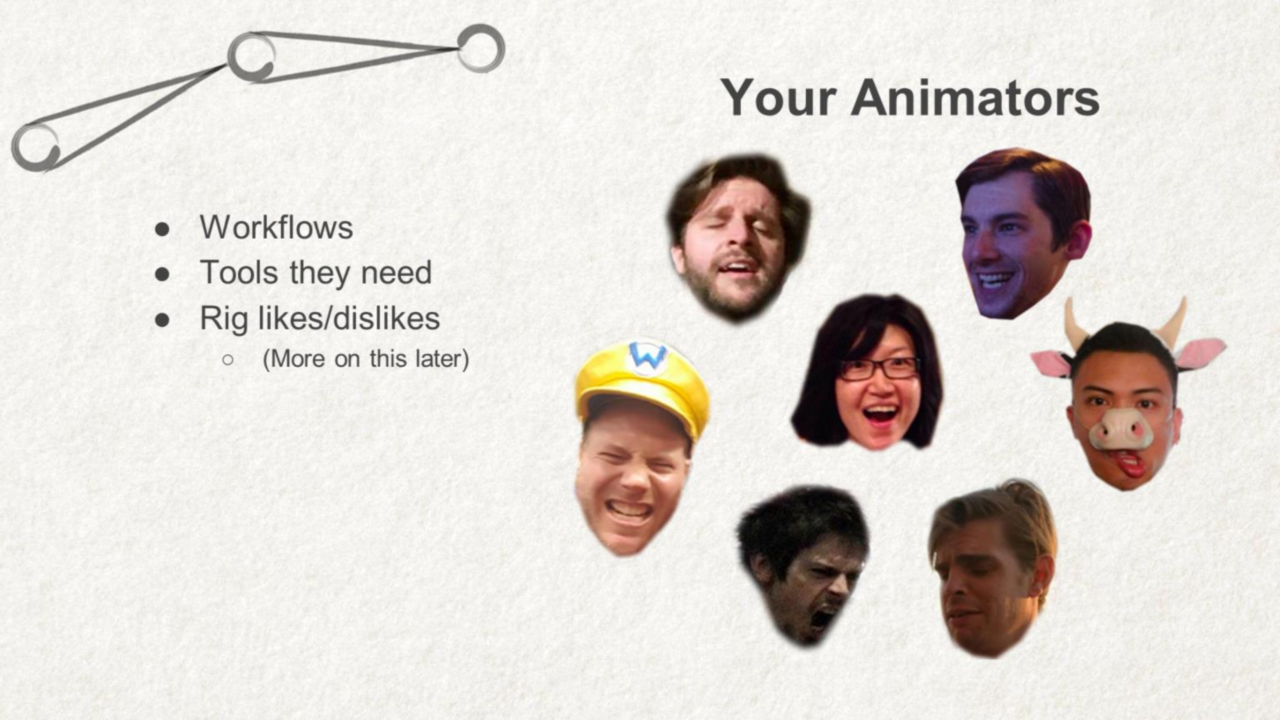


Software

- What are you using?
- Software capabilities

KNOW YOUR SOFTWARE

- But a Human Ik rig for use in MoBu as well
- Because of this there is a need for auto-rig tools to build both types of rigs
- And I also wrote a tool so they could easily transfer animation to and from each at the click of a button, which was used to get the results on screen now

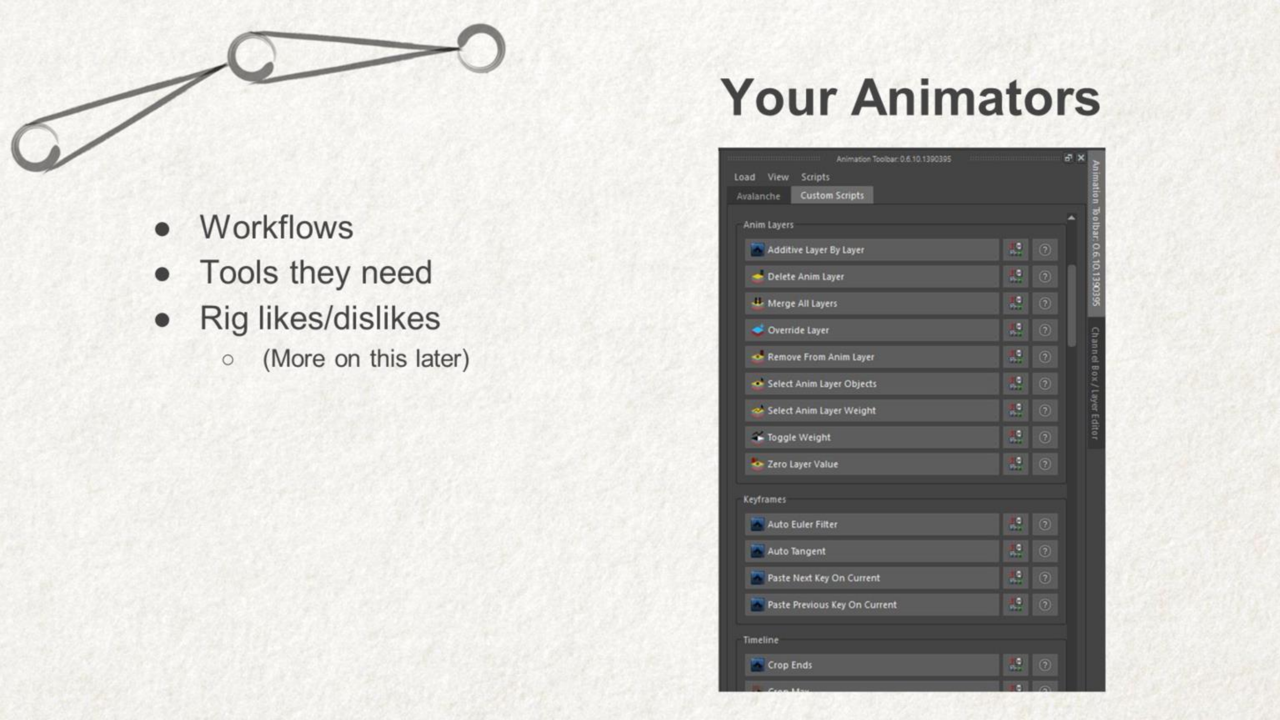


Your Animators

- Workflows
- Tools they need
- Rig likes/dislikes
  - (More on this later)

KNOW YOUR TEAM

- Speaking of rig flexibility, you will need to take into account and balance each animator’s preferred workflow to create a rig (or rigs) and toolset that everyone can work with
- You should ask the animators to tell you how they like to work or even live demo their workflows for you to help gauge their personal needs
- For instance at Avalanche…
- NEXT SLIDE

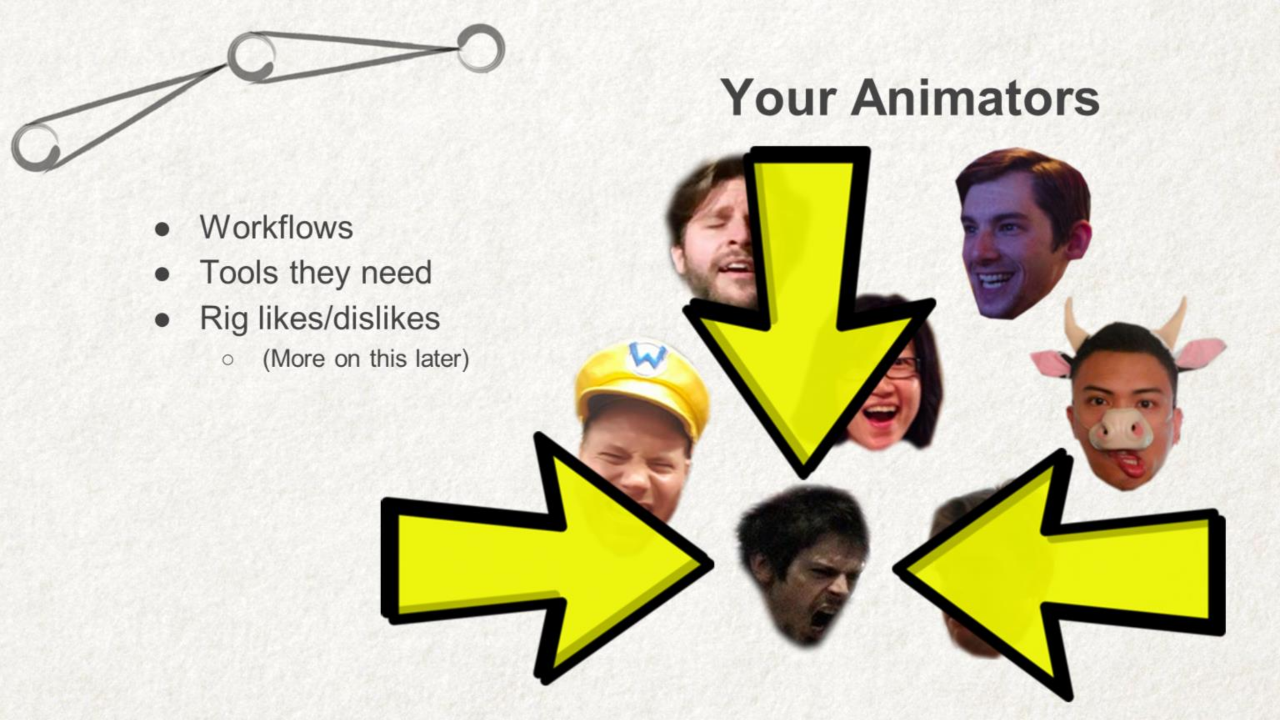


Your Animators

- Workflows
- Tools they need
- Rig likes/dislikes
    - (More on this later)

KNOW YOUR TEAM

- We have a nice, neat, dockable UI for all our animation tools
- However..There are always those that go against the grain
    - And there are SOME that don’t like using a UI

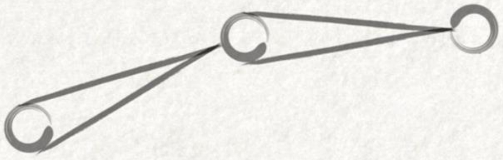


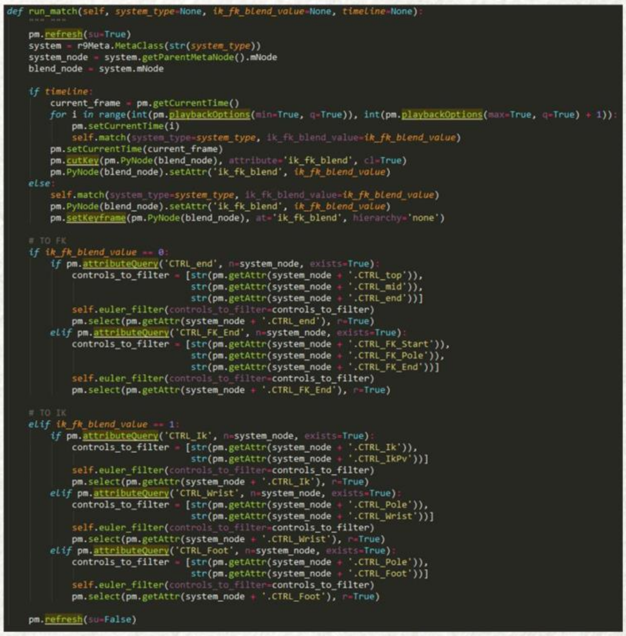
Your Animators

- Workflows
- Tools they need
- Rig likes/dislikes
  - (More on this later)

KNOW YOUR TEAM

- Pause 5 sec
- That’s okay though!
- NEXT SLIDE

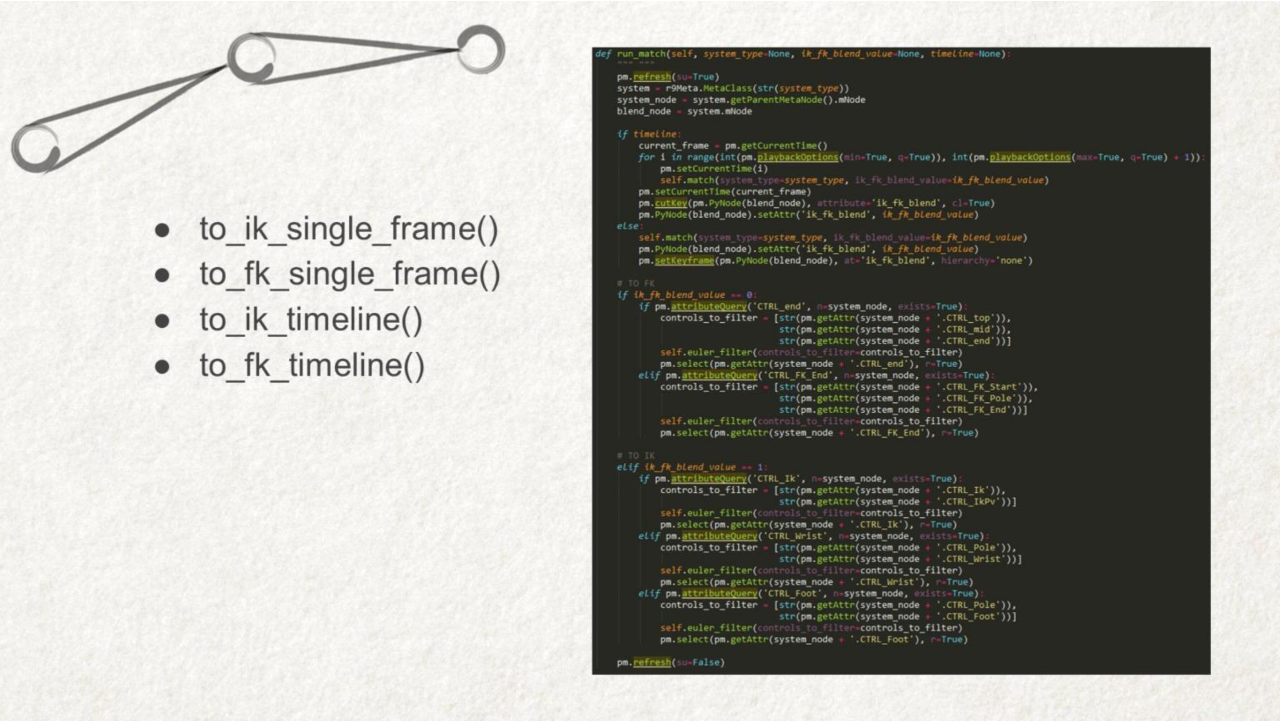




```python
def run_match(self, system_type=None, ik_fk_blend_value=None, timeline=None):
    """ """

    pm.refresh(su=True)
    system = r9Meta.MetaClass(str(system_type))
    system_node = system.getParentMetaNode().mNode
    blend_node = system.mNode

    if timeline:
        current_frame = pm.getCurrentTime()
        for i in range(int(pm.playbackOptions(min=True, q=True)), int(pm.playbackOptions(max=True, q=True) + 1)):
            pm.setCurrentTime(i)
            self.match(system_type=system_type, ik_fk_blend_value=ik_fk_blend_value)
        pm.setCurrentTime(current_frame)
        pm.cutKey(pm.PyNode(blend_node), attribute='ik_fk_blend', cl=True)
        pm.PyNode(blend_node).setAttr('ik_fk_blend', ik_fk_blend_value)
    else:
        self.match(system_type=system_type, ik_fk_blend_value=ik_fk_blend_value)
        pm.PyNode(blend_node).setAttr('ik_fk_blend', ik_fk_blend_value)
        pm.setKeyframe(pm.PyNode(blend_node), at='ik_fk_blend', hierarchy='none')

    # TO FK
    if ik_fk_blend_value == 0:
        if pm.attributeQuery('CTRL_end', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_top')),
                                  str(pm.getAttr(system_node + '.CTRL_mid')),
                                  str(pm.getAttr(system_node + '.CTRL_end'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_end'), r=True)
        elif pm.attributeQuery('CTRL_FK_End', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_FK_Start')),
                                  str(pm.getAttr(system_node + '.CTRL_FK_Pole')),
                                  str(pm.getAttr(system_node + '.CTRL_FK_End'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_FK_End'), r=True)

    # TO IK
    elif ik_fk_blend_value == 1:
        if pm.attributeQuery('CTRL_Ik', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_Ik')),
                                  str(pm.getAttr(system_node + '.CTRL_IkPv'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_Ik'), r=True)
        elif pm.attributeQuery('CTRL_Wrist', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_Pole')),
                                  str(pm.getAttr(system_node + '.CTRL_Wrist'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_Wrist'), r=True)
        elif pm.attributeQuery('CTRL_Foot', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_Pole')),
                                  str(pm.getAttr(system_node + '.CTRL_Foot'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_Foot'), r=True)

    pm.refresh(su=False)
```

KNOW YOUR TEAM

- I just have to make sure tools like this ik_fk_match

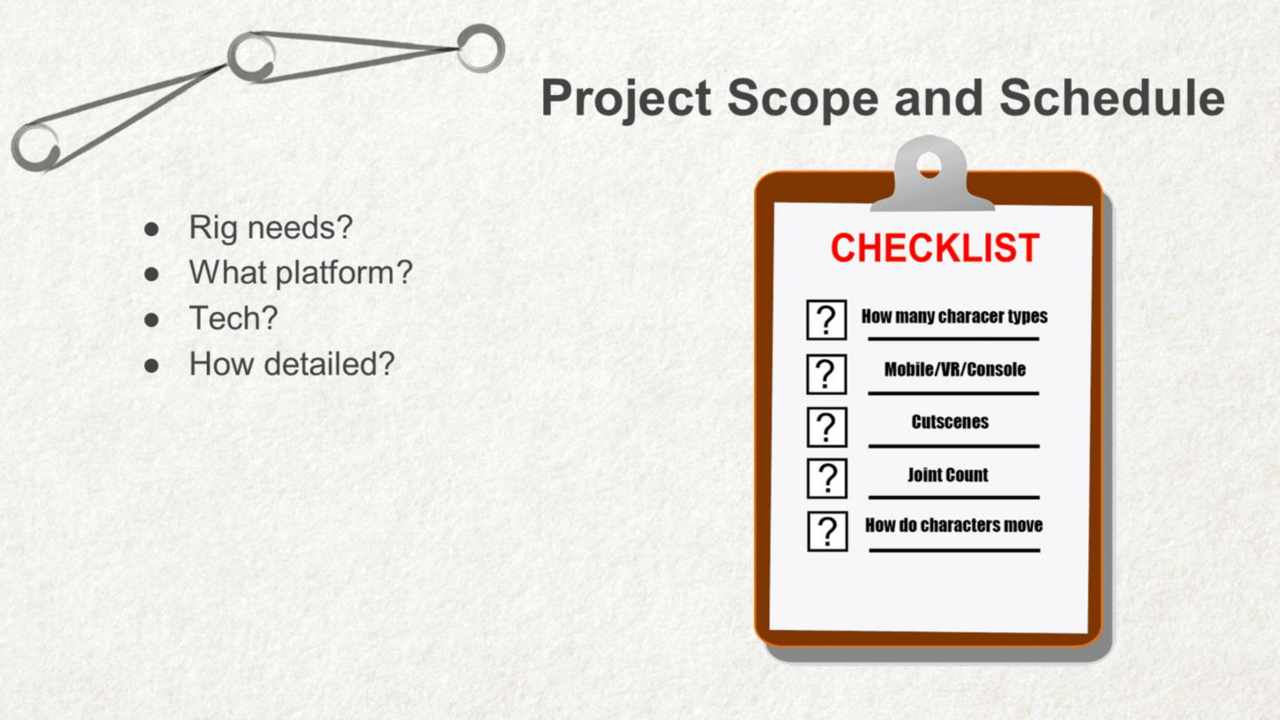


- to_ik_single_frame()
- to_fk_single_frame()
- to_ik_timeline()
- to_fk_timeline()

```
def run_match(self, system_type=None, ik_fk_blend_value=None, timeline=None):
    """ """
    pm.refresh(su=True)
    system = r9Meta.MetaClass(str(system_type))
    system_node = system.getParentMetaNode().mNode
    blend_node = system.mNode

    if timeline:
        current_frame = pm.getCurrentTime()
        for i in range(int(pm.playbackOptions(min=True, q=True)), int(pm.playbackOptions(max=True, q=True) + 1)):
            pm.setCurrentTime(i)
            self.match(system_type=system_type, ik_fk_blend_value=ik_fk_blend_value)
        pm.setCurrentTime(current_frame)
        pm.cutKey(pm.PyNode(blend_node), attribute='ik_fk_blend', cl=True)
        pm.PyNode(blend_node).setAttr('ik_fk_blend', ik_fk_blend_value)
    else:
        self.match(system_type=system_type, ik_fk_blend_value=ik_fk_blend_value)
        pm.PyNode(blend_node).setAttr('ik_fk_blend', ik_fk_blend_value)
        pm.setKeyframe(pm.PyNode(blend_node), at='ik_fk_blend', hierarchy='none')

    # TO FK
    if ik_fk_blend_value == 0:
        if pm.attributeQuery('CTRL_end', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_top')),
                                  str(pm.getAttr(system_node + '.CTRL_mid')),
                                  str(pm.getAttr(system_node + '.CTRL_end'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_end'), r=True)
        elif pm.attributeQuery('CTRL_FK_End', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_FK_Start')),
                                  str(pm.getAttr(system_node + '.CTRL_FK_Pole')),
                                  str(pm.getAttr(system_node + '.CTRL_FK_End'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_FK_End'), r=True)

    # TO IK
    elif ik_fk_blend_value == 1:
        if pm.attributeQuery('CTRL_Ik', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_Ik')),
                                  str(pm.getAttr(system_node + '.CTRL_IkPv'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_Ik'), r=True)
        elif pm.attributeQuery('CTRL_Wrist', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_Pole')),
                                  str(pm.getAttr(system_node + '.CTRL_Wrist'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_Wrist'), r=True)
        elif pm.attributeQuery('CTRL_Foot', n=system_node, exists=True):
            controls_to_filter = [str(pm.getAttr(system_node + '.CTRL_Pole')),
                                  str(pm.getAttr(system_node + '.CTRL_Foot'))]
            self.euler_filter(controls_to_filter=controls_to_filter)
            pm.select(pm.getAttr(system_node + '.CTRL_Foot'), r=True)

    pm.refresh(su=False)
```

KNOW YOUR TEAM

- Have simple, specific functions for easy hotkey and marking menu mapping



Project Scope and Schedule

- Rig needs?
- What platform?
- Tech?
- How detailed?

CHECKLIST

? How many characer types

? Mobile/VR/Console

? Cutscenes

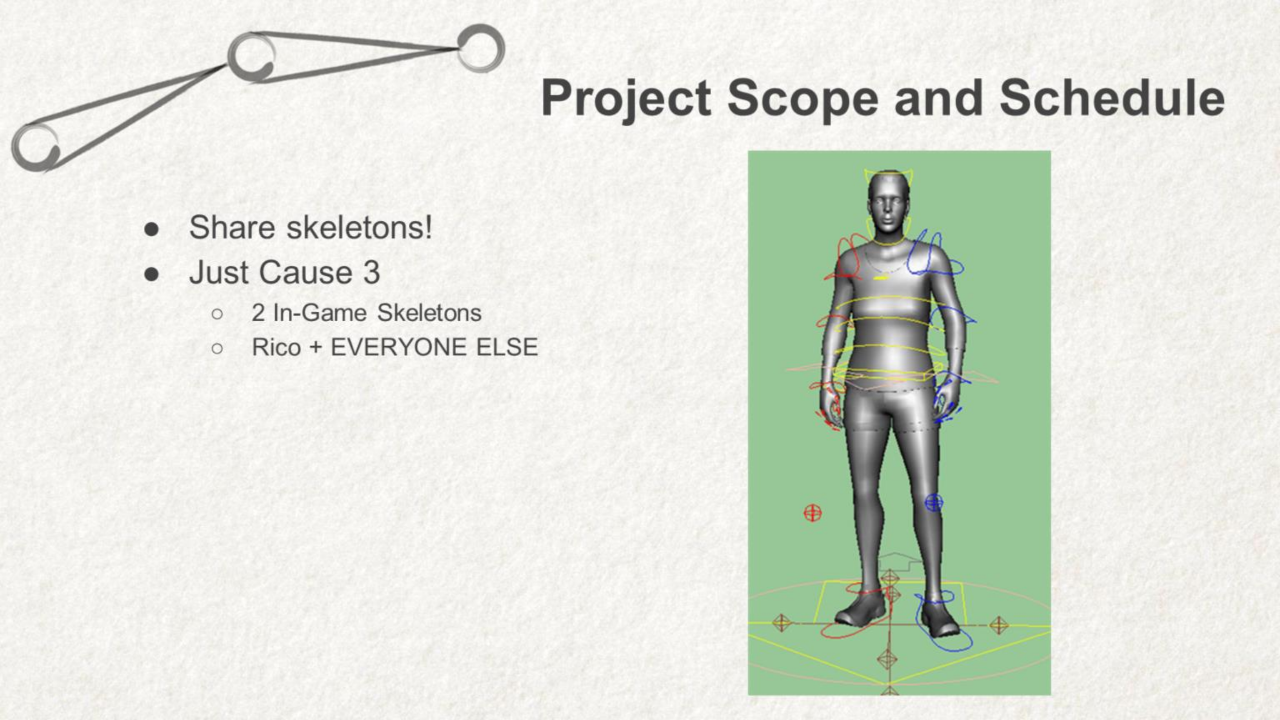
? Joint Count

? How do characters move

KNOW YOUR PROJECT SCOPE AND SCHEDULE

- You'll most certainly have specific animator requests for rig needs that you'll have to prioritize
- Past that, are you working on VR? Mobile? PC/Console? High end, pre-rendered cinematics?
- What kind of tech does your engine have?
  - Do you have run-time capabilities?
- Being aware of things
  - As simple as Joint counts
  - Or as advanced as being able to set up a full blown run-time rig
  - Will allow you to figure out where and what your needs are
  - And you, your team, AND your producers will be happy because you can better plan out your schedule based off of your estimates
- One huge tip to save time on your scope and

schedule...

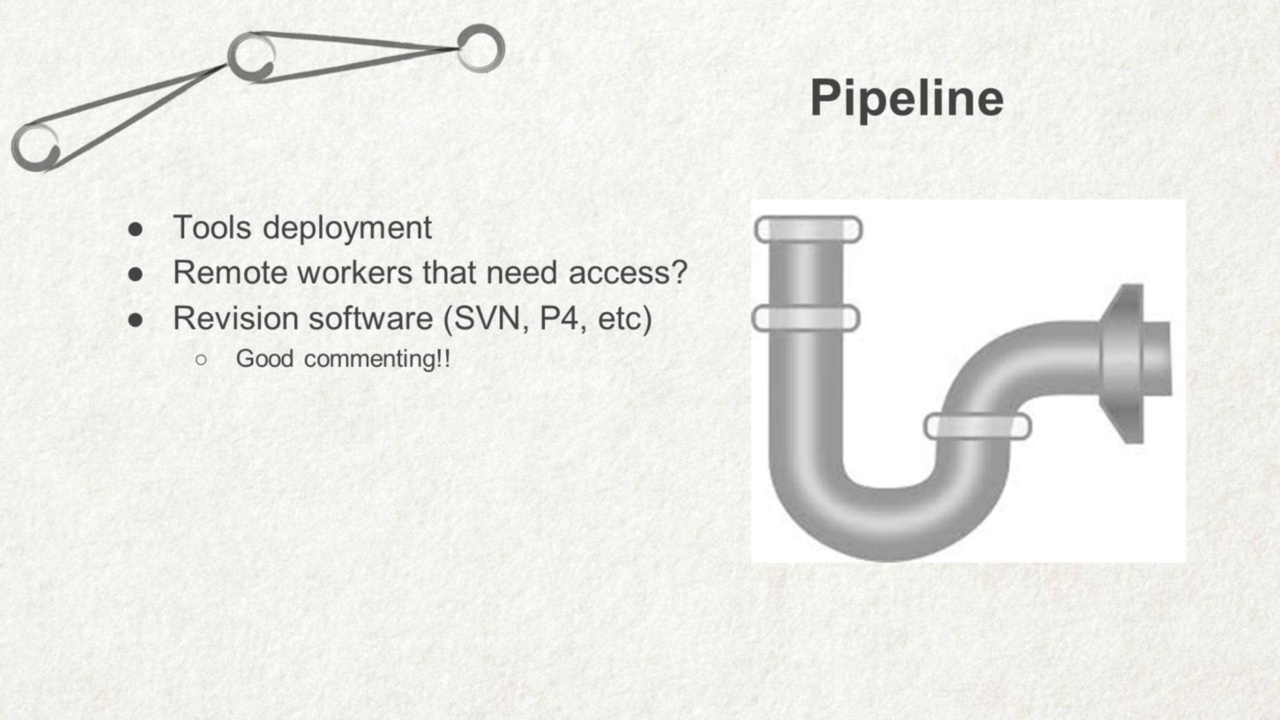


Project Scope and Schedule

- Share skeletons!
- Just Cause 3
  - 2 In-Game Skeletons
  - Rico + EVERYONE ELSE

KNOW YOUR PROJECT SCOPE AND SCHEDULE

- Share skeletons!
- On JC3 we had 2 skeletons…
  - One was Rico
  - One was everyone else
    - Female world sim
    - Male work sim
    - Enemies
    - Rebels and so on
- Sharing animation was a breeze
- And it cut down on amount of content as well as time spent on animating

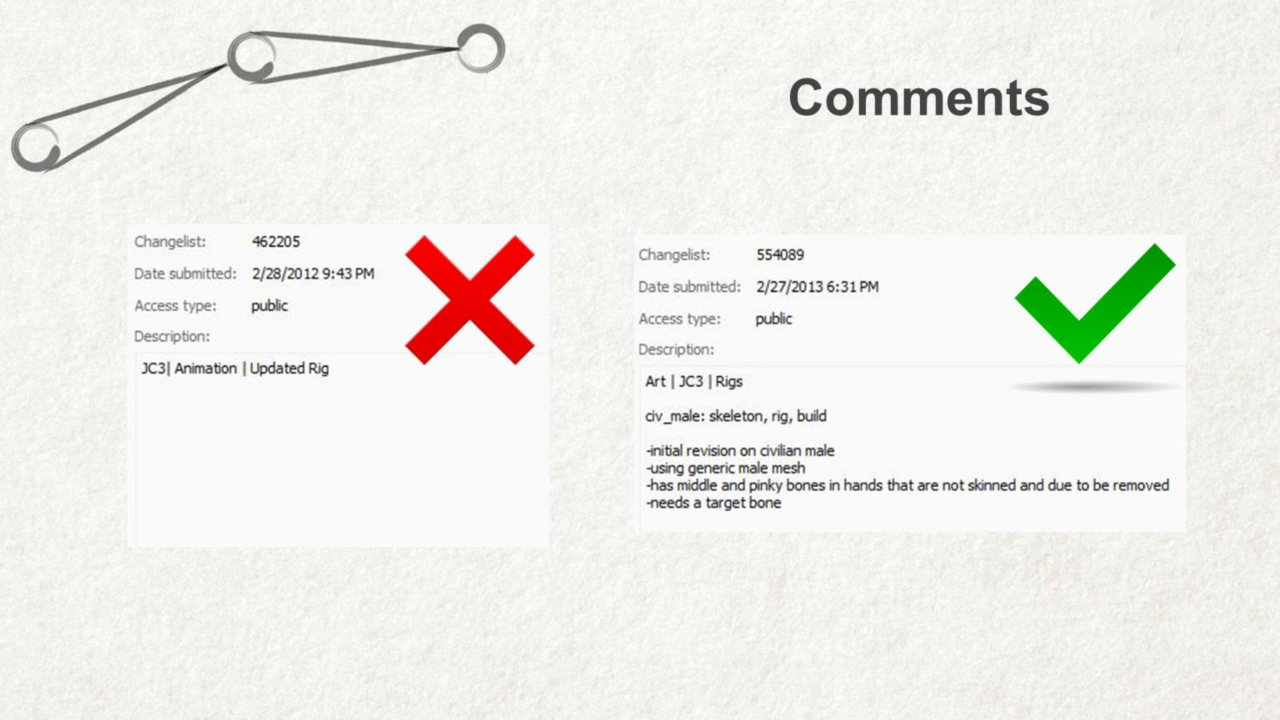


Pipeline

- Tools deployment
- Remote workers that need access?
- Revision software (SVN, P4, etc)
    - Good commenting!!

KNOW YOUR PIPELINE

- How are you going to deploy tool/rig updates to your team?
- Do you have remote workers that will need rigs and tools?
- You need to make sure you have a fast and efficient way to deliver your updates and something that works for all members of your animation team, in-house or not
- Revision software is an excellent way to push updates, and most studios will hopefully already be using use some form
    - One thing I need to stress to you...please write good comments
    - NEXT SLIDE



Comments

Changelist: 462205
Date submitted: 2/28/2012 9:43 PM
Access type: public
Description:
JC3| Animation | Updated Rig

✗

Changelist: 554089
Date submitted: 2/27/2013 6:31 PM
Access type: public
Description:
Art | JC3 | Rigs

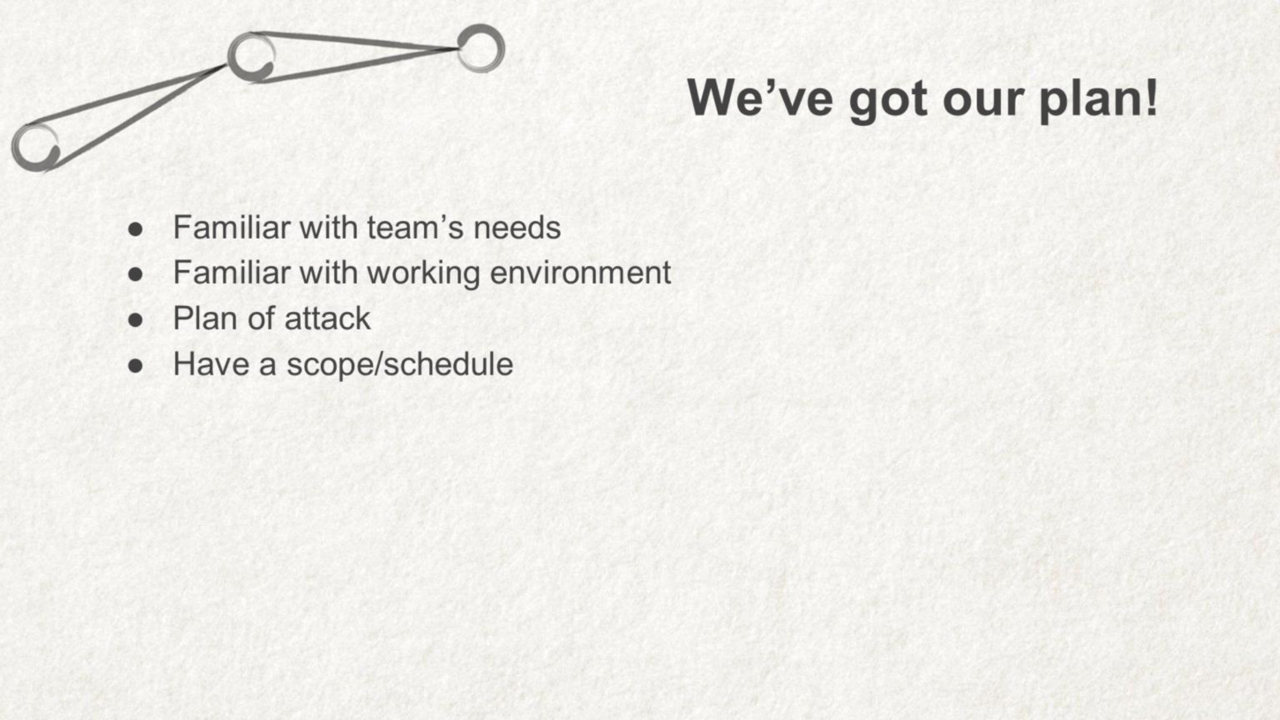
civ_male: skeleton, rig, build

-initial revision on civilian male
-using generic male mesh
-has middle and pinky bones in hands that are not skinned and due to be removed
-needs a target bone

✓

KNOW YOUR PIPELINE

- If you say “updated rig” it’s so redundant and vague... you’re submitting a file, of course you’re updating something
- This is sooo frustrating because it is such an easy thing to do and takes literal seconds to do
- It takes all of 30 seconds to write out a descriptive comment
- You’ll thank yourself later when you have to go back and figure out what the heck you did!

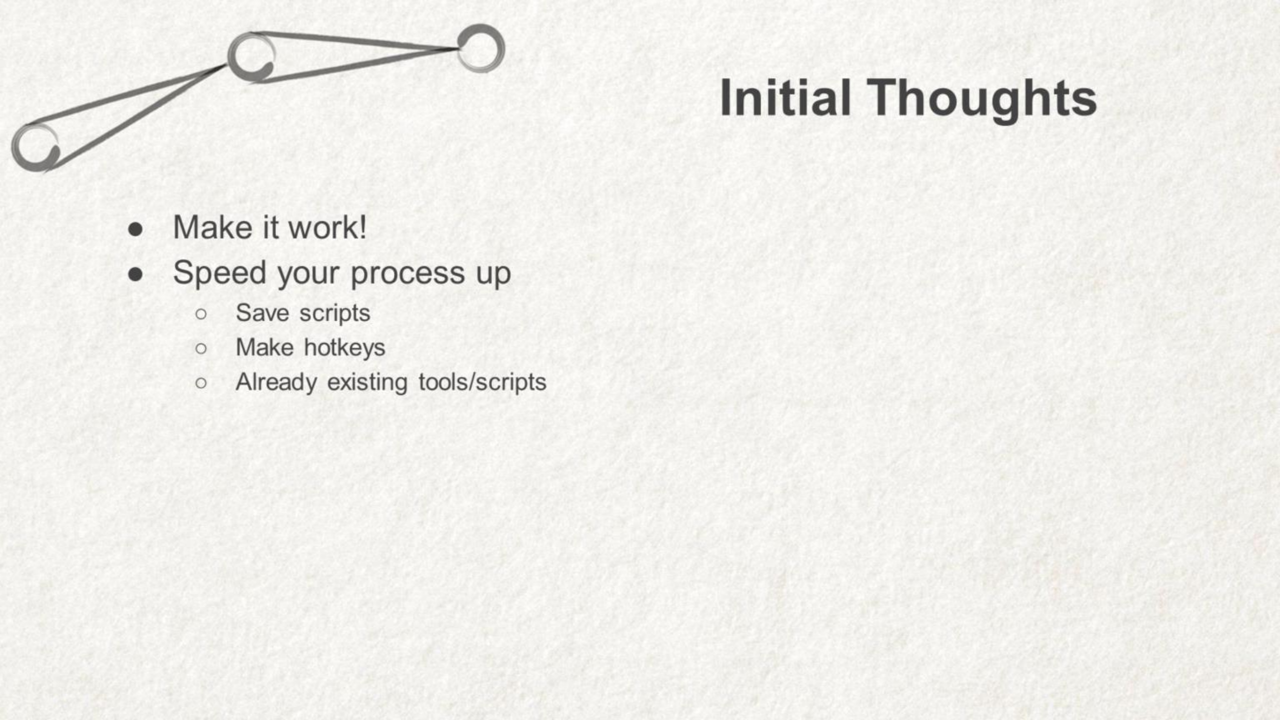


We’ve got our plan!

- Familiar with team’s needs
- Familiar with working environment
- Plan of attack
- Have a scope/schedule

WE’VE GOT OUR PLAN

- You’re familiar with your teams needs
- You’re familiar with all aspects of your working environment
- You have a plan of attack moving forward
- You’ve scoped and scheduled



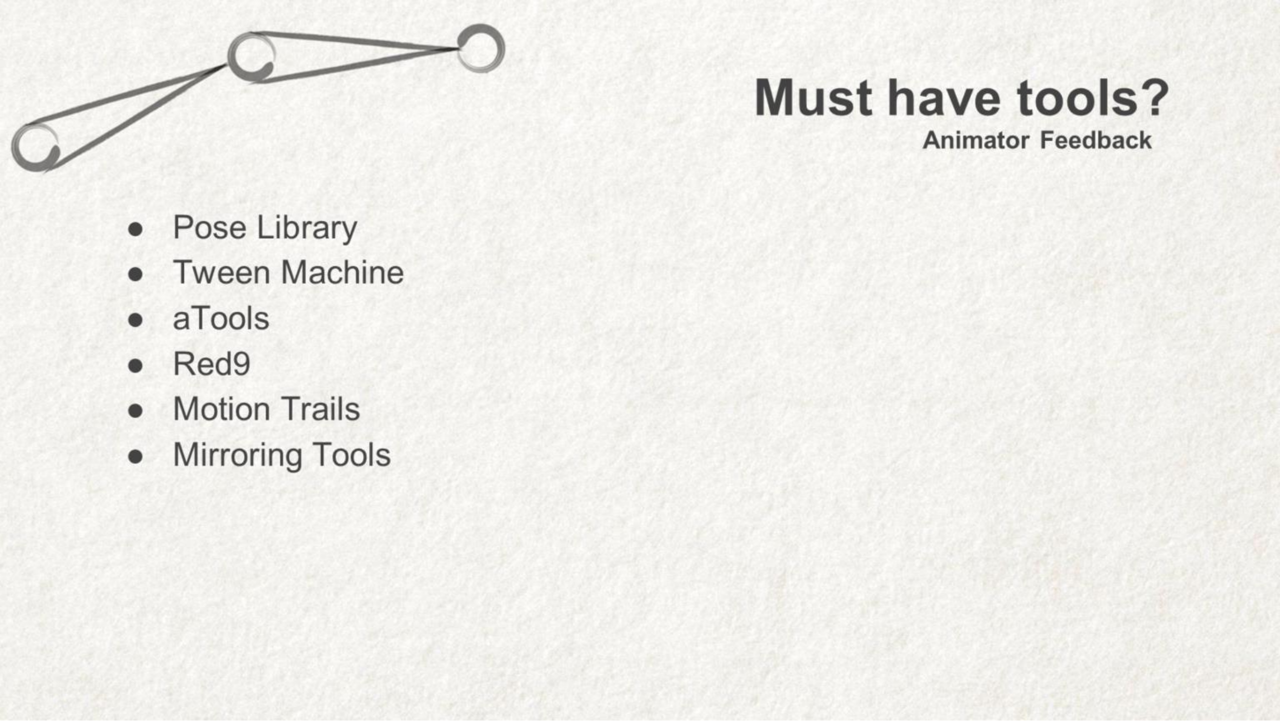
Initial Thoughts

- Make it work!
- Speed your process up
  - Save scripts
  - Make hotkeys
  - Already existing tools/scripts

INITIAL THOUGHTS

- A few overall pointers before we really get started with the hands on work
- Make your rig work before you worry about all the extras
- build your base, make it work well, get animators up and running ASAP, then build on top of that base.
  - This is why I love using Maya, referencing is such a powerful tool for rigs
- You can help yourself by speeding up your process along the way.
- Scripts and snippets you make now to make your process easier will help you again and again...keep them safe and save them!
- Use hotkeys for things you do often
  - I personally do this for copy/paste vert weights for my skinning phase

- Also, don’t rewrite tools that already exist.
    - Plenty of tried and true tools already available
- Speaking of which...



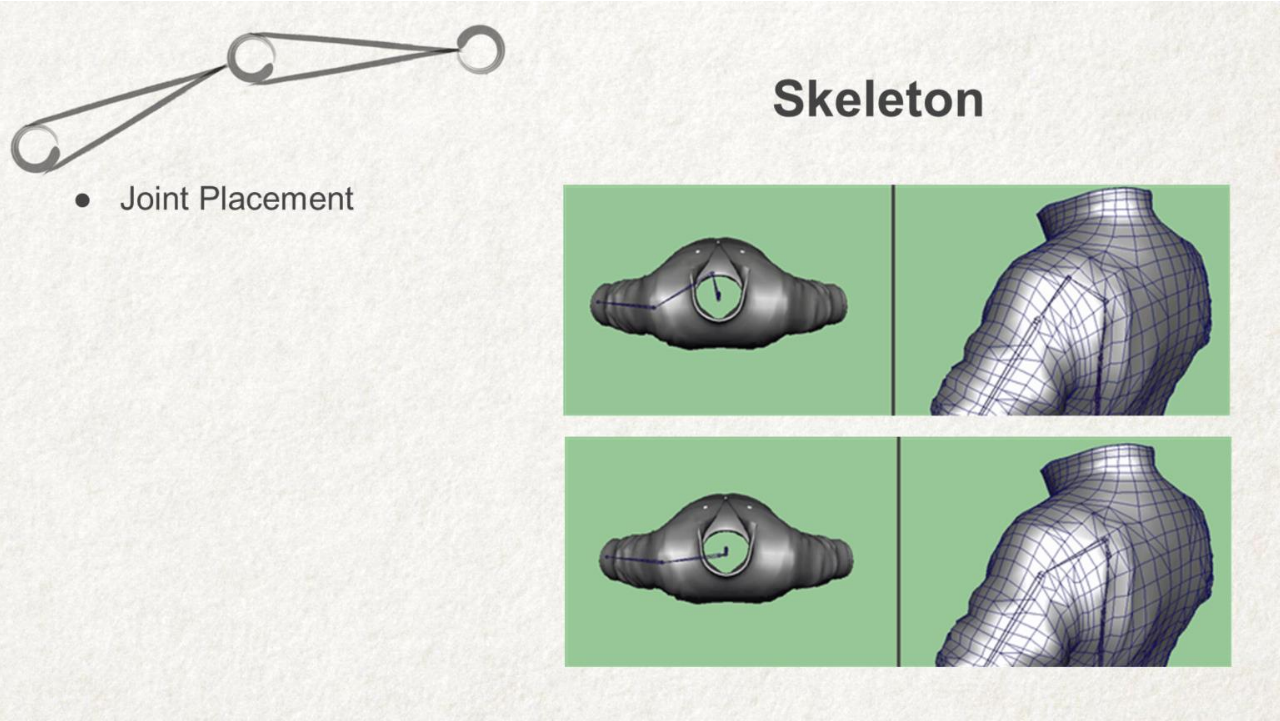
Must have tools?

Animator Feedback

- Pose Library
- Tween Machine
- aTools
- Red9
- Motion Trails
- Mirroring Tools

ANIMATOR FEEDBACK: What tools/scripts can you not work without

- It’s time for some animator feedback!
- This time for the question “What are your must have tools?”
- All of these can be found for free either online or are already built into Maya
- We actually have animators using every single one of these specific things as well at avalanche
- The nice thing about integrating these tools into your pipeline is that animators kept happy because they’re using tools that they’ve probably been using already for some time
- And YOU are kept happy because these are just a few things that you don’t have to write yourself!



Skeleton

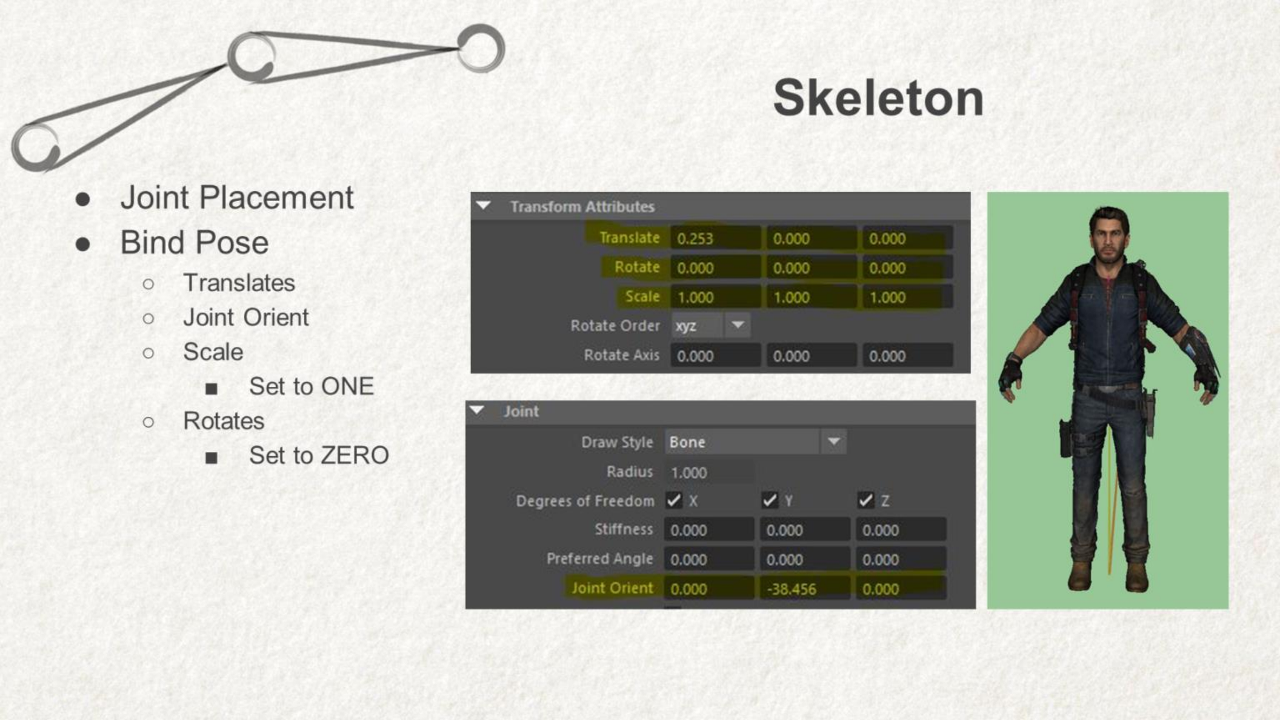
- Joint Placement

THE SKELETON

- The first thing you have to do is create your joints
  - And where you place your joints really matters!
  - Where and how to place your joints will be different...
    - Dependant on your asset’s shape and size
    - And more so if multiple meshes will be sharing the same skeleton
    - Joint placement may need a few passes as you start testing out character deformations
      - So be prepared for a little bit of tedious back and forth with that
- This is Rico’s jacket from JC3, on first try I placed his clavicle in a more realistic pivot position, which you

can definitely see stretches the shoulder in a way that kind of translates the mesh out when simply rotating

- I corrected this by going back and placing that joint much further back, and just from that placement change alone, his shoulder was looking muuuch better without the need for any additional joints



Skeleton

- Joint Placement
- Bind Pose
  - Translates
  - Joint Orient
  - Scale
    - Set to ONE
  - Rotates
    - Set to ZERO

Transform Attributes

Translate	0.253	0.000	0.000
Rotate	0.000	0.000	0.000
Scale	1.000	1.000	1.000
Rotate Order	xyz		
Rotate Axis	0.000	0.000	0.000

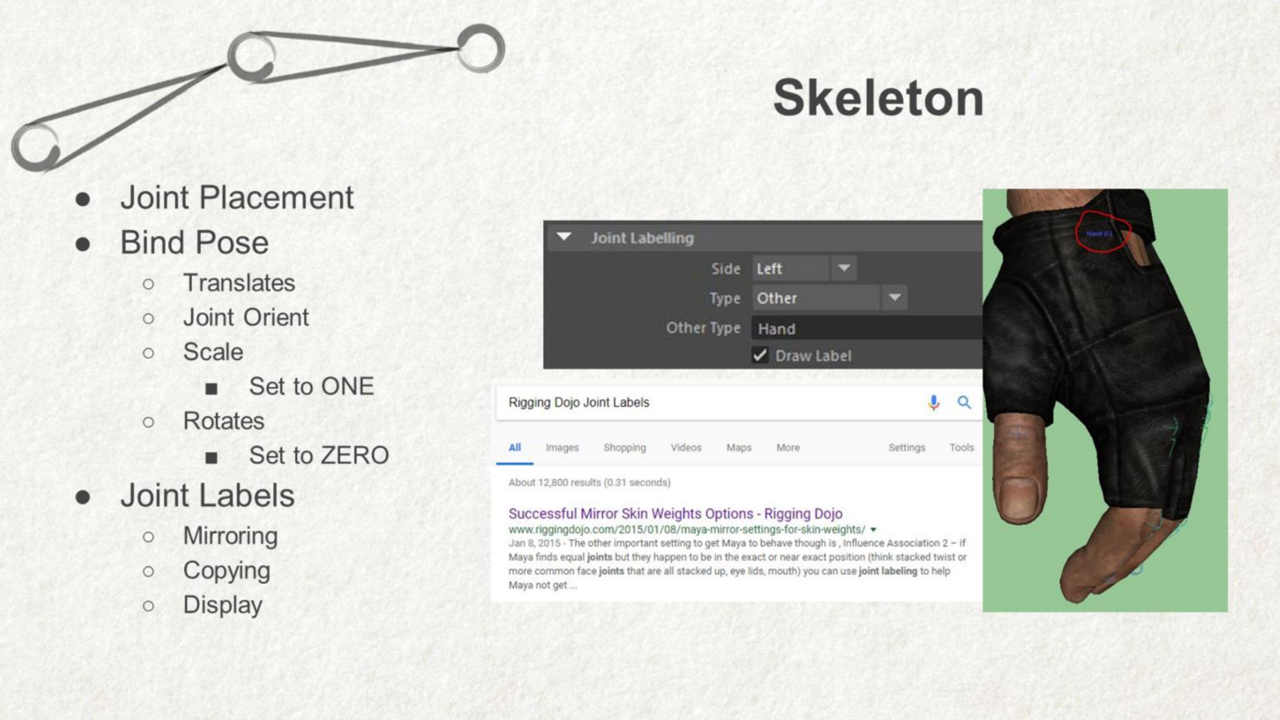
Joint

Draw Style	Bone		
Radius	1.000		
Degrees of Freedom	☑ X	☑ Y	☑ Z
Stiffness	0.000	0.000	0.000
Preferred Angle	0.000	0.000	0.000
Joint Orient	0.000	-38.456	0.000

THE SKELETON

- Your bind pose, also referred to as an a-pose
  - Is the very base of your rig, and if this is set up wrong, you’re already starting off on the wrong foot
- The things to pay most attention to here are to make sure you
  - Only have unique data on your Translates and Joint Orients
  - Scale is set at 1
  - Rotate is ZERO across the board
  - If you’re working on something that uses IK like a limb
    - Your middle joint should only have translate data on ONE translate channel

- And orientation on ONE joint orient channel
- This keeps your limb nice and flat for IK solvers later



Skeleton

- Joint Placement
- Bind Pose
  - Translates
  - Joint Orient
  - Scale
    - Set to ONE
  - Rotates
    - Set to ZERO
- Joint Labels
  - Mirroring
  - Copying
  - Display

Joint Labelling

Side: Left

Type: Other

Other Type: Hand

☑ Draw Label

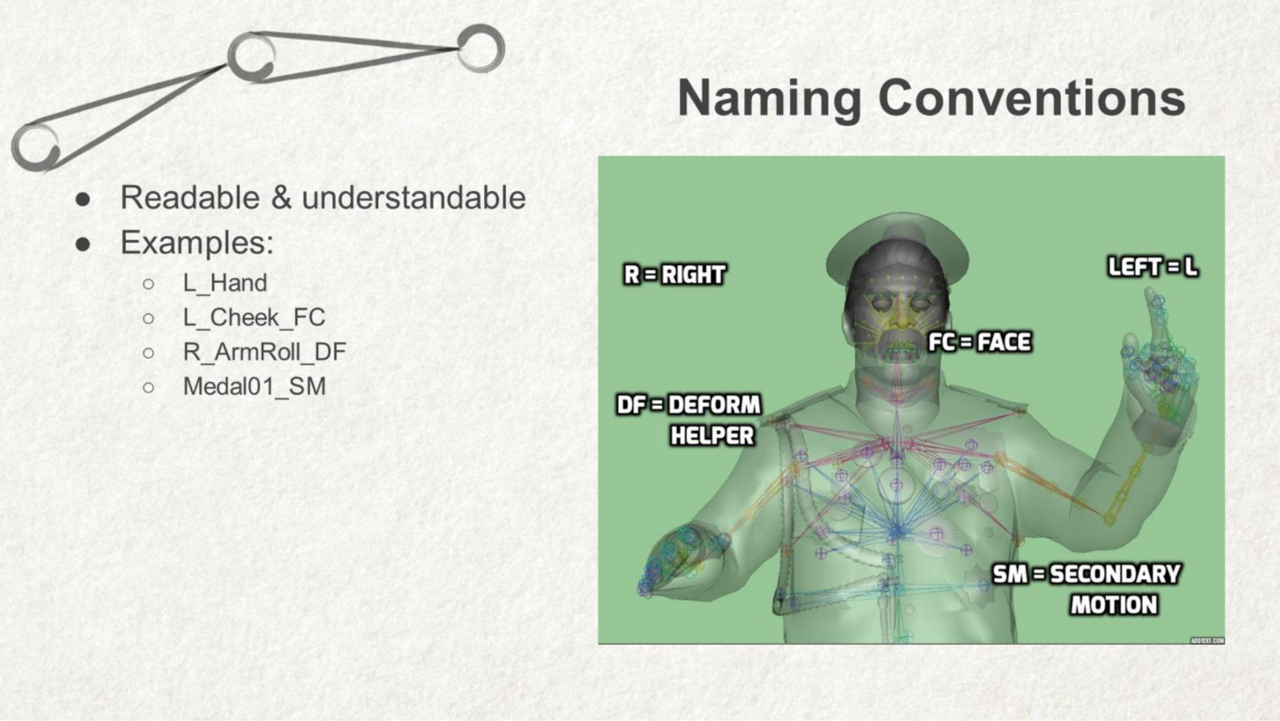
Rigging Dojo Joint Labels

About 12,800 results (0.31 seconds)

Successful Mirror Skin Weights Options - Rigging Dojo
www.riggingdojo.com/2015/01/08/maya-mirror-settings-for-skin-weights/
Jan 8, 2015 - The other important setting to get Maya to behave though is , Influence Association 2 – if Maya finds equal **joints** but they happen to be in the exact or near exact position (think stacked twist or more common face **joints** that are all stacked up, eye lids, mouth) you can use **joint labeling** to help Maya not get ...

THE SKELETON

- Finally, you want to label your joints
  - Not only does it add a nice layer of organization but, it gives you the best results for mirroring and copying skin weights
  - You can also draw label for a nice visual overlay of your joint names
  - For my own labels I use left/right/center for side, other for type, and then fill in the exact joint name sans side…
  - For a slightly more expanded explanation on this just google “rigging dojo joint labels” and your first hit should be for Rigging Dojo’s blog post titled “successful mirror skin weight options”

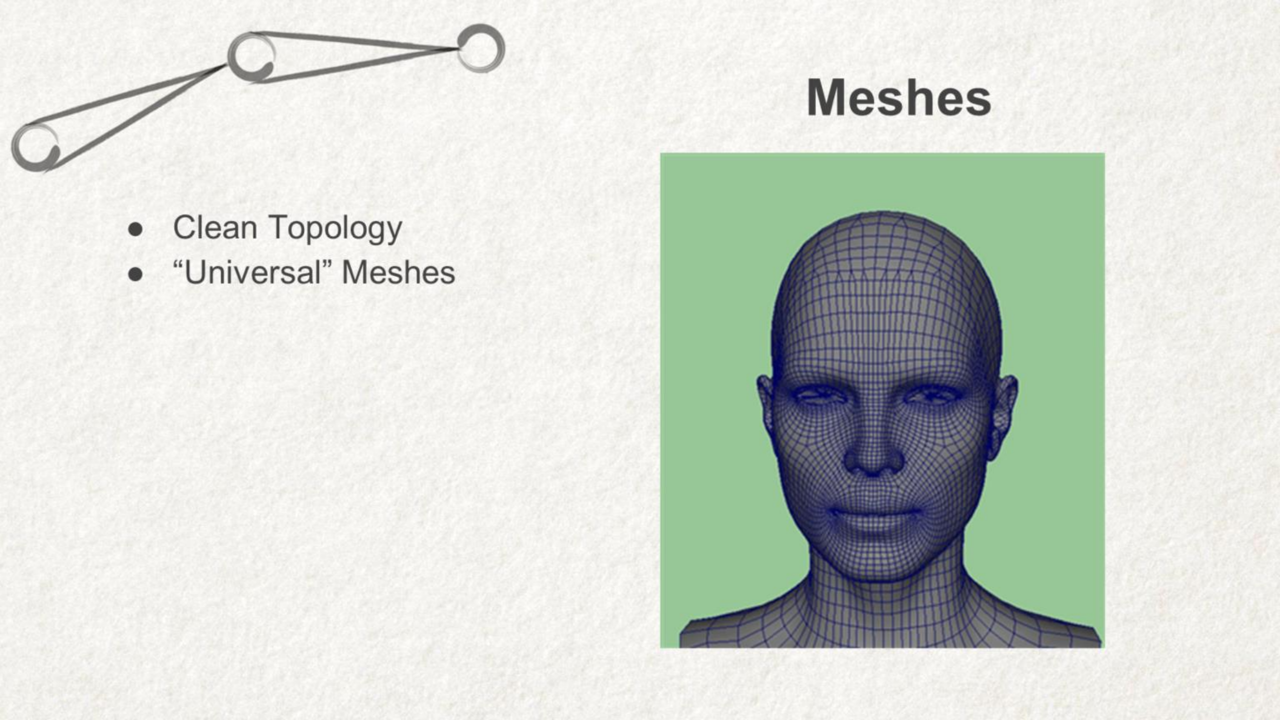


Naming Conventions

- Readable & understandable
- Examples:
  - L_Hand
  - L_Cheek_FC
  - R_ArmRoll_DF
  - Medal01_SM

NAMING CONVENTIONS

- You should be using good naming conventions for everything, starting with the skeleton
  - And they need to be readable and easily understood by anyone
- A few personal tags I like to use are
  - Of course your typical L and R for left and right
  - And I like to tag face joints, deformation helpers, and secondary motion joints with unique 2 letter tags
  - This allows me to easily select all joints with the FC tag in order to return all face joints and nothing else



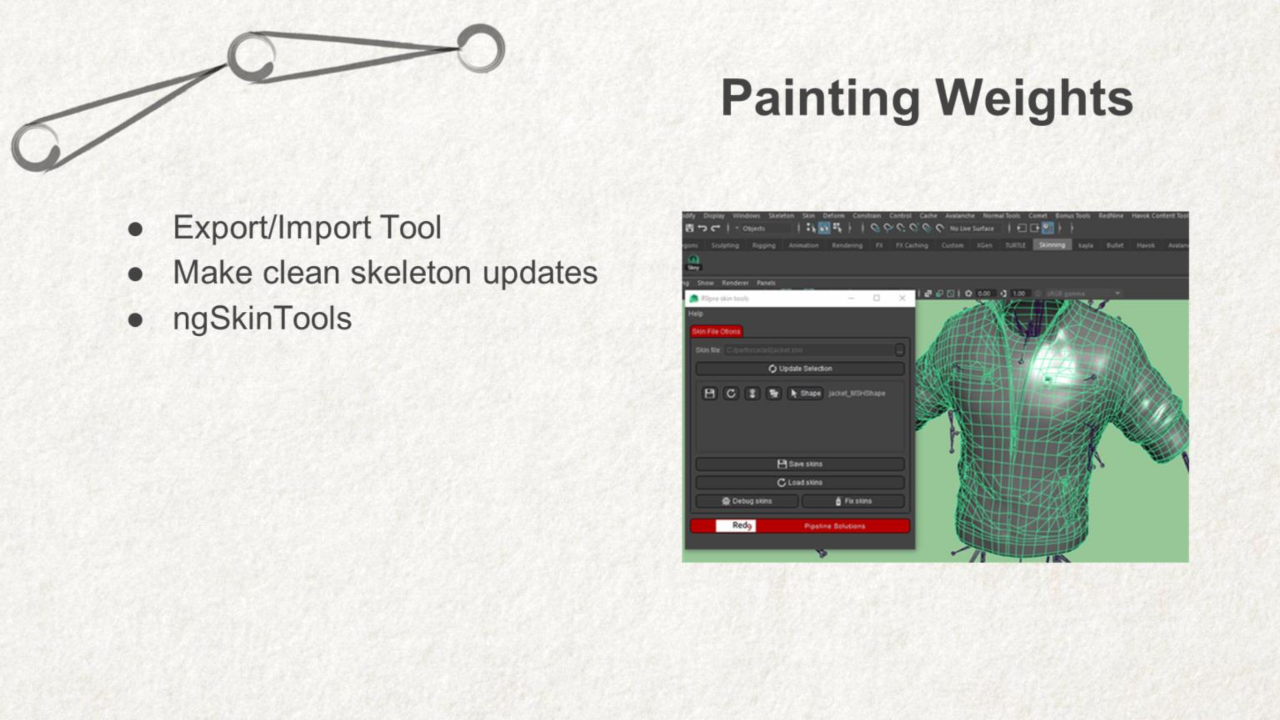
Meshes

- Clean Topology
- “Universal” Meshes

MESHES

- First and foremost, make sure you get clean topology from your artists
- Working with messy poly cuts and edges is not fun when skinning so if you run into a problem area, let your artist know. In my experience, when I bring something up to an artist they’re more than happy to make a fix for you and simply aren’t aware that a particular geo cut or shape isn’t deformation friendly, so let them know
- Also, make sure your artists are sharing meshes as often as possible.
- Using a “universal head mesh” is essential to save lots of skinning and setup time
    - What you see now are some of our JC3 main characters who all look comepletely different but use the same exact topology

  - Skin one, export the weights out and import them to the next head
  - Past this only small tweaks around the eyelids and lips were actually necessary!
- You can also apply this concept to things like hands, arms, legs
  - Pretty much anything with a similar surface type



Painting Weights

- Export/Import Tool
- Make clean skeleton updates
- ngSkinTools

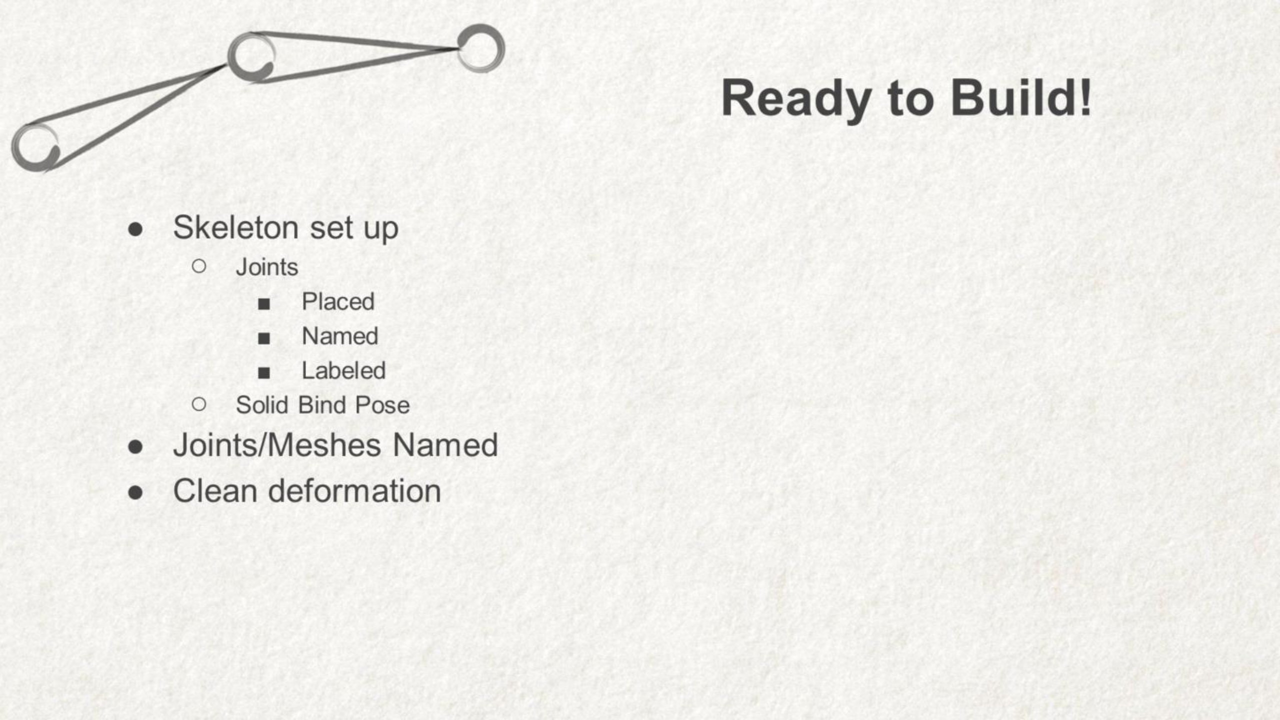
PAINTING WEIGHTS (Everyone’s favorite)

- Honestly there’s not a ton I can tell you here since there are a few different approaches to skinning and as long as the end result looks good it doesn’t really matter the approach you take
- My advice though, is to make sure you have a good Export/Import tool.
  - This allows you to easily make skeleton updates and retain your weights
  - It’s also a good failsafe if something goes wrong in your file
  - I happen to use the Red9 pro skin transfer tool because it’s so robust
  - But there are free tools out there that still work well if you don’t have time to write your own
- What I mean by make clean skeleton updates
  - Exporting weights, fully unbinding all meshes,

updating the skeleton, and importing your weights back in after a rebind ensures that your bind pose is exactly as you expect

    - I definitely advise against any of the “add influence” type of features in the software in order to assure that your bind pose is the same exact pose that you actually bind at

- There’s a skinning plugin called ngSkinTools...look it up, get it, love it...skinning will never be the same again

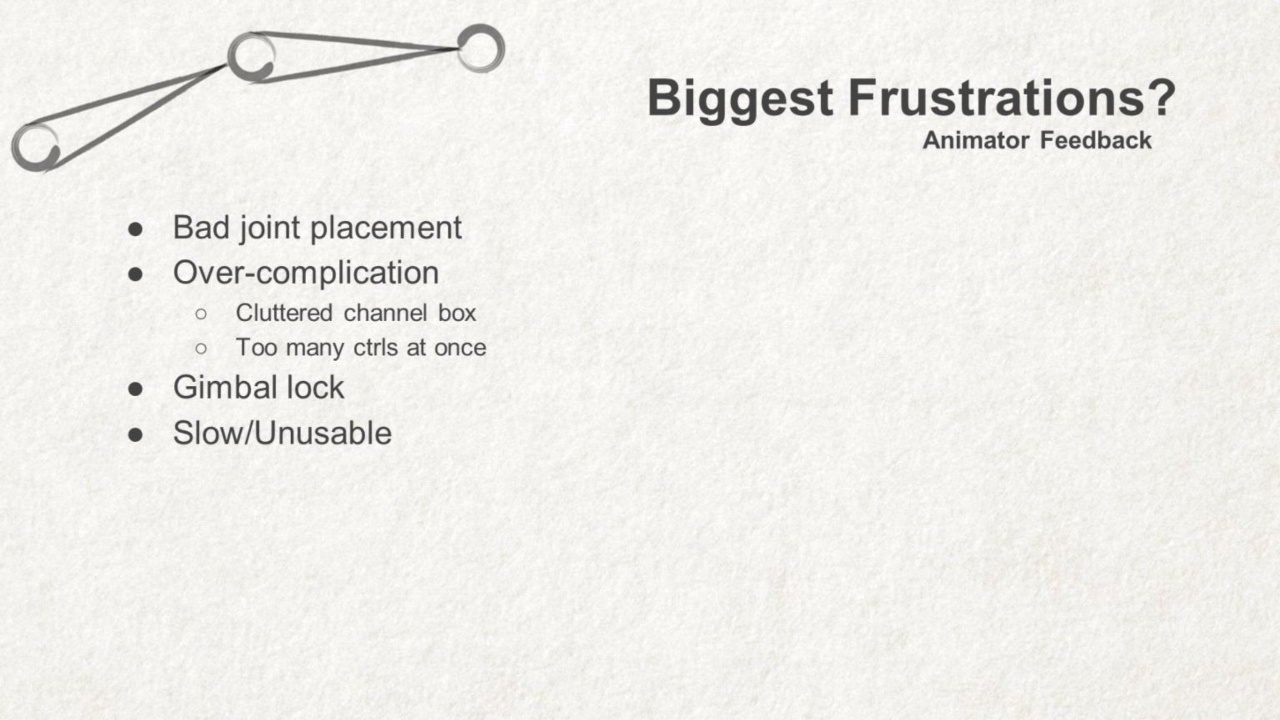


Ready to Build!

- Skeleton set up
    - Joints
        - Placed
        - Named
        - Labeled
    - Solid Bind Pose
- Joints/Meshes Named
- Clean deformation

WE’RE READY FOR BUILDING

- Your skeleton is set up
    - Joints are placed, named, and labeled
    - You have a solid bind pose as your base to build on
- Everything is named nicely
- You have clean and good looking deformations



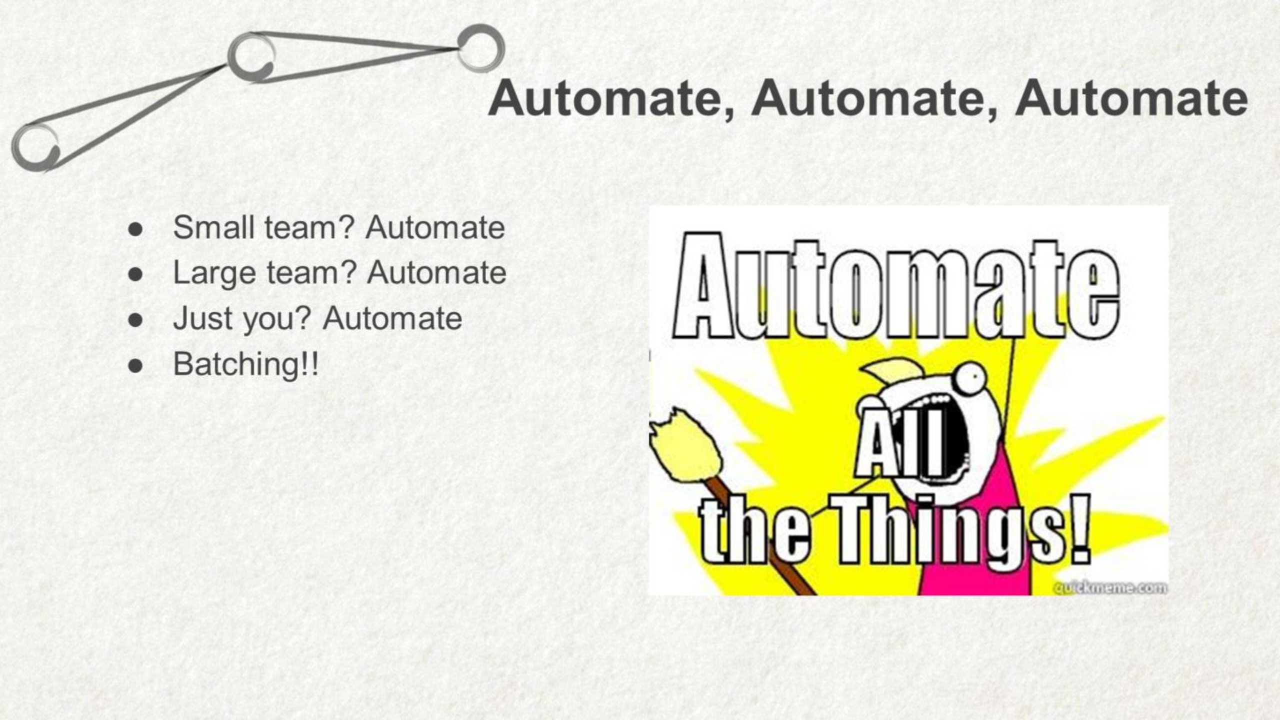
Biggest Frustrations?

Animator Feedback

- Bad joint placement
- Over-complication
    - Cluttered channel box
    - Too many ctrls at once
- Gimbal lock
- Slow/Unusable

MOST FRUSTRATING THINGS IN RIGS

- This seems like a good time to see what animators find frustrating in rigs
- Something we just talked about not long ago! Bad joint placement
- Over complication with a cluttered channel box or too many things to look at all at once
    - This causes a rig to get overwhelming making it hard to focus or know where to begin at times
- Gimbal lock...we can do our best to avoid this with good rotation orders...and guess what, more about this very soon
- Lastly, too many features make a rig slow and unusable...and hey, more on this in a bit too!



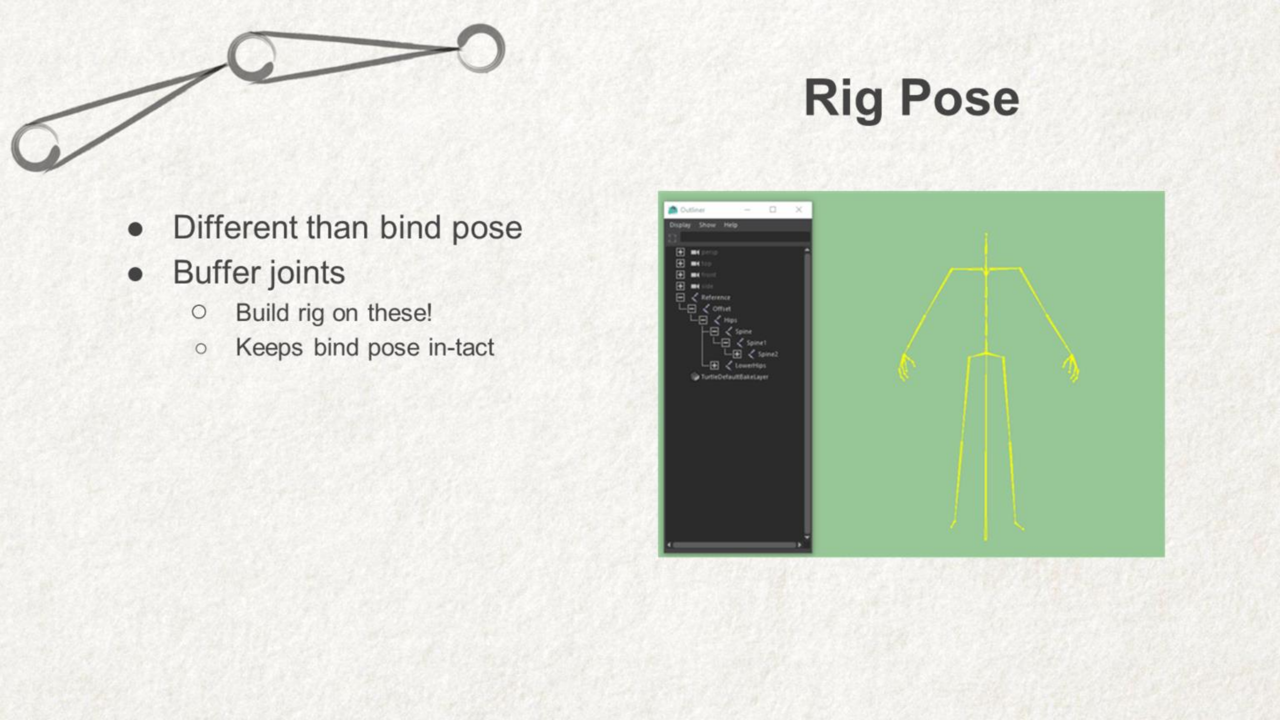
Automate, Automate, Automate

- Small team? Automate
- Large team? Automate
- Just you? Automate
- Batching!!

AUTOMATE

- If you take nothing else away but one thing from me...please make sure that it is ALWAYS AUTOMATE
- Doesn’t matter if you’re a small or large group working on a project
  - For starters, building an auto-rigger and having the ability to make quick and fast updates/changes to any aspect of your rig is absolutely essential
- Allow you to add/remove joints and control rig features very quickly
- Better prepared for character design changes as well when your AD decides to completely change the way your main character looks!
  - You make yourself and your rigs adaptable to all these sudden changes

- Make sure you also have a process to batch things
- If you you make a rig update
  - Easily apply that update to all rigs very quickly
- This also expands past just rigging…
  - Batching is super useful for Exporting, Transferring, and Retargeting animation as well



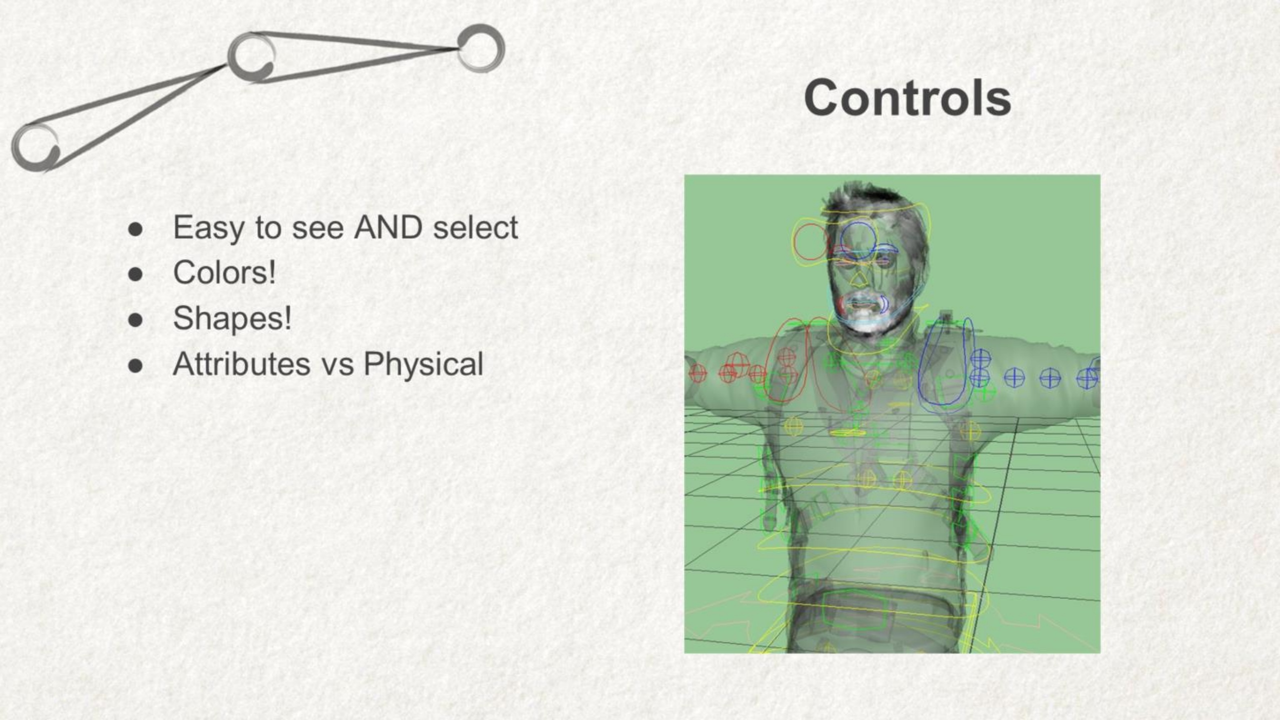
Rig Pose

- Different than bind pose
- Buffer joints
  - Build rig on these!
  - Keeps bind pose in-tact

RIG POSE

- Personally, I like to use the traditional T-Pose as a base when building a rig
  - This is different than the skeleton bind pose (visually show A vs T pose)
  - And it allows me to cleanly maintain that bind pose
  - It also really helps me create much more stable IK arms and legs since they are nice, flat, and parallel with the world
  - Since there is a need for HumanIK rigs at Avalanche, I also need to characterize my skeletons in order to build this, and having all rigs T-posed allows me to do that
- I achieve the T-pose through buffer joints
  - By duplicating the skeleton and adding the “_BUF” suffix to each joint

    - constraining the bind skeleton to it
    - And tposing
    - Now I can manipulate the buffer joints, build controls on top of them, and my bind skeleton follows and remains clean!

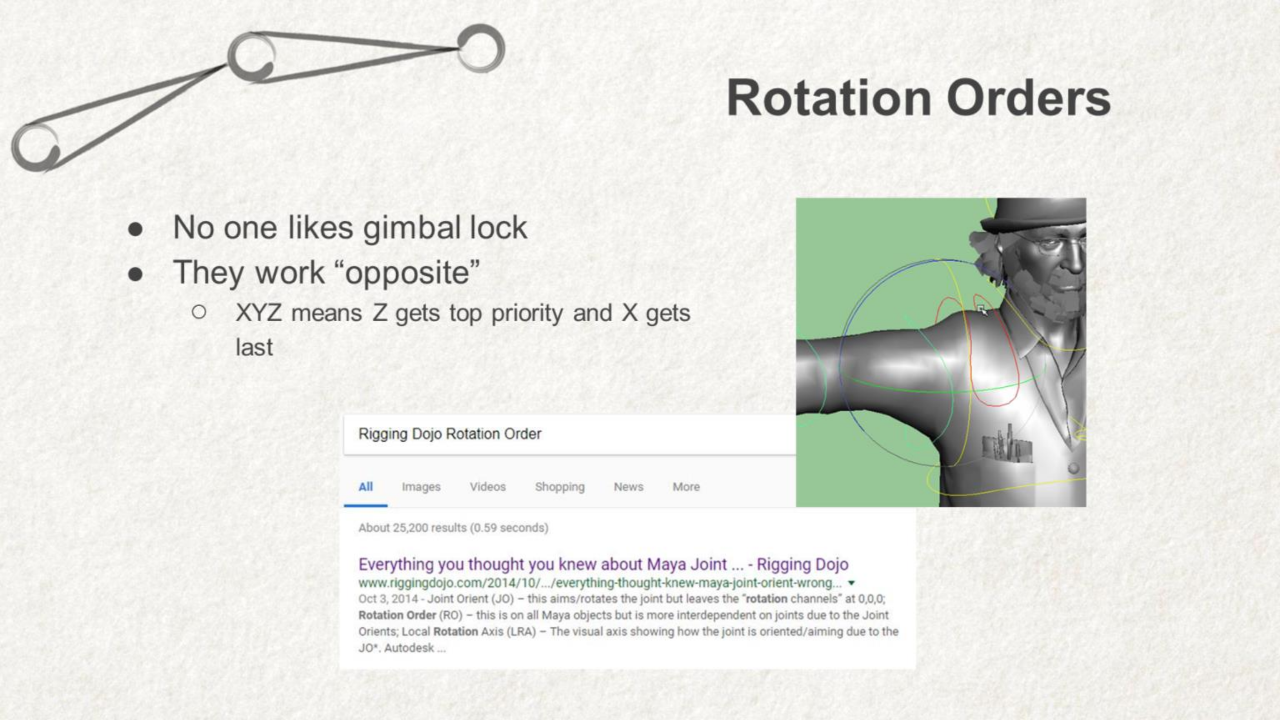


Controls

- Easy to see AND select
- Colors!
- Shapes!
- Attributes vs Physical

CONTROLS

- Controls need to be as animator friendly as possible
  - This means their functions should be easy understood
- This can be achieved through colors and shapes
- There’s also a never-ending debate between animators when it comes to using attributes or physical controls, most notably in foot controls and facial rigs...make sure you know what your team prefers



Rotation Orders

- No one likes gimbal lock
- They work “opposite”
  - XYZ means Z gets top priority and X gets last

Rigging Dojo Rotation Order

About 25,200 results (0.59 seconds)

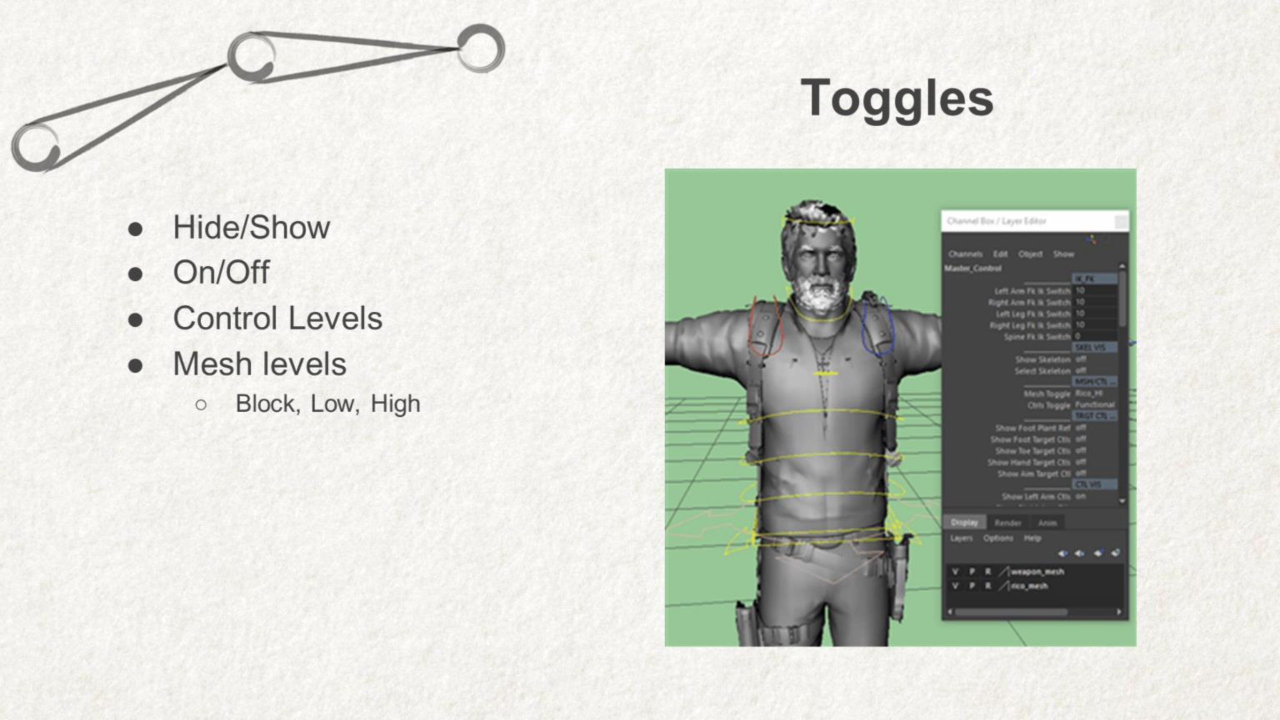
Everything you thought you knew about Maya Joint ... - Rigging Dojo
www.riggingdojo.com/2014/10/.../everything-thought-knew-maya-joint-orient-wrong... ▾
Oct 3, 2014 - Joint Orient (JO) – this aims/rotates the joint but leaves the “**rotation** channels” at 0,0,0; **Rotation Order** (RO) – this is on all Maya objects but is more interdependent on joints due to the Joint Orients; Local **Rotation** Axis (LRA) – The visual axis showing how the joint is oriented/aiming due to the JO*. Autodesk ...

ROTATION ORDERS

- Having good rotation orders for your rig will make your animators extra happy and your rig will be that much more usable
- When I try and figure out rotation orders I usually send a rig off to the animation team for their personal feedback
- Even better if you can create a rig that allows animators to switch their RO much like a space switch
- At Avalanche I need to keep my joint ROs at XYZ because the exported data in game only works with that
  - BUT the controls that drive those joints are rarely in XYZ

- Rigging Dojo has another great write up all about joint orients and rotation orders that you should check out

- ○ Google search “Rigging Dojo Rotation Order”



Toggles

- Hide/Show
- On/Off
- Control Levels
- Mesh levels
  - Block, Low, High

Master_Control

	IK_FK
Left Arm Fk Ik Switch	10
Right Arm Fk Ik Switch	10
Left Leg Fk Ik Switch	10
Right Leg Fk Ik Switch	10
Spine Fk Ik Switch	0
	SKEL VIS
Show Skeleton	off
Select Skeleton	off
	MSH/CTL ...
Mesh Toggle	Rico_HI
Ctrls Toggle	Functional
	TRGT CTL ...
Show Foot Plant Ref	off
Show Foot Target Ctls	off
Show Toe Target Ctls	off
Show Hand Target Ctls	off
Show Aim Target Ctl	off
	CTL VIS
Show Left Arm Ctls	on

V P R weapon_mesh
V P R rico_mesh

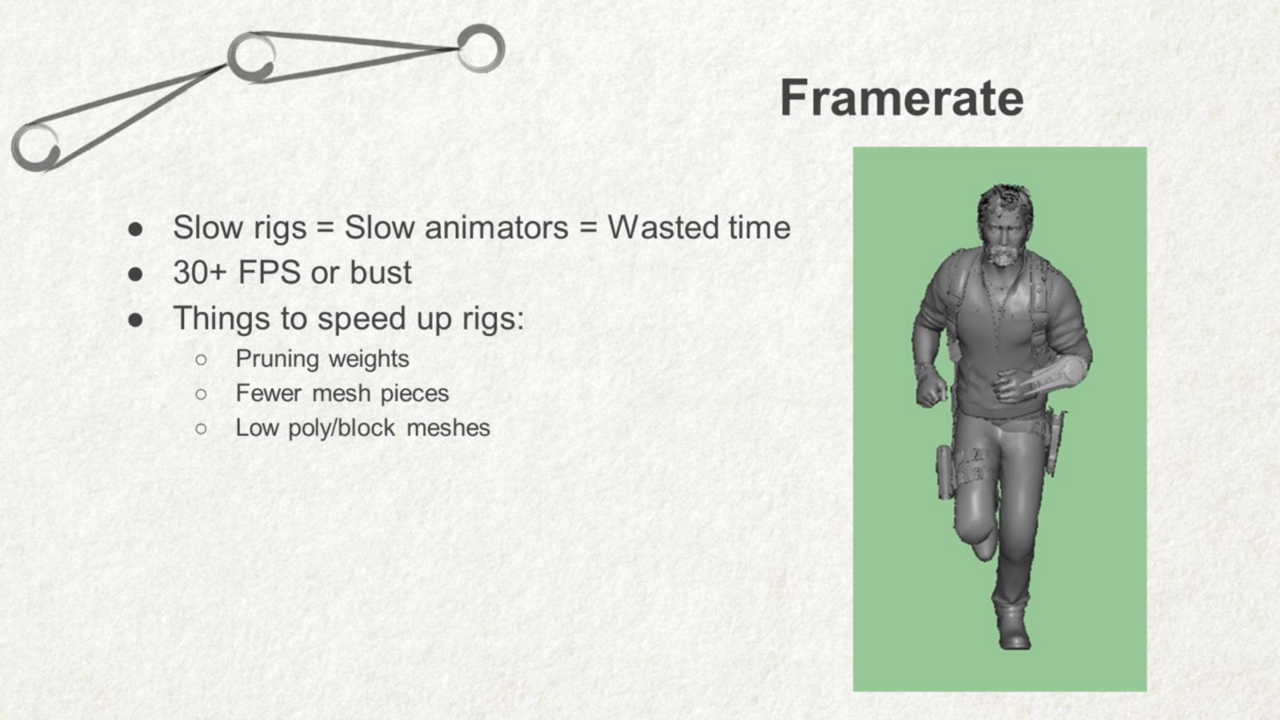
TOGGLES

- Toggles are nice to have for animators to hide/show things or turn features on and off
  - A simple setAttr command can be also applied to hotkeys for an even faster workflow
- Allowing animators to isolate just the head, legs, or torso for example can help them focus on key areas as they work
- Having Mesh Levels that an animator can toggle through can
  - Boost playback performance
  - As well as make each phase of animation easier
    - From block-in
    - To in-betweens and basic interpolation
    - To polish
- Having a block character where you make simple boxes for your basic body parts and parent constrain

them 1:1 with their bones can help with speed if necessary since you’re using not skin clusters.

  - This makes for ultra fast playback

- A low poly is great when you need to look at silhouettes
- A high res is of course good for as close to in-game quality as you can expect

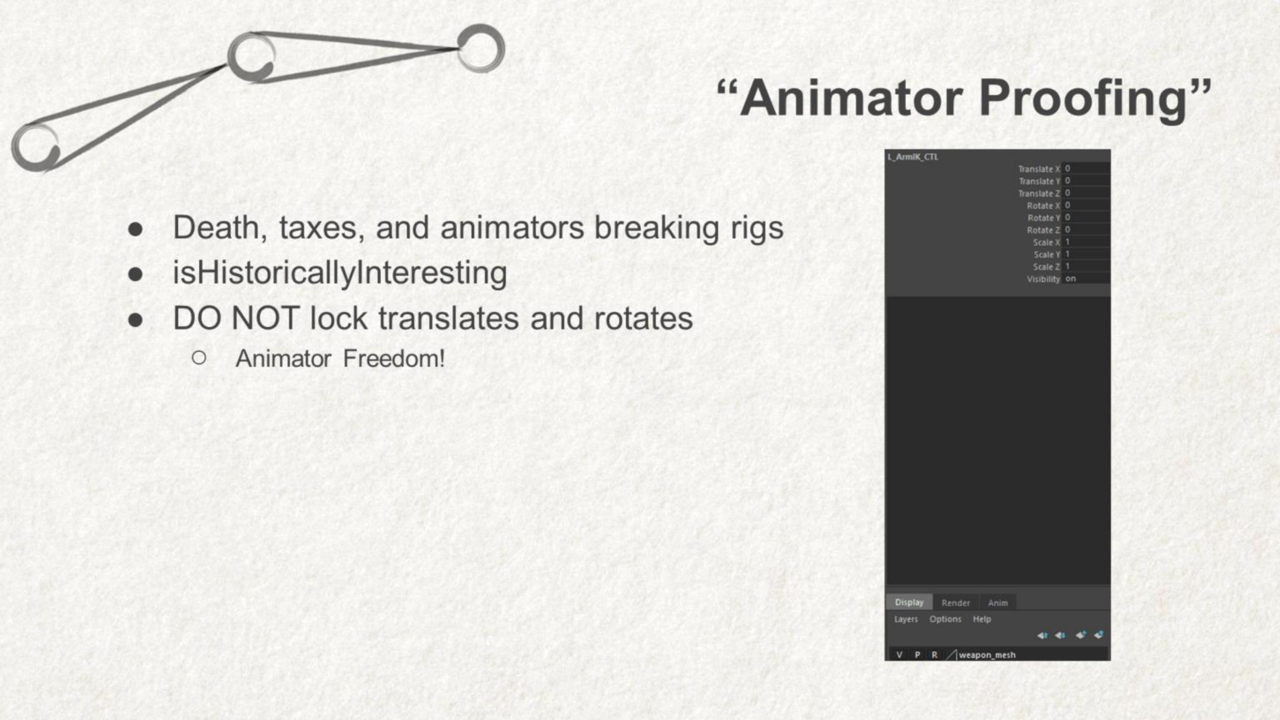


Framerate

- Slow rigs = Slow animators = Wasted time
- 30+ FPS or bust
- Things to speed up rigs:
  - Pruning weights
  - Fewer mesh pieces
  - Low poly/block meshes

FRAMERATE

- Speaking about playback
  - Framerate is the BIGGEST deal for an animator
  - Slow rigs mean more time spent trying to get animations done
- You really want to try and make sure that your rigs can playback at 30 FPS or more
- Things to help speed up a rig
  - Getting rid of unused skin inf in your mesh/Pruning weights
  - Using as few separate mesh pieces as possible
  - Have low-poly/block characters in rigs as just mentioned



“Animator Proofing”

- Death, taxes, and animators breaking rigs
- isHistoricallyInteresting
- DO NOT lock translates and rotates
  - Animator Freedom!

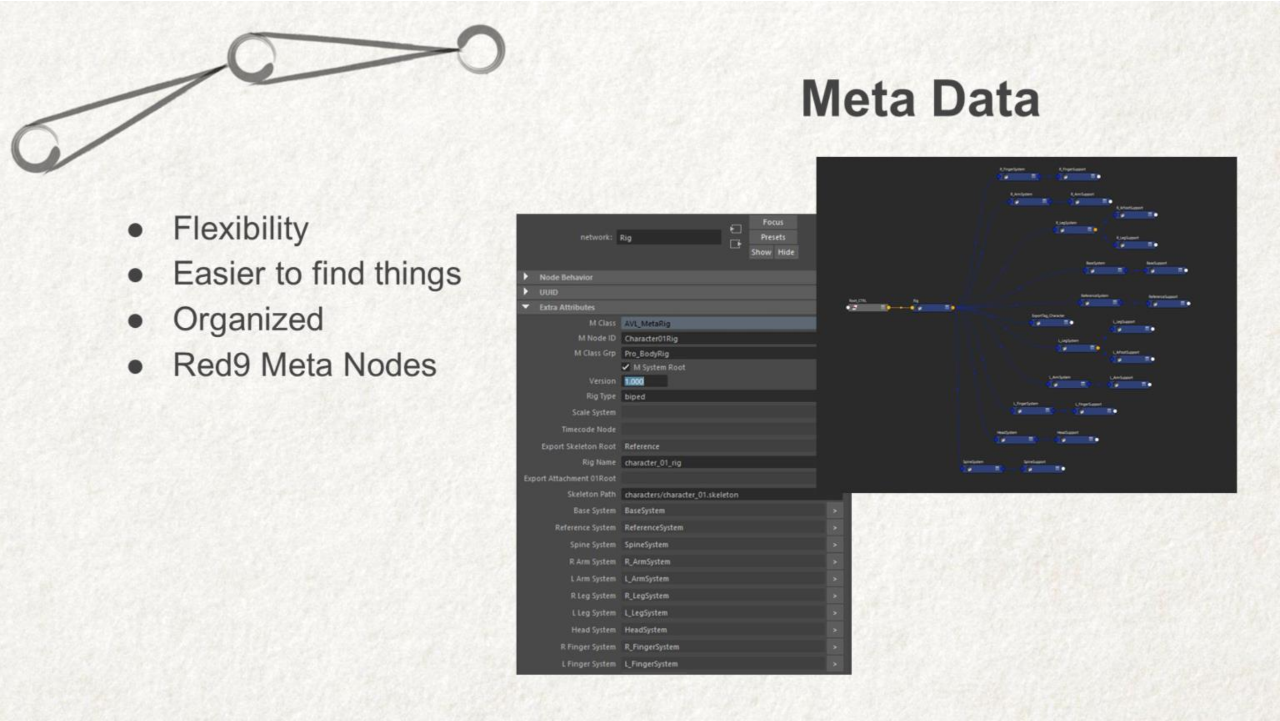
L_ArmIK_CTL

```
Translate X 0
Translate Y 0
Translate Z 0
   Rotate X 0
   Rotate Y 0
   Rotate Z 0
    Scale X 1
    Scale Y 1
    Scale Z 1
 Visibility on
```

ANIMATOR PROOFING

- The late Benjamin Franklin once said that “In this world nothing can be said to be certain, except death and taxes”
  - Well I’d like to make an amendment to that and say that I can be certain that Animators can and will ALWAYS break your rigs
  - Your job is to try and make it happen as little as possible
- Don’t let animators get their hands on components they don’t need to touch
  - You may name a group “DO NOT TOUCH” but they WILL TOUCH IT
  - Lock and hide anything and everything the animator doesn’t need
  - If they need access to certain random components

- Try to create a custom component they can use as a driver
  - Gives you more control
  - You can find it far easier when debugging
- Use command isHistoricallyInteresting to hide channel box outputs in Maya
- DO NOT lock translates and rotates though
  - You do want to make sure that animators have these channels open as to not limit their workflows



Meta Data

- Flexibility
- Easier to find things
- Organized
- Red9 Meta Nodes

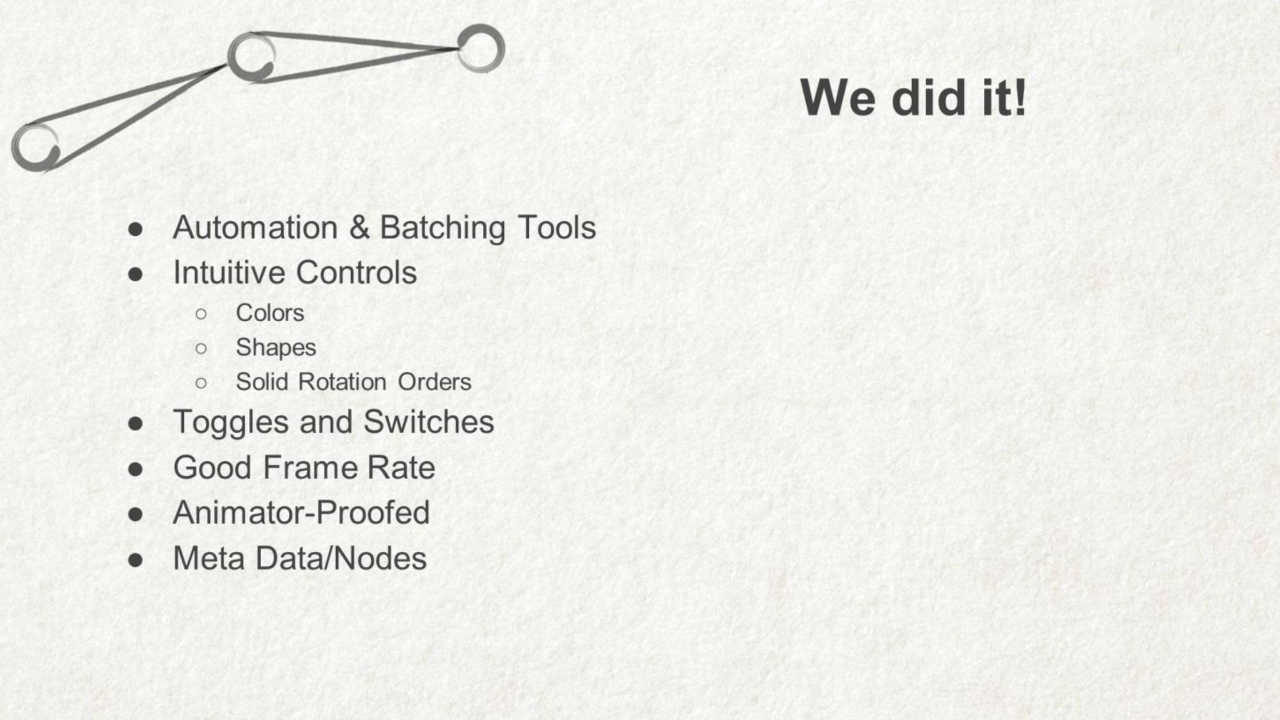
network: Rig

M Class: AVL_MetaRig
M Node ID: Character01Rig
M Class Grp: Pro_BodyRig
☑ M System Root
Version: 1.000
Rig Type: biped
Scale System:
Timecode Node:
Export Skeleton Root: Reference
Rig Name: character_01_rig
Export Attachment 01Root:
Skeleton Path: characters/character_01.skeleton
Base System: BaseSystem
Reference System: ReferenceSystem
Spine System: SpineSystem
R Arm System: R_ArmSystem
L Arm System: L_ArmSystem
R Leg System: R_LegSystem
L Leg System: L_LegSystem
Head System: HeadSystem
R Finger System: R_FingerSystem
L Finger System: L_FingerSystem

META DETA

- Hook everything on your rig up using meta deta
- Adds flexibility for tools (you can name controls differently on different characters for instance, but hook them into the same meta tag if they do the same thing)
- Allows you to find the same elements that are named differently rig to rig
  - And Whether you're searching
  - Or trying to write tools
  - You know exactly where and what things are
- Makes the visual appearance of your node networks nice and organized
- I use Red9 Meta Nodes at Avalanche because they're really robust
  - You can download the Red9 Studio tools for free and integrate Meta Nodes into either new or

already existing rigs really

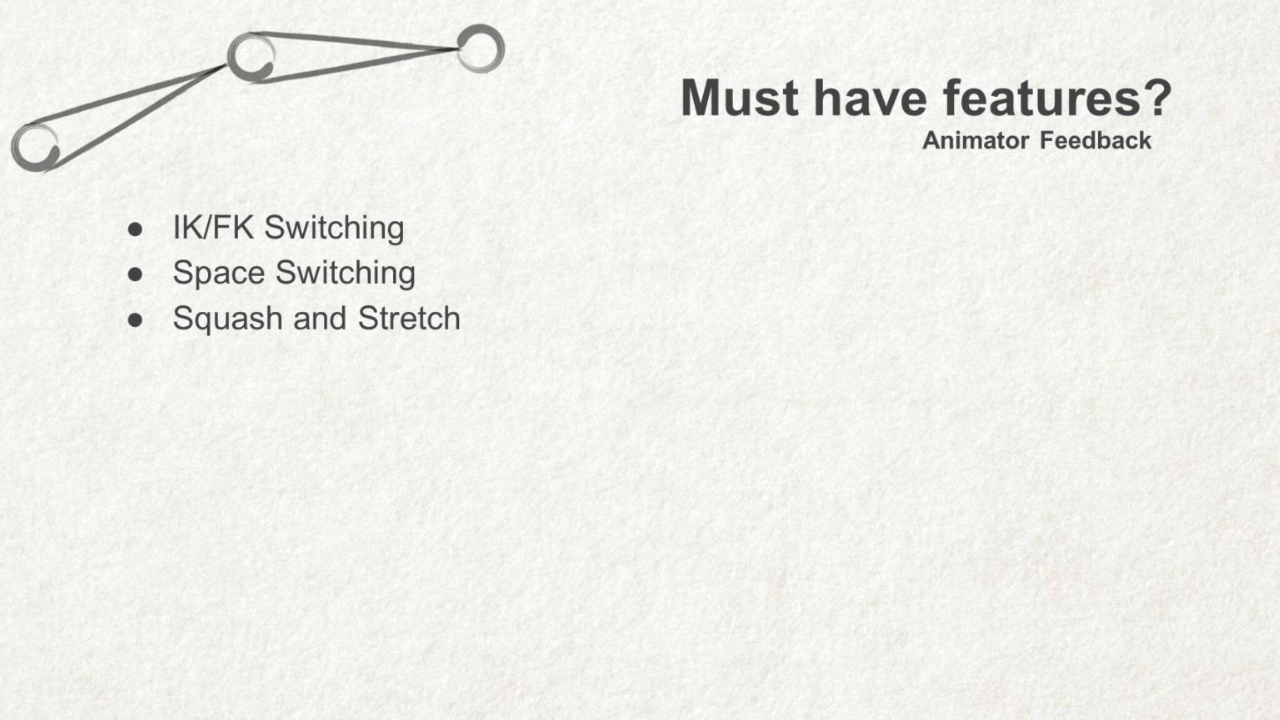


We did it!

- Automation & Batching Tools
- Intuitive Controls
  - Colors
  - Shapes
  - Solid Rotation Orders
- Toggles and Switches
- Good Frame Rate
- Animator-Proofed
- Meta Data/Nodes

WE DID IT!

- We have automation and batching
  - Most notably an auto rigger
- We have intuitive controls with
  - Organized colors
  - Shapes that are easy to select and find
  - And animator approved rotation orders
- There are toggles and switches to make your rig more dynamic and animator friendly
- And our playback speeds are fast and responsive!
- We’ve animator proofed as best as we can
- We’ve got a snazzy meta data setup for our rig components and controls
- Congratulate yourself for building a good solid rig!



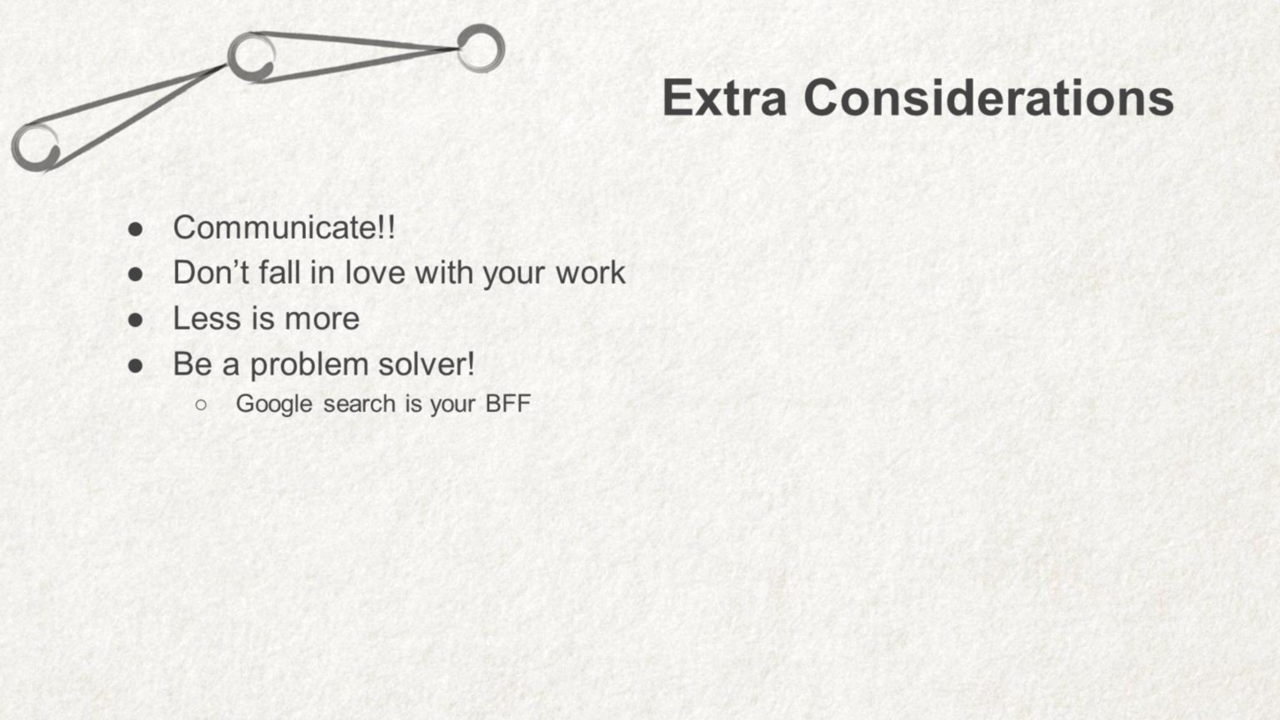
Must have features?

Animator Feedback

- IK/FK Switching
- Space Switching
- Squash and Stretch

MUST HAVE FEATURES

- Ah, and now to wrap up the last bit of animator feedback!
- None of this says anything about crazy features or awesome auto-driven things
- Every single animator said IK/FK and Space Switching
- A handful said squash and stretch as well (helps eliminate limb pops, and even realistic animation can benefit from juuuust a little bit of exaggeration)
- All simple yet extremely effective features



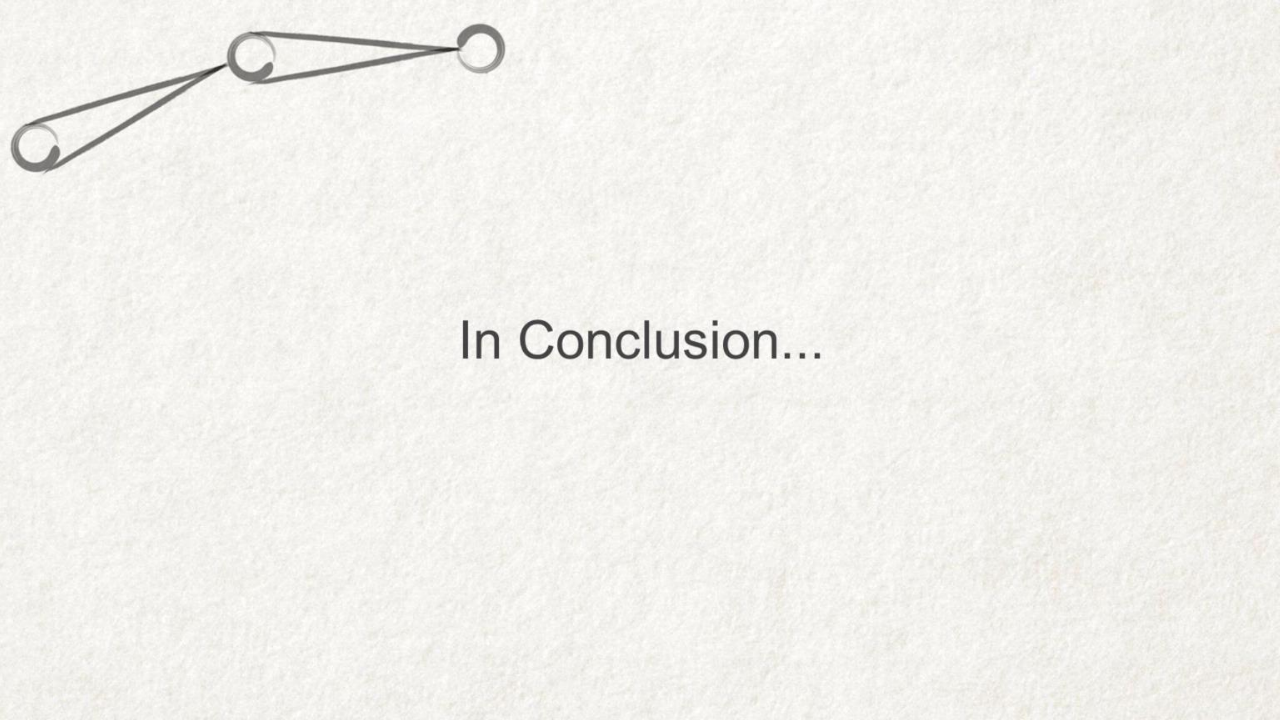
Extra Considerations

- Communicate!!
- Don’t fall in love with your work
- Less is more
- Be a problem solver!
  - Google search is your BFF

EXTRA CONSIDERATIONS

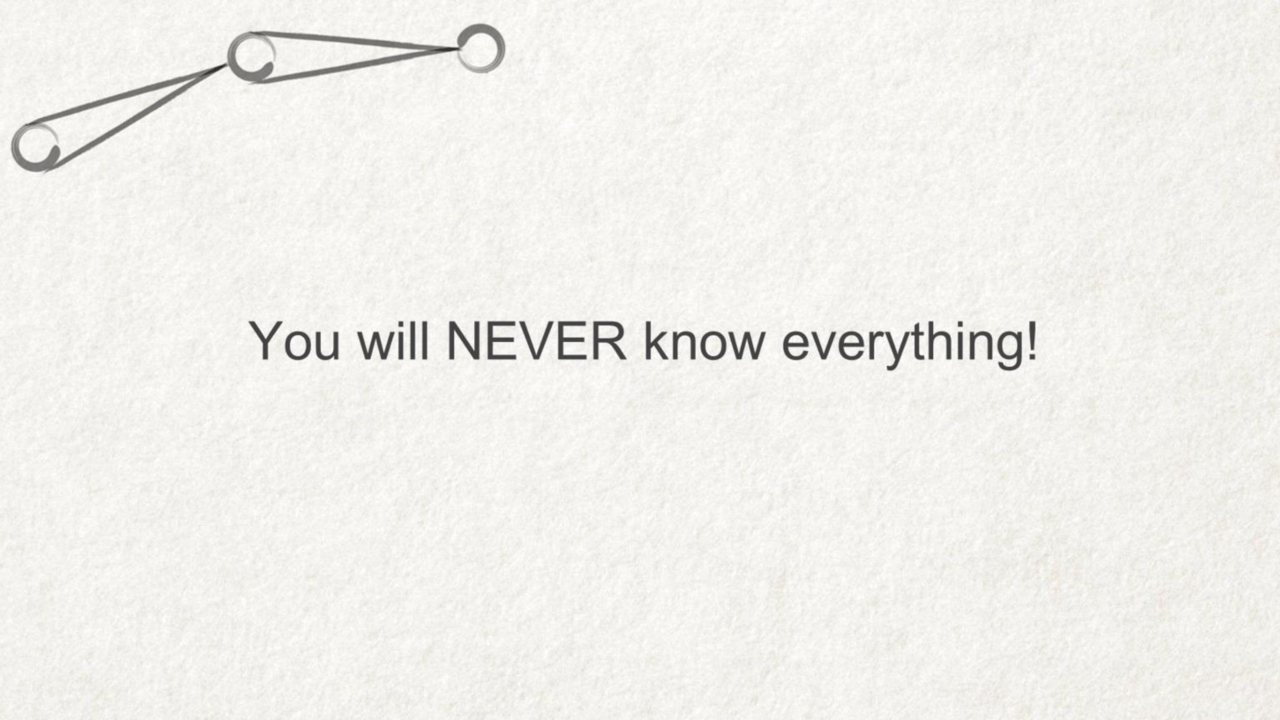
- Now that we’ve got a rig built here are a few extra considerations to remember before I get to the finale
- Your job is to give animators what they want (within reason of course).
- Your job is NOT to build a bunch of cool rig features and tools on your own without COMMUNICATING with the animators that these are what they NEED and WANT. As with everything in life, communication is key.
- Never fall in love with your work
  - A cool feature or tool may end up not being necessary in the end
- More doesn’t mean better… sometimes the simplest solution is the best solution..and as we just learned, animators mostly care about the simple things over the complicated ones!

- You need to be a problem solver more than anything else
    - Find your issues, figure out why they’re happening, and create your solutions
    - As a TA, you are an excellent google searcher
    - Don’t be ashamed to say “I don’t know how to do that” because you can follow that up with “but I will figure out how to!”



In Conclusion...

IN CONCLUSION...THE MOST IMPORTANT THING TO REMEMBER WHEN TRYING TO FIND ZEN IN THE ART OF RIGGING IS

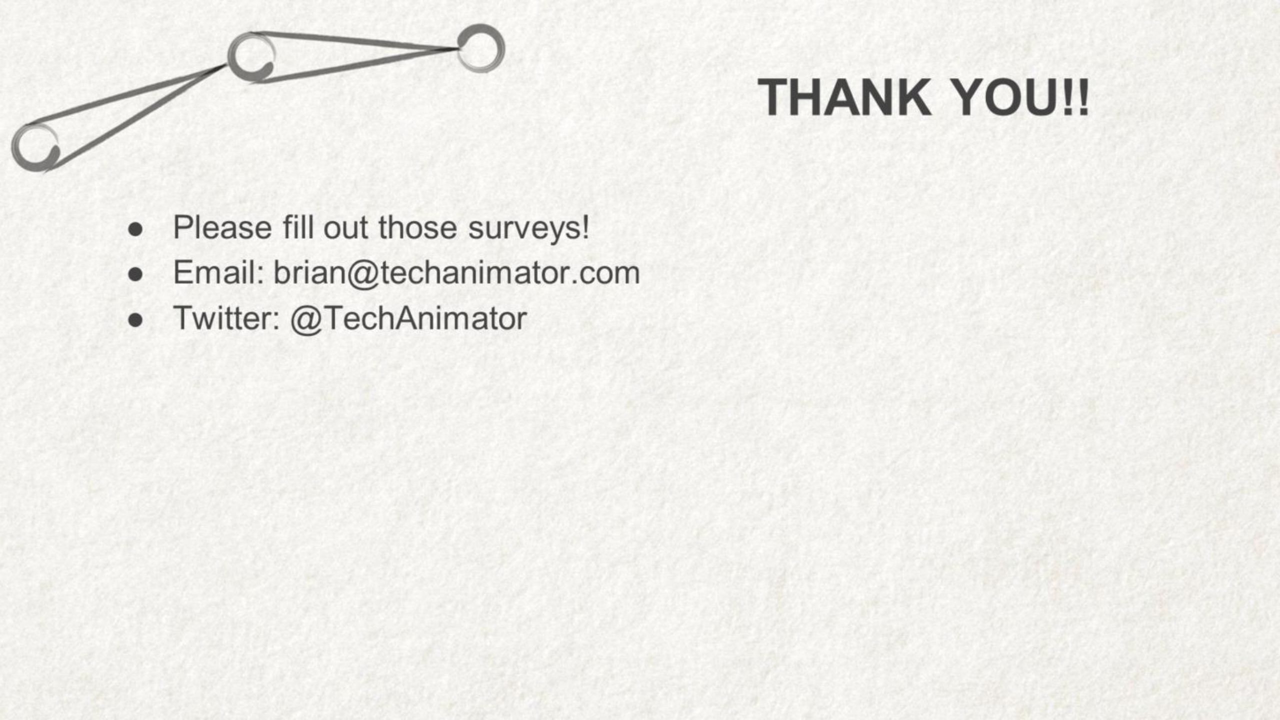


You will NEVER know everything!

YOU'LL NEVER KNOW EVERYTHING

- You'll simply never be able to know everything there is to know
- Don't worry or feel down on yourself because someone posted an awesome video on a matrix based solution to building IK/FK switchable chains when you're sitting there using the good ole' 3 chain blend.
- If you're building rigs and writing tools that work and you're achieving good results that animators are happy to use, who cares?
- There are way too many cool things out there to learn and know about
- Just keep yourself brushed up and learn new things as best you can, and don't stress trying to know it all!
- Contrary to popular belief, rigging doesn't have to be a painful ordeal

- It actually CAN be fun!



THANK YOU!!

- Please fill out those surveys!
- Email: brian@techanimator.com
- Twitter: @TechAnimator

- Please fill out your surveys
  - I appreciate any kind of feedback because I enjoy doing this
  - I want to continue to improve my speaking skills and figure out more fun things to talk about and share
- Also, Follow me on the Twitters! @TechAnimator