Animator Friendly Rigging
Part 1

Creating animation rigs which solve problems, are fun to use, and don't cause nervous breakdowns.

Jason Schleifer

AFR**Part 1**   Jason Schleifer

CONTENTS

I. INTRODUCTION

There is at least one indisputable fact known to animators all over the world: achieving quality animation with a sub-standard rig can cause more heartache, headache, hair loss, and general insanity than trying to run a marathon backwards in flip-flops while tossing jellybeans into your mouth.

Imagine you're driving a car and every time you turn the wheel to the right, the windshield wipers go off. Then you turn the wheel to the left and the car accelerates. When you press the gas—the stereo gets louder. When you try and change stations, the car reverses slowly. Sure, you could eventually drive somewhere, but the frustration that would ensue would be sure to make your life a living hell.. and in fact, you'd probably never want to drive again.

A good animation rig, on the other hand, will work the way you expect. The controls will make sense, be easy to understand, placed in a location that you expect, and work in a consistent manner. A rig that has been created with the animator's needs in mind, will be a joy to work with, and allow the animator to think more about their performance and less about how to wrestle the darn thing into submission.

What is Animation Rigging?

Simply put, animation rigging is:

A process by which an "object" is made ready for animation.

This means that animation rigging is the act of taking any object, character, prop, *anything* that the animator is going to animate, and making it possible for them to move around and set keyframes. Notice that the definition above states *objects* instead of *characters*. It's easy to forget that anything the animated character has to pick up and move *also* needs to be set up for animation by the rigger.

You can liken an animation rigger to a Marionette Maker. A person who builds marionettes creates a puppet, attaches strings to the arms, head and body, and then attaches those strings to little bits of wood. They prepare the puppet to be moved around by the puppeteer.

An animation rigger *does not have to be a puppeteer*, but they *do* have to understand what a puppeteer *needs* in order to create a believable performance.

A Riggers Role is to Balance..

1. The performance requirements of the character or prop.
2. The manipulation/interaction needs of the animator.
3. The technical requirements of the character or prop.

Many times, these needs will conflict with each other. For example, imagine you have an extremely realistic creature that is being animated, but the animator wants to be able to stretch the character's body in a physically impossible way in order to make the pose read from the camera angle. What do you do? Provide non-realistic animation controls that may break the rig? Tell the animator "no, you can't do that"? That's your job, to determine what exactly is possible and to balance the positives and negatives of each situation.

Of course, as an animator I would say that my needs were more important than any silly bio-mechanical necessities of the skinning rig.

Important Criteria for Each Animation Rig

In order to create the best rigs possible for any production situation, it is important to figure out exactly what problems you're trying to solve before just throwing solutions at them. The choices you make in determining how your animation rigs work should all meet the following criteria:

- Consistent/Predictable behavior
- Simplified control structure
- Easy to use
- Fast interaction
- Minimal counter-animation
- Solve the problem
- Allows the animator to do their job *without* compromising their ideas.

If these criteria aren't met, the rig will not work as the animator expects, and will cause frustration.

Animating is hard enough *without* having to fight the tool every step of the way.

II. DEVELOPING AN ANIMATOR FRIENDLY RIG

Creating a great animation rig isn't as simple as just adding a series of controls and handing it off to the animator. I've found that the best way to solve the problem is to really think about it in a logical way, not just technically, but also artistically. What works as the best "technical" situation, may not always work for the animator. Conversely, what the animator wants may not always be technically feasible.

Developing the Guidelines

What are the guidelines for how you should determine what's important about how a rig should work? There are two sets of requirements that you must always pay attention to:

Animation Requirements
* How the character should act – what does the director want? What kinds of movements will the animator need?
* What does the character need to be able to do?

Rig Requirements
* Standard animation Rig Specs – these are standards that you must follow to ensure consistency with all the animation rigs.
* How does the animator want to work with the rig.

Solving the Problems

You will have to answer questions and solve problems in order to achieve the requirements specified. The best thing you can do is develop a system that makes it easier to solve these problems. I like to use the following steps:

1. **Analyze the problem** – what are you trying to solve? What is the expected result? Is it similar to any other solution you've come up with before? How can you take advantage of previous solutions? What is the trade-off between functionality and speed? Does it really solve the problem or create other problems?

2. **Create a test of the solution** – Don't try and solve the problem elegantly yet, just figure out a way to get it to work correctly and give you the result you're looking for. *It doesn't have to be fast, it just has to work*.

3. **Analyze the solution** – is this the best workflow for the animator? Is the solution clean? What other problems does it introduce? Are they fixable? What is the best control structure?

4. **Re-create the solution cleanly** – try and remove any extra steps and nodes. Consolidate everything to the bare minimum. Make sure there is nothing "extra" needed. Ensure that every node and expression is necessary.

5. **Re-create again**, this time using code that is easy to modify. Document each step so you can easily modify the steps if necessary. Also, take advantage of an "object oriented" approach.. i.e. make sure that everything is done in a way that it can be re-used again. * *Note: This is not always necessary, but it can be extremely helpful if you ever need to create that rig again!*

III. Example I – Bouncy Ball

To demonstrate how important analyzing the needs of the rig are to creating a production-quality control structure, let's take a look at one of the most simple examples: a bouncing ball.
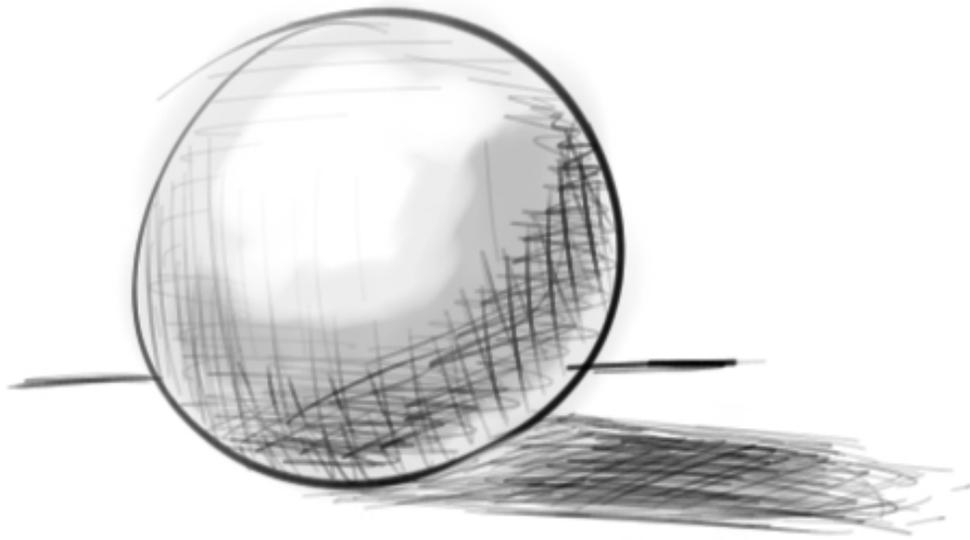

Figure 1 – The ball. A ball rig is seemingly simple, but can get complicated rather quickly.

If I were to simply say to three riggers, "create me a bouncy ball rig", I would get three different animation rigs. Maybe none of which would meet my exact needs. For example, how bouncy should the ball be able to be? What kind of material is it made of? What are the rules?

By speaking with animators before creating the rig, you can set up certain "rules" or guidelines as to how the rig should work. This will save you time and energy by producing only what is necessary.

Gathering Reference

Steps 1, 2, 5, 7, and 32 – GET SOME REFERENCE!

For the bouncy ball, let's take a look at the requirements, both animation and rig-based.

The **animation** requirements will be based on what the animator and director want to be able to do with the bouncy ball—how they want it to move around, to act, to interact with it's environment.

The **rig** requirements will be a combination of standard requirements (i.e. easy to use, etc) and specific requirements for this particular creature or object.

Of course, the best way to determine these requirements is to do exactly what an animator does: *get some reference*. You can't be certain of the way an object should move or what it should be able to accomplish without actually seeing or thinking about it performing the way you want.

So for a bouncing ball, if you have a ball available *get it out and start bouncing it around*. This will give you direct experience with the ball, which will help you determine what you need to have it do.

If the ball is meant to just bounce up and down, we'll have to add the ability to move it around *(Figure 2)*.
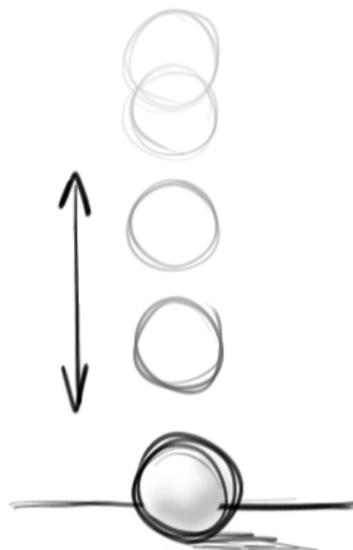
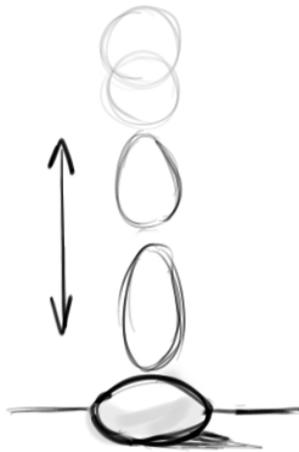**Figure 2 – moving the ball up and down**

What if the ball is supposed to be a bit rubbery? In this case, we'll need to squash it when it hits the ground, and perhaps stretch it out a bit while it's moving through the air *(Figure 3)* .
Now this would be a pretty boring animation if the ball just bounced up and down, so we probably need to add the ability to travel around horizontally a bit as well.

However, if we can do this, then we'll need to be able to angle the stretch and squash to make sure the ball stretches along the angle of movement (*Figure 4*).
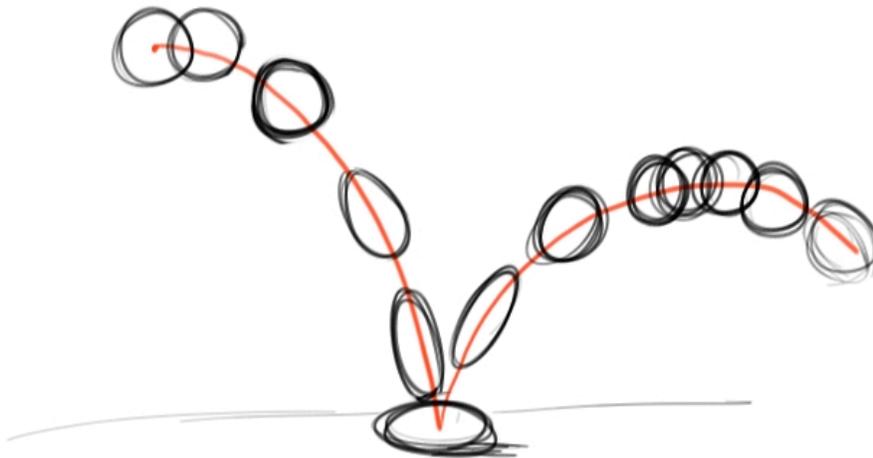
Figure 3 – adding squash and stretch

Figure 4 - allowing the ball to stretch along the path

So far this allows for quite a bit of animation control. What if the animator decides they want to bounce off of something that isn't the floor? Like a wall? In

this case, it is important for us to allow the ball to squash against any surface (*Figure 5*).
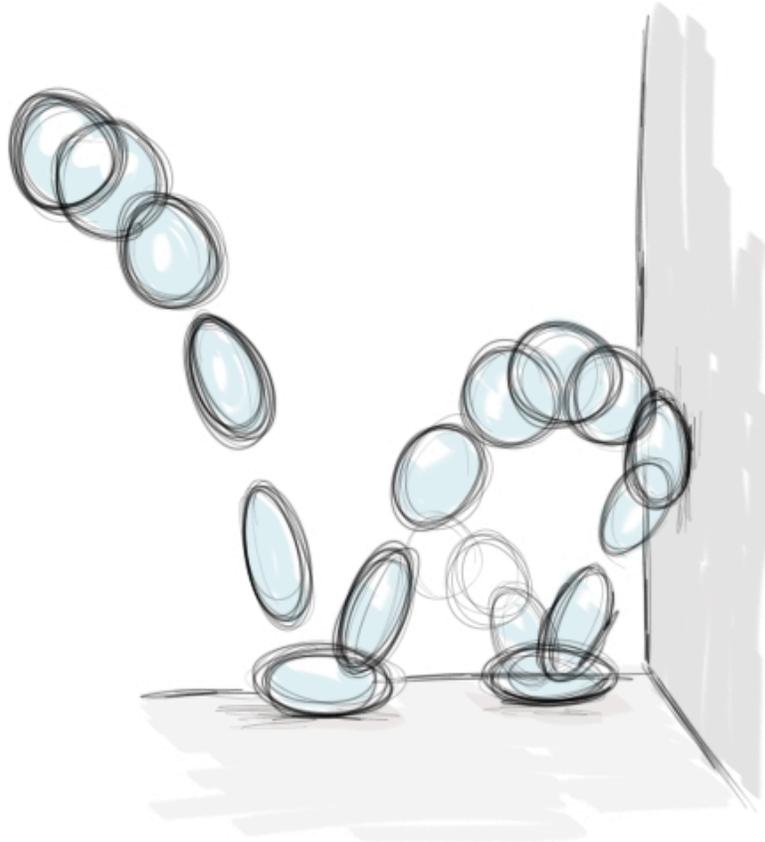

Figure 5 – Allowing the ball to squash against any obstacle, no matter what angle it is.

This is a pretty powerful rig! However, it's not quite finished.

What happens if the ball has a texture on it? We'll need to rotate the ball in order to make sure it's moving properly.

One really important note with this.. we need to be able to rotate the ball *independently* from the direction it's stretching. If the ball rotation and stretch-ation (it's a word, honest) are tied together, it will always look like the ball is physically jumping from spot to spot instead of rotating naturally.
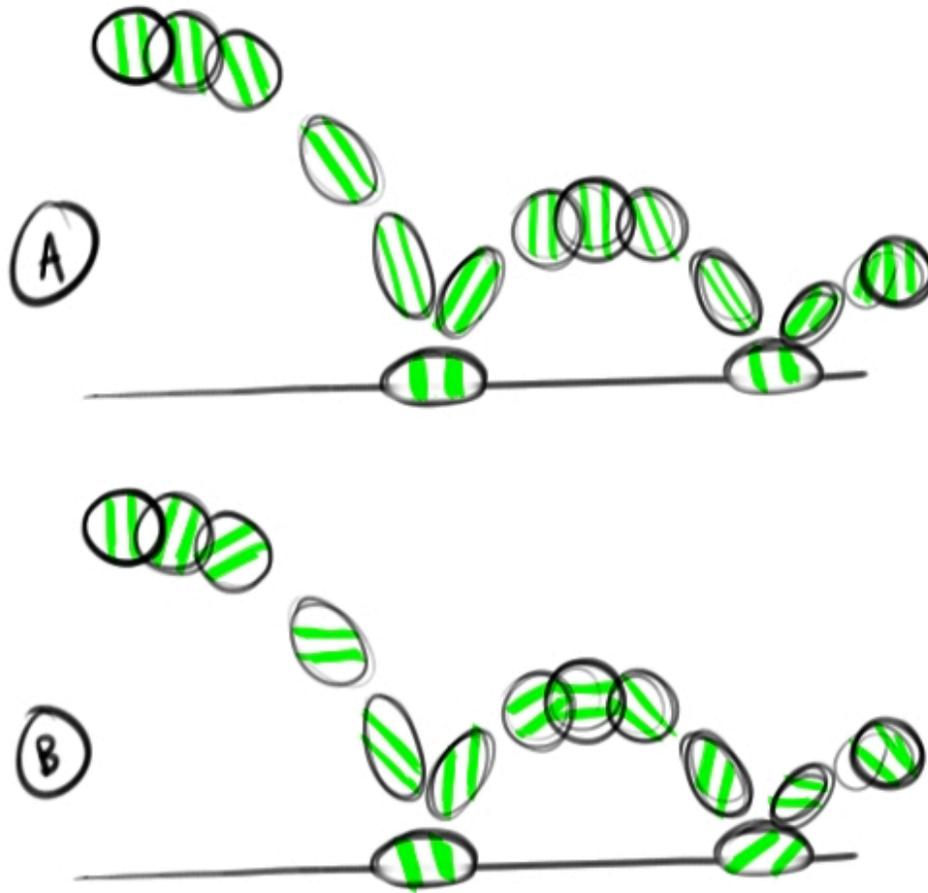
**Figure 6 - In drawing A, the ball is angled to follow the path that it's moving.
This makes it appear as if it's jumping from spot to spot. In drawing B, the ball
rotates naturally. The stretching is completely independent of the rotation.**

Let's review the rules that we have determined.

Animation requirements:
- The ball should be able to **move** around
- The ball should be able to **squash** and **stretch**.
- The **squashing** and **stretching** should be able to be **angled**.
- The ball needs to be able to **rotate**.
- The **rotation** and **angle** of **squash** and **stretch** should be **independent** from each other.

Now that we have a list of all the animation requirements, we can start to look at the rig requirements. These contain specific control requirements, general requirements that all animation rigs should have, and may also contain specific requirements for the rig in question. On this ball rig, our requirements are pretty basic:

Rig Requirements – CONTROL

These are control-specific requirements. Every control that the animator uses should follow these rules. We'll get into these in greater detail as we go through them, but here there are as a brief introduction.

- **Simple controls**
  Controls should make sense and be limited to only what's necessary. It's important to limit the amount of controls that the animator can work with, and try not to make controls that compete with each other. Your goal is to give enough control that the animator can do everything they need, but do not try and overwhelm them with options.

- **Animation should be easily transferable.**
  It's a painful fact that animators will have to start animating before the models are final, and sometimes even before you're finished rigging. Therefore, it's important to make it easy to swap in new models and to make updates to your rigs.

- **Controls should be unique and make immediate sense.**
  A rig may have anywhere from 2 to 60 animation controls. If every single control looks exactly the same, the animator will be confused and spend important time trying to figure out what control to select in order to do whatever it is they want. Instead, give them unique looking controls that help give them hints as to their intended use.

- **Controls should have the correct rotation orders.**
  When rotating objects, it's extremely important that the order of rotations lend themselves to intuitive animation. It's extremely important to analyze the order that the rotations evaluate to determine what is best for the animator. For example, if you're rotating a character around the Y-axis, it is imperative that X and Z move with that rotation in order to ensure the character can bend forward and backward.

- **Controls should be named correctly.**
  Animators will be searching through various interfaces for their controls: the Outliner, GraphEditor, Dope Sheet, channel box, etc. Five controls named nurbsCurve1, nurbsCurve2, nurbsCurve3, nurbsCurve4, and nurbsCurve5 will quickly become confusing for the animator. Which one is

the control for the left hand? Make it easy on them, name your controls appropriately.

- **Only be able to set keyframes on controls we want animators using.**
  If it's possible to set a keyframe on it, animators will. Limit what they have access to in order to 1) reduce clutter and 2) simplify their interaction.

Rig Requirements – ALL

These are general rig requirements that all animation rigs should have. Usually these are done at the end of the rigging process.

- **Only be able to select what the animators can use to animate.**
  Animators should be spending their time animating and not fighting their selection lists. Don't let them pick things you don't want them picking. If they can grab it, they'll move it. Pretend you're a parent and lock away anything that may hurt them.

- **Clean outliner when finished.**
  A messy scene will make it more difficult for the animator to work with their shot. A good rigger will make sure that everything is clean and tidy and easy to understand.

- **Can move the rig to any position and orientation and have it work.**
  This is a big one. Quite frequently I'll find rigs that work great when animating at the origin, but as soon as you try and move and orient the rig another way, it breaks. Test your rigs and make sure the animator can move them as far away from the origin as they want and that they can orient them in any direction and still have the rig work.

Let's take a look at how we can achieve both our animation and rig requirements in order to develop the most useful bouncy ball rig.

Want to learn more?
Purchase the entire DVD at:

http://jasonschleifer.com