Teaching 3D Computer Animation to Illustrators: The Instructor as Translator and Technical Director

Wobbe F. Koning
Montclair State University

Illustrators are not necessarily technically challenged, but teaching them to use technology can be a challenge. I teach 3D computer animation to bachelor-of-fine-arts students majoring in animation and illustration. Only a few of them aspire to become professional 3D animators; most of them prefer drawing with pencil and paper to moving vertices and faces around with a mouse or tablet pen.

My courses function as studio art classes, not software training exercises. The goal is to teach the principles behind the technology rather than button pushing. If students only learn to achieve specific results by clicking on the proper icons, they do not develop the most important skill needed to creatively use the technology—problem solving.

There are two primary areas in which my students' lack of a technical background complicates instruction of 3D computer animation, even as a studio art class. The first is the concepts and terminology needed to effectively communicate about 3D computer animation's technical side. These concepts and terms are alien to my students; their meaning eludes the students, and the words do not seem to want to stick. The second area is the software's complexity. Students are often discouraged by the steep learning curve and get lost in the plethora of tools, options, and settings.

Terminology

When I ask advanced animation students what a normal is, they might answer, "the little line sticking out of a polygon," remembering the visual I showed them in one of the beginning courses. If I ask what that line signifies, I often get a blank stare. I frequently encounter other similar disconnects. Students seem responsive when I refer to the tangent of a curve as the curve manipulation handle they use in a vector-based image creation program such as Adobe Illustrator, and they understand that it shapes the curve. But because they are not familiar with reading graphs, they do not seem to make the connection that the slope of the tangent indicates the rate of parameter change in an animation curve editor. In the previous sentence, I would probably lose them at "slope of the tangent."

I have not found a satisfactory solution for this issue. I do not subscribe to a contextualistic worldview[1] or consider myself a constructivist in that sense. However, teaching technology in context does appeal to me as a way to engage students.[2] I do not go as far as to focus the curriculum around topics that are close to the students' reality and interests, but I do try to provide context—for instance, by showing popular or successful animations that put the covered concepts to good use. When creating assignments, I try to relate them to the reality of work in an actual studio. Luckily, art classes usually have an intrinsic context.[3] In my classes, student projects involve creating virtual objects and eventually entire animations based on the students' own design and story ideas. However, to stimulate their creative process, I find it effective to sometimes show something far removed from their reality and interests—for instance, to broaden their horizon with a piece of abstract procedural animation.

Students' fear of mathematics still amazes me, and the term "mathematics" seems to encompass anything having to do with numbers. Creating 3D computer animation involves many numbers; they cannot be avoided. Because even basic concepts such as a vector or tangent still have to find a way into the students' vocabulary, I have much ground

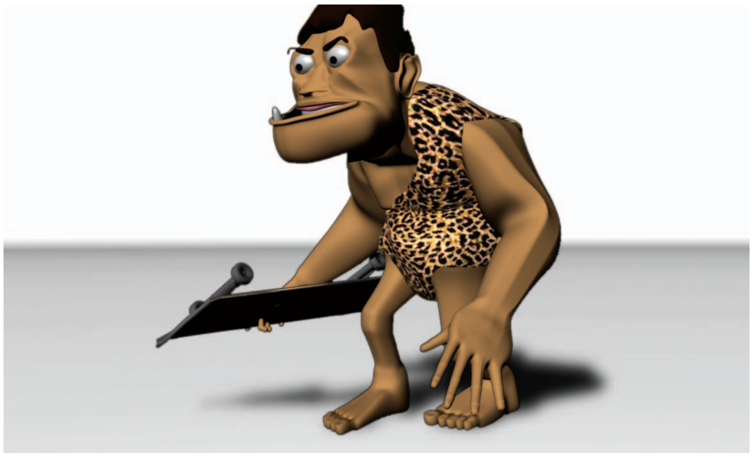

Figure 1. Teaching 3D to artists might be as hard as teaching a caveman to skate. Undergraduate student Brian Kapp designed this character and animated it using the supplied automated-rigging script.

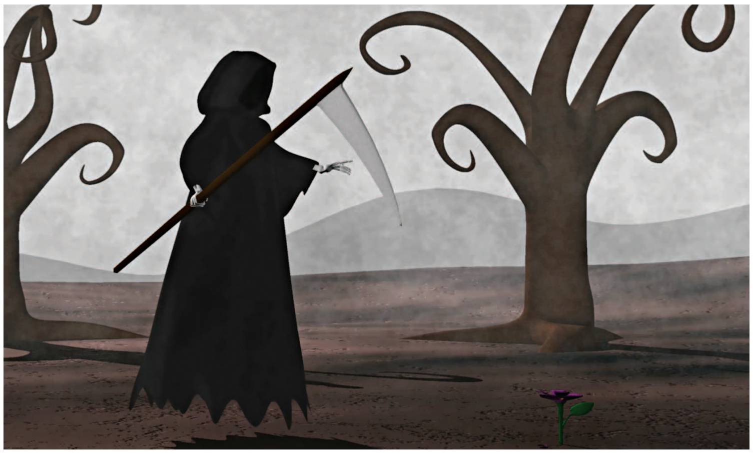

Figure 2. Even though this floating death character has no legs, advanced animation student Melanie Farnsworth animated it by modifying a script-created rig.

to cover before I can introduce something such as a shading algorithm. Translating computer graphics language into plain English is not something I have been trained to do, but being aware that it is one of my tasks as an instructor helps me prepare my lectures. A complicating factor is that, after covering a 3D animation concept, I often have to explain that although what I just laid out is true in theory, the implementation in the software we use is slightly or even radically different.

Software Complexity

We use the de facto (industry) standard software Autodesk Maya, which is not known for having a user-friendly interface. In several instances, Maya's use of terminology makes sense in the light of computer graphics history but is unclear and confusing for people new to the field. Unfortunately, the latter is probably the case with all similar software. However, I do not want to gear the way I cover the concepts to any specific software package. Software comes and goes, while the underlying theory remains relatively constant. It would be helpful if a textbook were available that covers the theory in a software-independent and concise, clear, and consistent way, without being bone dry. If you know of such a book, I would love to hear from you!

To address the software's complexity, I have been giving my students tools that can simplify certain tasks, taking some of the more technical work out of their hands so that they can focus on the creative process. Taken to the extreme, I would basically work as technical director on the students' projects.

One of the more complex aspects of 3D computer animation is *rigging*—the technical setup required to animate a 3D model. When you are teaching animation to nontechnical students, it might make sense to skip rigging altogether and have them animate with existing, fully rigged models. Although I do assign such exercises, limiting the scope of animation to that would not satisfy my students. They want to breathe life into models of their own design. This makes sense because they, being illustrators and thus good draftsman, are more likely to end up working on the design, concept art, and storyboard side of animation rather than doing the actual animation. They want to show off their own characters in motion. So, eliminating rigging is not an option. Unfortunately, rigging gets very complex very quickly (inverse-kinematics solvers, weight painting, animation controllers, and so on).

In spring 2011 I supplied my animation students with a script that can generate a complete rig for their characters with minimal user intervention—that is, an automated-rigging script. All the students must do is position all the placeholders for the joints. The script then generates the skeletal structure and sets up the appropriate inverse kinematics and the associated animation controls.

Because this was an experiment, I relied on freely available tools. After testing several of them, I found that Rapid Rig: Basic best suited my needs. (Rapid Rig: Basic is available at www.creativecrash.com/maya/downloads/scripts-plugins/animation/c/-rapid-rig-basic-for-maya--2.) A major reason for choosing this script was that it generates a rig with all the features the students normally need, set up in a manner not too complex for them to understand. So, they can perform minor modifications.

The results have been mixed but encouraging. (For two successful examples, see Figures 1 and 2.) Unfortunately, this script is geared toward human anatomy and supports only bipedal characters. Students who created stories about cats, dragons, or fish still had to build their rigs from the ground up. Most students whose stories involved bipeds used the script, even if their character was just a stick figure or did not quite have human proportions. The resulting animations had a level of refinement that would have been harder to achieve with a rig they built themselves, which would have been far more basic. For example, one important functionality in the script-generated rig is foot and toe roll. This functionality enables students to more easily give characters a proper foot roll while maintaining floor contact.

What Rapid Rig (and most other) automatic-rigging tools does not provide is a perfect "bind." Defining the relation between the skeleton and the skin the rig needs to deform—a process called *weight painting*—unfortunately remains primarily a manual exercise. For instance, if your character is a muscular caveman (as in Figure 1), weight painting can be quite involved and extremely cumbersome. Better weight-painting tools are next on my list of what I would like to offer my students, although I am not aware of any available (fully) automated tools.

Because I have not yet found the perfect tool, I have considered writing one myself or updating and expanding on an existing script. Developing within a fully open source environment such as Blender (www.blender.org) probably makes the most sense in an academic environment. However, as long as students (and even their parents) strongly believe that knowledge of Maya is essential to get a job in the industry, switching to free and open source software will be a hard sell, even though it would be good for our budget. In a closed-source environment, I am hesitant to invest much time in tool development.

One implication of the approach I just outlined is a shift in my role from instructor to facilitator. Taking some of the technical aspects out of the students' hands means I must do more research to find the right tools and make sure they work in the current lab configuration. It does, however, fit my aim of providing students a work environment that somewhat resembles that of an actual animation studio—something that, hopefully, they can relate to.

What would really help me reach that goal is actually not something technical. Finding a way to have all students work on a group project or group projects would be to their advantage. I have had some success with this during my chairmanship of our local animation festival MetroCAF (the New York City Metropolitan Area College Computer Animation Festival). Students had the clear goal of creating the opening animation for the show, and it worked out well. But that is a different story.

Overall, the experience of offering the script to students was quite positive, and I continue to provide this tool. An increasing number of students have been taking advantage of it. Because using automated-rigging tools continues to prove useful for my classes, I might consider purchasing more advanced tools in the future. However, even when provided with the perfect rigging tools, students still must understand the concept and practice of rigging to solve problems that might arise during animation, to figure out why something did not work as expected or completely stopped working when the rig broke.

Even when provided with the perfect rigging tools, students still must understand the concept and practice of rigging to solve problems that might arise during animation

Of course, I could again play the technical director who magically fixes things. However, because students tend to complete assignments the night before the deadline, I probably will not be available when they need help the most. So, the extra tools do not render obsolete the teaching of the general concepts and their implementation, but they can help students achieve higher-quality animation.

References

1. E. Fox, "Contextualistic Perspectives," *Handbook of Research for Educational Communication on Technology*, 3rd ed., J.M. Spector et al., eds., Lawrence Erlbaum, 2007, pp. 56–66.
2. S. Cooper and S. Cunningham, "Teaching Computer Science in Context," *ACM Inroads*, vol. 1, no. 1, 2010, pp. 5–8.
3. C. Case and S. Cunningham, "Teaching Computer Graphics in Context: Computer Graphics Education 09 Workshop," 2009; http://education.siggraph.org/media/reports/CGE09-Workshop-Report.pdf.

Wobbe F. Koning *is an assistant professor of art and design at Montclair State University. Contact him at koningw@mail.montclair.edu.*

Contact department editors Gitta Domik at domik@uni-paderborn.de and Scott Owen at sowen@gsu.edu.