Chapter 5

Animation

5.1 Introduction

In most uses of 3D graphics, animation is very close at hand at all times, simply because everything about a scene is just numbers in the memory of a computer, and data is very easy to change. Video is comprised of a sequence of still images, and rendering animated content is really just a matter of rendering several still images with small changes between frames. The concept of *motion blur* often becomes more important for animated content than for still images, but motion blur is not unique to animated content. Even a still image camera has a shutter speed that might be long enough for moving objects to be blurred, and computer generated still images may also need motion blur to be believable.

One question is: what can be changed? The answer to that is simple. *Everything* can be changed. Common animations involve changing the position and orientation of objects, but size and shape can be changed just as easily. Light sources may have their position, direction, color and intensity changed. The position, orientation and field of view of a camera can be animated to create a dynamic view. Less obvious scene properties like the color, transparency or reflection properties of a surface may be changed over time as well. Changes in surface appearance can be created by animating the parameters of a procedural texture, or making changes to the texture coordinates. Even the quality settings for the rendering process may change between frames if some parts of an animation require more attention to detail than others. In short: anything and everything about a scene can be animated.

A more relevant and more complex question is: *how* can these changes be created and edited in a reasonably efficient manner? Many interactive, automatic and semi-automatic methods for this have been created over the

years, and the rest of this chapter will be an outline of the most important among them. The classification is somewhat loose, the categories overlap to some extent, and some of the terminology might differ from other authors, but the overview should serve as a broad and reasonably complete introduction to current methods in animation.

5.2 Interactive methods

Interactive methods involve manual labour, in the sense that most every detail about the animation is specified explicitly by the user. The computer can of course assist greatly in this process by removing some of the most tedious and time-consuming work. One often overlooked advantage of computer animation is the fact that an animation can be watched, tweaked and watched again until it looks good without any significant extra cost, and several versions of a scene may be created with relative ease. This in itself is a big advantage to traditional animation techniques, where the time and cost for remaking a meticulously animated sequence can be prohibitively high.

5.2.1 Stop motion

Stop motion is the animation technique used for traditional cartoons and puppet animations. The exact position for every object is specified for every frame. All motions are performed in small incremental steps one frame at a time, and for each frame, a still image is captured. While this method is still perfectly possible to employ for computer animation, it is a very inefficient and slow process. Stop motion gives animators complete control over every aspect of the animation, but they also have to care about absolutely everything in every frame. There are too many things that need attention, and stop motion is a *lot* of work. However, the method is still in active use for movie production with clay figures, poseable puppets and traditional hand-drawn cartoons.

5.2.2 Keyframing

Keyframing is a term borrowed from traditional cartoon animation. Back when animated cartoons were always hand-drawn, the head animators drew only some frames, the *keyframes*, and left the rest of the frames for other artists. These people were called *inbetweeners*, and their job was simply to draw a number of intermediate positions between the endpoints of a motion. When keyframes are specified in a computer animation, software can do the job of inbetweeners. This process is called *interpolation*. It is not immediately obvious how best to interpolate between two keyframes to create a smooth and plausible motion, but for many animations it is enough

to specify only a small number of keyframes and let the interpolation do most of the job. Some manual editing may be required to specify how the interpolation should be performed at certain points, and for detailed and rapid movements the user might have to specify so many keyframes that the process almost resembles stop motion animation, but for the most part, keyframing and interpolation is a lot less work than traditional stop motion animation. Moreover, the main portion of the work can be spent on parts of the scene that matter the most for the end result, while secondary and peripheral motion can be animated using fewer keyframes.

Keyframing is one of the oldest forms of computer animation, but it is still a very important method, and it is sufficient to make almost any imaginable kind of animation. Animated motion pictures are routinely made with keyframing as the main expressive tool, and a skilled animator can create very life-like animations using only keyframing. It's certainly not the only tool available to animators today, but quite often it's still the right tool for the job.

5.2.3 Path scripting

Sometimes it's more natural for an animator to draw the 3-D path an object is supposed to follow through a scene, rather than setting keyframes at regular intervals. This is a common method for camera animations and moving vehicles, because it gives explicit control over where the object goes. The path is often drawn as a Béziér curve, but other types of splines may be used as well.

5.2.4 Expression scripting

Some animations are easily described as a simple time-dependent formula, like rotations around an axis or a back and forth swinging motion. In these cases, the best tool might be to write an expression in some scripting language to determine the motion. Depending on the implementation, such expressions may also include dependencies on other animations in the scene. Some animation packages allow detailed programmatic control of every aspect of the scene through a tightly integrated programming language, while others offer more limited scripting tools.

5.2.5 Hierarchical motion

Moving objects are often attached to other moving objects. The wheels of a car can spin, but they stay attached to the car. A human arm can bend at the wrist and the elbow, but when the upper arm moves around the shoulder joint, the lower arm and the hand both move with it. This kind of hierarchy is expressed as a "parent-child" relation, where the motion of one object directly influences another object, but where the second object may

also move independently of the first. This is often referred to as "linking", and most animations use it in some way or another.

Skeletal animation

One hierarchical motion of particular interest is *skeletal animation*, where a set of linked *bones* are used to control the motion of an articulated character. The animation is performed on a simple structure of linked, rigid objects with joints between them, a *skeleton*, and the model is a general mesh where each vertex is transformed according to the position of its associated bone. Vertices are generally associated with more than one bone, and the weights for which bones influence a certain vertex vary gradually across the mesh to create smooth, bending deformations from the individual motion of the separate, rigid objects in the skeleton. Performed correctly, this kind of "skin and bones" animation can create believable animations of characters resembling humans and animals. The rotations for the joints of the skeleton are often subject to constraints to prevent, for example, an elbow joint to bend along the wrong axis or bend backwards.

5.3 Semi-automatic methods

A few animation methods can be described as semi-automatic, in the sense that they require manual input but also create a lot of the motion automatically by some extra computations.

5.3.1 Inverse Kinematics

The term *inverse kinematics*, or *IK* for short, makes better sense if we first explain what is meant by *forward* kinematics. Forward kinematics is basically skeletal animation, as presented above. The motion of, say, a hand, is determined by the rotations of the joints in the arm. The rotations in the shoulder joint, the elbow and the wrist are combined to determine the position and orientation of the hand relative to the body. While this is perfectly useful in some cases, it doesn't really match our brains' concept of "moving the hand". In humans, the *motor cortex*, the parts of our brains that are involved in moving our body, is a deep layered structure with a hierarchy of motion planning and regulatory feedback. Our conscious plan for a motion sits at the top of this hierarchy and is very general, like "move the hand to over there and pick up that object". The execution of that motion requires a complex, well timed and well balanced sequence of muscle activations, but we are usually not consciously aware of the details, neither while we are planning the motion nor while we are performing it. Our motor cortex takes care of the low level details without requiring us to care about them. If something goes wrong, e.g. if we hit an obstacle or drop the object

we try to pick up, we need to come up with a new plan, but if things go well, it happens pretty much automatically.

In much the same manner as the motor cortex, inverse kinematics attempts to take care of the details in a complex motion and leave only the essentials to the animator. By describing the constraints of a skeletal structure like, say, a human arm, it's possible to solve equations to determine which motion is required for each individual joint to execute a motion of the hand along a specified path. The hand in this case is called the *end effector*, and the bones and joints up to the shoulder are called the *IK chain*. The constraints usually form a rather complicated system of simultaneous equations that seldom have a unique solution. If there are multiple valid solutions, the IK solver needs to try to determine which is the "best" solution. The criteria for what is best can include using as little energy as possible, avoiding extreme force or extreme flexing, or some other criteria. If an exact solution can't be found, approximations can be used, or special cases can kick in. If, for example, a hand cannot quite reach to the position we ask for, it might be good user interface design to at least stretch it as far as it goes towards that position without separating the bones in the arm from each other. A special case could instead be to also tilt the body when required, to allow the hand to stretch a little further.

Designing a skeleton, setting up the relevant IK chains for the limbs and connecting the skeleton to a mesh model is called *rigging*. It is a complex task which requires a lot of technical skill, good knowledge of the motions that are to be performed, and quite a lot of patience.

5.3.2 Secondary motion

Some motion in a scene is driven by other motion, like the antenna of a radio controlled car flailing about as the car drives along a path. Such secondary motion can often be expressed as fairly simple mathematical expressions, using e.g. the second derivative of the position to determine the acceleration of an object and thereby compute the forces acting upon objects that are attached to it. Most animation software contains a set of tools to compute such secondary motion.

Secondary motion can also involve idle motion from things that are not in active focus, but still need to move about slightly in order not to appear frozen in place. "Unfreezing" objects that would otherwise have been perfectly still can make a lot of difference for the perceived quality of a scene. Such secondary motion can often be specified with simple scripted expressions. In particular, using Perlin noise to specify small back-and-forth rotations and small changes in position turns out to be surprisingly effective to create secondary motion with a seemingly random, organic feel to it. Animation software with any self-esteem has at least some means for doing this.

5.4 Automatic methods

In many situations, animation can be created by computer simulations without any detailed user input. Such simulations can range from solving a few simple mechanical equations for a small number of rigid objects to performing very complex simulations of e.g. turbulent flow of a gas or a liquid. Common to all these simulations is that computer graphics often take significant short-cuts in order to speed up the computations, at the expense of accuracy. The primary goal is not to get correct results, but to keep the errors within visually acceptable limits.

5.4.1 Physics simulation

Physics simulations share the common trait that they use mathematical models from physics to create animations. The models differ greatly depending on the type of objects, but the common trait is that they are *systems of differential equations with constraints*.

Rigid bodies

Perhaps the most straightforward kind of simulation is *rigid body simulation*. Rigid body dynamics is a very mature and thoroughly studied field of mechanics, and the equations which govern a system of rigid bodies have been known for centuries. The most important properties to consider in the simulation are mass, surface friction, elasticity and inertia. External forces acting on the objects, such as gravity and air friction, need also be taken into account.

However, while the rigid body equations are reasonably easy to write down, they are surprisingly difficult to solve. Collision and contact between objects introduce hard constraints into the system that make it a *stiff problem*. Rigid objects may not penetrate each other, meaning that collisions and contact happen at a micrometer scale. Strictly speaking, interactions between objects happen at an atomic level with a timing in the order of nanoseconds. Hence, the corresponding equations might need to be solved with a very high accuracy to get an acceptable result.

Instead of doing the computations with a high precision and solving the equations iteratively using many steps, computer graphics often takes an approximative approach to the problem and defines objects as having a thin "soft shell" at their surface. Collisions are then no longer hard and instant with forces acting in sharp spikes (impulses), but slow and gradual with weaker collision forces applied over a longer time. This kind of "soft collisions" can create results that look downright terrible in close-ups views, but it's usually good enough. The approximation saves a lot of hassle and eliminates a lot of errors.

Mechanical systems

Rigid bodies can be connected by rotational and translational joints, dampers and springs. Many mechanical contraptions like vehicles can be accurately modelled by this kind of idealized system, which is often called a *mass-and-spring system*. To create motion, force actuators are used to push the model around, and rotational actuators simulate motors creating torque around an axis. With modern computers, it's perfectly possible to simulate a reasonably accurate model of a vehicle with wheels, a motor and suspension in real time, even for a game. It would be difficult to use other, simpler methods to simulate in a reasonably convincing manner e.g. how a vehicle behaves when driving on a bumpy road.

Soft bodies

Not all objects are rigid. Real world scenes include a lot of soft and deformable objects, one of the most important among them being *cloth*. Cloth simulations are not strictly limited to woven cloth, but can be used for any kind of deformable object that can be characterized as a thin sheet of some deformable material. Real world objects that can be successfully simulated using cloth simulation include cloth, leather, plastic foil and sheets of paper. Hair and fur present a slightly different problem, requiring different simulation models. Actual soft bodies, as in deformable blobs of fat or objects made of elastic materials like rubber, require yet another kind of simulations. Cloth and hair simulations are considered the most important soft body simulations in today's applications of computer graphics.

At the implementation level, most soft body simulations actually use mass-and-spring models, with a mesh of small nodes connected with springs and dampers.

Fluid simulation

Another important class of "objects" in virtual scenes is *fluids*, a term which includes both gases and liquids. Gaseous phenomena that are suitable for simulations include smoke, clouds and fire. Liquid phenomena that often need to be simulated for best visual results include water surfaces with strong waves and splashes, and flowing or pouring water interacting with objects. "Gooey" substances like mud or jelly can be simulated in this manner as well.

Fluid simulation for computer graphics is often performed as a set of particles in a grid of cells. The particles are often spheres or generalizations of spheres (such as a sphere with a soft shell), and the resolution of the grid is chosen such that each grid cell contains only a small number of particles. The simulation can be performed at a relatively coarse resolution, with details filled in by noise-like procedural textures. For example, only

the general shape and density of a billowing cloud may be simulated, while the fine details are a texture with the general appearance of a cloud but only a superficial connection to the simulation.

5.4.2 Particle systems

Fluid simulation with particles requires that the particles interact with each other in some way. Liquids are mostly incompressible, meaning that the particles may not come too close to each other. Gases can compress and expand, but a local high or low pressure will push or pull nearby particles, so there is still interaction between particles, at least indirectly.

If we remove the interactions between particles altogether, we get what is referred to as a *particle system* simulation. Particle systems can contain millions of particles, because they obey very simple rules for their animation. In the simplest case, they just preserve their speed and direction and move in straight lines, but they can also be influenced by external forces such as gravity, friction and collisions with larger objects, as well as by internal forces like a random variation in speed or direction.

In the simulation, particles are *spawned* with a position, speed and direction, usually with some variation around a mean, and are allowed to exist for a time that is either predetermined at their creation or decided based on interactions with the environment. The visual impact comes not from the individual particles, but from the large number of particles. The sheer number of particles can obscure the fact that the individual particles don't actually behave in a very realistic manner.

Particles for particle systems are simulated as points or spheres for simplicity, but they can be drawn as any kind of geometry. Particles simulating rain can be drawn as lines along their direction of motion to create a cheap motion blur effect. Particles simulating snowflakes can be drawn as small white blobs with an added random rotation to simulate tumbling. Smoke and fire can be simulated by particle system where the particles are drawn as small blobs that overlap on the screen. Sprays of water can be simulated by particles drawn as metaballs, which creates a solid jet near the source but breaks up into drops when the particles spread out.

Finally, particle systems may be generalized to have more complex behavior for each particle. For example, a crowd of people or a flock of animals can be simulated as particles where each particle tries to stay close to the others, but not too close, and move in the same general direction as its neighbors while also avoiding obstacles in the environment. This kind of *crowd simulation* can be very convincing, even with relatively simple behaviours for each particle.

A lot more can be said about particle systems, but in conclusion, particle systems are very useful, and very much used in animation.

5.5 Motion capture methods

In recent years, the requirements for realism and expressiveness in computer animation has created a trend towards using recorded motion from real actors, and applying it to computer models. This is called *motion capture*, or "mocap" for short. Most current commercial applications of animation make rather heavy use of motion capture, both in off-line rendering and real time applications. Motion data can describe the entire body or some part of the body, and may include small details like facial expressions.

5.5.1 Full-body motion capture

Motion capture studios for full-body motion capture are currently large and expensive installations with a high cost of maintenance and support. They are often run by companies that specialize in motion capture and offer their services for hire.

Capturing the motion of an entire human body requires measurements of the angles of all joints in the skeleton, or the position of a large number of points on the body. Depending on the application and the method used for capturing and processing the data, motion capture files use either angles or positions, or a combination of both, to describe the pose of an actor. The file contains data for each joint in each frame, and the files can be quite large.

Direct measurement

A very direct and simple method for motion capture is to make direct measurements of joint angles by sensors placed on an exoskeleton that is strapped to the actor. This is relatively cheap and very robust, but a bit constricting for the actor, and more indirect measurements are often preferred.

Camera-based methods

A very common setup is to capture video from a number of cameras placed around the actor, and use image processing to determine the pose. For best results, cameras with a high resolution and high frame rate should be used, because it improves the timing, accuracy and robustness of the image processing, all of which are crucial to the quality of the end result. The best systems on the market today use reflective markers on the actor and infra-red flashes from the camera to make the markers stand out against the background. Some systems use active infra-red LED markers to be able to resolve which marker is which for a complicated pose. Range cameras like the Microsoft Kinect can capture the pose of a human being without special markers, using only a single image and some smart image processing algorithms, although with limited resolution and accuracy.

Accelerometers

With the advent of smartphones, sensors have been developed that measure tilt and acceleration with reasonable accuracy at a very low cost. This kind of sensors have been used successfully for motion capture. The precision is not as good as for camera-based methods, but accelerometer-based systems don't require the actor to stay within the field of view of a camera, and they can generate measurement data at a very high rate.

5.5.2 Track motion capture

Full-body motion capture using complicated and expensive equipment is so common that one often forgets an important special case which is a lot cheaper. Motion capture can be performed by something as simple as mouse (x, y) input being mapped to the movements of the head or the eyes of a character. An animator can create several such tracks in the animation and use them together to create a convincing life-like motion.

This kind of motion capture can be extended to making a puppet with a few sensors in it, and have an actor move the puppet instead of acting in person. The ability to act through a puppet is a universal human trait. Professional puppeteers have trained to control hand puppets with special controllers to perform several different motions simultaneously, and hand puppets are often controlled by several people acting together. While learning to express both motion and emotion through an abstract controller can be difficult, it is still a useful, low cost method for creating animation. The simple input interface makes the animation data immediately available, which means that a virtual puppet controlled by one or more people can respond in real time to a live audience.

5.5.3 Lip sync

A special case that deserves mention is *lip sync*, the art of making a virtual character move its mouth as if it were saying the words that have been recorded from a voice actor. Lip sync can be done entirely by key framing with good quality and reasonable speed, but in some cases automatic methods are preferred. There may be lots of dialogue, like in a game with lots of alternate responses for a large number of characters. In such situations, there are now good voice recognition algorithms which can simply listen to what is being said and create reasonably good mouth movements accordingly. Facial animation requires a lot more than lip sync data, for example facial expressions, changes in the direction of gaze and movements of the head, but lip sync is the fastest motion, and the motion that is most tedious to animate with manual methods.

If the speech is not pre-recorded but streamed in real time, voice recognition methods do not perform well, even if considerable processing power

is spent on doing it quickly. The reason for this is that the mouth moves in anticipation of the phoneme that comes next. The mouth looks different for the ”s” in ”so” than for the ”s” in ”see”, and the change of expression to pronounce the vowel happens slightly *before* the vowel is heard. A solution is of course to add a slight delay to the audio stream to allow time both for the audio processing and the anticipation in the lip sync.