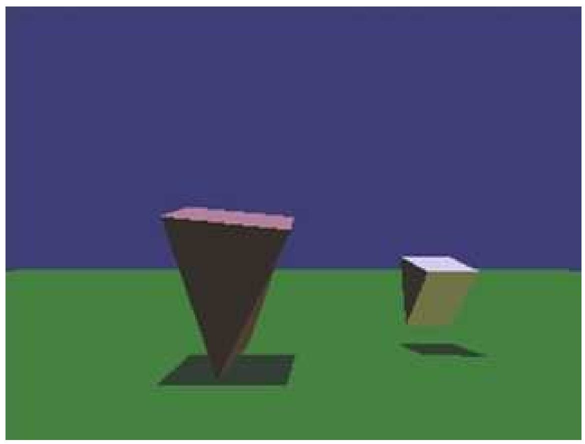
15 Animation

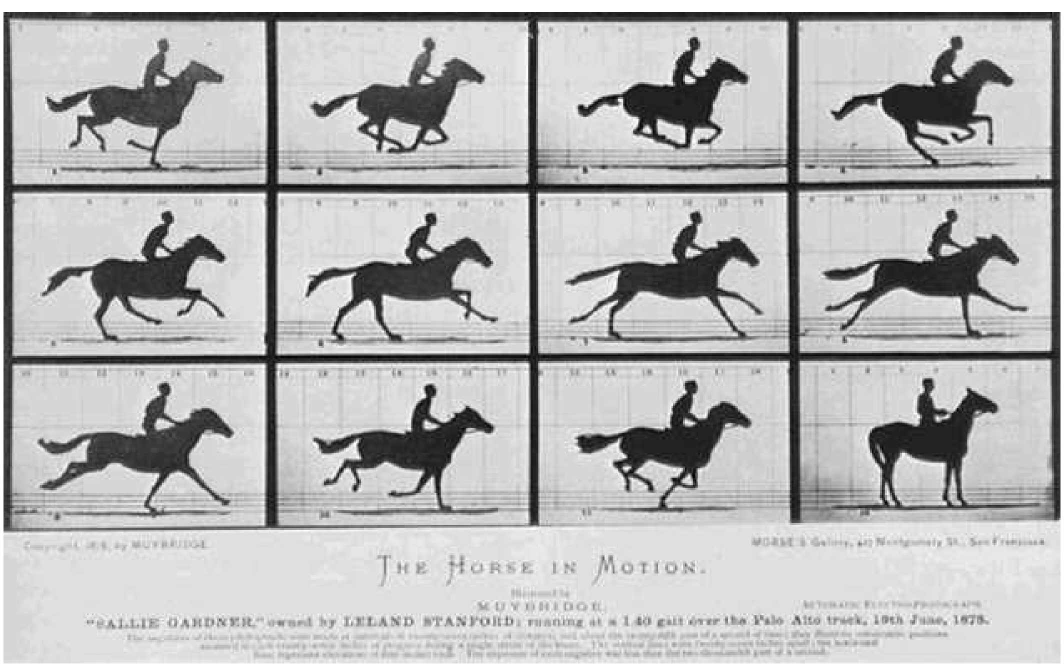
15.1 Overview

Motion can bring the simplest of characters to life. Even simple polygonal shapes can convey a number of human qualities when animated: identity, character, gender, mood, intention, emotion, and so on.



Very simple characters (image by Ken Perlin)

A movie is a sequence of frames of still images. For video, the frame rate is typically 24 frames per second. For film, this is 30 frames per second.



In general, animation may be achieved by specifying a model with n parameters that identify degrees of freedom that an animator may be interested in such as

- polygon vertices,
- spline control,
- joint angles,
- muscle contraction,
- camera parameters, or
- color.

With n parameters, this results in a vector $\vec{q}$ in n-dimensional state space. Parameters may be varied to generate animation. A model's motion is a trajectory through its state space or a set of motion curves for each parameter over time, i.e. $\vec{q}(t)$, where t is the time of the current frame. Every animation technique reduces to specifying the state space trajectory.

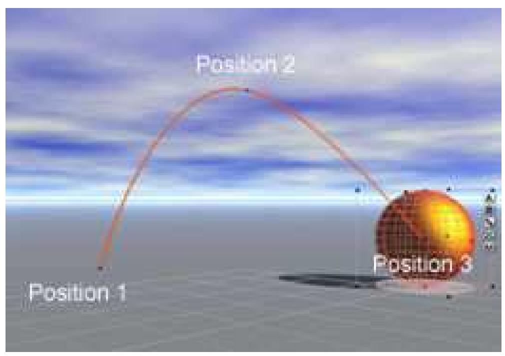
The basic animation algorithm is then: `for` $t{=}t_1$ `to` t_{end}: `render(`$\vec{q}(t)$`)`.

Modeling and animation are loosely coupled. Modeling describes control values and their actions. Animation describes how to vary the control values. There are a number of animation techniques, including the following:

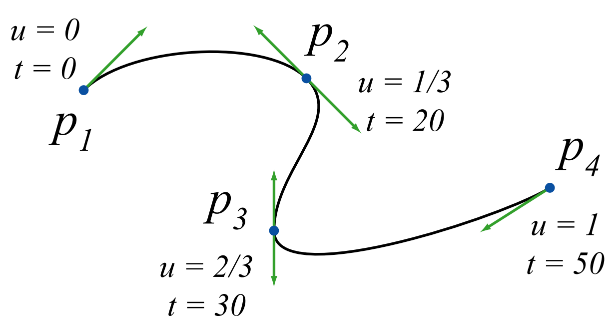
- User driven animation
  - Keyframing
  - Motion capture
- Procedural animation
  - Physical simulation
  - Particle systems
  - Crowd behaviors
- Data-driven animation

15.2 Keyframing

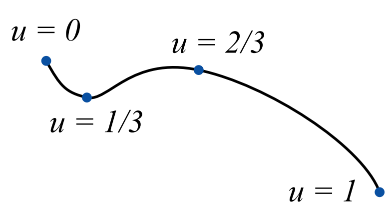
Keyframing is an animation technique where motion curves are interpolated through states at times, $(\vec{q}_1, ..., \vec{q}_T)$, called keyframes, specified by a user.



Catmull-Rom splines are well suited for keyframe animation because they pass through their control points.



One problem is that the spline control does not correspond to time. Instead we have $p(u)$, the position with regard to the spline control, when we need $p(t)$, the position with regard to time.



The solution is to determine $u(t)$, the spline control with regard to time. Then $p(t) = p(u(t))$. We can go even further: define $s(t)$ to be the length traveled at a given time, and let $u(s(t))$ be length with regard to the spline control. Then $p(u)$ is the position with regard to the spline control, and $p(t) = p(u(s(t)))$. We can use the arc length (distance traveled along a curve) formula: $s(t_1, t_2) = \int_{t_1}^{t_2} \left|\frac{dp}{dt}\right| du$.

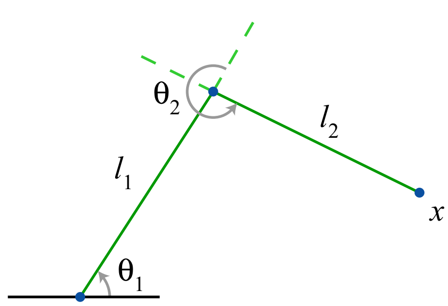
- Pros:
  - Very expressive
  - Animator has complete control over all motion parameters
- Cons:
  - Very labor intensive
  - Difficult to create convincing physical realism
- Uses:
  - Potentially everything except complex physical phenomena such as smoke, water, or fire

15.3 Kinematics

Kinematics describe the properties of shape and motion independent of physical forces that cause motion. Kinematic techniques are used often in keyframing, with an animator either setting joint parameters explicitly with **forward kinematics** or specifying a few key joint orientations and having the rest computed automatically with **inverse kinematics**.

15.3.1 Forward Kinematics

With forward kinematics, a point $\bar{p}$ is positioned by $\bar{p} = f(\Theta)$ where Θ is a state vector $(\theta_1, \theta_2, ...\theta_n)$ specifying the position, orientation, and rotation of all joints.



For the above example, $\bar{p} = (l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2), l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2))$.

15.3.2 Inverse Kinematics

With inverse kinematics, a user specifies the position of the end effector, $\bar{p}$, and the algorithm has to evaluate the required Θ give $\bar{p}$. That is, $\Theta = f^{-1}(\bar{p})$.

Usually, numerical methods are used to solve this problem, as it is often nonlinear and either underdetermined or overdetermined. A system is underdetermined when there is not a unique solution, such as when there are more equations than unknowns. A system is overdetermined when it is inconsistent and has no solutions.

Extra constraints are necessary to obtain unique and stable solutions. For example, constraints may be placed on the range of joint motion and the solution may be required to minimize the kinetic energy of the system.

15.4 Motion Capture

In motion capture, an actor has a number of small, round markers attached to his or her body that reflect light in frequency ranges that motion capture cameras are specifically designed to pick up.

With enough cameras, it is possible to reconstruct the position of the markers accurately in 3D. In practice, this is a laborious process. Markers tend to be hidden from cameras and 3D reconstructions fail, requiring a user to manually fix such drop outs. The resulting motion curves are often noisy, requiring yet more effort to clean up the motion data to more accurately match what an animator wants.

Despite the labor involved, motion capture has become a popular technique in the movie and game industries, as it allows fairly accurate animations to be created from the motion of actors. However, this is limited by the density of markers that can be placed on a single actor. Faces, for example, are still very difficult to convincingly reconstruct.

- Pros:
  - Captures specific style of real actors
- Cons:
  - Often not expressive enough
  - Time consuming and expensive
  - Difficult to edit
- Uses:
  - Character animation
  - Medicine, such as kinesiology and biomechanics

15.5 Physically-Based Animation

It is possible to simulate the physics of the natural world to generate realistic motions, interactions, and deformations. **Dynamics** rely on the time evolution of a physical system in response to forces.

Newton's second law of motion states $f = ma$, where f is force, m is mass, and a is acceleration. If $x(t)$ is the path of an object or point mass, then $v(t) = \frac{dx(t)}{dt}$ is velocity and $a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2}$ is acceleration. Forces and mass combine to determine acceleration, i.e. any change in motion.

In **forward simulation** or **forward dynamics**, we specify the initial values for position and velocity, $x(0)$ and $v(0)$, and the forces. Then we compute $a(t)$, $v(t)$, $x(t)$ where $a(t) = \frac{f(t)}{m}$, $v(t) = \int_0^t a(t)dt + v(0)$, and $x(t) = \int_0^t v(t)dt + x(0)$.

Forward simulation has the advantage of being reasonably easy to simulate. However, a simulation is often very sensitive to initial conditions, and it is often difficult to predict paths $x(t)$ without running a simulation—in other words, control is hard.

With **inverse dynamics**, constraints on a path $x(t)$ are specified. Then we attempt to solve for the forces required to produce the desired path. This technique can be very difficult computationally.

Physically-based animation has the advantages of:

- Realism,
- Long simulations are easy to create,
- Natural secondary effects such as wiggles, bending, and so on—materials behave naturally,
- Interactions between objects are also natural.

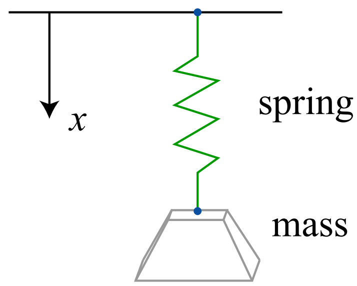
The main disadvantage of physically-based animation is the lack of control, which can be critical, for example, when a complicated series of events needs to be modeled or when an artist needs precise control over elements in a scene.

- Pros:
  - Very realistic motion
- Cons:
  - Very slow
  - Very difficult to control
  - Not expressive
- Uses:
  - Complex physical phenomena

15.6 Spring-Mass Systems

15.6.1 Single 1D Spring

Spring-mass systems are widely used to model basic physical systems. In a 1D spring, $x(t)$ represents the position of mass, increasing downwards.



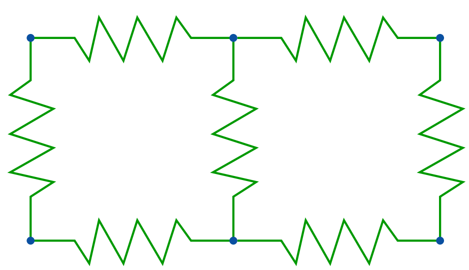
A spring has resting length l and stiffness k. Deformation force is linear in the difference from the resting length. Hence, a spring's internal force, according to Hooke's Law, is $f^s(t) = k(l - x(t))$.

The external forces acting on a spring include gravity and the friction of the medium. That is, $f^g = mg$ and $f^d(t) = -\rho v(t) = -\rho \frac{dx(t)}{dt}$, where ρ is the damping constant.

Hence, the total force acting on a spring is $f(t) = f^s(t) + f^g + f^d(t)$. Then we may use $a(t) = \frac{f(t)}{m}$ with initial conditions $x(0) = x_0$ and $v(0) = v_0$ to find the position, velocity, and acceleration of a spring at a given time t.

15.6.2 3D Mass-Spring Systems

Mass-spring systems may be used to model approximations to more complicated physical systems. Rope or string may be modeled by placing a number of springs end-to-end, and cloth or rubber sheets may be modeled by placing masses on a grid and connecting adjacent masses by springs.



Let the ith mass, m_i, be at location $\bar{p}_i(t)$, with elements $x_i(t)$, $y_i(t)$, $z_i(t)$. Let l_{ij} denote the resting length and k_{ij} the stiffness of the spring between masses i and j.

The **internal force** for mass i is

$$f^s_{ij}(t) = -k_{ij} e_{ij} \frac{p_i - p_j}{\|p_i - p_j\|},$$

where $e_{ij} = l_{ij} - \|p_i - p_j\|$.

