Overview

- *Animation:* Rapid display of slightly different images create the illusion of motion.
- *Traditional approach:* Keyframed animation. Need to consider arc-length parameterization, interpolation of orientation, camera animation.
- *Supporting techniques:* Squish-box deformations and skeletons, motion capture, kinematics and inverse kinematics.
- *Other animation approaches:* Physically-based dynamics, constraint-based animation, procedural animation, behavioral animation.

Traditional 2D Cel Animation

An overview of 2D cel animation, introducing the concept of keyframing. Some discussion of the "art" of animation, á la Disney.

2D Cel Animation

To Do

Keyframes and Inbetweening

The Art of Animation

Issues in Automated Keyframed Animation

A general overview of keyframed animation to ferret out the issues: we end with a discussion of transformation matrix animation and show why and how it fails. This motivates the rest of the module.

Automated Keyframing

- Replace the human inbetweener with interpolation algorithms.
- Keyframes correspond to setting of parameters at different points in time.

\+ The computer provides repeatability and automatic management of keyframes.

– Interpolation is not as smart as a human inbetweener. The animator may have to provide more key frames and/or additional information to get a good result.

Utility of Key Parameters

A good keyframe animation computer system limits the number of key parameters and maximizes the utility of each.

- Parameters should have immediate geometric or visible significance. Entries in a transformation matrix are *not* good parameters.
- Properties of objects that are static should be maintained automatically.
- Relationships between parts of articulated objects should be maintained automatically.
- Interpolation routines should be optimized to particular parameter types. Motion paths, orientation, camera attitude, and surface properties all require different types of control.
- A real-time interface should be provided to interactively set parameters and iteratively improve the animation.

Functional Animation

- An independent scalar function is specified for each "keyframed" parameter.
- Functional animation is a basic capability that supports all others.
- Most useful for truly scalar quantities: brightness of a light source, for example.
- Splines are useful for representing functions.
- Continuity control is a must: both its automatic maintenance and selective breaking.
- The functions can be edited both *explicitly*, as graphs, or *implicitly*, through a direct manipulation interface.

Linear Interpolation

- The most basic interpolation technique for supporting animation is linear interpolation, or the *lerp*.
- Given two parameters p_0 and p_1 at times t_0 and t_1, an intermediate values if given by

$$p(t) = \frac{t_1 - t}{t_1 - t_0} p_0 + \frac{t - t_0}{t_1 - t_0}$$

- Disadvantage: Discontinuities in derivative exist at all key frame point.
- Advantage: The rate of change within a segment is constant (and so can easily be controlled).

Velocity Control

- Given a constant rate of change, we can create a specific *variable* rate of change.
- Given an function $\tau = f(t)$, reparameterize with τ.
- For the lerp,

$$p(\tau) = p(f(t)) = \frac{f(t_1) - f(t)}{f(t_1) - f(t_0)} p_0 + \frac{f(t) - f(t_0)}{f(t_1) - f(t_0)} p_1$$

Spline Interpolation

- Instead of linear interpolation, spline interpolation can be used.
- Advantages:
  - Continuity control can be obtained.
  - Fewer keyframes may be required for a given level of "quality."
- Disadvantages:
  - Extra information may be required at keyframes, such as tangent vectors.
  - Splines are more expensive to evaluate.
  - Splines are more difficult to implement.

Spline Types and Animation

Different spline types will have different tradeoffs and capabilities:

- *B-splines:* Give automatic continuity control, but only approximate their control points.
- *Nonuniform B-splines:* Needed to break continuity.
- *Nonuniform rational B-splines (NURBS):* Provide additional local control over the shape of a curve.
- *Bézier splines:* Require additional information for each segment and do not automatically maintain continuity. However, they do interpolate the ends of segments.
- *Hermite splines:* Require tangent vectors at each keyframe.
- *Catmull-Rom splines:* Provide only first derivative continuity.
- *Beta-splines:* Provide automatic continuity, control over the tangent and approximate arc length of each segment.

Transformation Matrix Animation

- One way to support an animation capability: interpolate a transformation matrix.
- Keyframes would be "poses" of objects given by the animator.
- Functional animation would be applied independently to all entries of the transformation matrix.
- Disadvantage: This does not support animation of rigid bodies.
  Suppose two poses are given of an object which differ by a 180° rotation. Under linear interpolation, the object will not rotate but will turn inside out, collapsing to a plane in the middle of the interpolation.

Rigid Body Animation

- Rigid body transformations only have 6 degrees of freedom, while general affine transformation have 12.
- To obtain good interpolation we have to consider separately the three degrees of freedom that result from translation and the three that result from orientation.

Motion Path Animation

Motion path animation in the context of spline interpolation of position. Deals mostly with problems of arc-length-reparameterization and velocity control.

Motion Path Overview

As a basic capability, we would like to translate a point through space along a given path. We would like:

- Independent control of velocity along path.
- Continuity control.

While splines can easily support continuity control, *velocity* control is more difficult.

Spline Interpolation Problems

Spline position interpolation will give us continuity control over changes position, but:

- An equal increment in the spline parameter does not usually correspond to an equal increment in distance along the spline.
- Different segments of the spline with the same parametric length can have different physical lengths.
- If we parameterize the spline directly with time objects will move an a nonuniform speed.

Arc Length Parameterization

1. Given a spline path $P(u) = [x(u), y(u), z(u)]$, compute the arclength of the spline as a function of u:$s = A(u)$.
2. Find the inverse of $A(u)$: $u = A^{-1}(s)$.
3. Substitute $u = A^{-1}(s)$ into $P(u)$ to find a motion path parameterized by arclength, s: $P(s) = P(A^{-1}(s))$.

Note that u (and thus s) should be global parameters, extending across all segments of the original spline.

Velocity Control

To control velocity along a spline motion path,

- Let $s = f(t)$ specify distance along the spline as a function of time t.
- The function $f(t)$, being just a scalar value, can be supported with a functional animation technique (i.e., another spline).
- The functions $f(t)$ may be specified as the integral of yet another function, $v(t) = df(t)/dt$, the *velocity*. Note that the integral of a spline function can be found analytically, through a manipulation of the control points.
- The motion path as a function of time is thus given by $P(t) = P(f(t)) = P(A^{-1}(f(t)))$.

Some Nasty Problems

There are some nasty problems with the arc-length parameterization approach:

1. The arc-length $s = A(u)$ is given by the integral

$$s = A(u) = \int_0^u \sqrt{\left(\frac{dx(v)}{dv} \right)^2 + \left(\frac{dy(v)}{dv} \right)^2 + \left(\frac{dz(v)}{dv} \right)^2} dv$$

   which has no analytic solution if the motion path is a cubic spline.
2. Since $A(u)$ has no analytic form, $A^{-1}(s)$ has no analytic form.

We have to use numerical techniques:

1. Find $A(u)$ with numerical quadrature. Precompute some values of s, at least at the boundaries of segments, to make this easier.
2. Since s is a monotonic function of u, for a given s^* we can find the solution u^* to $s^* = A(u^*)$ by bisection (binary search).

Real Time Issues

- Exact arc-length parameterization may not be feasible.
- An alternative: compute points on the spline at equally-spaced parametric values, and use linear interpolation along these chords.
- The linear interpolation should consider the distance between samples to maintain constant velocity.

Animating Camera Motion

Animation issues specifically dealing with camera motion. Some cinematic perspectives are developed through a look at the terminology used to describe camera motion.

Overview of Camera Animation

While cameras could be animated using a combination of motion path and orientation interpolation tools, the requirements for camera motion are somewhat different:

- The camera should always be level, unless we specify otherwise.
- The image of objects of interest should be stable on the film plane.
- Specification of camera motion has a long cinematic tradition that should be respected.

Cinematics of Camera Motion

- An animated zoom changes perspective, which can be disturbing and/or distracting. A dolly should be used instead to enlarge the image of an object of interest.
- The camera should almost never be rotated about its view direction—unless a sea sick audience is the objective.
- Changing the focal depth can be used to track objects of interest, but a sophisticated renderer is required to simulate focus.
- Depth of field is rarely changed in real cinematography due to technical limitation, but might be useful in computer animation.
- Smooth spline animation should almost always be used for camera animation. Quick camera moves should normally be replaced by cuts (instantaneous changes of scene) unless a special effect is desired.
- Animation of spotlight sources is similar to the animation of a camera. Support of a "lights'-eye-view", therefore, is often useful.

Terminology of Camera Motion

Dolly: Move forward, along the line of sight of the camera (towards the object of interest).

Track: Move horizontally, perpendicular to the line of sight of the camera. More generally, move in a horizontal plane.

Crane: Raise the camera vertically.

Tilt (Bank): Rotate about the horizontal axis perpendicular to the line of sight of the camera.

Pan (Yaw): Rotate about the vertical axis of the camera (after tilt).

In addition, cameras have other parameters that can be animated:

Zoom (in, out): Change the angular field of view of the camera.

Focus: Change the focal depth, i.e., the distance at which objects are in focus.

Depth of Field: Changing the aperture size has the effect of changing the depth of field, i.e., the range of depths over which objects can be considered to be in focus.

Specification of Camera Motion

Since with a camera we are usually more interested in what it's looking at than its exact orientation, camera animation may be specified by

1. A motion path for the camera itself.
2. A motion path for a look-at point (possibly derived from the motion of an object of interest).

The camera would be moved along the motion path and the orientation of the camera would be determined by the look-at point. Ideally, the "focus would be pulled" to match the distance to the object of interest at the look-at point. Tilt might be constrained to lie within certain bounds.

Skeletons

- Given a model with a large number of vertices, vertices can be grouped together and manipulated as a unit. For example, in an arm, all points describing the forearm move more-or-less as a rigid body.
- The connectivity between groups of parameters can be represented by tying each group to a "bone" and arranging the bones in an articulated "skeleton."
- The bone and all its associated vertices are treated as a single rigid object for the purposes of transformation.
- Movement of the bones is constrained by the joints in the skeleton.
- Different kinds of joints (revolute, hinge, ball) can support specific types of motion.
- An animation can be previewed using only the bones, speeding rendering time.
- In the final rendering, the model is shown without bones.
- As an enhancement, vertices may be influenced by more than one bone in a weighted fashion. This results in a more flexible and realistic surface near joints. For example, a kneecap and its skin surface might be affected by both the femur and the shinbone, and assume a position halfway between either.

Free-Form Deformations

- Sometimes, a skeleton cannot capture the desired shape change of an object. An example is the "squash and stretch" of a bouncing ball as it hits the ground. Such global changes in shape can be expressed using a *deformation*.
- A deformation changes the *space* around an object with a nonlinear transformation. New coordinates for every point in a space are determined as functions of the old coordinates.

1. A rectangular "squish box" is usually placed around a part of the model that needs to be animated.
2. The coordinates of all points within the box are determined relative to the frame given by the box. Suppose a vertex P can be expressed as $[u, v, w]$ relative to the coordinates of the box.
3. The new coordinates of P are given by a tensor product Bézier spline $B^{n_1,n_2,n_3}(u, v, w)$ with control points $P_{i,j,k}$.
4. If the $P_{i,j,k}$ have coordinates

$$P_{i,j,k} = [i/(n_1 + 1), j/(n_2 + 1), k/(n_3 + 1)],$$

the transformation is given by the identity:

$$[u, v, w] = B^{n_1, n_2, n_3}(u, v, w).$$

Normally, then the control points are initially set to these values and are moved to effect a deformation.

5. Continuity conditions can be enforced. For example, if a hand is being deformed but the arm is not, the control points along the edge of the box that cuts the wrist should not be moved, nor the next layer. This maintains both position and derivative continuity across the wrist.
6. The object should be finely tesselated or radical deformations will not work properly. For realistic animation, typically only small deformations are used.

Using FFDs and Skeletons Together

Skeletons and free-form deformations can be used simultaneously. The number of control points in a free-form deformation can be large, but if they are moved in groups relative to a skeleton, excellent animation control can be obtained for arbitrary models.

Kinematics and Inverse Kinematics

Kinematics: The study of motion independent of the forces that cause the motion. Includes position, velocity, acceleration.

Forward kinematics: The determination of the positions, velocities, and accelerations of all the links in an articulated model given the position, velocity, and acceleration of the root of the model and all the transformations between links. Forward kinematics is a necessity for skeletal keyframed animation, and is, fortunately, easy to implement.

Inverse kinematics: The derivation of the motion of intermediate links in an articulated body given the motion of some key links.

Inverse Kinematics

- Unlike forward kinematics, inverse kinematics is often nonlinear, underdetermined or overdetermined, and possibly ill-conditioned.
- The complexity of determining a solution to an inverse kinematic model is proportional to the number of free links.
- One free joint between two fixed ones can be solved fairly efficiently, for example, with only one space degree of freedom.
- Extra constraints may be needed to obtain a unique and stable solution. For example, a joint may be required to point downwards in a gross approximation to gravity if it is not otherwise constrained. Joint motion constraints (hinge vs. revolute vs. ball) are also useful.
- An optimization objective can be used instead (or in addition to) extra constraints. The resulting optimization problem can be solved iteratively as an animation proceeds.

  Example optimization objectives: minimize the kinetic energy of a structure, minimize the total elevation of a structure, minimize the maximum (or average) angular torque.

Physically-Based Animation

Use of physical simulation for animation.

Comparison with keyframed animation, derivation of an example using a spring-mass system.

Physically-Based Animation

Idea: To obtain a physically plausible animation, *simulate* the laws of Newtonian physics acting on objects in the environment. In contrast to kinematics, this is called a *dynamics* approach.

- Have to control objects by manipulating forces and torques, either directly or indirectly.
- Animated objects may be passive: bounding balls, Jello, wall of bricks falling down, etc.

Because of the difficulty of inverse dynamics (finding solutions for forces to move a physical object along a desired path) most current applications are of the "passive" variety. This is still useful for secondary effects in a keyframed animation; e.g.,

- a mouse character's ears should flap when it shakes its head
- a T. Rex's gut should sway while it runs
- a feather in a cap should wave when wind blows on it
- cloth should drape around an actor and respond to the actor's movements

- a wall of bricks should fall down when a superhero crashes into it

The only problem with such simulated secondary effects is that they can make bad keyframed animation look even worse!

Simulation

Setting up and solving a physically-based animation proceeds as follows:

1. Create a model with physical attributes: mass, moment of inertia, elasticity, etc.
2. Derive differential equations by applying the laws of Newtonian physics.
3. Define initial conditions, i.e., initial velocities and positions.
4. Supply functions for external forces (possibly via keyframing).
5. Solve the differential equations to derive animation, i.e., motion of all objects in scene as a function of time.

During the solution of the differential equations we have to be prepared to deal with discontinuities due to collisions. As an example, we will discuss the simulation of spring-mass and point mass models. Both have useful applications and are relatively simple to simulate.

Spring-Mass Models

1. Create a model:
   - A spring-mass model is composed of a mesh of point masses m_i connected by springs with spring constants k_{ij} and rest lengths l_{ij}.
   - Let $P_i(t) = [x_i(t), y_i(t), z_i(t)]$ be the position of mass m_i at time t.
   - Let N_i be the set of all indices of masses which are connected to mass m_i. If $j \in N_i$, then m_i and m_j are connected by a spring.
   - The springs exert a force proportional to their displacement from their rest length, and in a direction which tends to restore the rest length:

$$\begin{aligned} \vec{f}^{s}_{ij}(t) &= -k_{ij}(l_{ij} - |x_i(t) - x_j(y)|) \ \frac{x_i(t) - x_j(t)}{|x_i(t) - x_j(t)|} \\ \vec{f}^{s}_{i}(t) &= \sum_{j \in N_i} \vec{f}^{s}_{ij}(t) \end{aligned}$$

   - The masses are assumed to be embedded in a medium which provides a damping force

of

$$\vec{f}_d = p_i \vec{v}_i(t)$$

where $\vec{v}_i(t)$ is the velocity of m_i at time t. (Damping could also be provided on the derivative of the change in rest length of the springs, which would be coordinate-system independent.)

2. The motions of each mass is governed by the following second order ordinary differential equation:

$$m_i \vec{a}(t) = -p_i \vec{v}_i(t) + \vec{f}_i^{\,s}(t) + \vec{f}_i^{\,e}(t)$$

where $\vec{f}_i^{\,e}(t)$ is the sum of all external forces acting on node i and

$$\vec{a} = \left[\frac{d^2 x_i(t)}{dt^2}, \frac{d^2 y_i(t)}{dt^2}, \frac{d^2 z_i(t)}{dt^2} \right],$$

$$\vec{v} = \left[\frac{dx_i(t)}{dt}, \frac{dy_i(t)}{dt}, \frac{dz_i(t)}{dt} \right]$$

3. Initial conditions: user supplies initial positions of all masses and velocities.
4. The user supplies external forces: gravity, collision forces, keyframed forces, wind, hydrodynamic resistance, etc., as a function of time.

5. Simulation.
   - Factor second-order ODE into two coupled first-order ODEs:

$$\begin{aligned}
\vec{v} &= [v_{xi}(t), v_{yi}(t), v_{zi}(t)] \\
&= \left[\frac{dx_i(t)}{dt} , \frac{dy_i(t)}{dt} , \frac{dz_i(t)}{dt} \right] \\
\vec{a} &= \left[\frac{dv_{xi}(t)}{dt} , \frac{dv_{yi}(t)}{dt^2} , \frac{dv_{zi}(t)}{dt} \right] \\
&= \frac{1}{m_i} \left(-p_i \vec{v}_i(t) + \vec{f}_i^{\,s}(t) + \vec{f}_i^{\,e}(t) \right)
\end{aligned}$$

   - Solve using your favorite ODE solver. The simplest technique is the Euler step: pick a Δt. Then compute the values of all positions at $t + \Delta t$ from the positions at t by

discretizing the differential equations:

$$\begin{aligned}
\vec{a}_i(t+\Delta t) &\leftarrow \frac{\Delta t}{m_i}\left(-p_i\vec{v}_i(t) + \vec{f}_i^{\,s}(t) + \vec{f}_i^{\,e}(t)\right) \\
\vec{v}_i(t+\Delta t) &\leftarrow v_i(t) + \vec{a}(t)\Delta t \\
P_i(t+\Delta t) &\leftarrow P_i(t) + \vec{v}(t)\Delta t
\end{aligned}$$

Collisions

- The simulation may be interrupted by a collision. When this occurs, a collision force should be generated to reflect the momentum of the colliding masses. It may be necessary to use a Δt much shorter than the frame time and/or back up the simulation after interpenetration to handle collisions reasonably robustly.
- As another example, orbital mechanics can be simulated using gravitational forces between all point masses, and no damping or spring forces. For a large number of point masses this can be computationally expensive.

Alternative

A collection of alternative animation techniques and technologies. Currently only motion capture is discussed, but behavioral and procedural animation should also be included.

Motion Capture

Rather than setting keyframes manually, the data can be captured from real objects, creatures and/or actors performing the desired motion. The positions of some key points on the actor are tracked and these motions are mapped to corresponding points in the model. It may often be necessary to generate secondary actions for the computer model, as tracking is expensive. Tracking technologies include:

1. Electromagnetic position and orientation sensors. Requires a wire to each sensor. Readings can be distorted by metal and magnetic fields. Limited working volume.
2. Ultrasonic rangefinder triangulation. Cheaper but less accurate than magnetic sensors.
3. Optical triangulation. Reflective or emissive markers are attached to objects, then two cameras triangulate position. Can track more points cheaply, without wires, but points can be occluded.
4. Body suit. Fibers in instrumented clothing detect angles of all joints.
5. Rotoscoping. Arbitrary video from two cameras is converted to motion paths either through manual selection of key points or use of computer vision. Prone to error and occlusion. Tedious if done manually.

6. Puppetry. Real-time data is collected from an arbitrary input device an mapped to a computer model, which is displayed in real time during capture. A special-purpose input device in the shape of the object being animated may be built.

Problems with Motion Capture

Motion capture tends to require more-or-less exotic technology. Even puppetry requires at least real-time graphics. Problems with motion capture include:

1. How do we map motion paths to objects of a different shape and scale (the ostrich-to-T. Rex problem)?
2. How do we deal with noise and missing data?
3. How can we reduce the number of samples?
4. How can we generalize motion (i.e., We have a walk and a turn, how do we get an actor to follow a path)?

Reading Assignment and News

See Physically Based Models and Procedural Techniques pg 512 - 513 in Recommended Text.

Please also track the News section of the Course Web Pages for the most recent Announcements related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics25/cs354/)