Lecture 05: Animation

I. Animation loop/concepts
   A. Clear → Draw → Show

   Update → Clear → Draw → Swap Buffers

   Change the world

   B. Double buffering - 2 refresh buffers: one shown and one for drawing.
      1. Animation - any time-sequence visual change in a picture
         i. Real-time - Viewed exactly at a rate capable of human + device refresh rate
         ii. Frame-by-Frame - Hypothetical offline generation then glued together in video
      D. Key-Frame - Detailed drawing at a certain time in animation

      key-frames for "Squash + Stretch" of a ball

      Timing

   E. Inbetweens - intermediate frames between key frames
      Example - total frame rate is 24 FPS for 1440 frames, so if 88 key frames desired then 58 inbetweens are added between key frames.
      Determined through interpolation - linear, cubic, Bézier
      Ch. 14.

II. Overview of animation techniques
   A. Squash and Stretch - for acceleration on non-rigid objects
   B. Timing - Strategic spacing in time between key frames. Close frames get slower motion.
   C. Morphing - transformation of shapes from one to another
   D. Cogwheel - spacing by track
   D. Motion capture - recording of human actors for motion
   F. Motion - repeated patterns, e.g., rotating object

III. Motion specification techniques
   A. Direct motion specification - directly specify all transformations or approximating through an equation, e.g.,

\[ y(x) = A \sin(wx + \theta) e^{-kx} \]  

(damped, rectangular, curve)

Bouncing ball
B. Goal-directed Systems - define tasks like "walk" or "run" or "pick-up"

Can include transition diagrams, e.g. state-machines

C. Kinematics and dynamics

i. Kinematics - give motion parameters of position, velocity, and acceleration
ii. Inverse kinematics - give start and goal, then define parameters
iii. Dynamics - specify forces on object, e.g. physically-based modeling (see below)

iv. Inverse dynamics - determining forces from initial/terminal positions

v. Newton's law

\[ F = \frac{d}{dt} (mv) \]
\[ F = ma \]

(D) Inertial frame - repeated motions, e.g. rotating object in series or flying motions

E. Articulated Figures

ii. Example

III. Articulated Figure - hierarchical model defining skeleton, then "skin" is wrapped around skeleton (beyond scope of class)

IV. Elecroen

A. Hierarchical models (Chapter 11)

- Model - single representation, e.g. one geometry

- Instance - one specific object of that model

- Hierarchical model - tree-based description or model system

- Ex. Tractor

\[ W = T_f R_f T_w R_w \]

- Degree of freedom - the axis or motion, e.g. wheel rotation, tractor x,y,z

- Keyframe defined as vector of

IV. *In groups of three*

Define hierarchical model they same system for 3D Stick Figure
IV. Particle Systems (Chapter 23-3)

A. Physically-based modeling - equations governing behavior of objects

1. Example - gravity on point

   Position: \( x \)  
   Velocity: \( v = \dot{x} \)  
   Acceleration: \( \ddot{x} = \dddot{x} \)

   as Forces

   \[ F = m \dddot{x} \]
   \[ \dddot{x} = \frac{F}{m} \]

   gravity: \( m \dddot{x} = \ddot{x} \)

   \[ \ddot{x} = g \]

   \[ \dddot{x} = \frac{F}{m} = \frac{m \dddot{x}}{m} = g \]

   \[ v = \int_0^t \dot{x} \, dt = v_0 + \dot{x} \, t \]

   \[ x = \int_0^t (v_0 + \dot{x} \, t) \, dt = x_0 + v_0 \, t + \frac{1}{2} \ddot{x} \, t^2 \]

ii. Euler Integration - many systems too complicated to solve equations explicitly, also that is slow. So we trade some time for error.

b. Derivation:

   Let \( \Delta t \) be a step in time.

   \[ v^{n+1} = v^n + \Delta t \dot{x} \]
   \[ x^{n+1} = x^n + \Delta t v^n \]

   written more generally:

   \[ \Delta \vec{s} = \vec{s}^{n+1} - \vec{s}^n \] where \( \vec{s} \) is state, for example:

   \[ \vec{s} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \]

   \[ \vec{s} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \]

   So you determine forces, solve for \( \dddot{x} \) and euler integrate.

   a. Air force \( \vec{F}_a = \vec{v} \)

   b. wind force \( \vec{F}_w = v \ddot{x} \)

   c. Total \( \vec{F}_t = \vec{F}_a + \vec{F}_w \) etc.

   d. Basic loop:

   1. Set initial conditions
   2. For \( t = 0 \) to total time step \( \Delta t \) do
   3. Compute acceleration
   4. Euler Integrate state
   5. Update state
   6. Draw frame

   e. With collisions loop gets much more complicated. See me for more details, or if you would like to handle

   f. Bouncing - when an object gets close to stopping, just stop simulating it. To save computation time. Avoid "perpetual" effect. Many stopping criteria.

   \[ \vec{v} \ll \vec{v}_0 \]

   \[ \vec{v} \ll \vec{v}_0 \]

   \[ \vec{v} \ll \vec{v}_0 \]
8. Particle systems
   a. Set of point masses under forces. Note: they do not interact at each other.
   b. Pre-allocate memory for system, e.g., array of 10,000 cells.
   c. Used for simulating effects, e.g., water, fire, smoke, etc. in physics N-body systems.
   iv. State:
      - Each particle has position \( \mathbf{x} \) and velocity \( \mathbf{v} \), but can also have:
        - Age, color, temperature, opacity, etc. Whatever you want to simulate.
   v. Initial conditions:
      - Want lots of particles that vary in size, but also want to regenerate particles after they die.
      - Don't generate more than you preallocated for.

a. Generators:
   - A value/vector creator based on distribution.
   b. Distributions:
      - Many available, but two simplest:
        - Uniform:
          - \( \mathbf{x} \) - mean
          - \( \sigma \) - standard deviation
          - Each is equally likely in a range.
        - Gaussian (normal):
          - \( \mu \) - mean
          - \( \sigma \) - standard deviation
          - More toward central value.

b. Point generator:
   - \( \mathbf{x_0} = \mathbf{0} \) (start point)
   - \( \mathbf{v_0} = \) random direction \( \times \mathbf{v} \)
   - \( \mathbf{v} \) - velocity

C. Directed generator:
   - \( \mathbf{x_0} = \mathbf{0} \)
   - \( \mathbf{v_0} = (\mathbf{n} + \text{small random direction}) \times \mathbf{v} \)

0. Disc:
   - \( \mathbf{v_0} = (r, \theta) \) (random point on circle)
   - \( \mathbf{v_0} = \) at direction away from disk normal

- Many more combinations, get creative!
VII. Choreography of forces - beyond gravity/wind

a. Gravity toward a point

\[ \vec{F} = -\frac{G m_1 m_2}{r^2} \hat{r} \]

Can:
1. Ignore second mass \(-\frac{m_2}{r^2} \hat{r}\)
2. Be proportional to \(-\frac{m_2}{r^2} \hat{r}\)

b. Repel from point

\[ \vec{F} = \frac{G m_1 m_2}{r^2} \hat{r} \]

C. Anything you can imagine
- Towards/away from ins/surfaces
- Potential foibles
- Etc.

C. Flocking - point masses that interact

D. Spring-mass systems - point mass connected by simulated springs

E. Rigid-body systems - volumes pick, force + torque