**Notes**

- Textbook: matchmove 6.7.2, B.9

**Match Move**

- For combining CG effects with real footage, need to match synthetic camera to real camera: “matchmove”
- Too unreliable to just measure camera movement mechanically
  - In some shots can actually use computer motor control of camera to follow path
  - Useful for its consistency, but bias makes it useless for match move
- Instead need to estimate camera parameters from footage

**Match points**

- Need to identify image space positions of enough world space points
  - 3 non-collinear if field-of-view known, 4 if not
  - More points can improve robustness
    - Also deal with camera distortions
- Typically identify points by hand
  - For difficult scenes (grass?) may need computer vision techniques, or just put stuff in the scene to track (and paint over later)

**Solving match move**

- Nonlinear equations can be difficult
- Probably need to use optimization to find robust solution from multiple uncertain points
- May use through-the-lens techniques to avoid nonlinearity - except for first frame - or at least to start the nonlinear solver on subsequent frames
- May need interactive help to lock on
  - Enter the matchmove artist
Particle Systems

- For fuzzily defined phenomena, highly complex motion, etc. particle systems provide a (semi-)automatic means of control
- Break up complex phenomena into many (hundreds, thousands, or more) component parts
  - E.g. fire into tiny flames
- Instead of animating each part by hand, provide rules and overall guidance for computer to construct animation

When in doubt...

- Used to model particle-like stuff: dust, sparks, fireworks, leaves, flocks, water spray...
- Also phenomena with many DOF: fluids (water, mud, smoke, ...), fire, explosions, hair, fur, grass, clothing, ...
- Three things to consider:
  - When and where particles start
  - The rules that govern motion (and additional attached variables, e.g. colour)
  - How to render the particles

What is a particle?

- Most basic particle only has a position $x$
- Usually add other attributes, such as:
  - Age
  - Colour
  - Radius
  - Orientation
  - Velocity $v$
  - Mass $m$
  - Temperature
  - Type
- The sky is the limit - e.g. AI models of agent behaviour
Seeding

- Need to add (or seed) particles to the scene
  - Where?
    - Randomly within a shaped volume or on a surface
    - At a point
    - Where there aren’t many particles currently
  - When?
    - At the start
    - Several per frame
    - When there aren’t enough particles somewhere
- Need to figure out other attributes, not just position
  - E.g. velocity pointing outwards in an explosion

Basic animation

- Specify a velocity field $v(x,t)$ for any point in space $x$, any time $t$
- Break time into steps
  - E.g. per frame - $\Delta t = 1/30$th of a second
  - Or several steps per frame
- Change each particle’s position $x_i$ by “integrating” over the time step (Forward Euler)
  $$x_i^{\text{new}} = x_i + \Delta t v(x_i,t)$$

Velocity fields

- Velocity field could be a combination of pre-designed velocity elements
  - E.g. explosions, vortices, ...
- Or from “noise”
  - Smooth random number field
  - See later
- Or from a simulation
  - Interpolate velocity from a computed grid
  - E.g. smoke simulation

Second order motion

- Real particles move due to forces
  - Newton’s law $F = ma$
  - Need to specify force $F$ (gravity, collisions, ...)
  - Divide by particle mass to get acceleration $a$
  - Update velocity $v$ by acceleration
  - Update position $x$ by velocity
  $$v_i^{\text{new}} = v_i + \Delta t \frac{F(x_i,v_i,t)}{m_i}$$
  $$x_i^{\text{new}} = x_i + \Delta t v_i^{\text{new}}$$
**Time integration**

- Really solving ordinary differential equations in time:
  \[
  \frac{dx_i}{dt} = v_i(x_i, t) \quad \text{or} \quad \begin{cases} 
  \frac{dx_i}{dt} = v_i \\
  \frac{dv_i}{dt} = \frac{1}{m_i} F(x_i, v_i, t)
  \end{cases}
  \]

- Methods presented before are called “Forward Euler” and “Symplectic Euler”
  - There are better numerical methods
  - These are the simplest that can work - but big issue is stability - more on this later

---

**Basic rendering**

- Draw a dot for each particle
- But what do you do with several particles per pixel?
  - Add: models each point emitting (but not absorbing) light -- good for sparks, fire, ...
  - More generally, compute depth order, do alpha-compositing (and worry about shadows etc.)
  - Can fit into Reyes very easily
- Anti-aliasing
  - Blur edges of particle, make sure blurred to cover at least a pixel
  - Particle with radius: kernel function

---

**Motion blur**

- One case where you can actually do exact solution instead of sampling
- Really easy for simple particles
  - Instead of a dot, draw a line
    (from old position to new position - the shutter time)
  - May involve decrease in alpha
  - More accurately, draw a spline curve
  - May need to take into account radius as well...