

CHAPTER 7

MOTION AND PARTICLE DYNAMICS

Contents

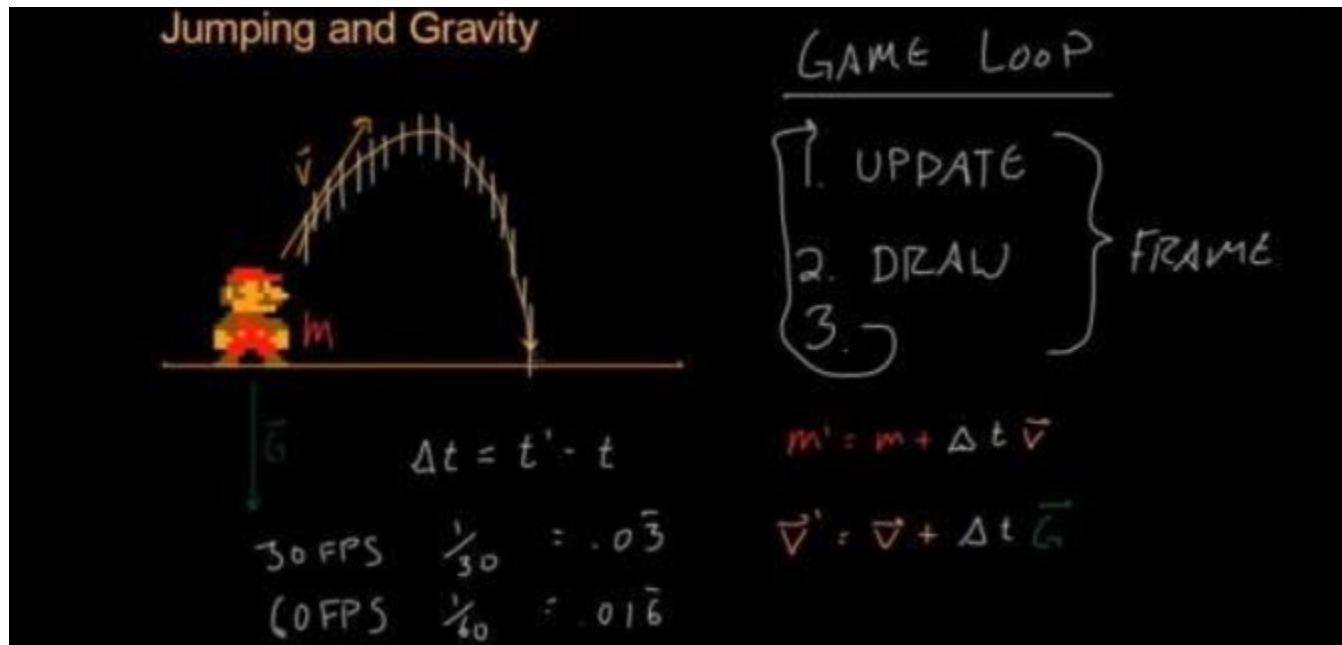
- **Motion detection**
- Motion is games
- Game loop in jumping and gravity
- Velocity vector
- **Motion blur**
- **Particle system**
- Particle simulation
- "Snowflakes" versus "Hair"
- A cube emitting 5000 animated particles

Motion detection

- Motion detection is the process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object.
- It can be achieved by either mechanical or electronic methods.
- When it is done by natural organisms, it is called motion perception.
- Methods of Motion can be detected by monitoring changes in:
 - Infrared light (passive and active sensors)
 - Visible light (video and camera systems)
 - Radio frequency energy (radar, microwave and tomographic motion detection)
 - Sound (microphones, other acoustic sensors)
 - Kinetic energy (triboelectric, seismic, and inertia-switch sensors)
 - Magnetism (magnetic sensors, magnetometers)
 - Wi-Fi Signals (WiFi Sensing)

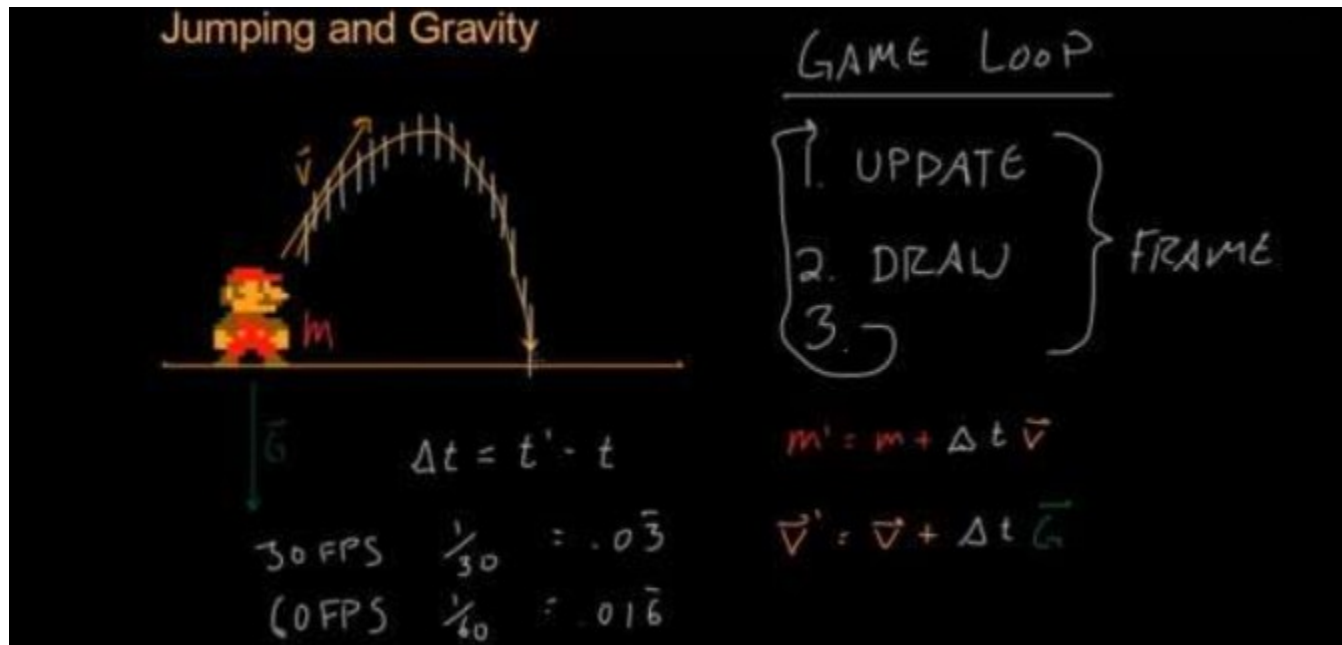
Motion is games

- Let us consider an object where a mario character jumps. It has the ability to jump, get flowers and avoid traps by jumping.
- The path he jumps is curved. The velocity vector $\vec{v}$ is the straight line. The gravity vector $\vec{g}$ makes the mario fall. How $\vec{v}$ and $\vec{g}$ is combined to take the curved path to jump?
- Cut the curved path into the bunch of small pieces. When they are put together it looks like a curved line.
- The mechanism used is game loop.



Game loop in jumping and gravity

- The game loop uses UPDATE for updating position, gravity, velocity, powerups and enemy position, etc..
- Then DRAW is used for drawing new things.
- How long this individual cuts should be? is the question.
- Delta t is the time passes between each frame. t' is t prime and t is the last/previous frame.



Velocity vector

- For example, if the game is running at 30 frames per second (FPS) then delta t is 1/30. For 60 FPS, delta t is 1/60 which is small. Hence the cut tiny pieces have really small numbers.
- Marios current position is m. new position is m prime m'. Delta t cuts the velocity vector in the equation.
- New velocity vector equals all velocity vector summed up with delta t and gravity.

The diagram is divided into two main sections: "Jumping and Gravity" on the left and "GAME LOOP" on the right.

Jumping and Gravity: A Mario character is shown on a ground line, labeled with a red m . A parabolic trajectory is drawn above it, with a velocity vector $\vec{v}$ at the start and a downward arrow labeled $\vec{g}$ representing gravity. Below the ground line, the time interval is defined as $\Delta t = t' - t$. Two calculations for delta t are shown: for 30 FPS, $\frac{1}{30} = .0\bar{3}$, and for 60 FPS, $\frac{1}{60} = .01\bar{6}$.

GAME LOOP: A list of three steps is enclosed in a bracket labeled "FRAME":
1. UPDATE
2. DRAW
3. (a small loop symbol)
Below the loop, two equations are written in red and green: $m' = m + \Delta t \vec{v}$ and $\vec{v}' = \vec{v} + \Delta t \vec{g}$.

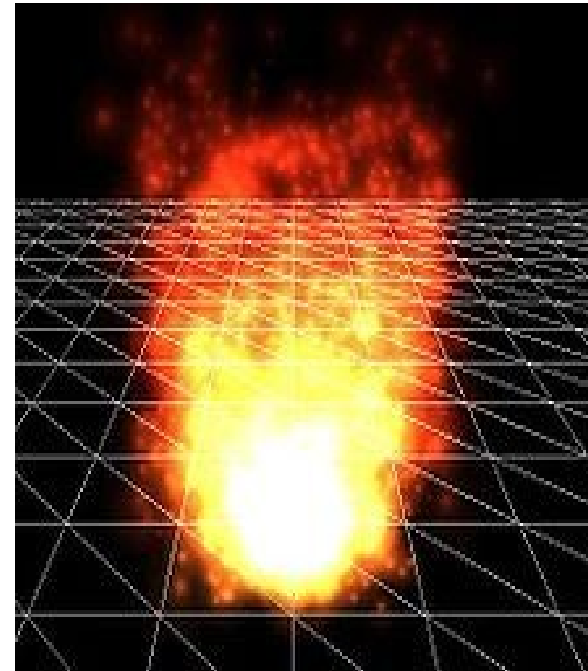
Motion blur

- **Motion blur** is the apparent streaking of moving objects in a photograph or a sequence of frames, such as a film or animation.
- It results when the image being recorded changes during the recording of a single exposure, due to rapid movement or long exposure.
- Earth's rotation causes motion blur in long-exposure photos of the night sky.
- This diurnal motion leaves star trails in exposures like this one taken at La Silla Observatory.



Particle system

- A **particle system** is a technique in game physics, [motion graphics](#), and [computer graphics](#) that uses many minute [sprites](#), [3D models](#), or other graphic objects to simulate certain kinds of "fuzzy" phenomena, which are otherwise very hard to reproduce with conventional rendering techniques - usually highly [chaotic](#) systems, natural phenomena, or processes caused by chemical reactions.
- A particle system used to simulate a fire, created in [3dengfx](#).



Particle simulation

- There is a red particle dust is a mass floating in the breeze. This dust is carried along the wind.
- At each point in the grid there are different velocity vector that specifies which direction the wind is going in

NUMERICAL INTEGRATION

DIFFERENTIAL EQUATION

$v = f(x, t)$

$\{a = \{-g\}$

$[v = -g[t] + v_0] = f(x, t)$

$x = -\frac{1}{2}gt^2 + v_0t + x_0$

$x_0 = x(t_0)$ $t_1 - t_0 = \Delta t$

$v_0 = v(t_0)$

$x_1 = x_0 + v_0 \cdot \Delta t$

$x_{n+1} = x_n + [v_n] \Delta t$

$x_{n+1} = x_n + f(x_n, t_n) \Delta t$

EULER'S METHOD



- These particles are carried along velocity vectors. The node of dust travelling path has to be determined.
- Velocity of a particle at any time is determined with specified location x and the current time t .

NUMERICAL INTEGRATION



DIFFERENTIAL
EQUATION

$v = f(x, t)$

$\{a = \} -g$

$|v = -g(t) + v_0| = f(x, t)$

$x = -\frac{1}{2}gt^2 + v_0t + x_0$

$x_0 = x(t_0)$ $t_1 - t_0 = \Delta t$

$v_0 = v(t_0)$

$x_1 = x_0 + v_0 \cdot \Delta t$

$x_{n+1} = x_n + [v_n] \Delta t$

$x_{n+1} = x_n + f(x_n, t_n) \Delta t$

EULER'S METHOD

- Tracing the path involve acceleration a .
- There is initial position of the object x and initial velocity v .

NUMERICAL INTEGRATION

DIFFERENTIAL EQUATION

$v = f(x, t)$

$a = -g$

$v = -gt + v_0 = f(x, t)$

$x = -\frac{1}{2}gt^2 + v_0t + x_0$

$x_0 = x(t_0)$ $t_1 - t_0 = \Delta t$

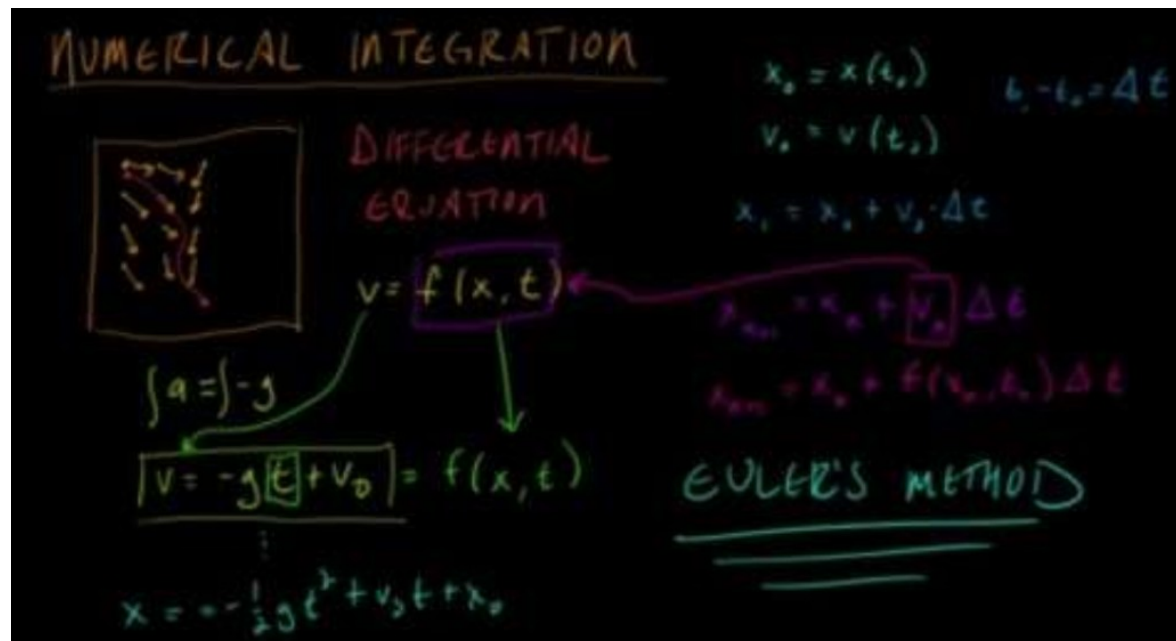
$v_0 = v(t_0)$

$x_1 = x_0 + v_0 \cdot \Delta t$

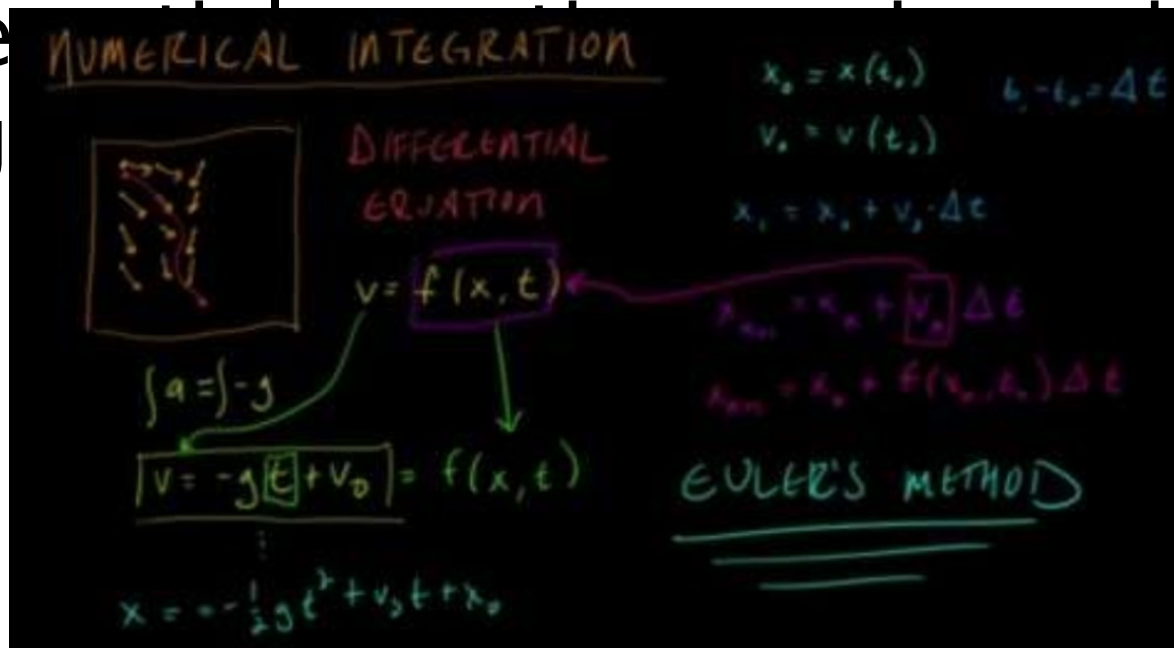
$x_{n+1} = x_n + v_n \Delta t$

$x_{n+1} = x_n + f(x_n, t_n) \Delta t$

EULER'S METHOD



- X_1 is the position of the object in next time is equal to position and velocity last time.
- V_n can be substituted with the



"Snowflakes" versus "Hair"

- Particle systems can be either **animated** or **static**; that is, the lifetime of each particle can either be distributed over time or rendered all at once.
- The consequence of this distinction is similar to the difference between snowflakes and hair - animated particles are akin to snowflakes, which move around as distinct [points](#) in space, and static particles are akin to hair, which consists of a distinct number of [curves](#).

A cube emitting 5000 animated particles

- A cube emitting 5000 animated particles, obeying a "gravitational" force in the negative Y direction.
- The same cube emitter rendered using static particles, or strands.

