Mechanics Simulations With JavaScript

Peter Krieg

Physics Fall Semester Thesis

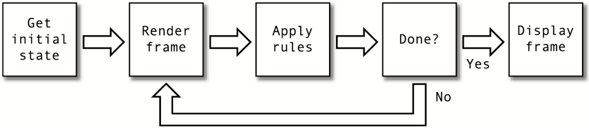
December 3, 2014

Overview - Why Did I Choose This Topic?

- I hope to use programming as a lens to view physics
- Examine mechanics in more detail
- Solve physics problems through simulations
- JavaScript high level language - viewable easily in web browser

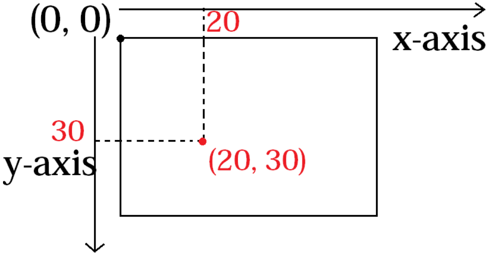
What is a simulation?

- Animation vs. Simulation
- Frames per second
- File size



Method of Basic Simulation

- HTML5 canvas application programming interface (API)
- Timer for each frame



Chapter 1: Basic kinematics and aerodynamic drag

- Three simulations
- Simulation #1: Basic bouncing ball
- Simulation #2: Bouncing ball with aerodynamic drag
- Simulation #3: Multiple bouncing balls

Simulation #1: Basic Bouncing Ball

- Realistic g value
- $9.81\ \frac{px}{s^2} = .1635\ \frac{\frac{px}{s}}{frame} \times \frac{60 frame}{s}$

- Coefficient of restitution (C_r)
- $C_r = \sqrt{\frac{KE_f}{KE_i}} = \sqrt{\frac{\frac{1}{2}mv_f^2}{\frac{1}{2}mv_i^2}} = \frac{v_f}{v_i}$
- $v_f = v_i * C_r$

Simulation #2: Bouncing Ball With Aerodynamic Drag

- $f_{drag} = -\frac{1}{2} C_d \rho A v^2$

- F_D = force of drag
- ρ = density of fluid
- v = speed of object relative to fluid
- C_d = drag coefficient (affected by texture, shape, viscosity, lift, etc)
- A = cross-sectional area of object

Simulation #3: Multiple Balls Bouncing

- Same physics as simulation #1
- Array of ball objects
- Each object has properties
- Each frame cycles through array, updating properties of each object

Chapter 2: Planetary Motion

- 3 Simulations
- Simulation #4: Orbits
- Simulation #5: Escape velocity
- Simulation #6: Kepler's 2nd law

Simulation #4: Orbits

- Newton's Law of universal gravitation
- $F_g = G\frac{m_1 m_2}{r^2}$

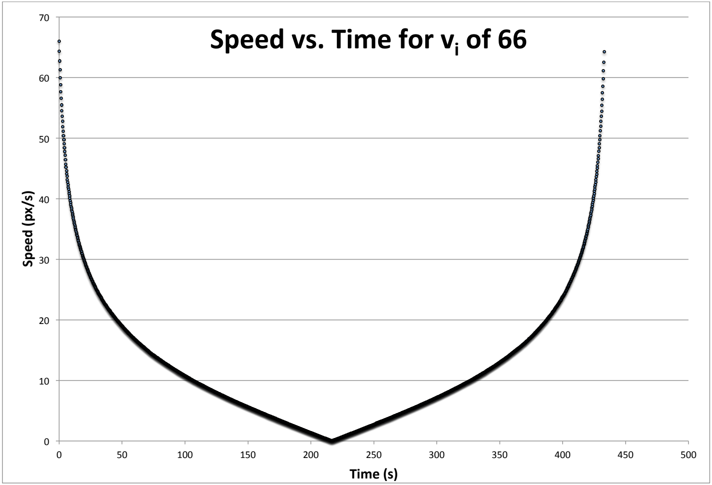
- Euler's Method to update velocity
- $x(t + dt) = x(t) + \frac{dx}{dt}(t)\,dt$

Simulation #5: Escape Velocity

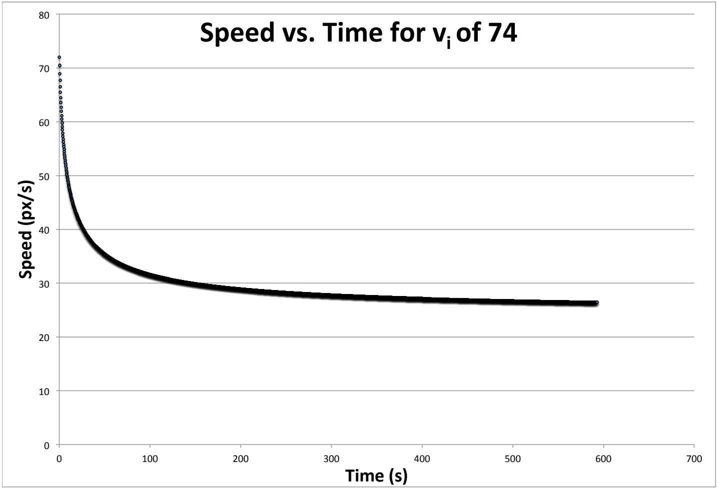
- $K_i + U_{g_i} = K_f + U_{g_f}$
- $\frac{1}{2} m v_{esc}^2 - \frac{GMm}{r} = 0 + 0$
- $v_{esc} = \sqrt{\frac{2GM}{r}}$
- $v_{esc} = \sqrt{\frac{2 * 1 \frac{px^3}{s^2} * 1000000}{410 px}} \approx 69.843 \frac{px}{s}$

- Used bigger canvas, and plotted velocities during planet's travel

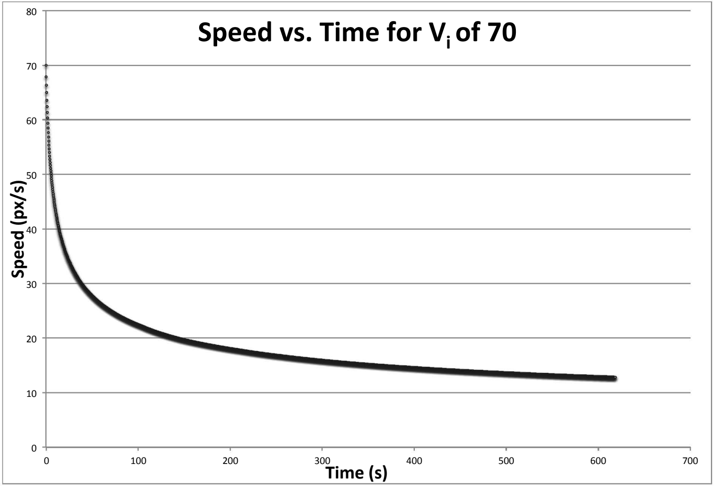
Simulation #5: Escape Velocity



Simulation #5: Escape Velocity



Simulation #5: Escape Velocity



Simulation #6: Kepler's 2nd law

- Early 1600's Johannes Kepler proposed laws explaining how planets orbit the sun
- Law #2: "The radius vector drawn from the Sun to a planet sweeps out equal areas in equal time intervals"
- Simulation shows constant $\frac{dA}{dt}$

Derivation of Kepler's 2nd Law

- Gravity force is *central* force
- $\vec{\tau} = \vec{r} \times \vec{F_g} = \frac{d\vec{L}}{dt}$
- $\vec{L} = \vec{r} \times \vec{p} = M_p \vec{r} \times \vec{v}$
- $L = M_p \left| \vec{r} \times \vec{v} \right|$

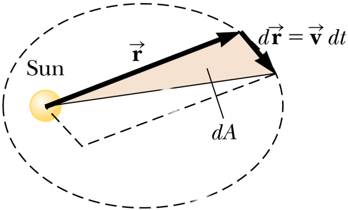


Figure : Relationship between $\vec{r}$ and $d\vec{r}$

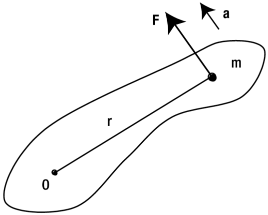
Derivation of Kepler's 2nd Law (Continued)

- $|\vec{r} \times d\vec{r}|$ area of parallelogram
- $dA = \frac{1}{2}|\vec{r} \times d\vec{r}| = \frac{1}{2}\left|\vec{r} \times \vec{v}dt\right| = \frac{1}{2}\left|\vec{r} \times \vec{v}\right| dt$
- $dA = \frac{1}{2}\left(\frac{L}{M_p}\right) dt$
- From before, $|\vec{r} \times \vec{v}| = \frac{L}{M_p}$

- $\frac{dA}{dt} = \frac{1}{2}\left(\frac{L}{M_p}\right)$
- L and M_p are constants

Chapter 3: Rotational Motion

- $\vec{\tau} = \vec{r} \times \vec{F}$
- $I = \int r^2 dm$
- $\vec{L} = I\vec{\omega}$



Chapter 3: Rotational Motion

- Newton's 2nd law for rotation
- $\vec{T} = I\vec{\alpha}$
- Program updates ω by calculating α from T and I.

Conclusion

- Simulations can be made very accurate with JavaScript
- Advantages of simulations involve sending instructions
- Future improvements could involve 3D

Thank You