Building a Better Jump

J. Kyle Pittman
@PirateHearts
Co-founder, Minor Key Games

Hi

- I'm Kyle
  - Hi Kyle

gearbox software

2007-2013

Minor Key Games

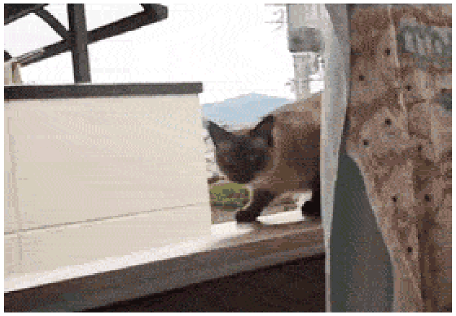
2013-20XX

Motivation

- Avoid hardcoding, guessing games
- Design jump trajectory on paper
- Derive constants to model jump in code

Motivation

- Has this ever happened to you?



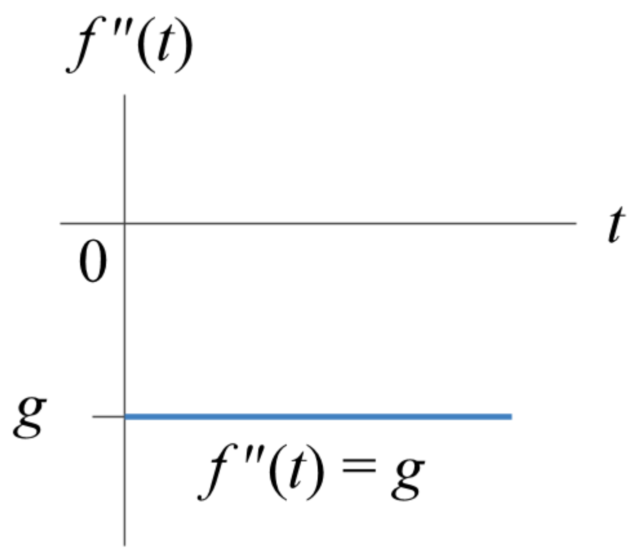
- There’s GOT to be a better way!!

Assumptions

- Model player as a simple projectile
- Game state
  - Position, velocity integrated on a timestep
  - Acceleration from gravity
- No air friction / drag

Gravity

- Single external force
- Constant acceleration over time

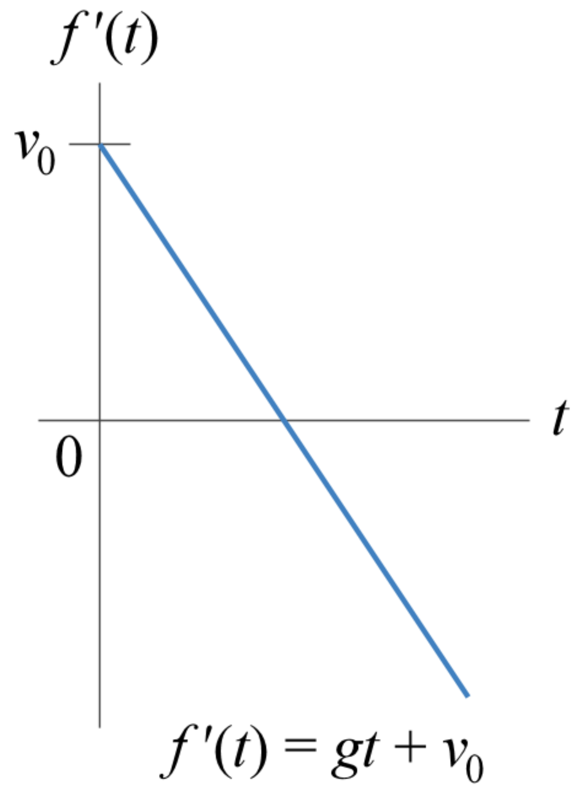


Integration

- Integrate over time to find velocity

$$\int g \, dt =$$

$$gt + v_0$$

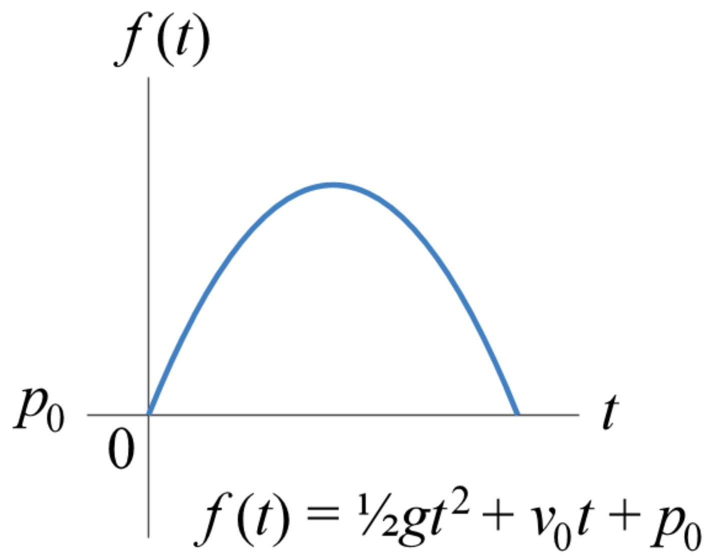


Integration

- Integrate over time again to find position

$$\int gt + v_0 \, dt =$$

$$\tfrac{1}{2} gt^2 + v_0 t + p_0$$



Projectile motion

$$f(t) = \tfrac{1}{2} g t^2 + v_0 t + p_0$$

- Textbox Physics 101 projectile motion
- Understand how we got there

Parabolas

- Algebraic definition
  - $f(x) = ax^2 + bx + c$
- Substituting

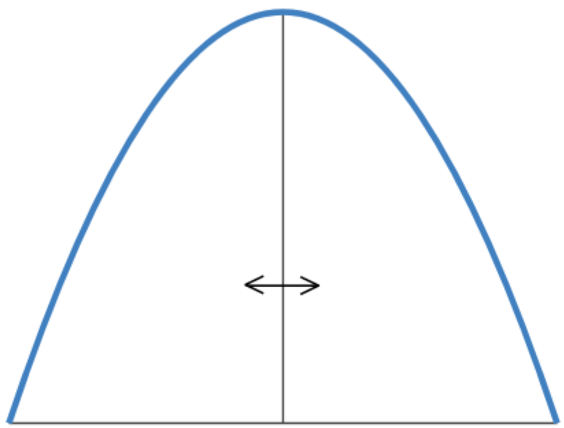
$$x \rightarrow t \qquad b \rightarrow v_0$$

$$a \rightarrow \tfrac{1}{2} g \qquad c \rightarrow p_0$$

$$f(t) = \tfrac{1}{2} g t^2 + v_0 t + p_0$$

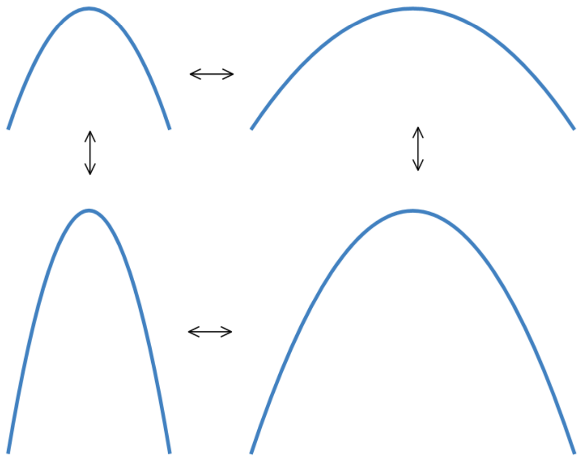
Properties of parabolas

- Symmetric



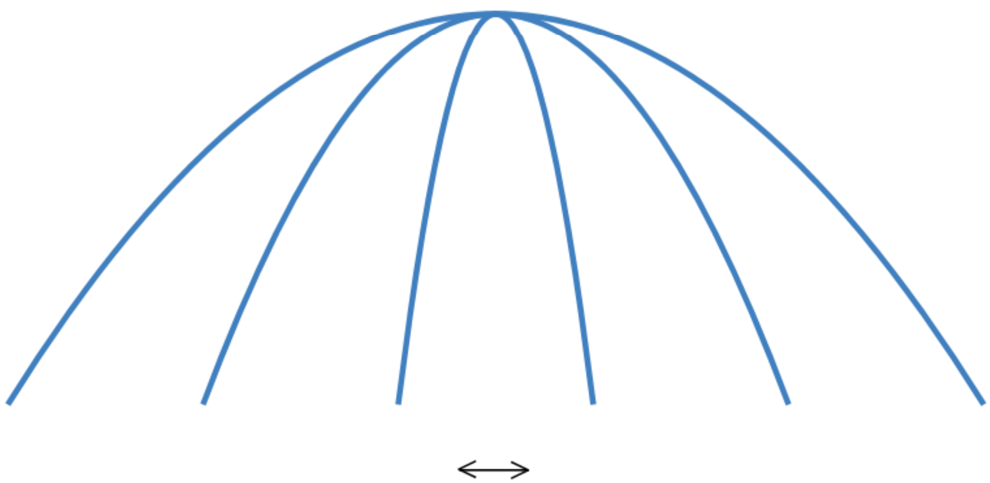
Properties of parabolas

- Geometric self-similarity

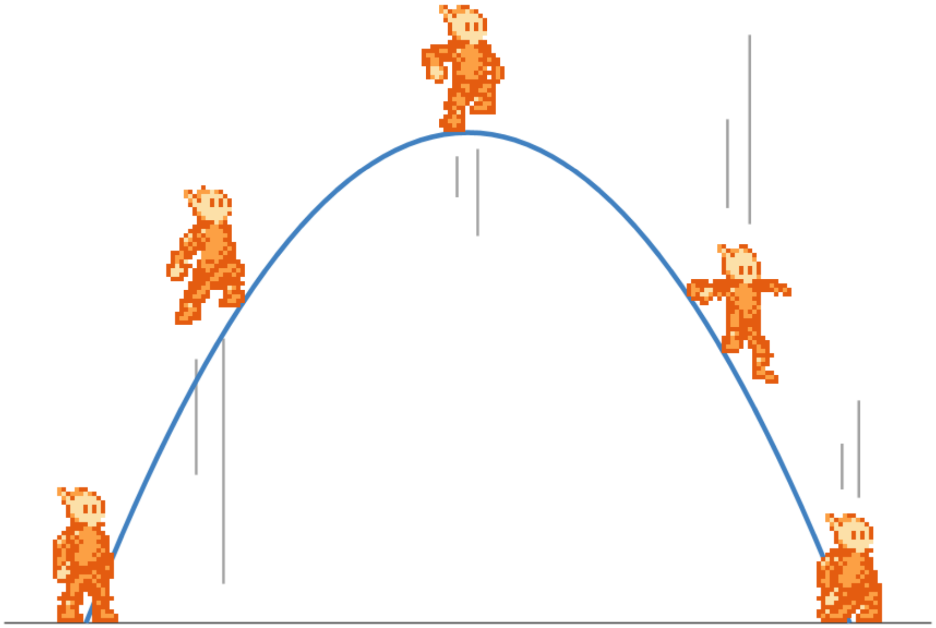


Properties of parabolas

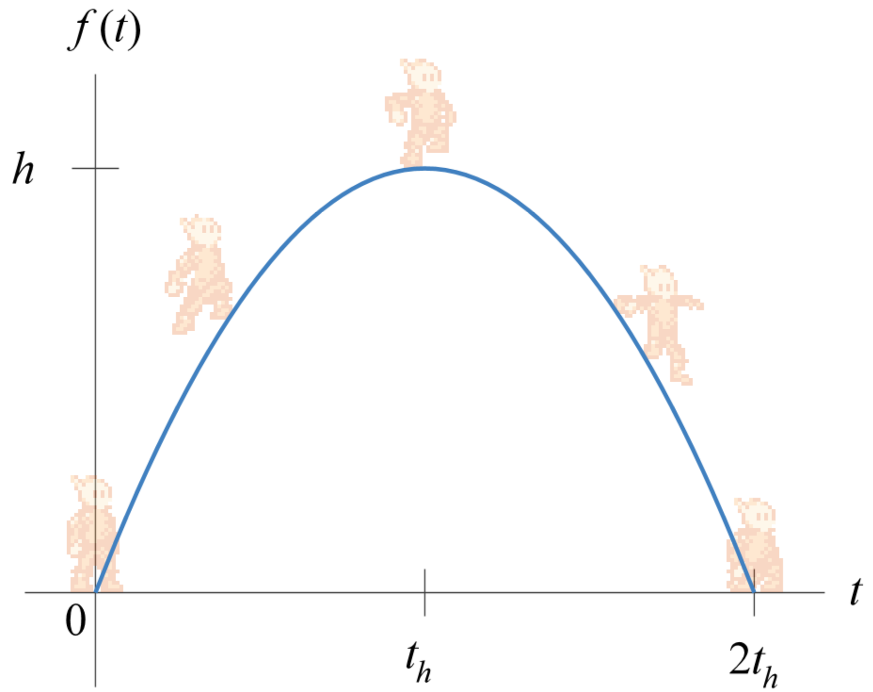
- Shaped by quadratic coefficient $a \to \frac{1}{2} g$



Design on paper

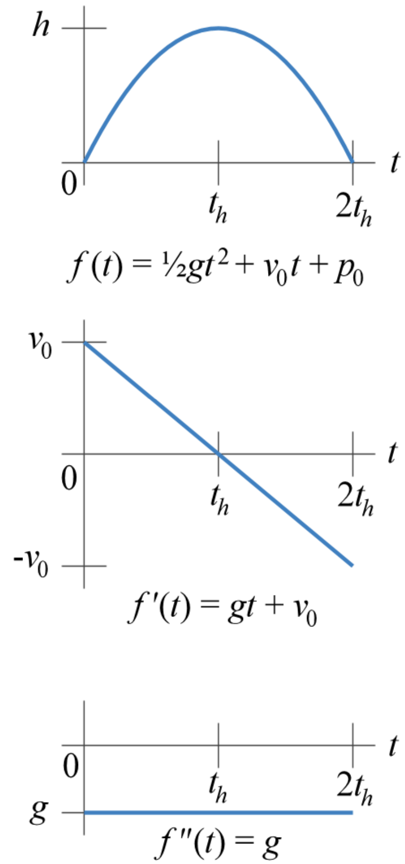


Design on paper

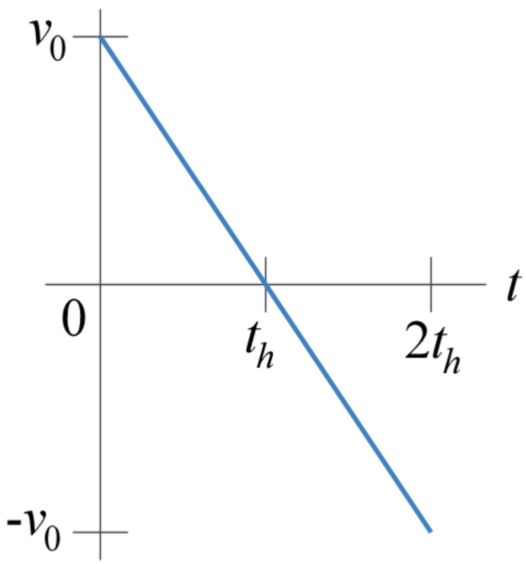


Maths

- Derive values for gravity and initial velocity in terms of peak height and duration to peak



Initial velocity



$f'(t) = gt + v_0$

Solve for v_0:

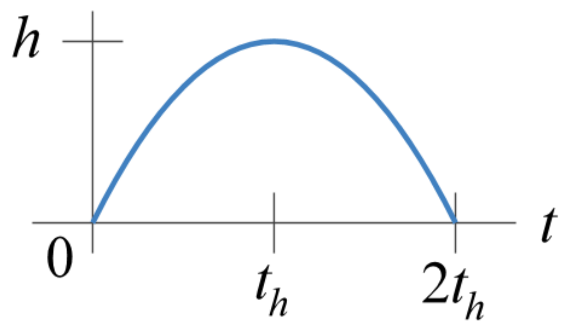
$$f'(t) = gt + v_0$$

$$f'(t_h) = 0$$

$$0 = gt_h + v_0$$

$$v_0 = -gt_h$$

Gravity



$$f(t) = \tfrac{1}{2} g t^2 + v_0 t + p_0$$

Known values:

$$v_0 = -g t_h$$

$$p_0 = 0$$

Solve for g:

$$f(t) = \tfrac{1}{2} g t^2 + v_0 t + p_0$$

$$f(t_h) = h$$

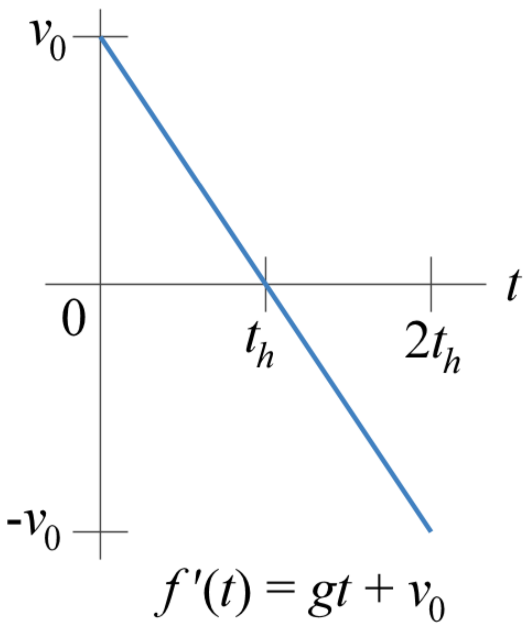
$$h = \tfrac{1}{2} g t_h^{\,2} + v_0 t_h + p_0$$

$$h = \tfrac{1}{2} g t_h^{\,2} + (-g t_h) t_h + 0$$

$$h = -\tfrac{1}{2} g t_h^{\,2}$$

$$g = \frac{-2h}{t_h^{\,2}}$$

Back to init. vel.



Solve for v_0:

$$v_0 = -gt_h$$

$$g = \frac{-2h}{t_h^{\ 2}}$$

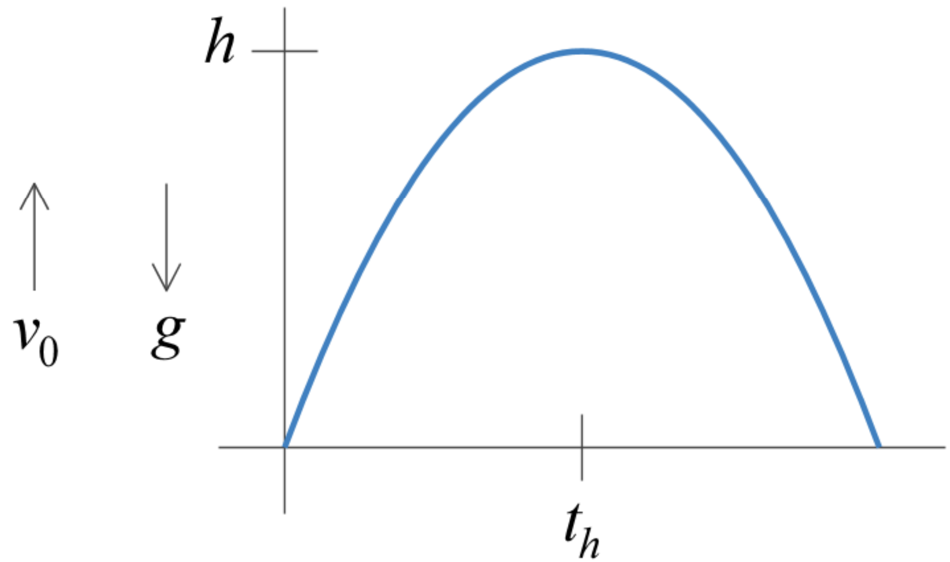
$$v_0 = -\left(\frac{-2h}{t_h^{\ 2}}\right)t_h$$

$$v_0 = \frac{2h}{t_h}$$

Review

$$v_0 = \frac{2h}{t_h}$$

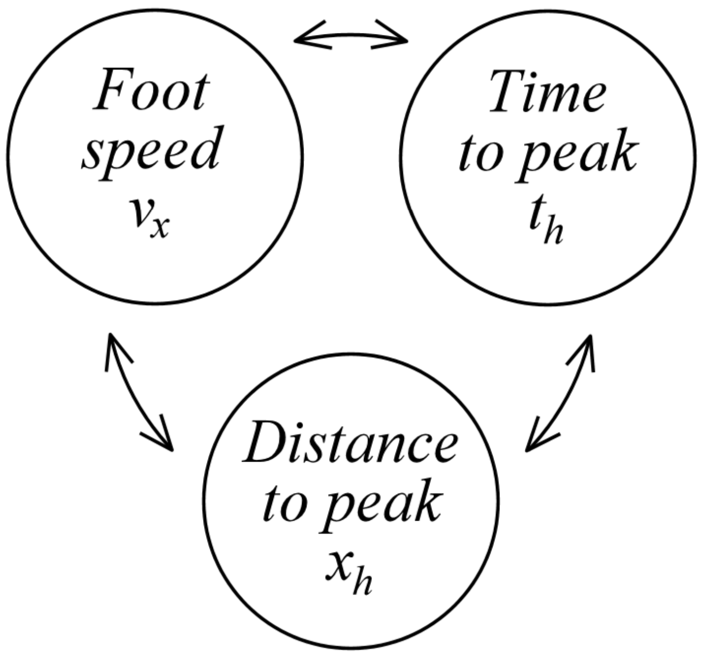
$$g = \frac{-2h}{t_h^{\,2}}$$



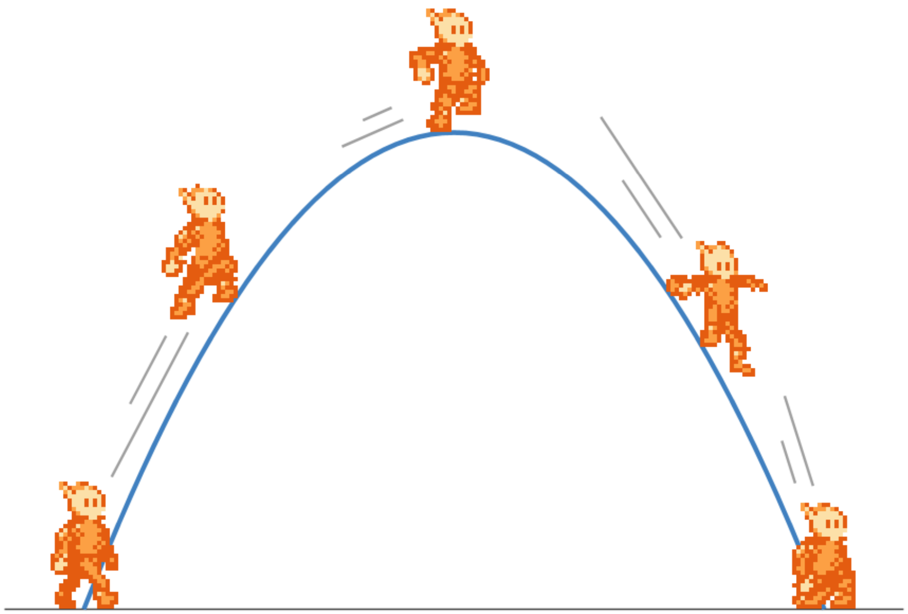
Time → space

- Design with x-axis as distance in space
- Introduce lateral (foot) speed
- Keep horizontal and vertical velocity components separate

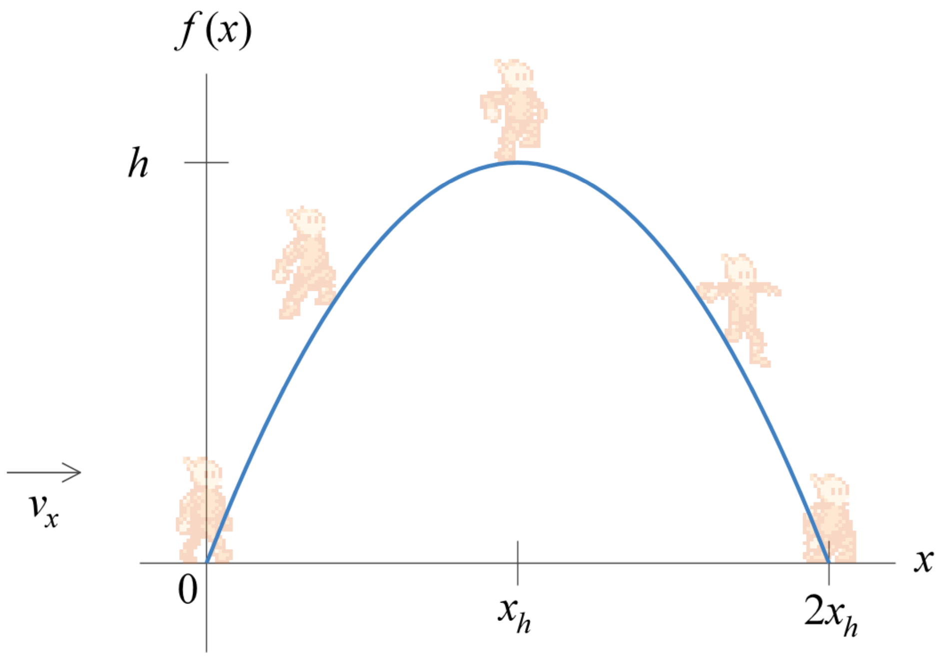
Parameters



Time → space



Time → space



Maths

- Rewrite gravity and initial velocity in terms of foot speed and lateral distance to peak of jump

Maths

$$v_0 = \frac{2h}{t_h}$$

$$g = \frac{-2h}{{t_h}^2}$$

$$t_h = \frac{x_h}{v_x}$$

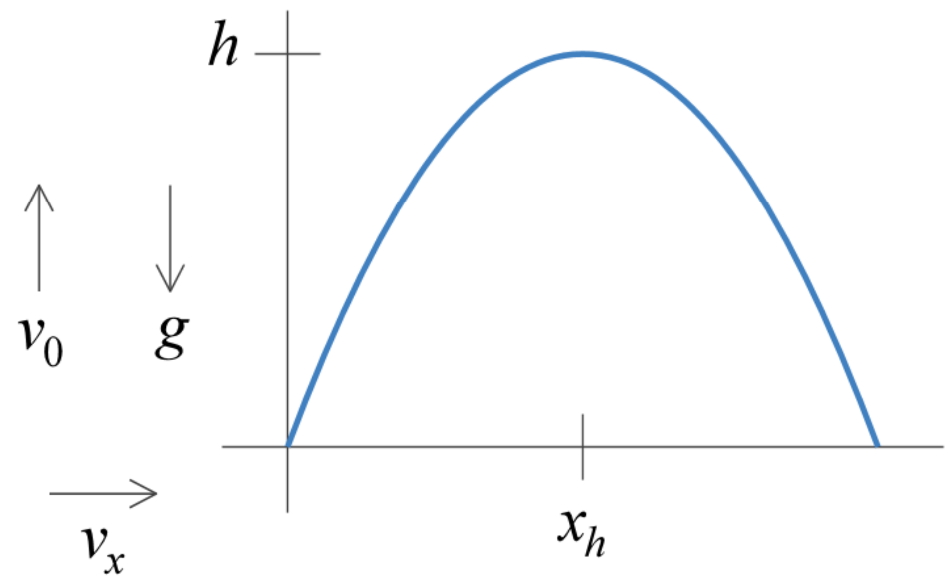
$$v_0 = \frac{2hv_x}{x_h}$$

$$g = \frac{-2h{v_x}^2}{{x_h}^2}$$

Review

$$v_0 = \frac{2hv_x}{x_h}$$

$$g = \frac{-2hv_x^{\,2}}{x_h^{\,2}}$$

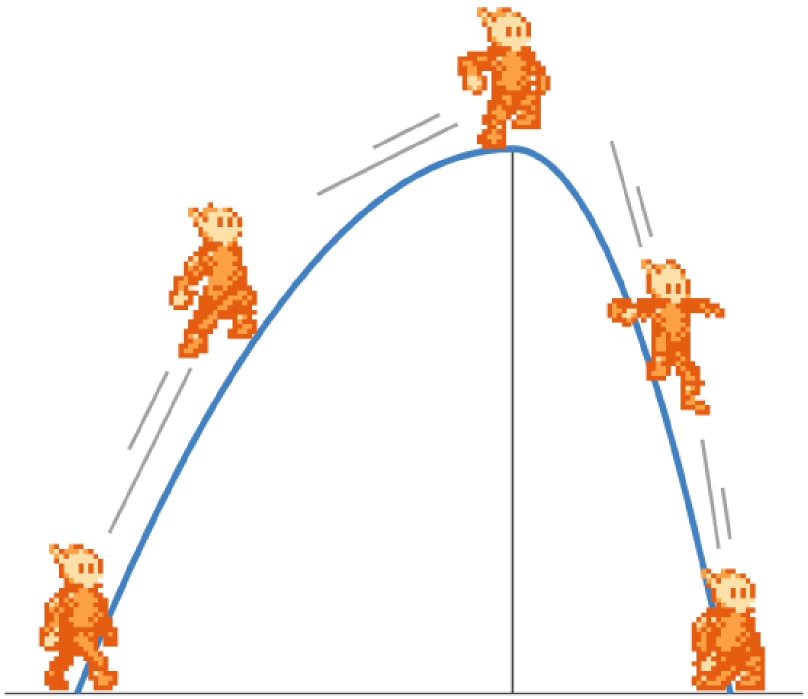


Breaking it down

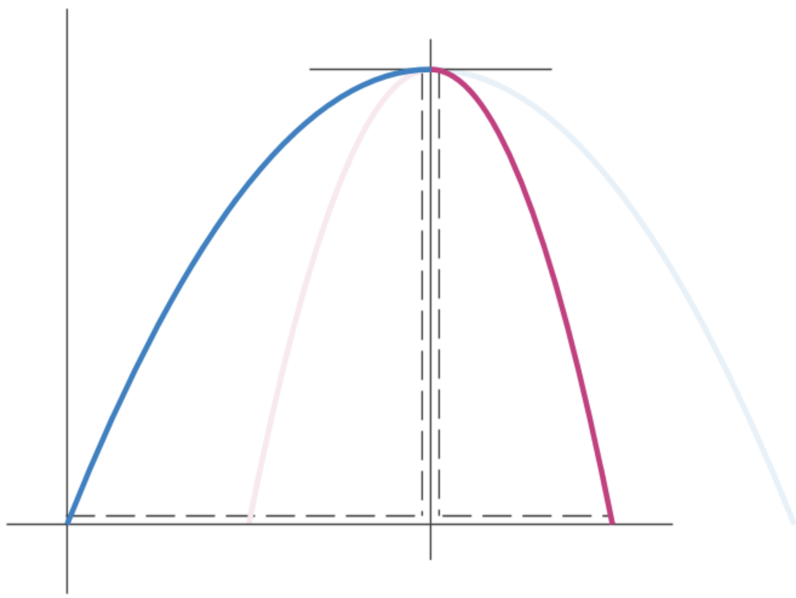
- Real world: Projectiles always follow parabolic trajectories.
- Game world: We can break the rules in interesting ways.
- Break our path into a series of parabolic arcs of different shapes.

Breaks

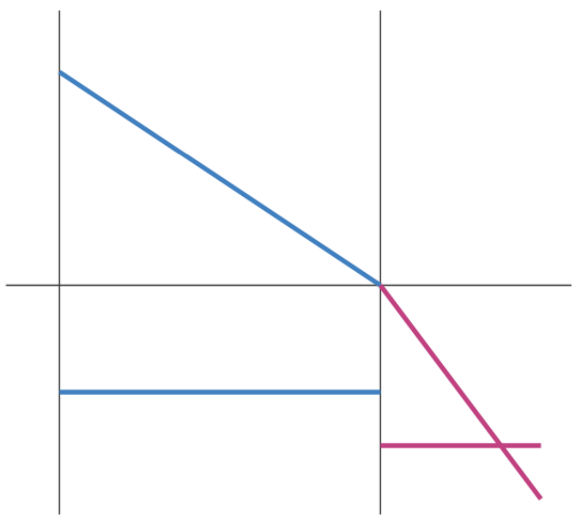
- Maintain continuity in position and velocity
  - Trivial in implementation
- Choose a new gravity to shape our jump



Fast falling

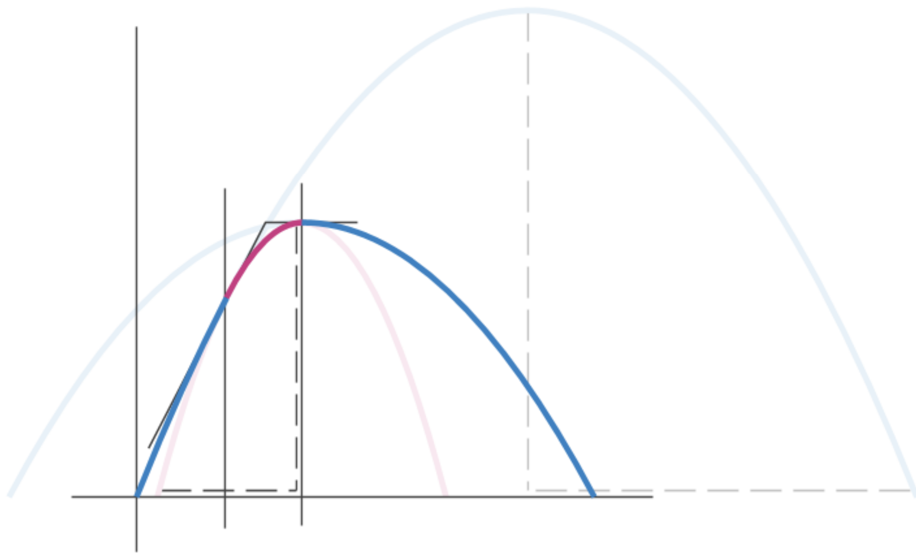


Position

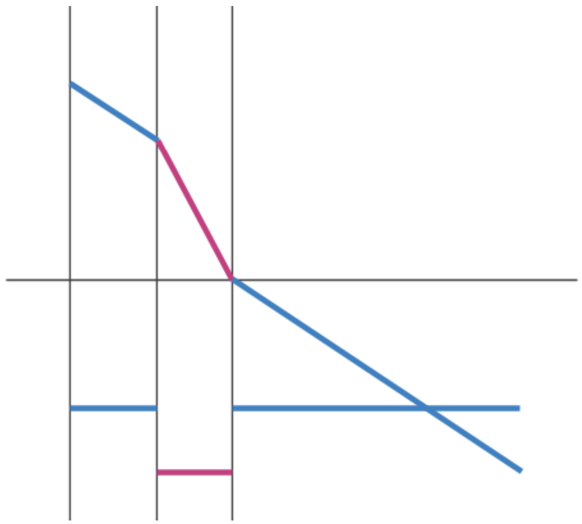


Velocity / Acceleration

Variable height jumping

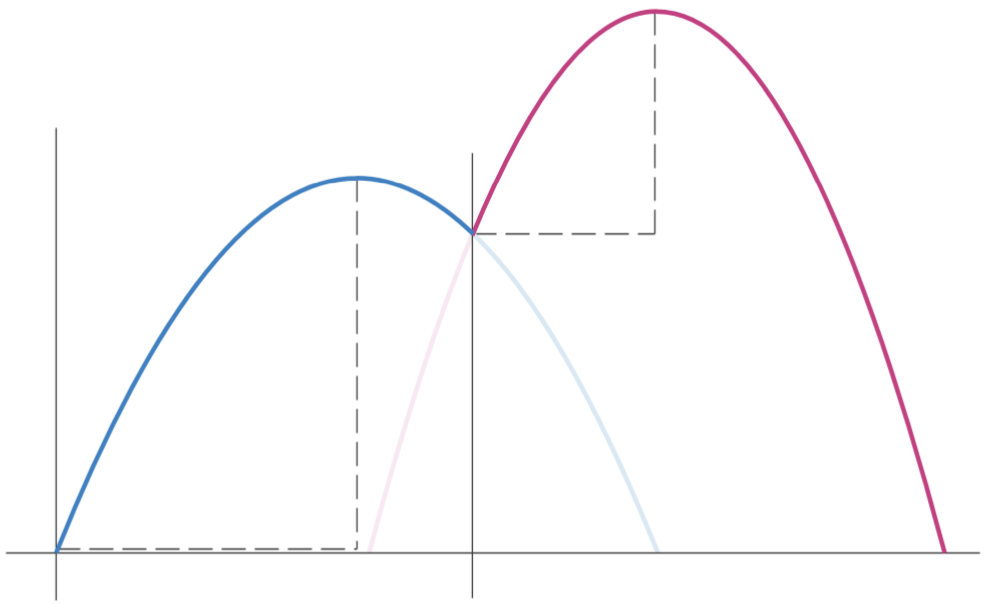


Position

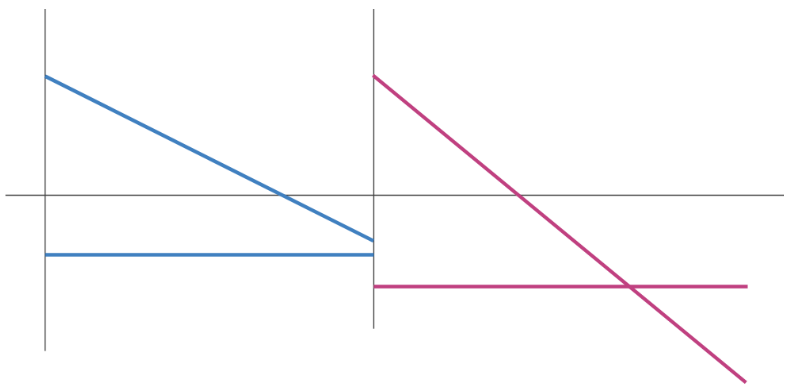


Velocity / Acceleration

Double jumping



Position

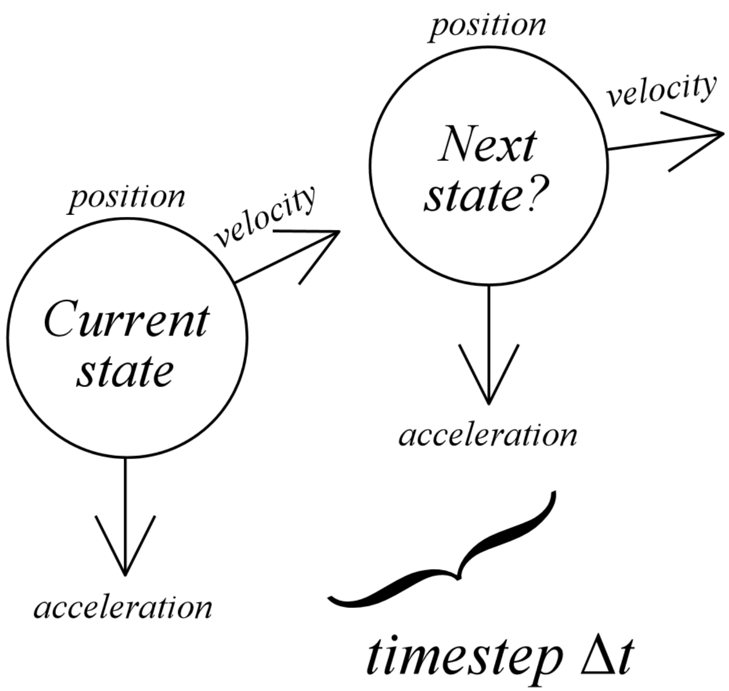


Velocity / Acceleration

Integration

- Put our gravity and initial velocity constants to use in practice
- Integrate from a past state to a future state over a time step

Integration



Euler

- Pseudocode

```
pos += vel * Δt
vel += acc * Δt
```

- Easy
- Unstable
- We can do better

Runge-Kutta (RK4)

- The “top-shelf” integrator.
- No pseudocode here. :V
- Gaffer on Games: “Integration Basics”
- Too complex for our needs.

Velocity Verlet

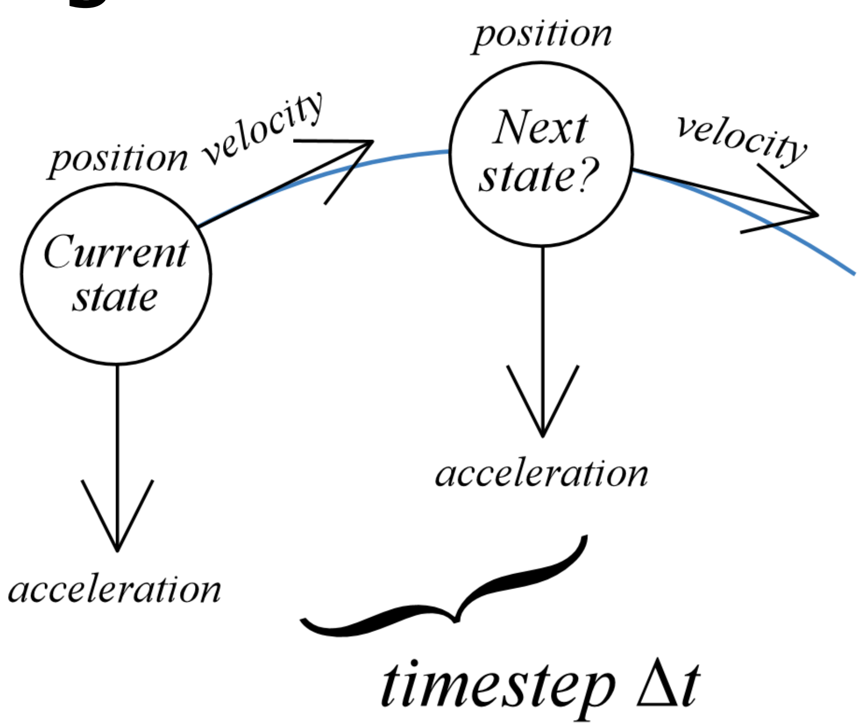
- Pseudocode

```
pos += vel*Δt + ½acc*Δt*Δt
new_acc = f(pos)
vel += ½(acc+new_acc)*Δt
acc = new_acc
```

Observations

- Similarity to projectile motion formula
- What if our acceleration were constant?
- We could integrate with 100% accuracy

Assuming constant acceleration



Assuming constant acceleration

- Pseudocode

```
pos += vel*Δt + ½acc*Δt*Δt
vel += acc*Δt
```

- Trivially simple change from Euler
- 100% accurate as long as our acceleration is constant

Near-constant acceleration

- What if we don't change a thing?
- The error we accumulate when our acceleration does change (versus Velocity Verlet) will be:
  - Δacc * Δt * Δt
  - Acceptable?

The takeaway

- Design jump trajectories as a series of parabolic arcs
- Can author unique game feel
- Trust result to feel grounded in physical truths

Questions?

- In practice: *You Have to Win the Game* (free game PLAY IT PLAY MY THING)
- The Twitters: **@PirateHearts**
- http://minorkeygames.com
- http://gunmetalarcadia.com