# CS 134

Platformer Physics

# Questions?

Questions about the homework?

# Super Mario World Example

Let's watch Super Mario World and
list out all the different behaviors

# Super Mario World Example

- Run on ground
- Run into wall
- Move in air
- Jump up through one-way floor
- Jump down onto one-way floor
- Hit enemy from above
- Hit enemy from side or below
- Hit a coin
- And many many more...

# Super Mario World Example

- These behaviors can get grouped into three categories:
  - Sprite Motion
  - Sprite / Background Collision Resolution
  - Sprite / Sprite Collision Resolution

# Super Mario World Example

- These behaviors can get grouped into three categories:
  - Sprite Motion
  - Sprite / Background Collision Resolution
  - Sprite / Sprite Collision Resolution
- This is what we are going to cover entirely today.

# Sprite Motion

- “Normal” motion behavior
- Walking around on the ground
- Free fall in air.

# Sprite Motion

- Ground motion:
  - Left / Right with input
  - Up via jumping
    - Hold down jump to go higher
  - Walk up / down inclined surfaces
  - Bump into walls
- Left / Right motion is just like before

# Sprite Motion

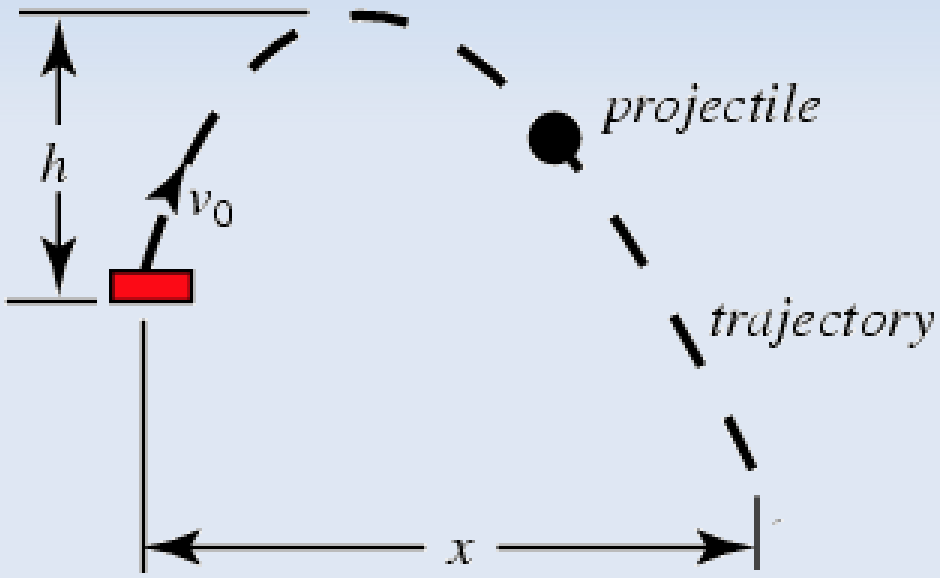
- Air motion:
  - Left / Right with input
  - Can't move faster up!
  - Fall down faster and faster due to gravity
- Left / Right motion may be slower than when grounded
- Falling down should accelerate due to gravity.

# Sprite Motion

- Two new states:
  - isGrounded
  - yVelocity
- When isGrounded is true, you will do “ground” motion
  - Jumping, stick to ground, etc.
- When isGrounded is false, you will do “air” motion
  - Air attacks, less air control, double jumps, etc.

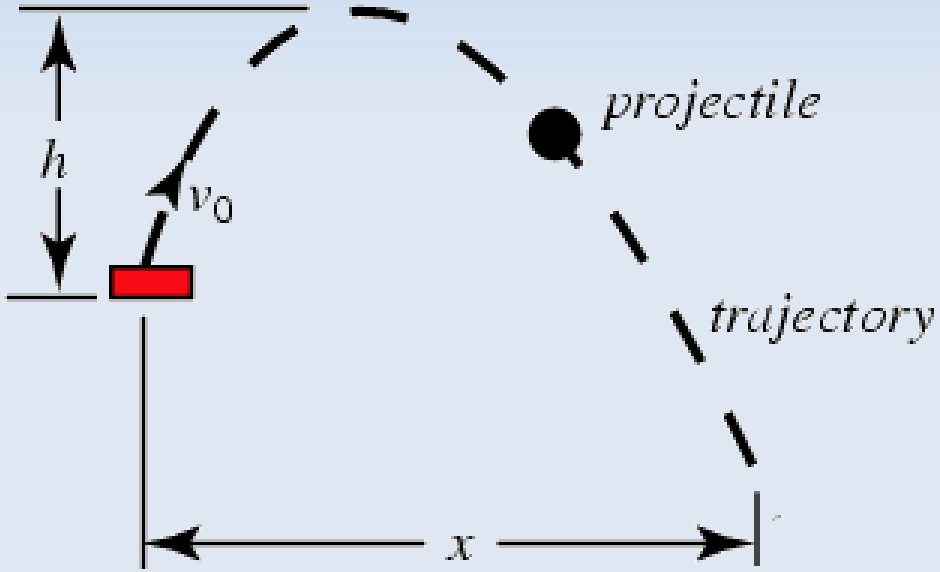
# Sprite Motion

- Storing yVelocity allows you to remember fall speed and have a sense of momentum.
- Objects that are airborne always follow a parabola, falling faster the more they are airborne.



# Sprite Motion

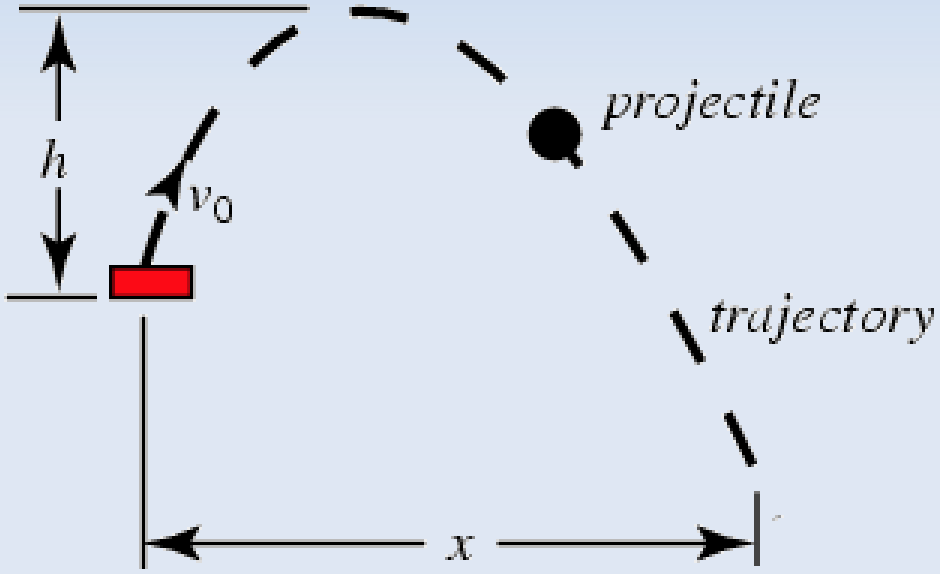
- To state it in calculus terms:
  - p(t) = position
  - v(t) = velocity
  - a(t) = acceleration
  - v(t) = dp/dt
  - a(t) = dv/dt = gravity



# Sprite Motion

- a(t) = 9.8
- v(t) = 9.8t + c
- p(t) = $4.9t^2 + ct + d$

- p is a quadratic equation, which always follows a parabola!



# Sprite Motion

- Games are frame based, so we don't need to do it algebraically.
- Solve based on previous state.
  - $v(t) = 9.8t + c$

- If $v(t) = 0$, then $v(t+1) = 9.8$.
- If $v(t) = 55$, then $v(t+1) = 64.8$
- $v(t+dt) = v(t) + 9.8dt$

# Sprite Motion

- We can do the same thing for position.
  - $p(t+dt) = p(t) + v(t)dt$
- Incidentally, we are literally doing calculus here.
  - $p(t+dt) - p(t) = v(t)dt$
  - $[p(t+dt) - p(t)] / dt = v(t)$
  - $dp/dt = v(t)$
- This step-based method is how you can code actor motion for falling.

# Sprite Motion

- All this comes together to build the basic Platformer update

- yPos = yPos + yVelocity * deltaTime
- yVelocity = yVelocity + yGravity * deltaTime

# Sprite Motion

- All this comes together to build the basic Platformer update

- **if (isGrounded) yVelocity = 0**
- yPos = yPos + yVelocity * deltaTime
- yVelocity = yVelocity + yGravity * deltaTime

# Sprite Motion

- All this comes together to build the basic Platformer update

- if (isGrounded) yVelocity = 0
- **if (isGrounded && jumpButtonPressed)**
- **    yVelocity = jumpVelocity**
- yPos = yPos + yVelocity * deltaTime
- yVelocity = yVelocity + yGravity * deltaTime

# Sprite Motion

- Other things to add:

- Adjustable jump heights
  - Set yVelocity to jumpVelocity as long as jump button is held
  - Need jumpTimeRemaining state
- Double jumps, triple jumps
  - Allow jumping as long as you have jumps remaining
  - Need jumpsRemaining state

# Sprite Motion

Questions?

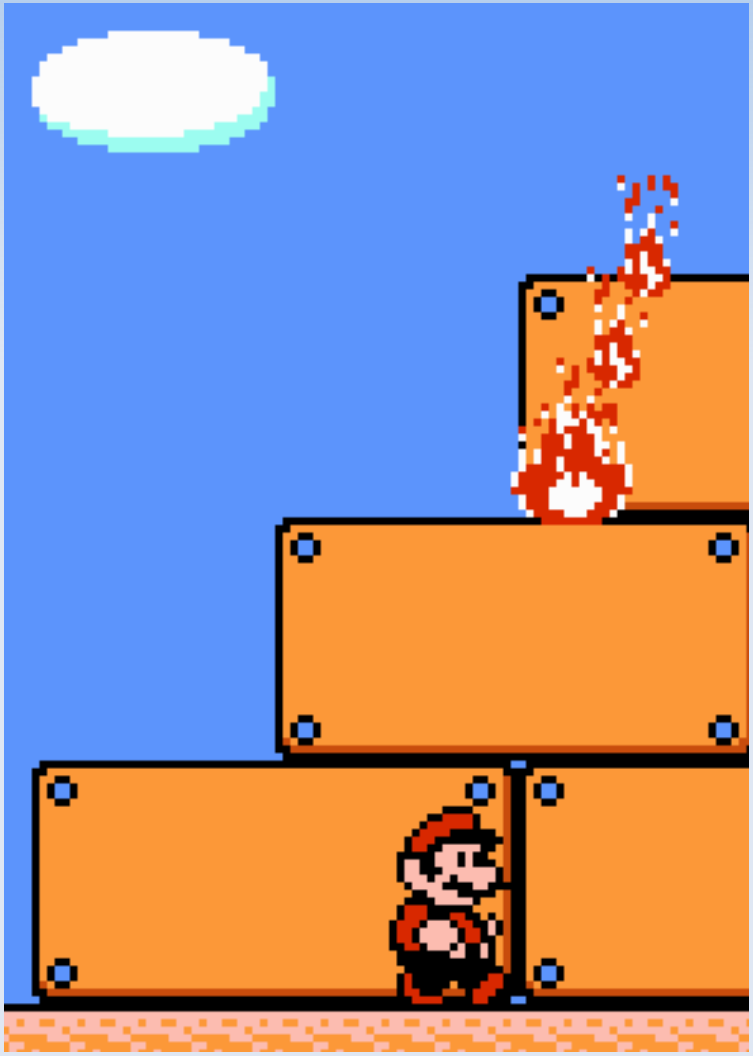
# Sprite / Background Resolution

- Very similar to the motion talked about so far.
  - Need to make sure we update isGrounded here
- Split motion into “X” pass and “Y” pass
- When resolving Y pass, set isGrounded based on if you resolve by pushing the character up

# Sprite / Background Resolution

# Sprite / Background Resolution

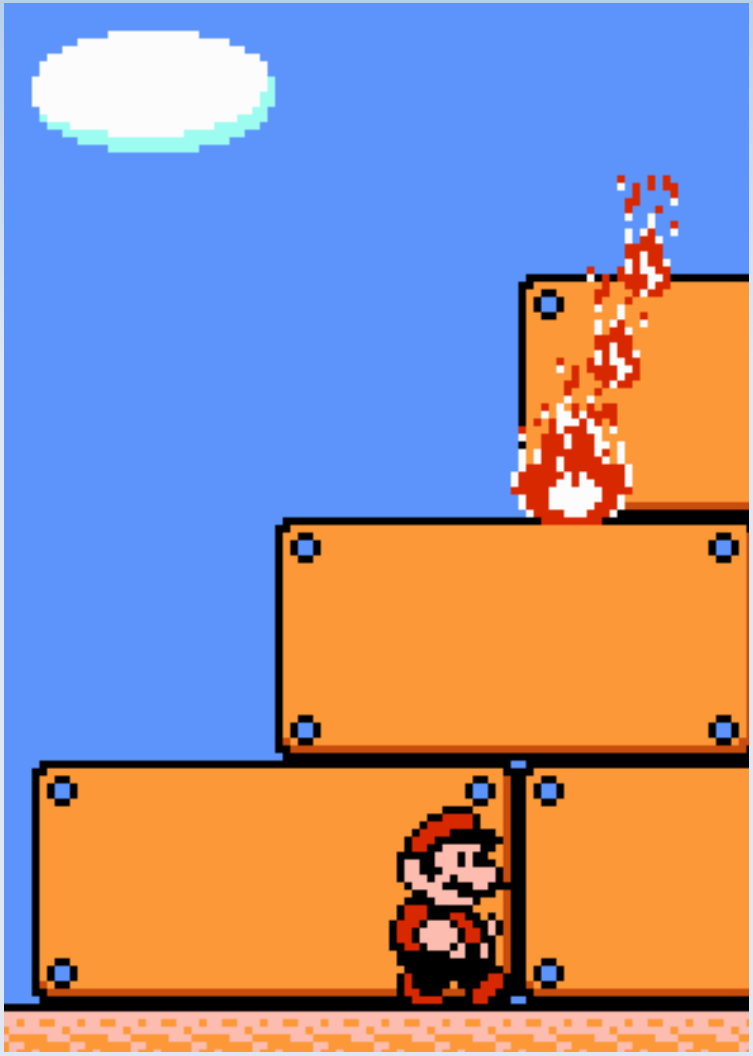
- One-way floors are needed if you want to allow to walk “in front” of platforms.
- Mario can jump “up” past the floor of each level here, but can't jump down.
- Often games have a way to jump down too.



# Sprite / Background Resolution

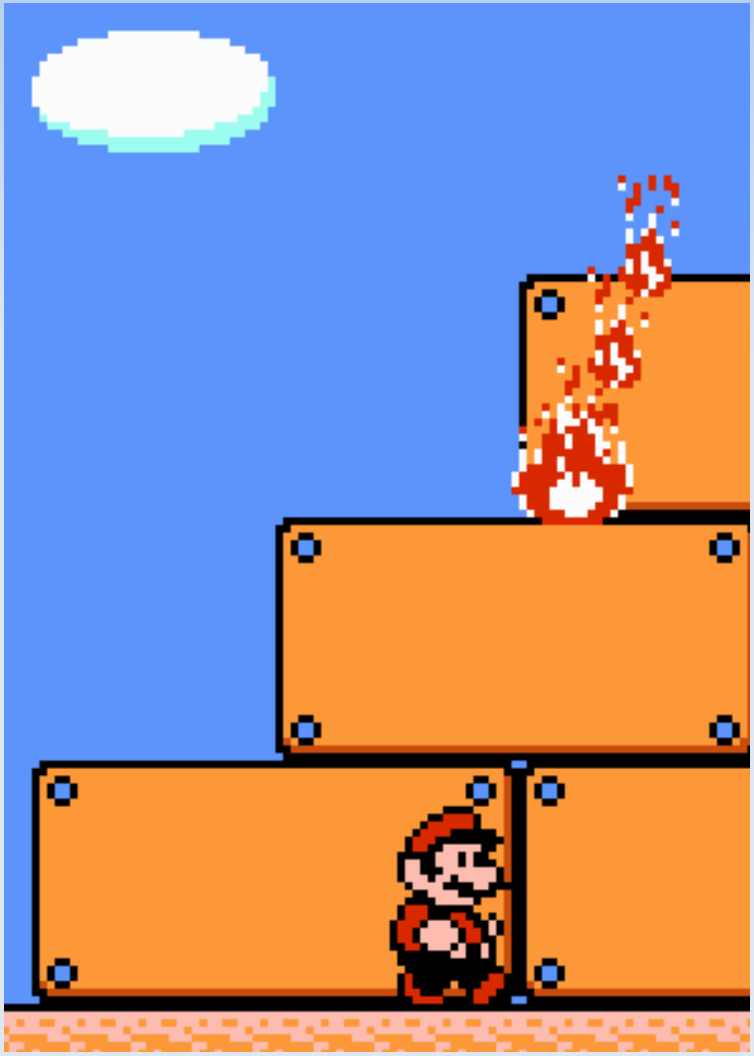
- Extra checks during collision resolution

- On down only floors:
  - X motion is allowed to collide with floor
  - Y motion backs up only if previous Y was above floor

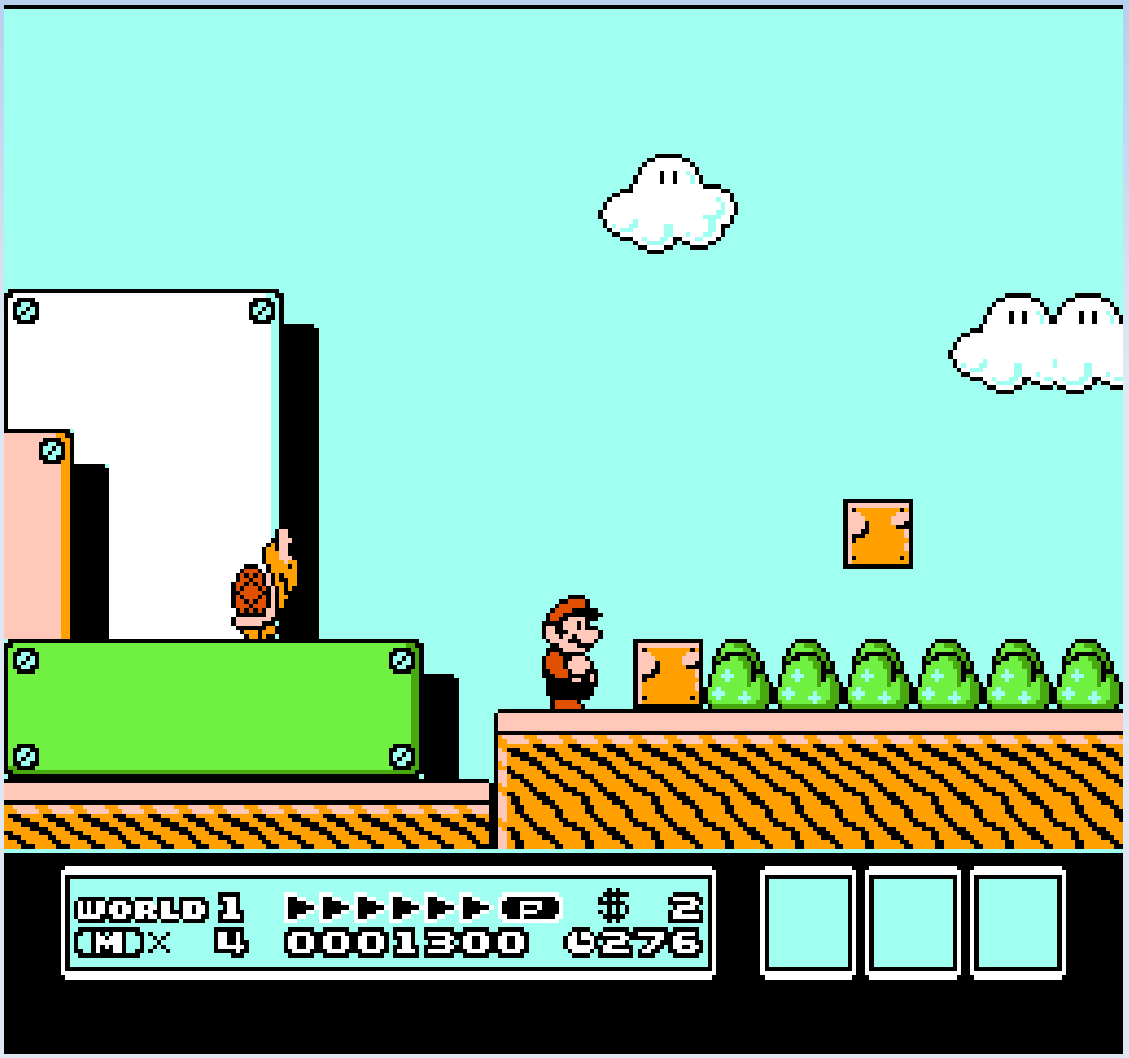


# Sprite / Background Resolution

- Take advantage of this to allow jumping down
- Move y down one pixel (outside of physics motion) so next frame is not above the floor



# Sprite / Background Resolution



# Sprite / Background Resolution

Questions?

# Sprite / Sprite Resolution

- This is where the fun game logic lives!
- There's lots and lots of special cases.

- Make sure the design is foolproof and every collision is fully resolved so the collision won't happen next frame.

# Sprite / Sprite Resolution

- Hit enemy from above
- Hit enemy from side or below
- Hit a coin

# Sprite / Sprite Resolution

- Hit enemy from above
  - Set enemy animation to “Dead”
  - Add 100 points to score
  - Set Mario's Y speed
- Hit enemy from side or below
  - Hurt Mario, make Mario invincible
- Hit a coin
  - Remove coin from world
  - Add 1 to Mario's coin count, if coin count is 100, set it to 0 and add 1 to Mario's life count

# Platformer Motion

- Other game-specific motion rules can be added
  - Wall jumping
  - Pushing objects
  - Double jump
  - Running up walls
- Any others you want covered?