Dr. Sarah Abraham
University of Texas at Austin
Computer Science Department

Platforming Mechanics

Elements of Graphics
CS324e
Spring 2018

Platforming in Video Games

- Genre of gameplay that focuses on jump timing and distance
- Requires physics, collision detection, and precise input handling

Physics

- Often implement a mix of **dynamic** (physically-based application of forces) and **kinematic** (control-based application of movement)
- "Forces" only applied during certain **player states**
  - Running (velocity plus friction applied)
  - Jumping (gravity and air control applied)

Running

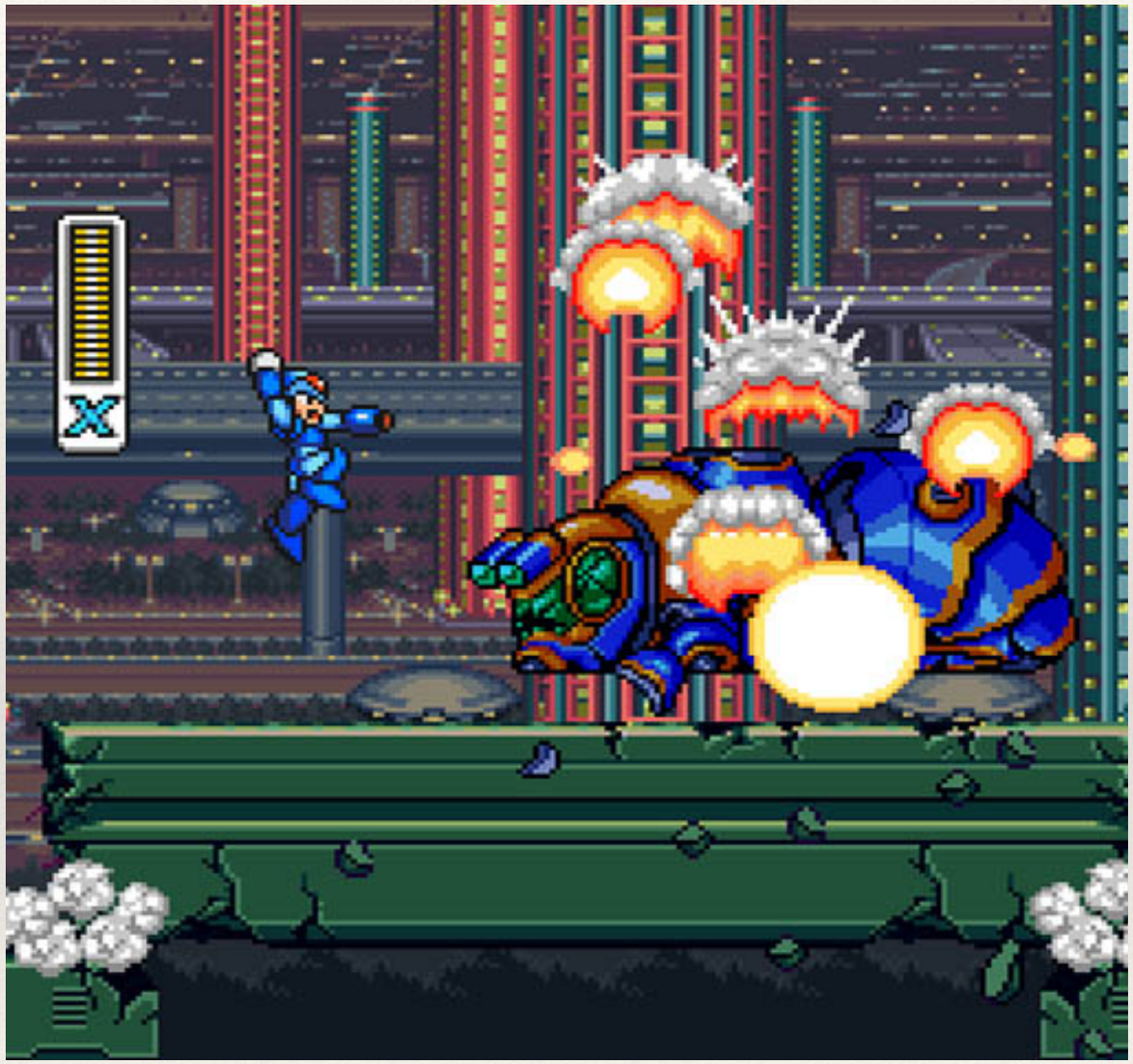
- Move in the horizontal direction (left and right)
- Applying position changes directly tends to feel clunky
  - Apply velocity with high friction to give tight controls
  - Apply acceleration then resolve velocity/position for slide-y controls

Jumping

- Apply impulse force to launch character upwards
  - Gravity also applied downwards
- Air controls provide player a degree of horizontal movement control

Control Feel

Super Mario Bros



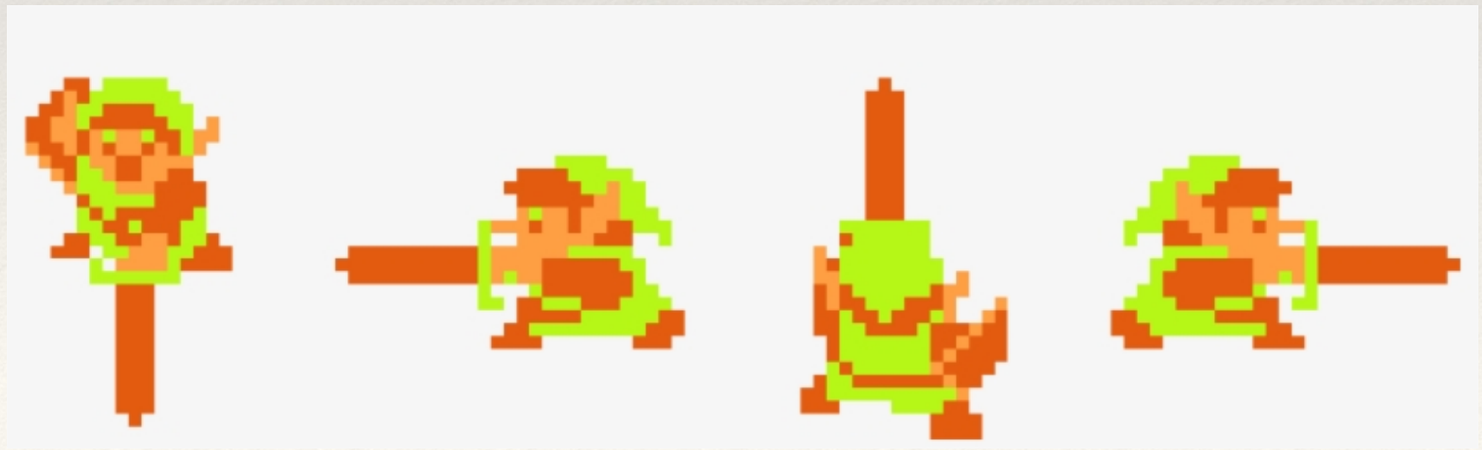
Megaman X

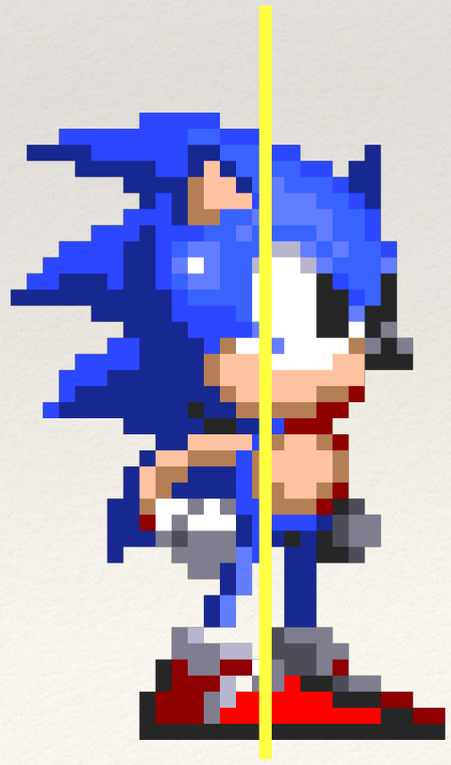
Detecting State

- Must determine **state** of player character to know:
  - Actions available
  - How to resolve physics
- Examples:
  - Is character moving left or right?
  - Is the character idling?
  - Is the character on the ground?
  - Is the character taking damage?

Set Left and Right State

- Use of scale (-1 or 1) to flip direction the sprite is facing
  - Won't work if you want to preserve asymmetry of character
- Vertical axis of sprite must be along the center
- Player Character class must track the current direction (boolean value)





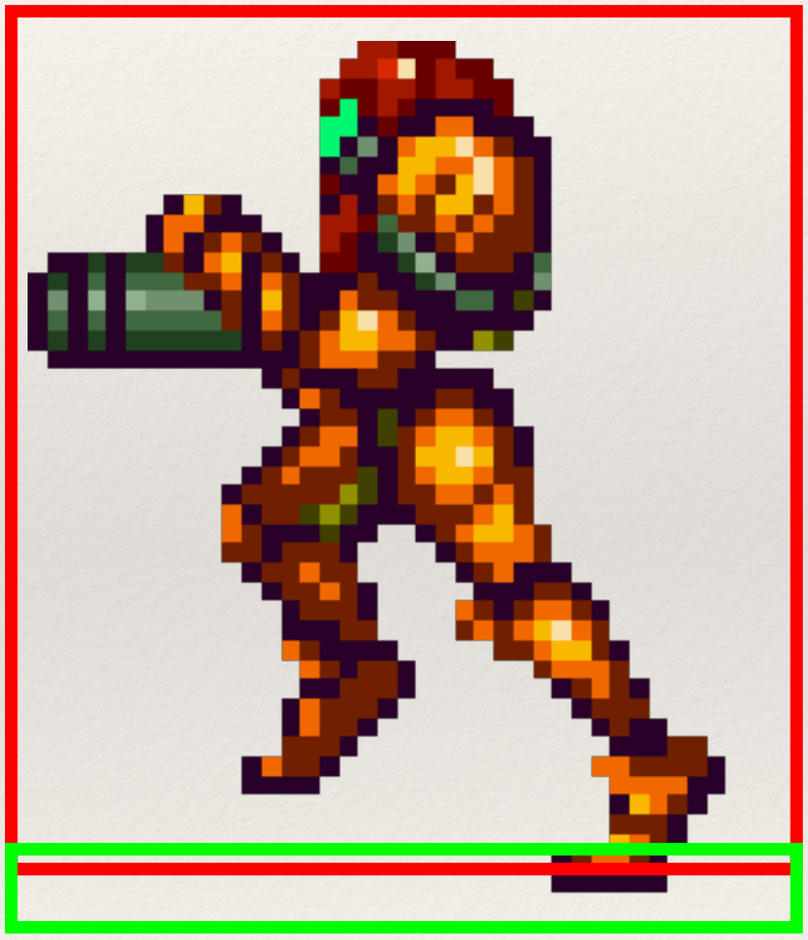
Checking for Ground

- Characters must begin jump from the ground
  - Double and triple jumps only useable in the air
  - Must track if moves are available then reset upon landing
- How to check if the character is on the ground?

Ending Fall State

- Must check whether the character's feet are **in contact** with the ground
  - Set "landed" character state
- Requires more complex collision volumes
  - Often called hitboxes / hurtboxes in video games

Hitbox detects interactions across whole sprite



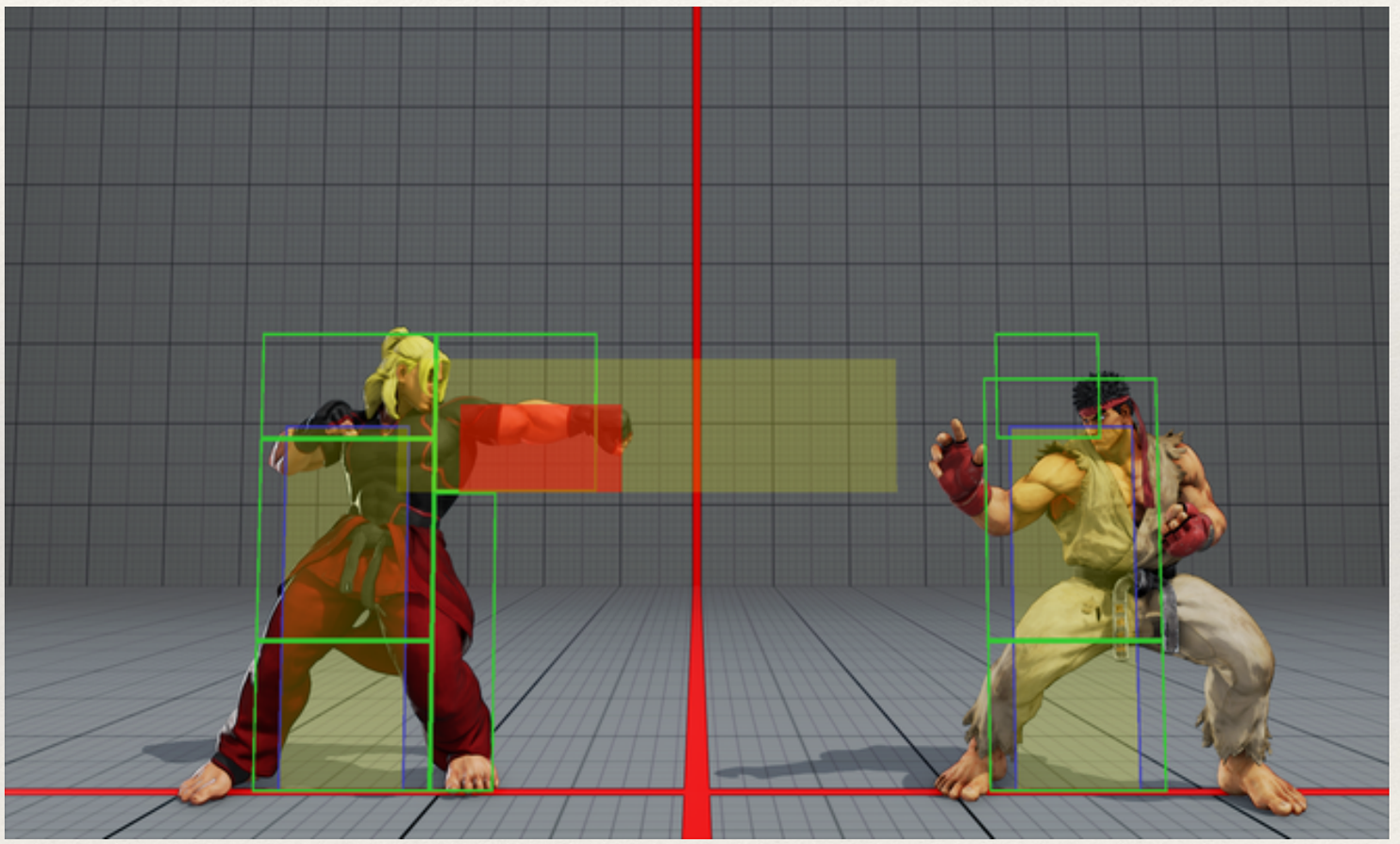
Ground box detects when sprite feet are in contact with ground geometry

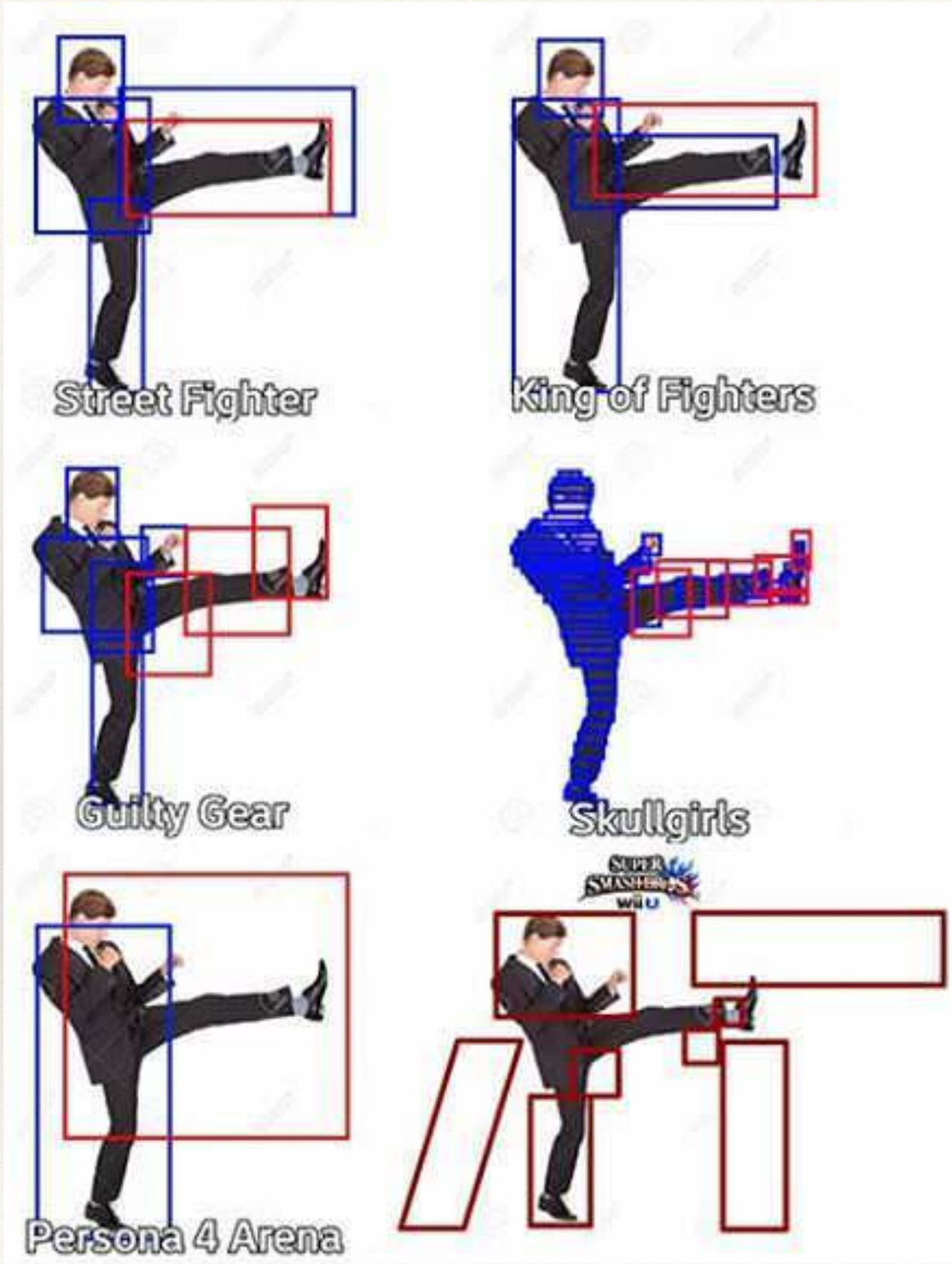
Multiple Hitboxes

- What are other benefits of multiple hitboxes?

Hitboxes vs Hurtboxes

- Hitboxes are collision boxes that register when a character's attack collides with an enemy
- Hurtboxes are collision boxes that register when attacks colliding with the character




Street Fighter
King of Fighters
Guilty Gear
Skullgirls
Persona 4 Arena


MONSTER HUNTER
DARK SOULS

Entering Hit State

- **Hit state** occurs when the character is performing an attack
  - Player inputs not processed until action completes
- **Stun state** occurs when the character is hit by an attack
  - Player inputs not processed until stun state ends
  - Often character is made **invincible** until the state ends
  - Often character is **knockbacked** away from attacking enemy

Input Buffer

- Players can often provide multiple inputs within the same frame
  - Essential in fighting and action games
- Even for simple interactions, necessary to **buffer** inputs in Processing
  - Use a boolean array to set when individual keys are pressed
  - `boolean [] inputBuffer = new boolean [4];`
  - `inputBuffer` holds information when left, right, up, down are pressed

Sound in Processing

- Add sound library via Sketch -> Import Library -> Add Library -> Sound
  - `import processing.sound.*;`
- Library contains variety of generators and filters
  - Noise
  - Oscillators
  - Audio input/output
  - Effects

SoundFile

- SoundFile allows playback and manipulation of sound files
  - `play()` starts playback
  - `loop()` starts looped playback
  - `stop()` stops playback
- Sound playback can be associated with start of program (background music) or triggered by specific actions (FX)

Minim Audio Library

- Easy-to-use pre-built classes
- Built into Processing
  - Sketch -> Import Library -> Add Library -> Minim
- http://code.compartmental.net/tools/minim/