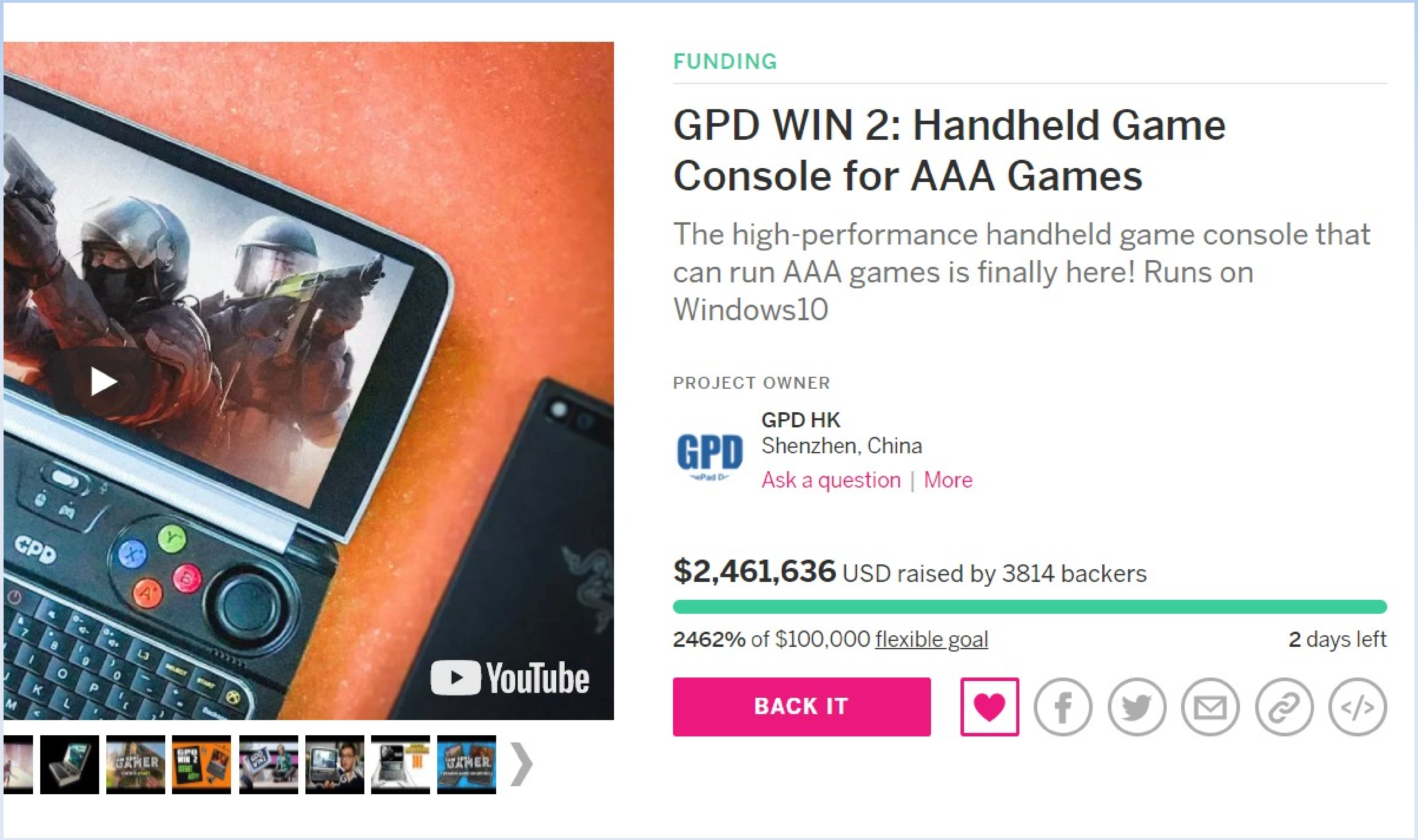
CS 134

MORE
Genre Specific Physics

Today in Video Games



FUNDING

GPD WIN 2: Handheld Game Console for AAA Games

The high-performance handheld game console that can run AAA games is finally here! Runs on Windows10

PROJECT OWNER

GPD HK
Shenzhen, China
Ask a question | More

$2,461,636 USD raised by 3814 backers

2462% of $100,000 flexible goal 2 days left

Genre Specific Physics

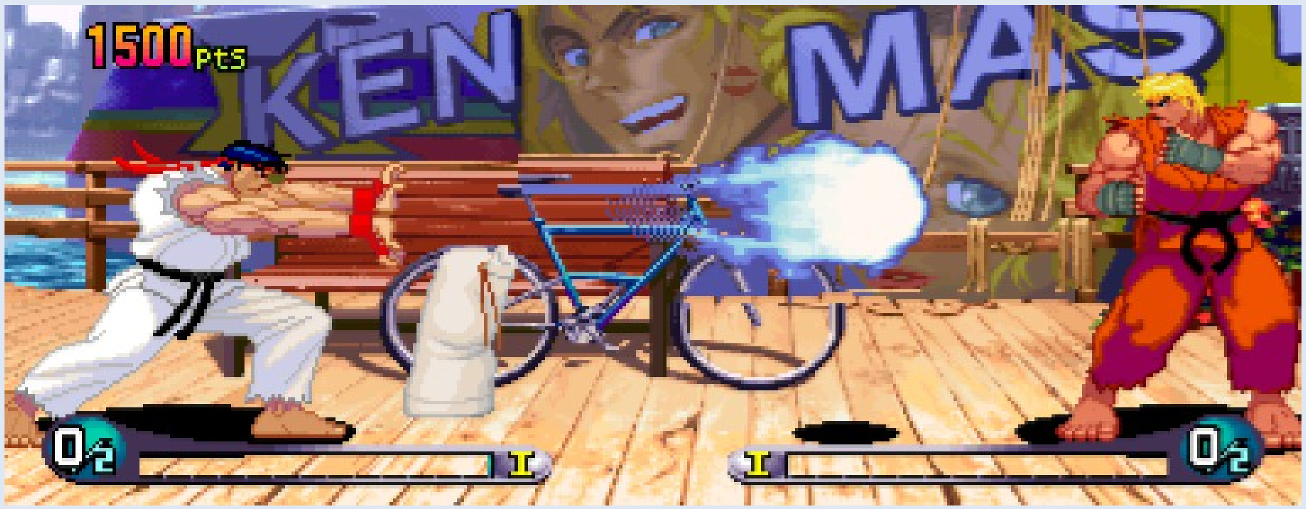
- Physics so far is enough for many different game genres
  - Platformer, RPG, Shooter, Metroidvania, Sports…
- Two key genres need more advanced physics
  - Fighting Game
  - 2D Brawler

Fighting Game Physics

- Only two sprites, both with really high quality art
- Many different moves for each character
- Each move is extremely unique

Fighting Game Physics

- Collision resolution is very simple!
  - Damage -OR- Push
- Nothing to interact other than the characters or projectiles



Fighting Game Physics

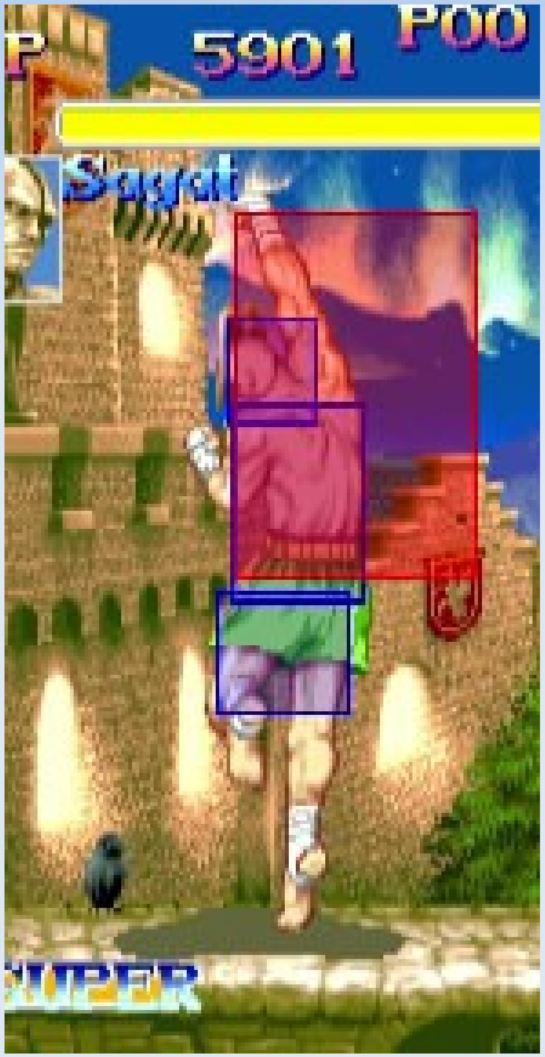
In fighting games, animation controls physics

- Most Games:
  - Input sets player state and movement
  - State set animations
  - Physics resolution sets state

- Fighting Games:
  - Input sets animations
  - Animations set movement and collision
  - Physics resolution sets animation

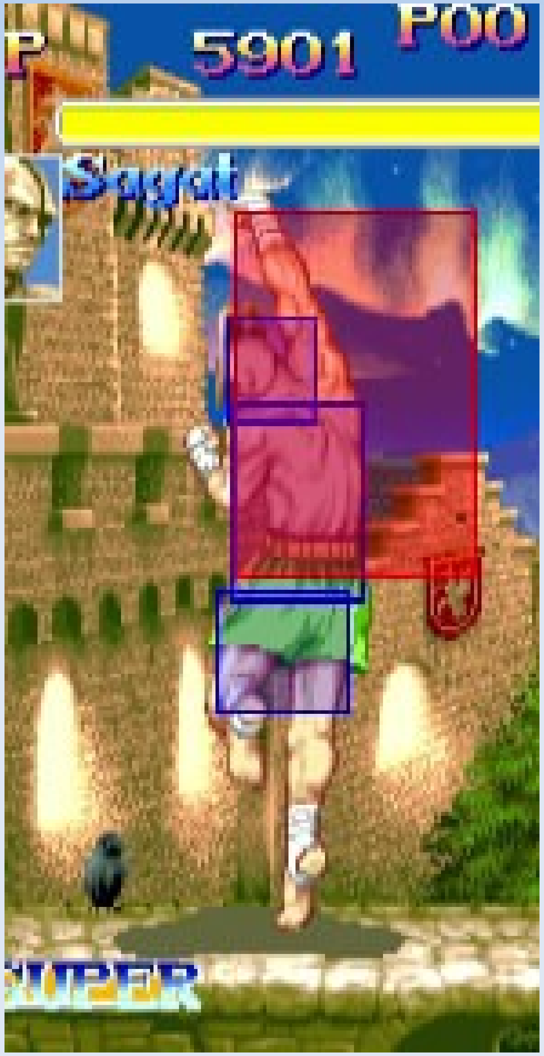
Fighting Game Physics

- Animation has additional gameplay data added
- Hitboxes
- Motion
- Let’s see a video...

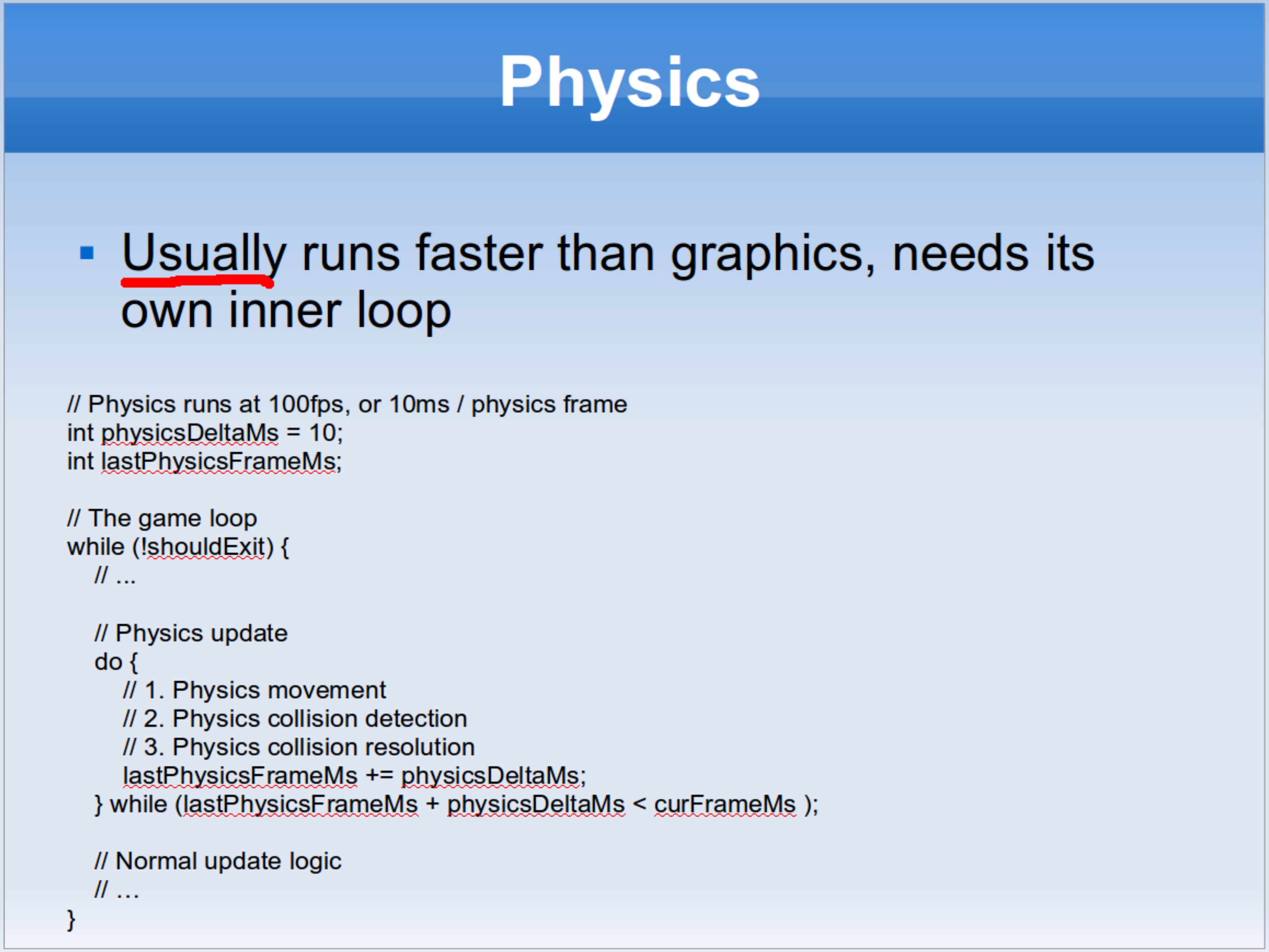


Fighting Game Physics

```
class AnimationDef {
    String name;
    FrameDef[] frames;

    float motionSpeed;
    boolean isJump;
    int damage;
}

class FrameDef {
    int image;
    float frameTimeSecs;
    int w;
    int h;

    AABB[] attackBox;
    AABB[] vulnerableBox;
    AABB collisionBox;
}
```



Fighting Game Physics

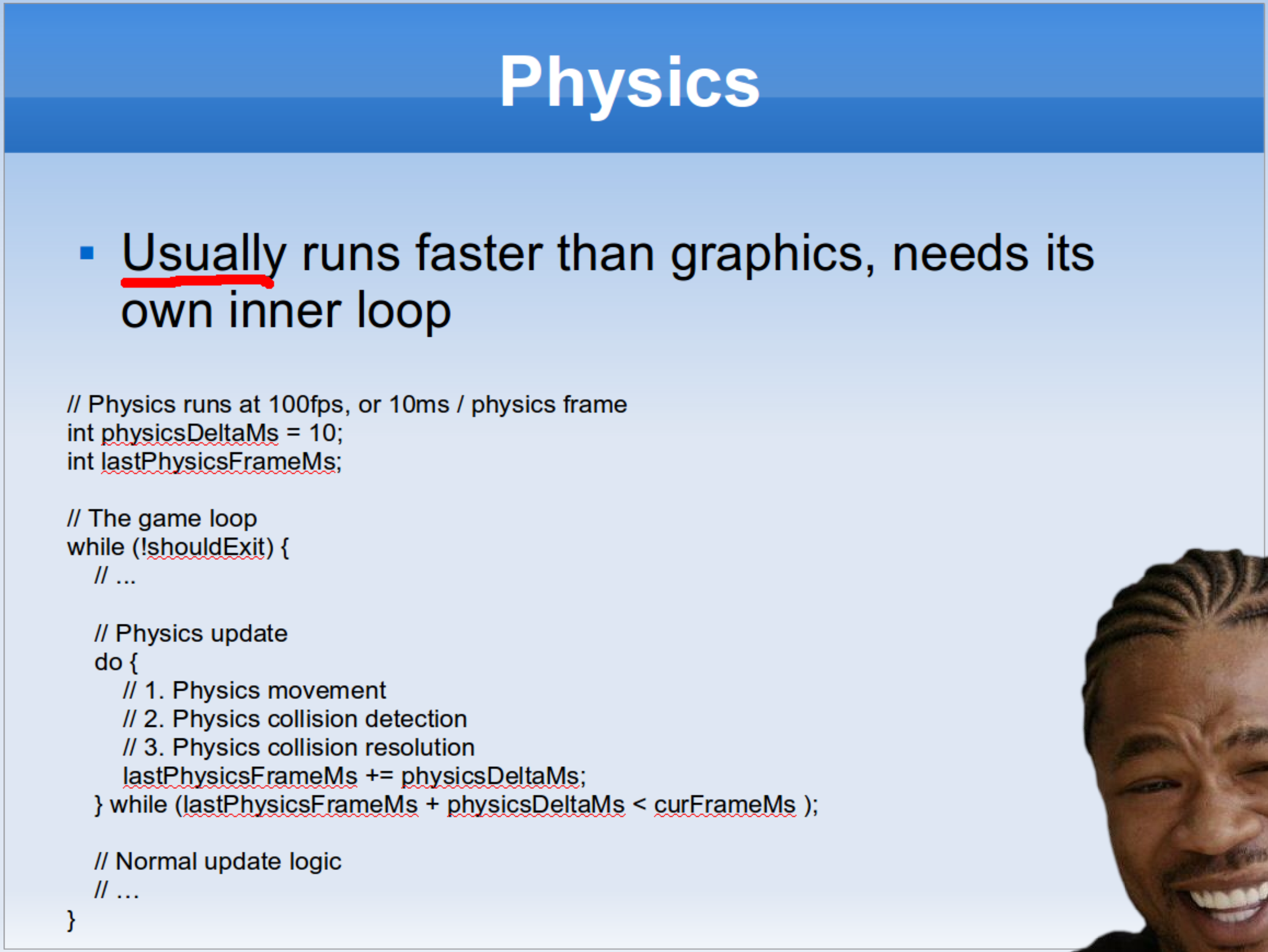


Physics

- Usually runs faster than graphics, needs its own inner loop

```
// Physics runs at 100fps, or 10ms / physics frame
int physicsDeltaMs = 10;
int lastPhysicsFrameMs;

// The game loop
while (!shouldExit) {
   // ...

   // Physics update
   do {
      // 1. Physics movement
      // 2. Physics collision detection
      // 3. Physics collision resolution
      lastPhysicsFrameMs += physicsDeltaMs;
   } while (lastPhysicsFrameMs + physicsDeltaMs < curFrameMs );

   // Normal update logic
   // …
}
```

Fighting Game Physics



Physics

- Usually runs faster than graphics, needs its own inner loop

```
// Physics runs at 100fps, or 10ms / physics frame
int physicsDeltaMs = 10;
int lastPhysicsFrameMs;

// The game loop
while (!shouldExit) {
   // ...

   // Physics update
   do {
      // 1. Physics movement
      // 2. Physics collision detection
      // 3. Physics collision resolution
      lastPhysicsFrameMs += physicsDeltaMs;
   } while (lastPhysicsFrameMs + physicsDeltaMs < curFrameMs );

   // Normal update logic
   // ...
}
```

Fighting Game Physics

- With physics tied so much to animation, it no longer is going faster than rendering
- Now animation will also be fixed framerate
  - Because animation now is physics
- Animation needs to change to be frame based instead of time based

Fighting Game Physics

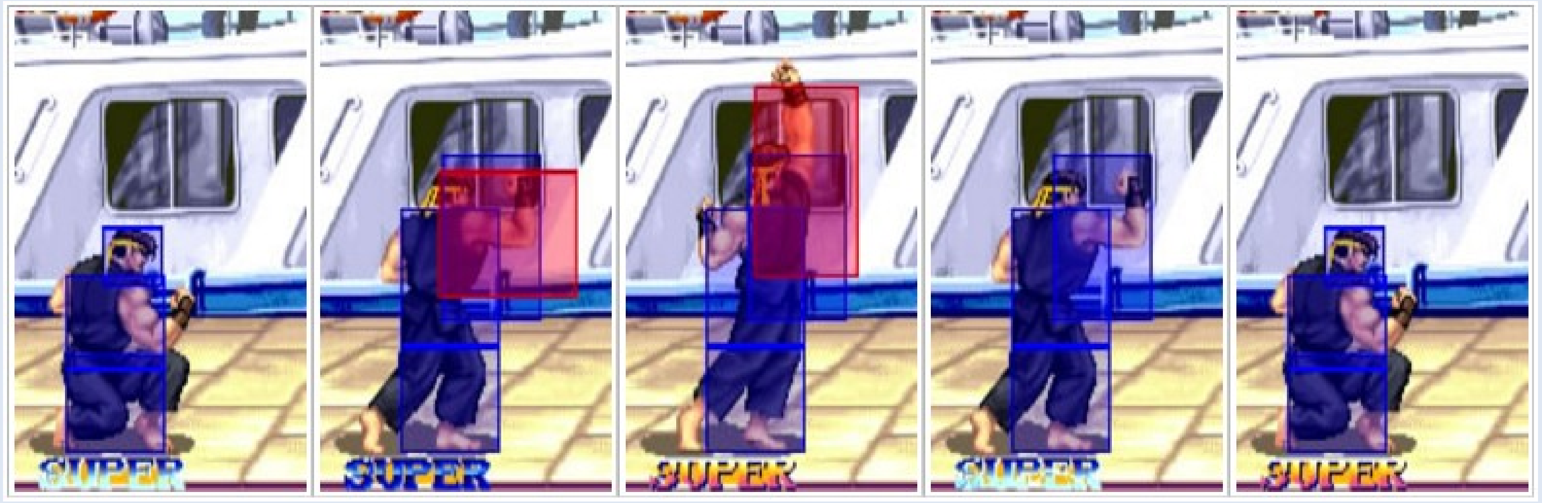
```
// Animation and Physics runs at 10fps, or 100ms / frame
int animationDeltaMs = 10;
int lastAnimationFrameMs;

// The game loop
while (!shouldExit) {
    // ...

    // Gameplay animation update
    while (lastAnimationFrameMs + animationDeltaMs < curFrameMs ) {
        // 1. Animation update
        // 2. Physics movement
        // 3. Physics collision detection
        // 4. Physics collision resolution
        lastAnimationFrameMs += animationDeltaMs;
    }

    // Normal update logic is much smaller, non-gameplay animations and timers
    // …
}
```

Fighting Game Physics

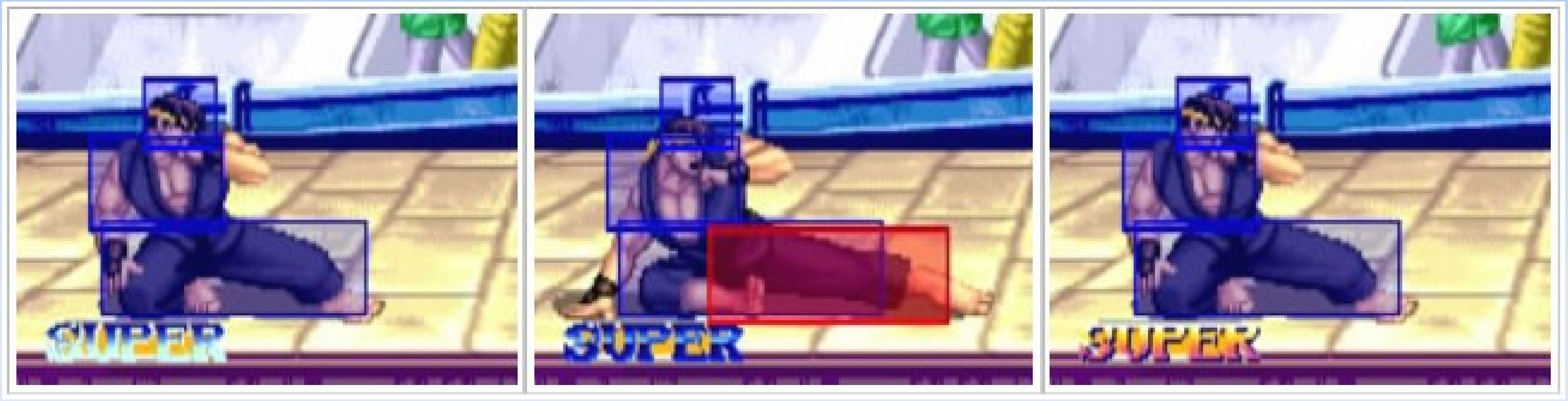
- For each piece of art, you need to define:
  - List of vulnerable boxes
  - List of attack boxes
  - A collision box
- If two collision boxes overlap, resolve by pushing the players apart
- If an attack box overlaps a vulnerable box, deal damage and set to a hit animation

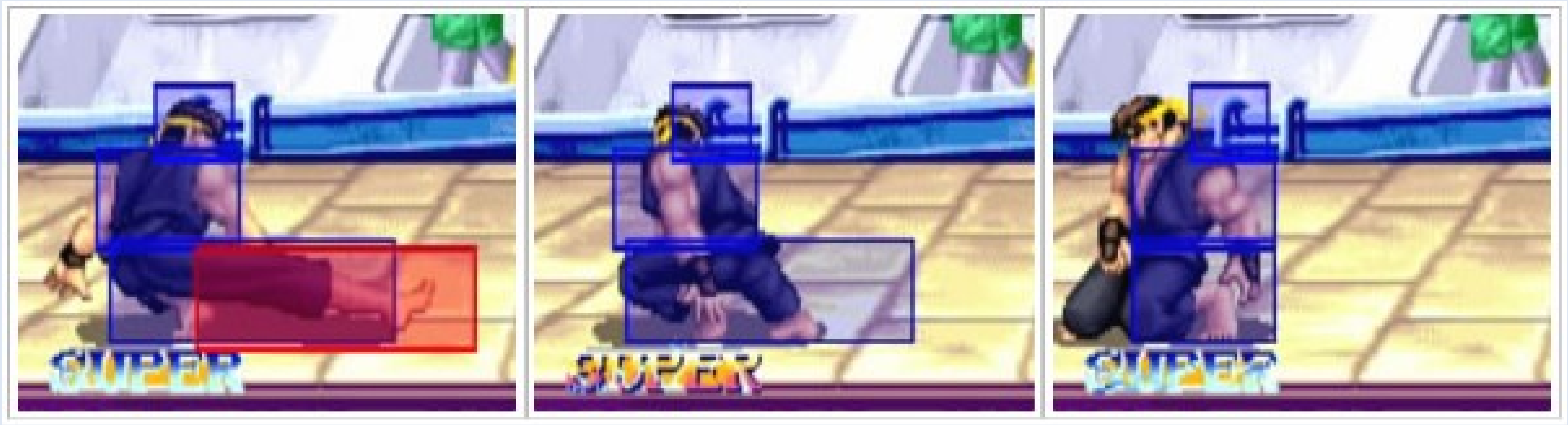
Fighting Game Physics

- And create lots and lots and lots of content

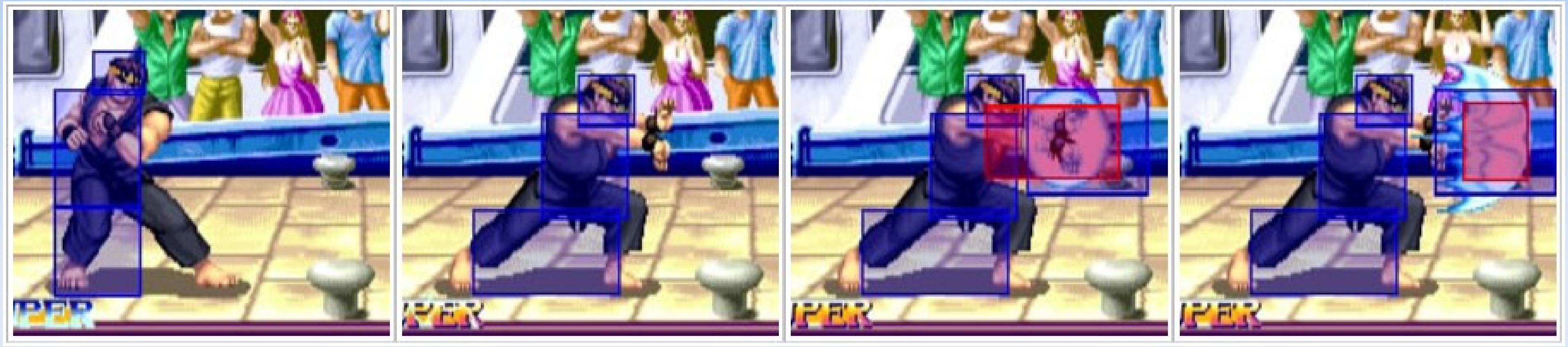


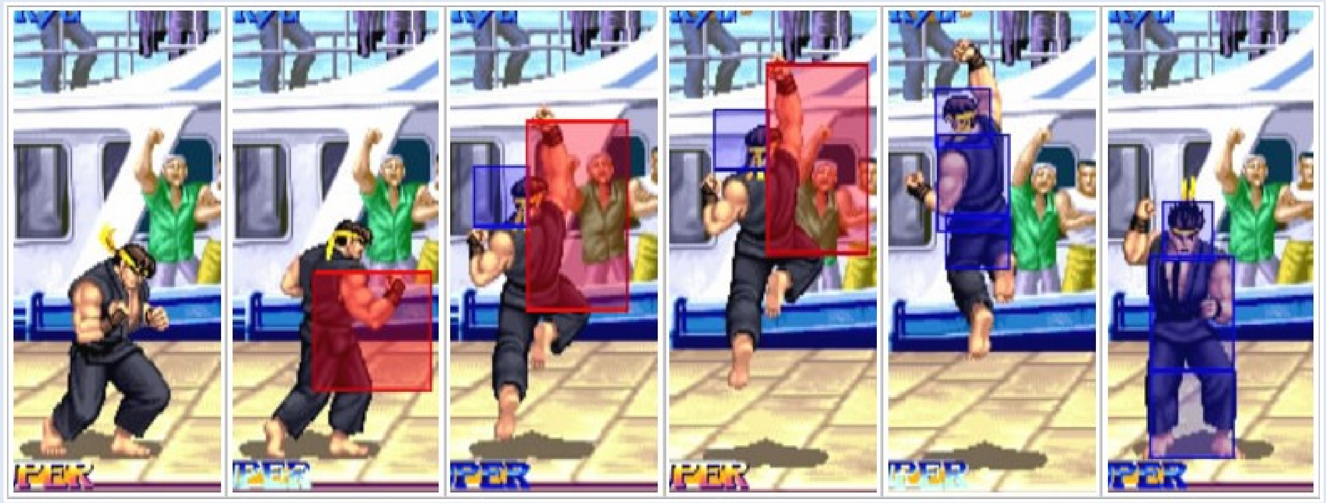
Fighting Game Physics



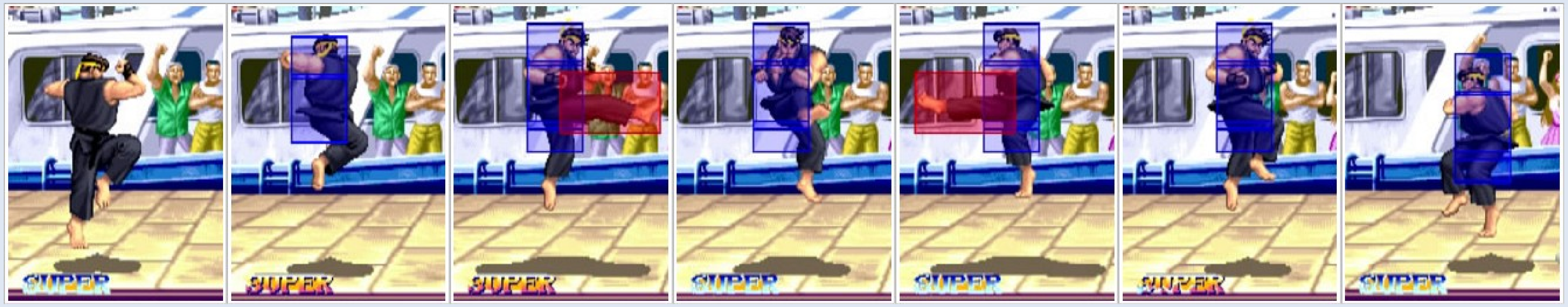


Fighting Game Physics





Fighting Game Physics



Fighting Game Physics

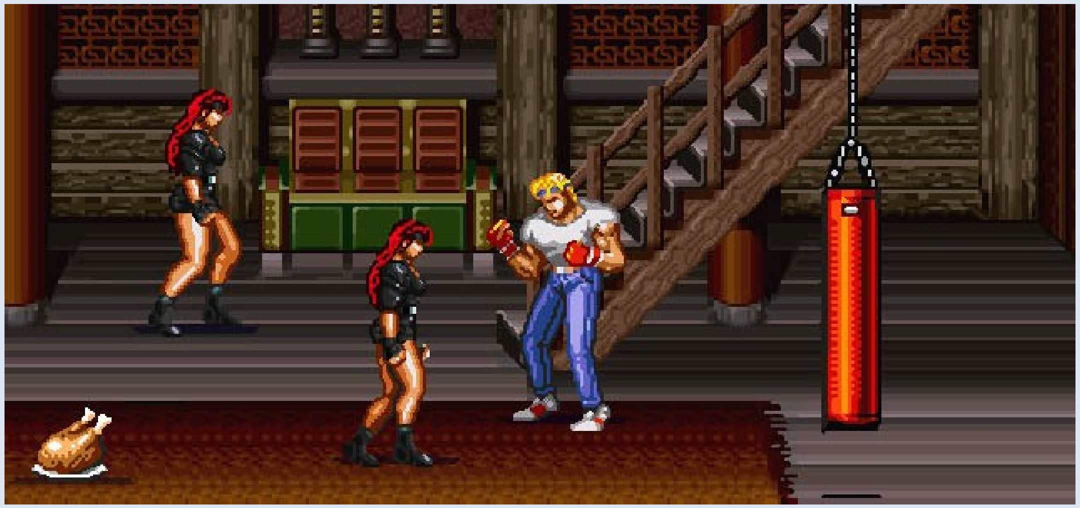
For a project in this class,
keep the art under control!

Fighting Game Physics

Questions?

Brawler Physics

- Side scrolling fighter where the players fight lots and lots of enemies.



Brawler Physics

- Borrow a lot of the techniques from fighting games
  - Animation based collisions, lots of different moves
- New need:
  - Entire game has to be playable in 3D
  - All bounding boxes should be 3D

Brawler Physics

- AABB3D / AABB3D collision detection
- Check in order:
  - Is box1 left of box2?
  - Is box1 right of box2?
  - Is box1 above box2?
  - Is box1 below box2?
  - **Is box1 in front of box2?**
  - **Is box1 behind box2?**

Brawler Physics

- To do this, you need two new fields in AABB3D, front and back.

- class AABB3D {

  public float left, right;

  public float top, bottom;

  public float front, back;

  }

Brawler Physics

That's it!

Brawler Physics

Questions?

Homework 6

- Due March 23rd

- Add background collision detection and resolution to your game.
- Everything you've added so far must collide “sensibly”
  - Player, enemies, and projectiles
- You only need top-down collision to be handled

Homework 6

- Extra credit:
- Add one or two of the advanced physics techniques talked about

- Platformer physics
- Actor motion
- One way walls
- Pixel perfect collision
- Fighting game collision