CS 134

Frame Update, Level Representation & Graphics

Homework 1

Due tonight!

Any questions?

Timing

- A common problem I’ve seen is that on SOME computers, the player moves slowly and on other computers, the player moves fast.

```
// In game logic update:
spritePos[0] += 2;
```

Timing

```
// The game loop
long lastFrameNS;
long curFrameNS = System.nanoTime();
while (!shouldExit) {
    System.arraycopy(kbState, 0, kbPrevState, 0, kbState.length);
    lastFrameNS = curFrameNS;
    curFrameNS = System.nanoTime();
    long deltaTimeMS = (curFrameNS - lastFrameNS) / 1000000;

    // Actually, this runs the entire OS message pump.
    window.display();

    if (!window.isVisible()) {
        shouldExit = true;
        break;
    }

    // How often is this called?
    spritePos[0] += 2;

    gl.glClearColor(0, 0, 0, 1);
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT);

    glDrawSprite(spriteTex, spritePos[0], spritePos[1], spriteSize[0], spriteSize[1]);
}
```

Timing

- Remember, the game loop is limited by graphics, so different computers will go through the loop at different speeds.
  - Graphics prowess
  - Vsync
- We need to make sure the sprite moves at a constant pixels / sec
  - We wanted 2 pixels at 60 fps, so 120 pixels / sec
  - Pixels/sec * sec/frame = pixels/frame

Timing

- System.nanoTime()
  - Returns time as a nanosecond count
  - Subtract the value since the last frame to figure out how much time has passed since the last frame
  - Needs to be consistent across entire frame.

  - Historically, games measure time in milliseconds, so I am used to converting nanoseconds to milliseconds
      - ms = ns / 1,000,000

Timing

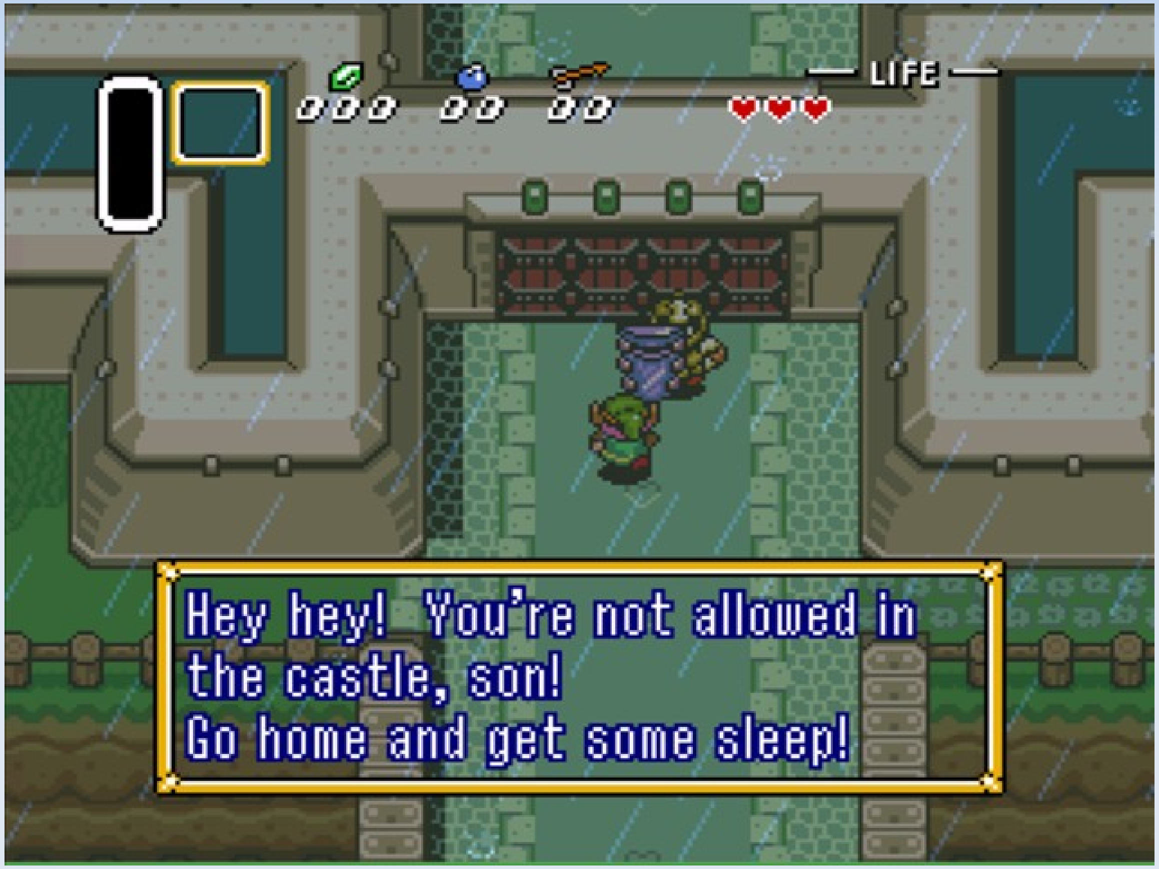
```
long lastFrameNS;
long curFrameNS = System.nanoTime();

while (!shouldExit) {
    System.arraycopy(kbState, 0, kbPrevState, 0, kbState.length);
    lastFrameNS = curFrameNS;

    // Actually, this runs the entire OS message pump.
    window.display();
    if (!window.isVisible()) {
        shouldExit = true;
        break;
    }

    currentFrameNS = System.nanoTime();
    int deltaTimeMS = (currentFrameNS - lastFrameNS) / 1000000;

    // Check keyboard input for player
    // Update positions and animations of all sprites

    gl.glClearColor(0, 0, 0, 1);
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT);

    // Draw background(s)
    // Draw sprites
    // Draw more background(s)
```

Timing

- For C, use SDL_GetTicks()
  - Returns time in milliseconds instead of nanoseconds.
  - Otherwise, identical.

2D Drawing

- Look at any Super Nintendo era game.
- Lots of cool graphical effects
- How did they make these worlds and draw them?

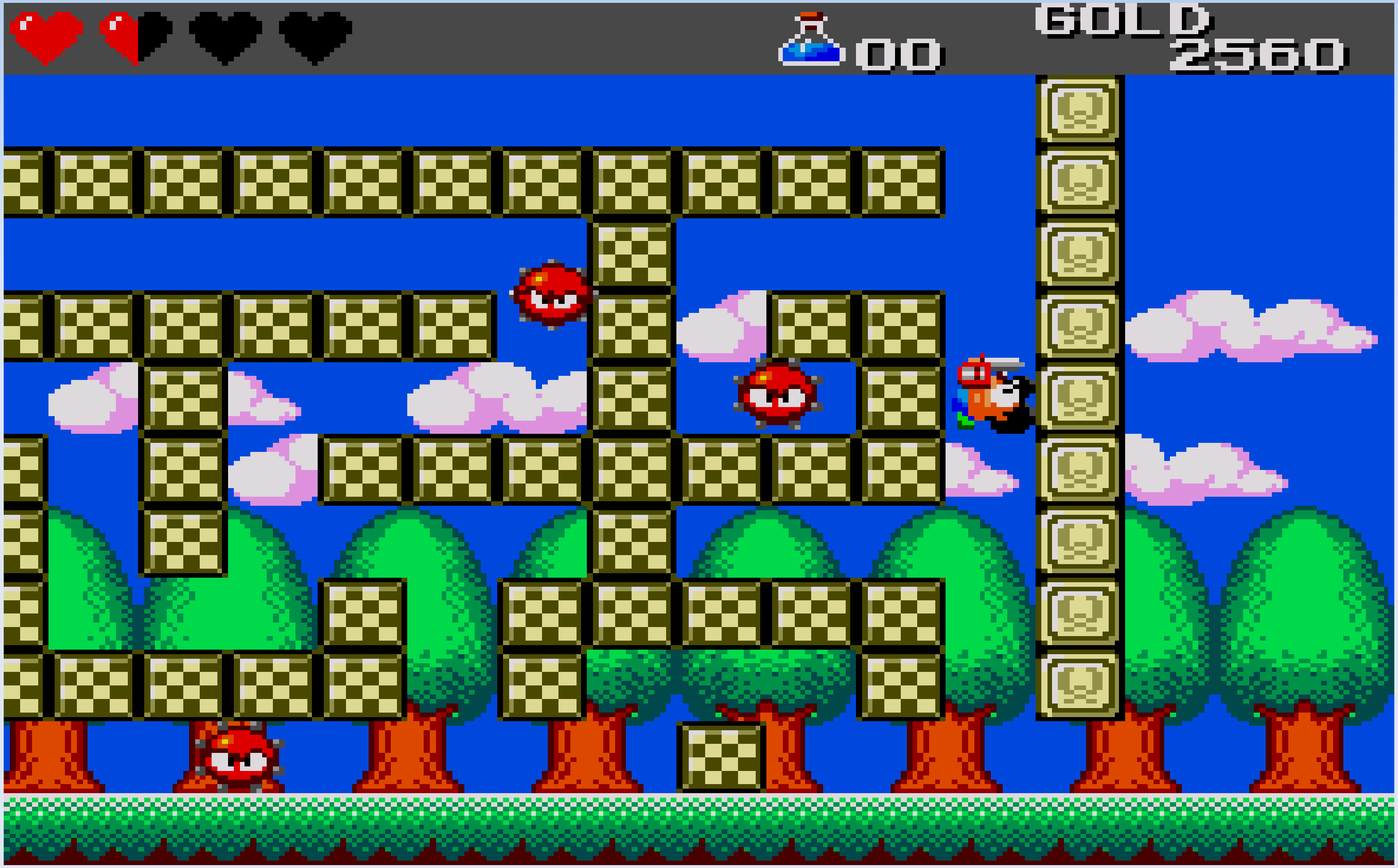

LIFE
Hey hey! You're not allowed in the castle, son!
Go home and get some sleep!

2D Drawing

- Static ”backgrounds”
- Animated ”sprites”
- Overlayed HUD info

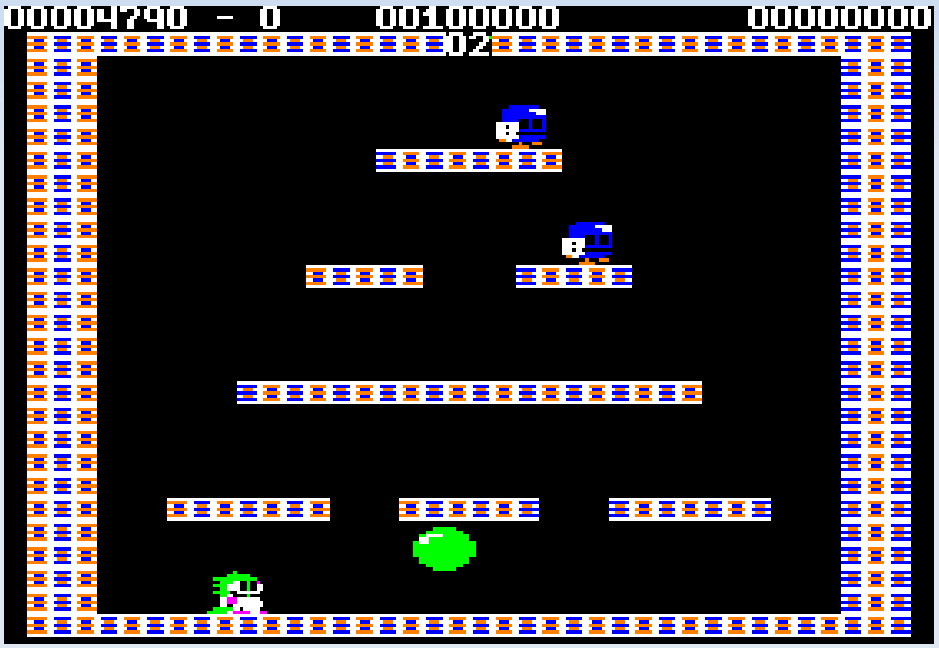
- No matter what, everything can be built off out of our single glDrawSprite function

2D Drawing



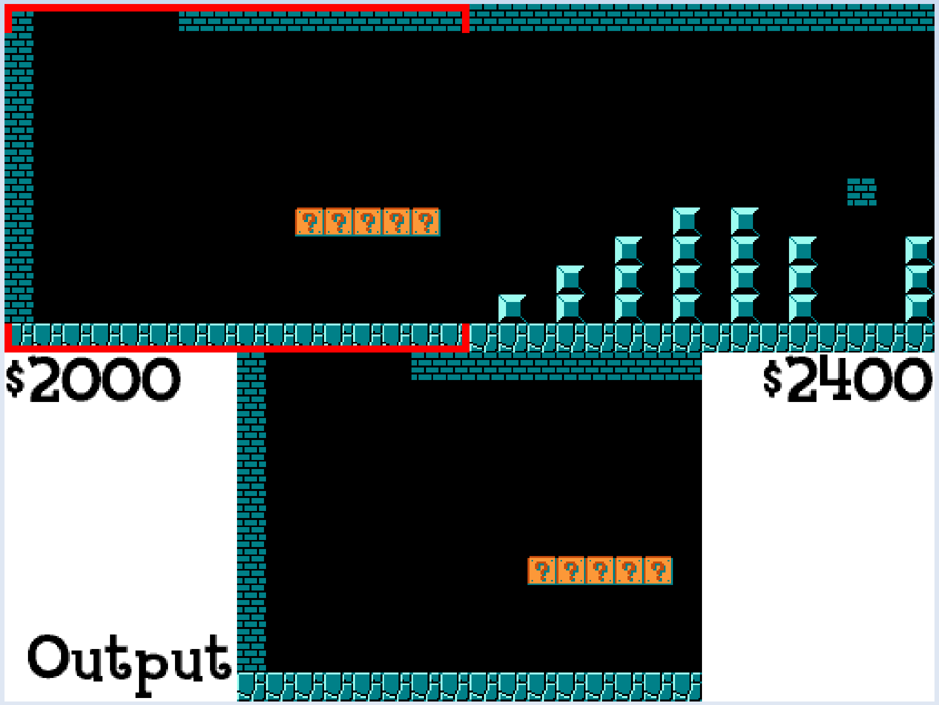
History

- Naive Implementation
  - Apple II (1977)
- One giant array
- Each pixel is a byte



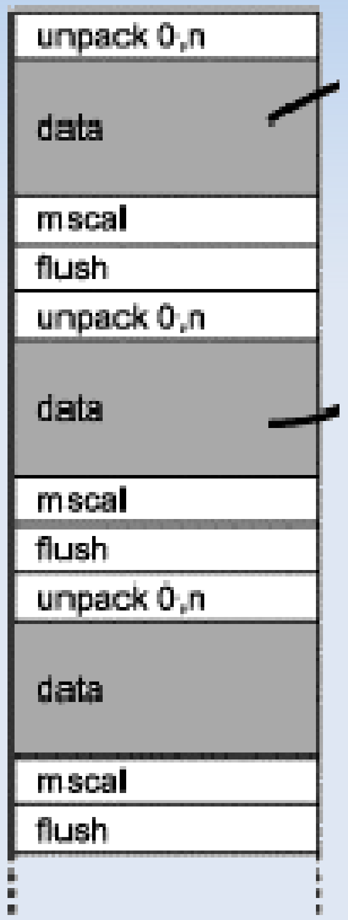
History

- Scroll BG w/ Sprites
  - NES (1985)
- Separate layers
  - BG, Sprite
- BG space is bigger than one screen



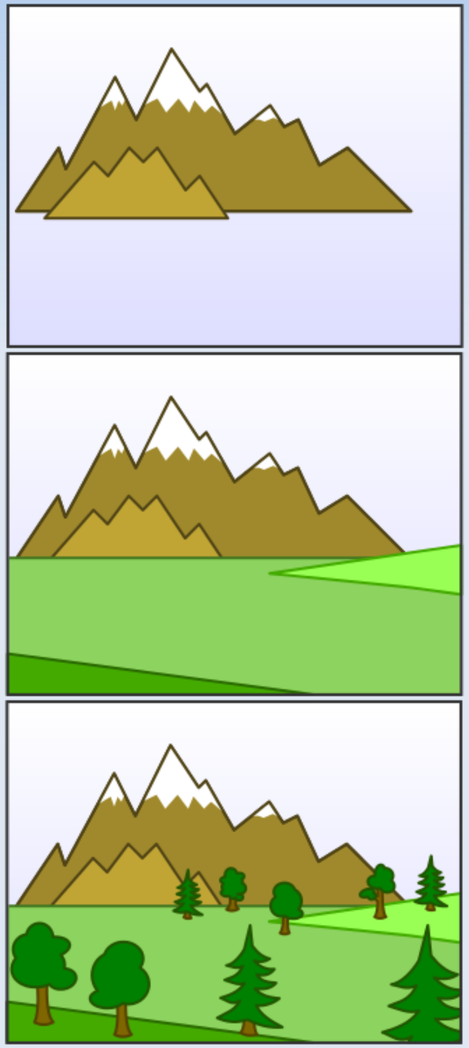
History

- Command Based
  - PlayStation (1995)
- Redraw everything every frame
- Huge array of draw commands



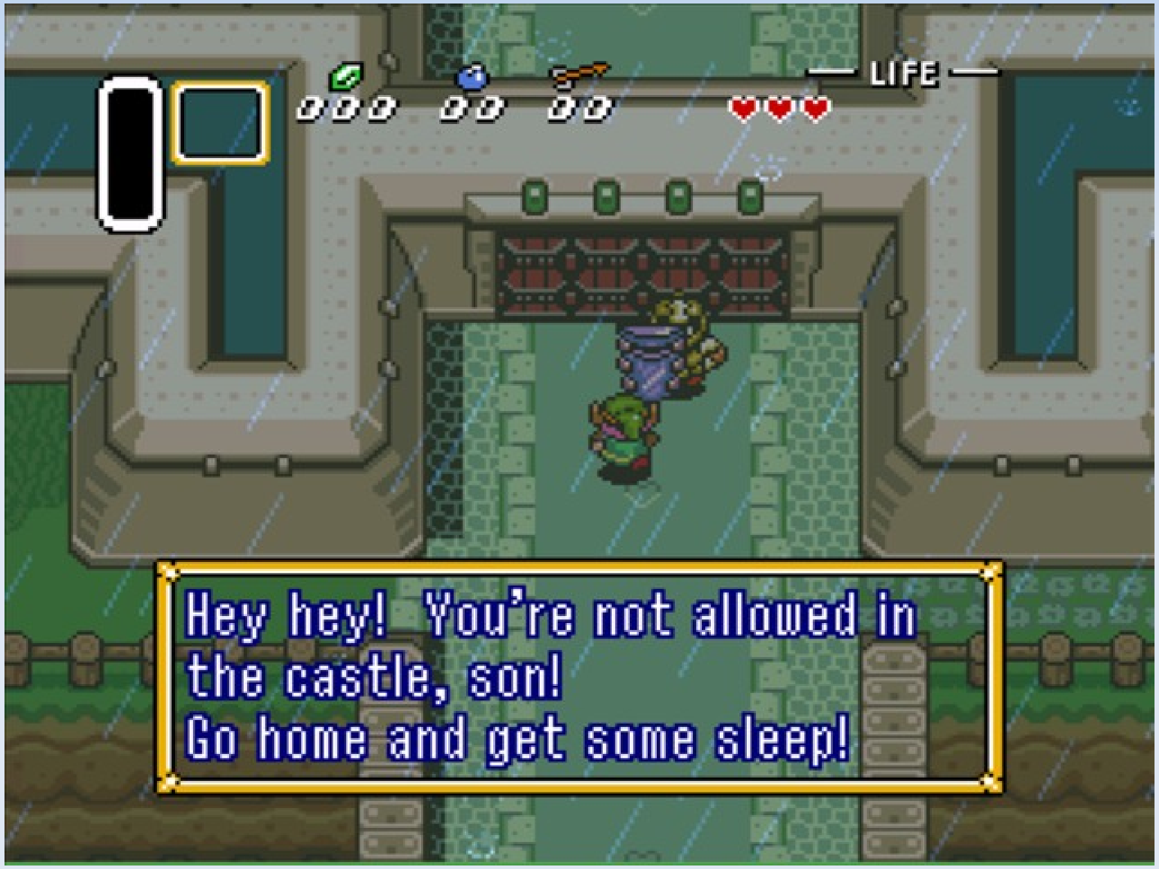
Painter's Algorithm

- Still using Command Based hardware today.
- For us, there's just one command – draw sprite
- Everything is completely redrawn every frame.
- Need to make sure you draw things in the right order.



Painter's Algorithm

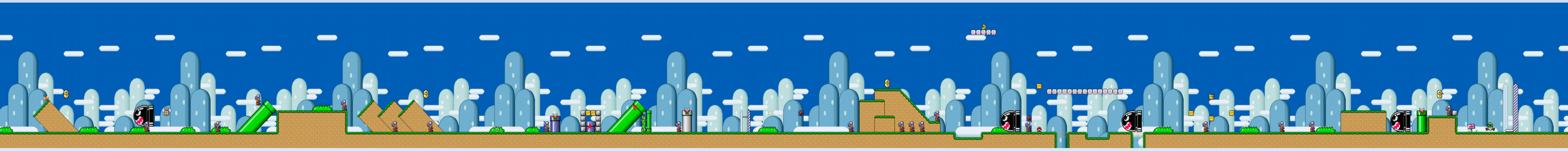
- Let's break this up into drawing order...



Hey hey! You're not allowed in the castle, son!
Go home and get some sleep!

Backgrounds

- You might want to have a texture for whole background.



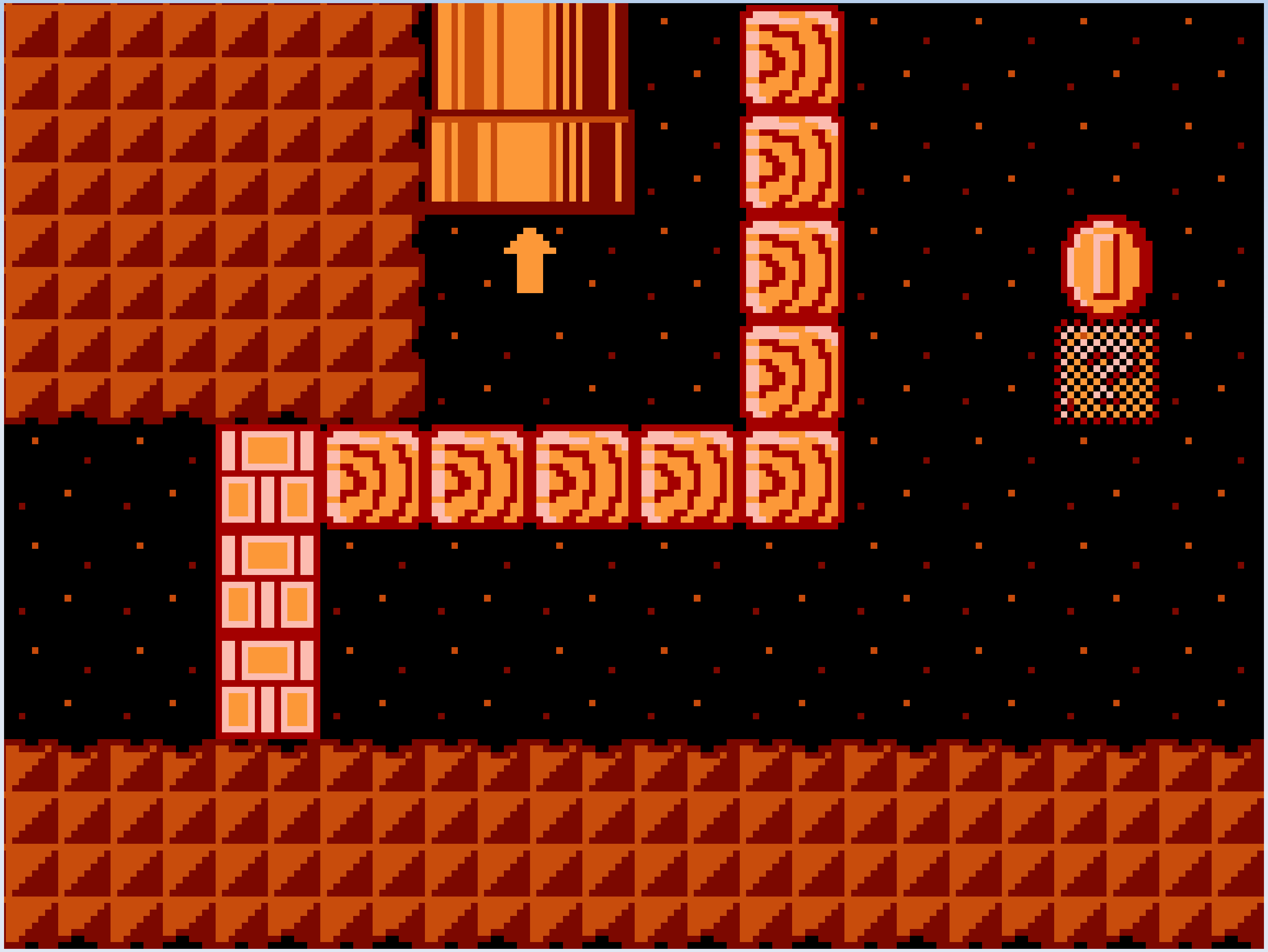
- This won't work.

Backgrounds

- Problems
- GL_MAX_TEXTURE_SIZE
  - glGetIntegerv(GL_MAX_TEXTURE_SIZE, &val);
  - Often 8196 or higher on PC, 2048 on mobile
- Art Time
  - Big textures take a lot of time to make
  - You need an artist to draw every single level

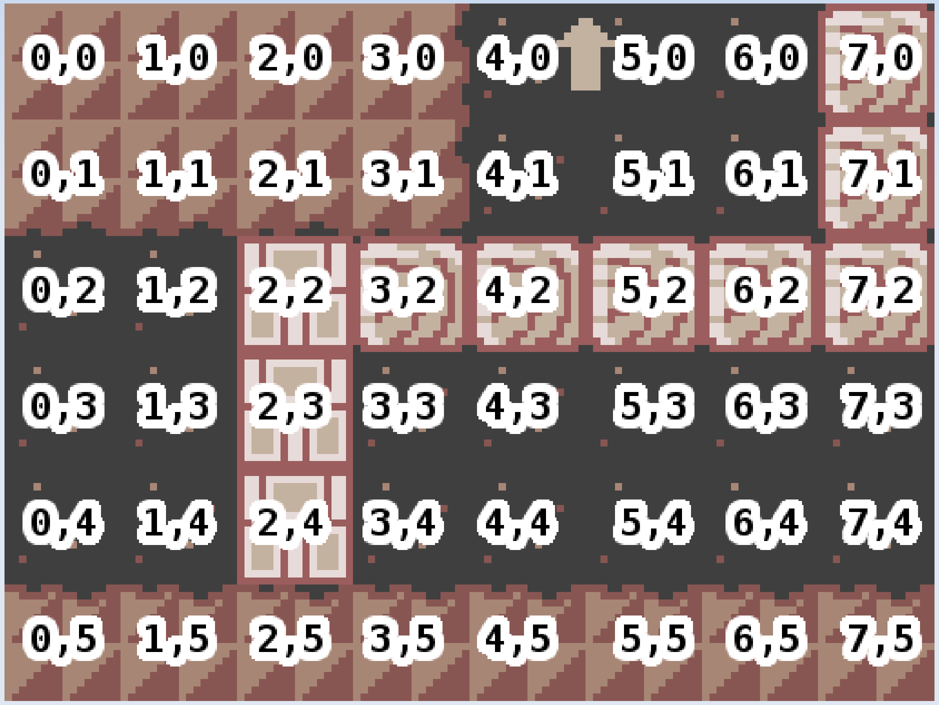
- Solution: Tiles

Backgrounds



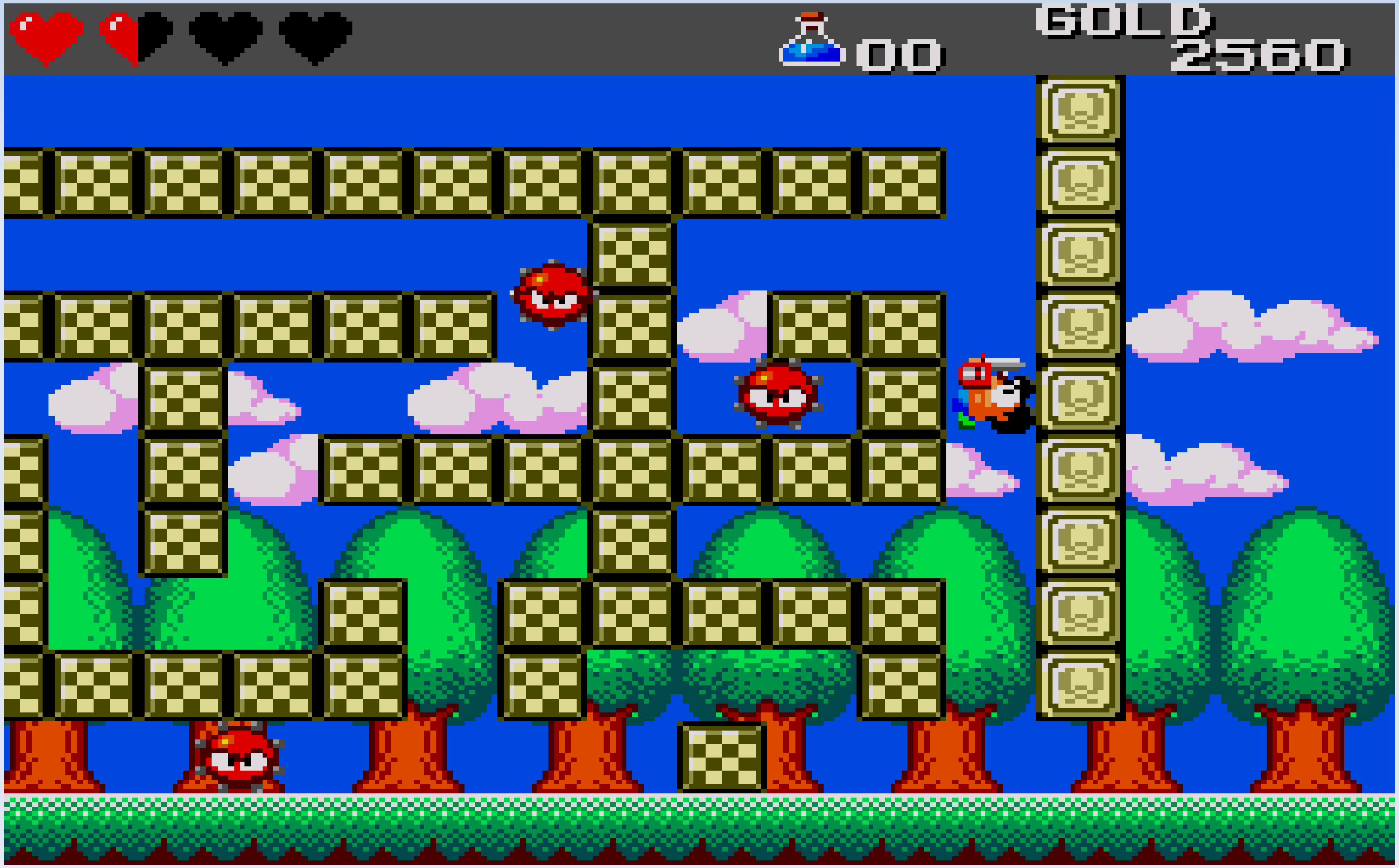
Backgrounds

- Level is 2D array of indexes
- Tile position:
  - x*w
  - y*h



Backgrounds

Can you see the tiles here?



Backgrounds

```
class BackgroundDef {
    int width;
    int height;
    int[] tiles;

    public int getTile(int x, int y) {
        return tiles[y * width + x];
    }
}
```

- You could write a file loader, but for now just define this in code, it's easier!

Backgrounds

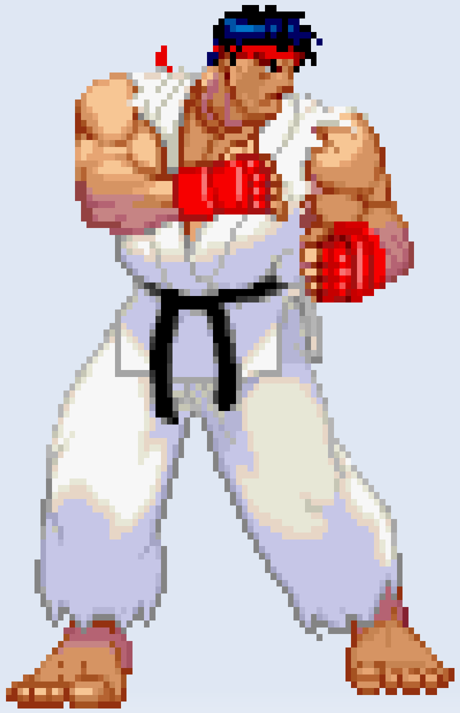
Questions?

Sprites

- Unlike Backgrounds, sprites change what you see from frame to frame
- But each animation is unchanging.

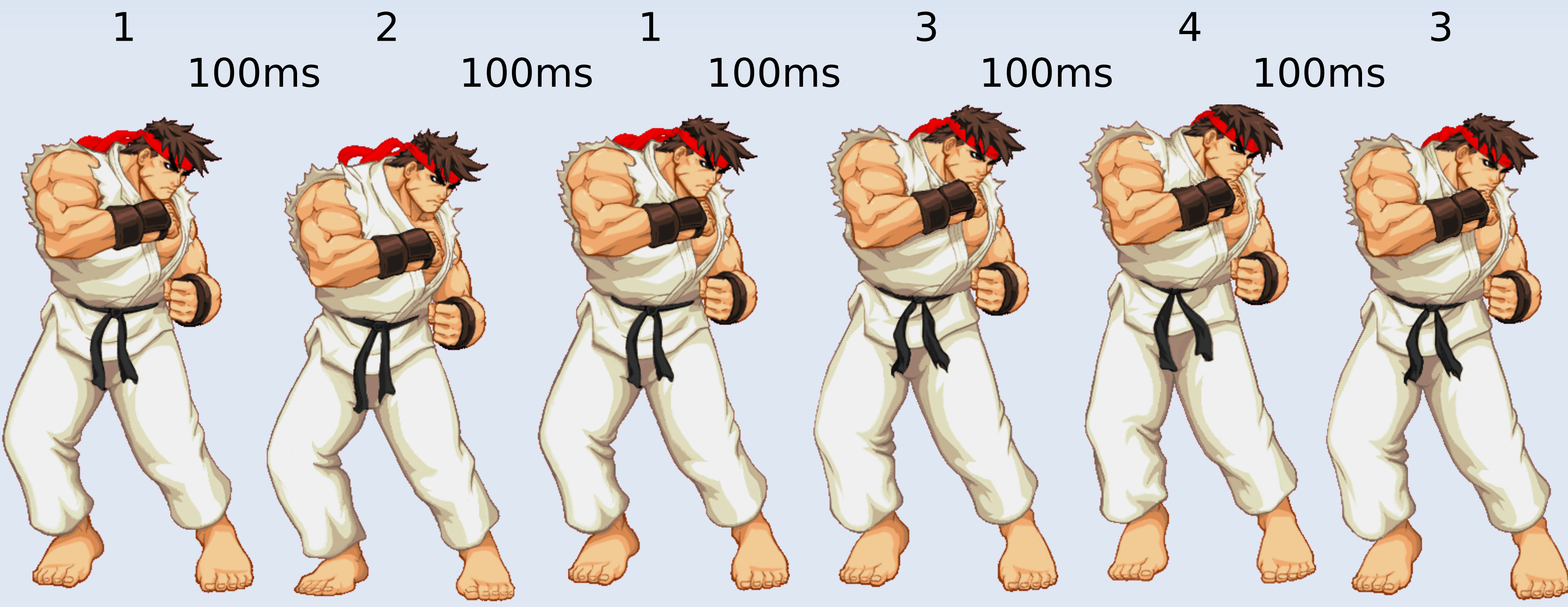
Sprites

- Idea: have a ”def” for the animation.



Sprites

- List of frames
- Each ”frame” has a time and an image



Sprites

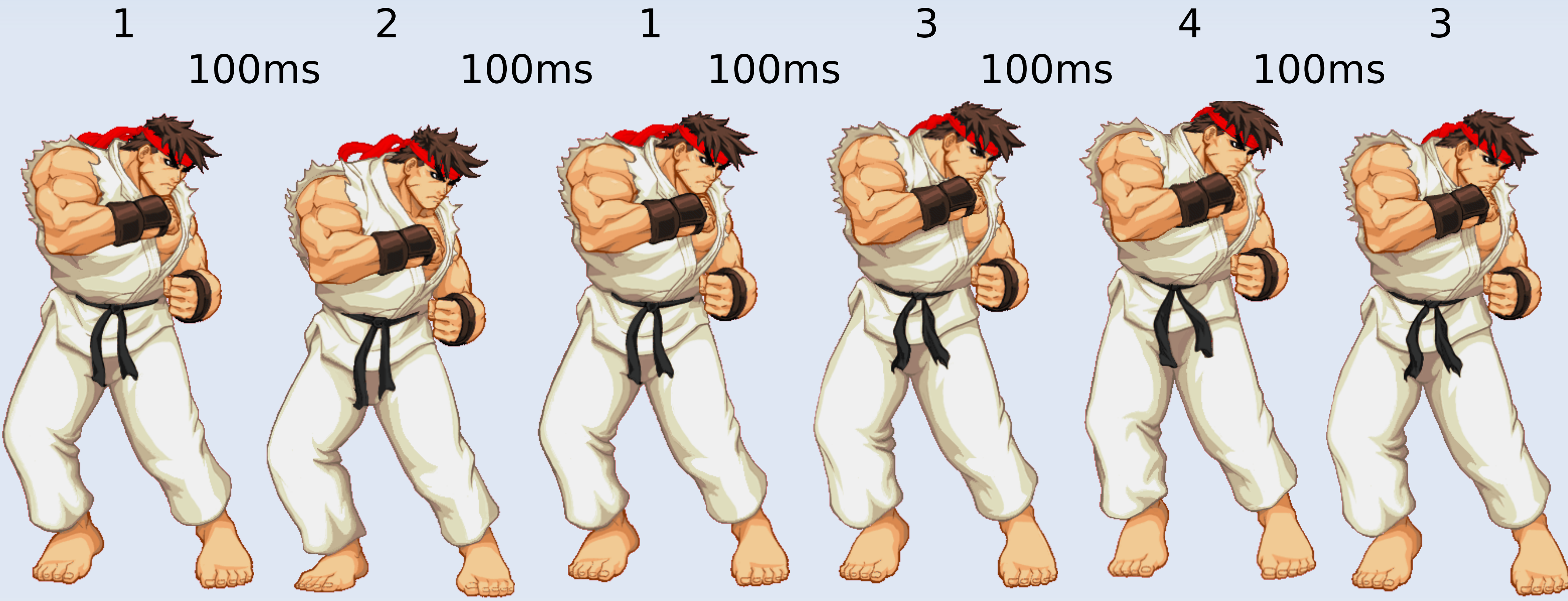
```
class AnimationDef {
    public FrameDef[] frames;
}

class FrameDef {
    public int image;
    public float frameTimeSecs;
}
```

Sprites

But wait, the AnimationDef alone is not enough to draw the current state of Ryu!

Sprites

- You also need to know:
  - Where in the animation you are
  - How much time until the next part of the animation



Sprites

```
class AnimationData {
    AnimationDef def;
    int curFrame;
    float secsUntilNextFrame;

    public void update(float deltaTime);
    public void draw(int x, int y);
}
```

- Every frame, the AnimationData for Ryu will change!

Level Representation

- Data / Defs
  - Split up actor information into changing (Data), unchanging shared (Defs). Share Defs among all actors.
  - Level data is Defs and per-actor data.
- Prototype based
  - Combine Data and Defs. Let both change.
  - Level data is Prototype and per-actor data.

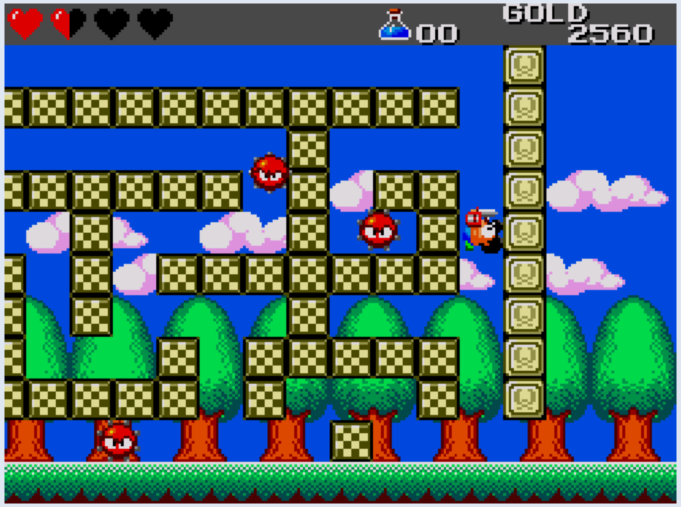
Summary

- Backgrounds are easy
  - Simple for loop!
  - Can have multiple backgrounds to have stuff in front of and behind sprites.
- Sprites are a bit harder
  - They have state!
  - But ultimately, you have a list of sprites and you call update() and draw() on each of them.

The Game Loop So Far

```
while (!shouldExit) {
    System.arraycopy(kbState, 0, kbPrevState, 0, kbState.length);

    // Actually, this runs the entire OS message pump.
    window.display();
    if (!window.isVisible()) {
        shouldExit = true;
        break;
    }

    // Check keyboard input for player
    // Update positions and animations of all sprites

    gl.glClearColor(0, 0, 0, 1);
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT);

    // Draw background(s)
    // Draw sprites
    // Draw more background(s)

    // Present to the player.
    window.swapBuffers();
}
```

LIFE
Hey hey! You're not allowed in the castle, son!
Go home and get some sleep!



Next Class

Homework

- Simple tiled background
- Controllable animating sprite moving around

- Character must now move ”sensibly” with arrows or WASD

Homework

- Sites with existing art:
  - http://www.spriters-resource.com/
  - http://spritedatabase.net/
- Note that sprites are usually all in one file, you will have to cut it up into pieces.

Homework

- Extra credit options:
  - Have the character have ”appropriate” animations for its motion (idle, move left, move right, etc.)
  - Have multiple non-player controlled characters move around the world and animate.