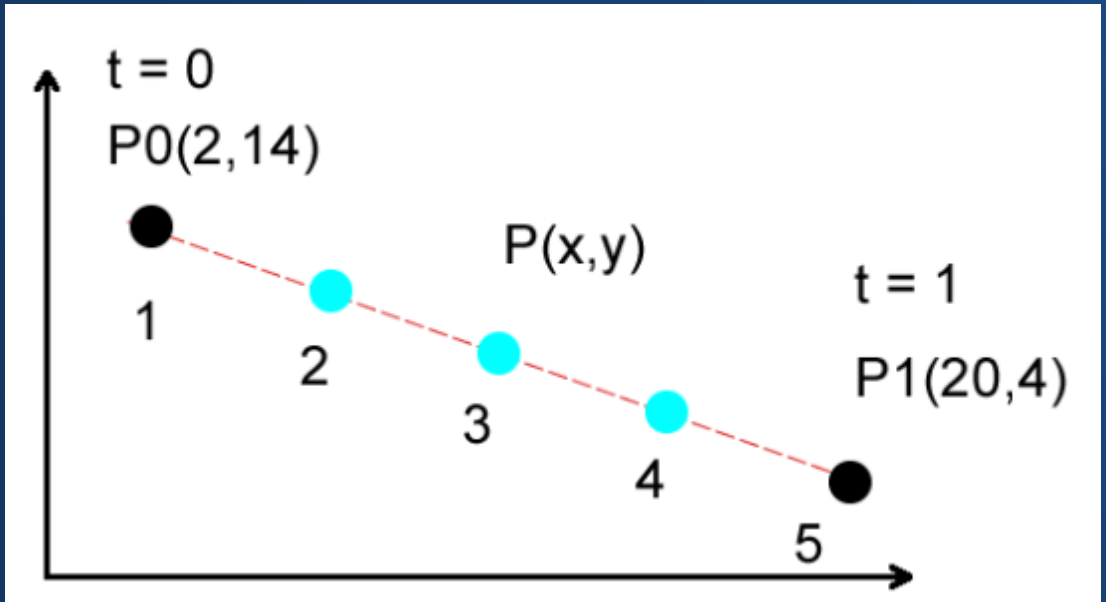
Timers and Animation

Animation

Made by drawing a sequence of images in succession, giving the impression of motion when played back.

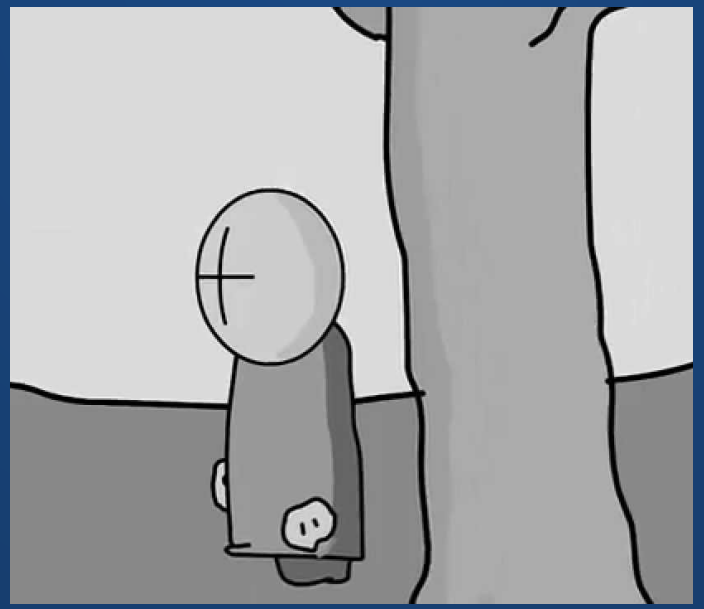


One way to generate these images is to interpolate between keyframes. This is mostly how we've animated things in this class so far.

Image-Based Animation

We can also just load and display a bunch of images!

These images can be hand-drawn (as in the case of flipbook animations) or created in animation programs like After Effects or Flash



Loading Animations in Processing

Same principle as loading a single image into a `PImage`, but now we need multiple of them!

Load images into an array, storing them in animation order.

```
1 PImage animation[100];
2 animation[0] = loadImage("anim-0.png");
3 animation[1] = loadImage("anim-1.png");
4 animation[2] = loadImage("anim-2.png");
5 // Yuck
```

But uh...

Image Animations in Processing

The `nf()` function formats a number into a string. This is a convenient way to avoid having to hardcode image names.

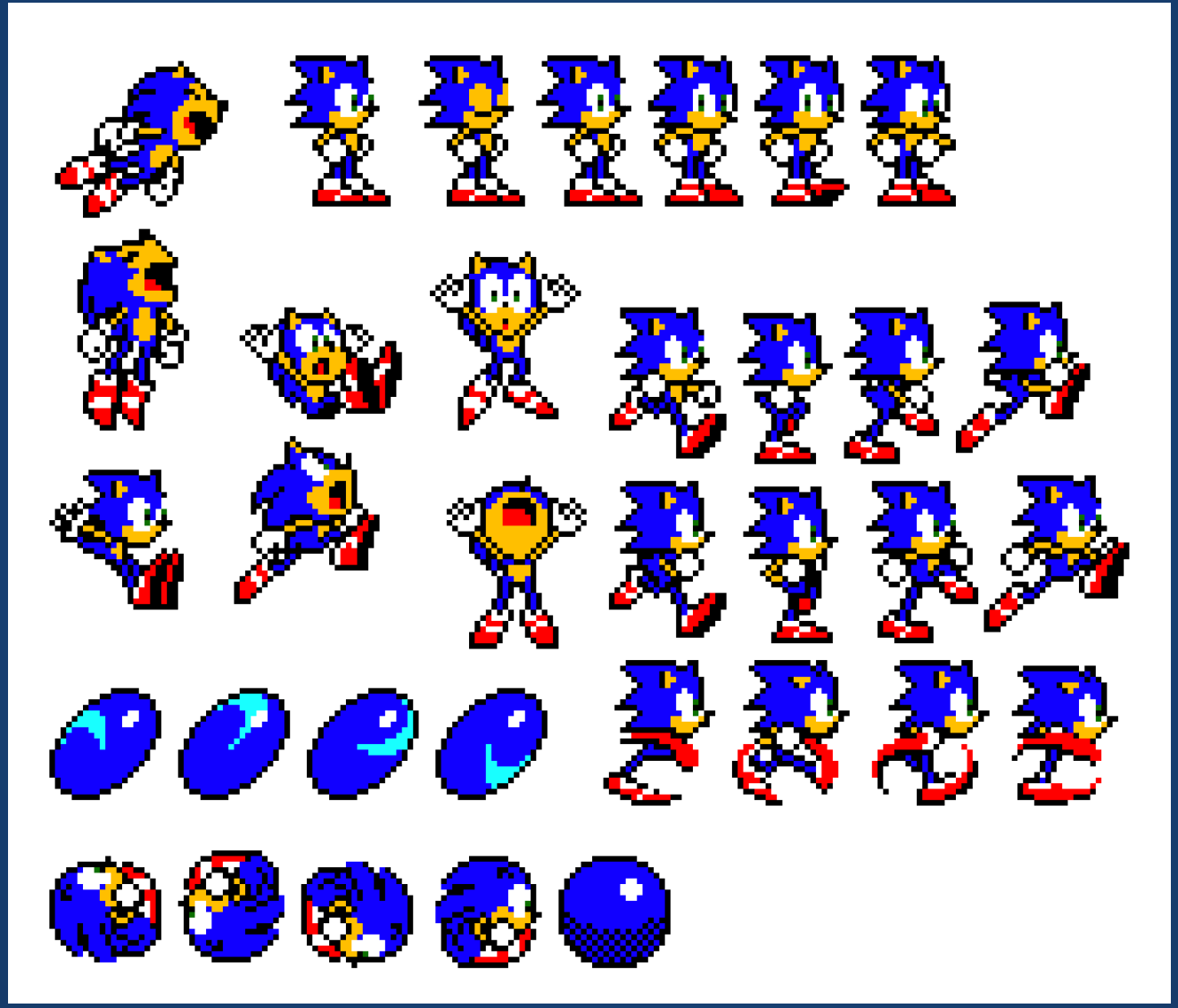
`nf()` can also zero-pad numbers, which is great for keeping the frames in sorted order e.g. in a file browser.

To play the animation, loop over the animation array.

Can use the modulus operator to wrap the animation.

Processing has a global variable `frameCount` that auto-increments every time `draw()` is called.

Sprite Sheets



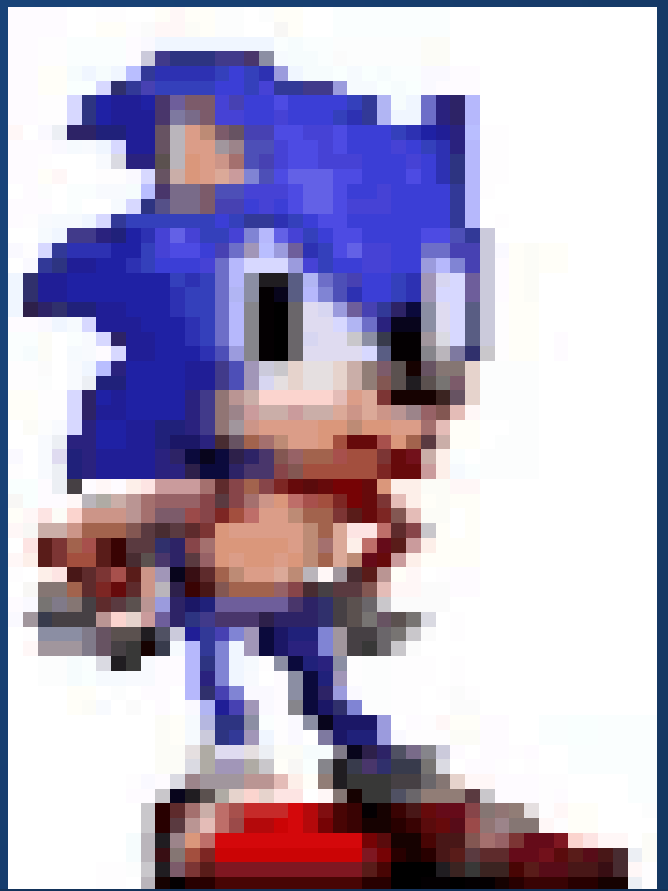
Hands-On: Sprites

1. Collect or create a sequence of images to use as a sprite.
2. Within `setup()`, load these images into an array. Use a loop (potentially with `nf()`) instead of manually loading images one-by-one.
3. Within the `draw()` function, display the images in sequence at a given location
4. Use `frameCount` and the modulus operator to make the sprite loop infinitely.
5. Experiment with `frameRate()` to change the speed of the animation. Use one you think looks good.

Timers

We tied our sprite animation speed to the frame rate.

This isn't always a good idea. In fact, I'm going to go ahead and say that it's a ~~bad~~ lazy idea, even though it's convenient and everyone does it.



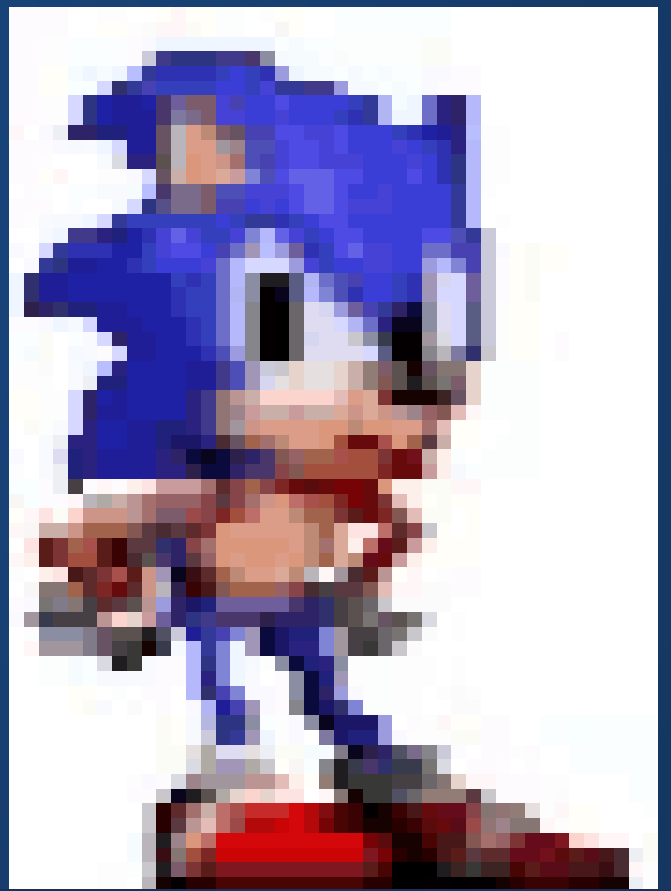
Timers

Instead of tying ourselves to the draw rate (i.e. do something every draw), we can look at a clock, and only take action if it has been enough time since the last time it happened.

Example for Sonic:

We want to advance the animation every 0.2 seconds (200ms).

Advance the frame if it has been at least 200ms since the frame was last advanced.



We can use millis() to read how many milliseconds have passed since the program started

```
1  int animationTimer = 0;
2  int animationTimerValue = 200;
3  int currentFrame = 0;
4  void draw() {
5    image(x_sprite[currentFrame], 20, 250);
6    // If it has been more than 200ms since the last "tick"
7    if ((millis() - animationTimer) >= animationTimerValue) {
8      currentFrame = (currentFrame + 1) % numFrames;
9      // Set time of current tick
10     animationTimer = millis();
11   }
12 }
```

Hands-On: Timers

1. Experiment with the code example for a timer-based sprite
2. Turn this into a `Timer` class, which has an interval for activation and has a method which returns whether it's been long enough since it was last activated.
3. Add a method to pause your timer.
4. Use the Timer class to animate your sprite. Pause/unpause your timer when the user clicks, so that the user can pause your animation by clicking.

If Time

Talk with your group about the final project.

Index Cards!

1. Your name and EID.

2. One thing that you learned from class today. You are allowed to say "nothing" if you didn't learn anything.

3. One question you have about something covered in class today. You *may not* respond "nothing".

4. (Optional) Any other comments/questions/thoughts about today's class.