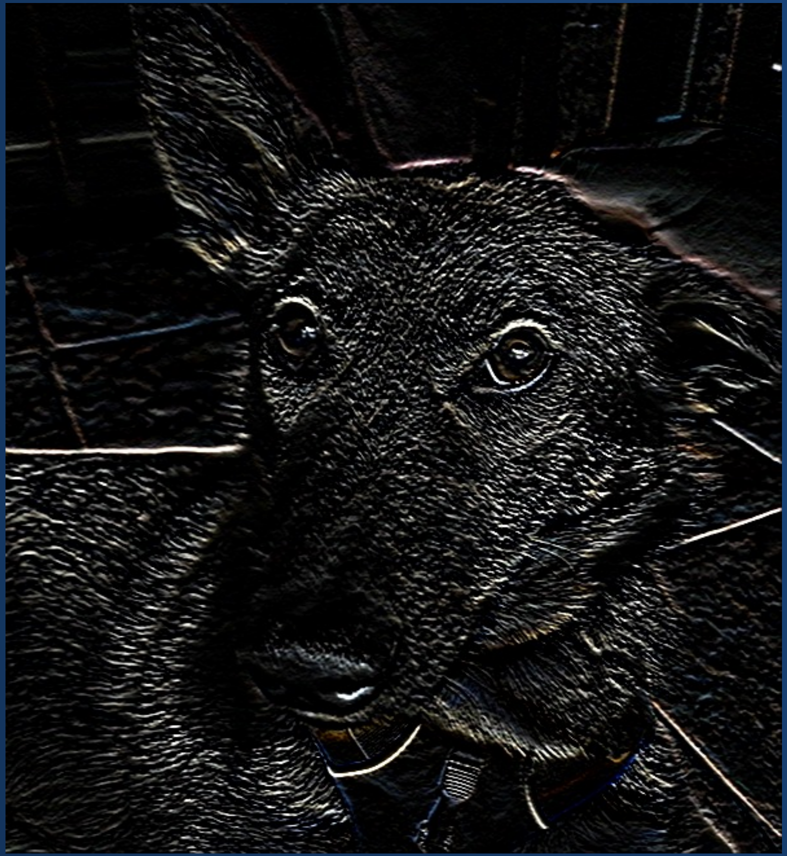
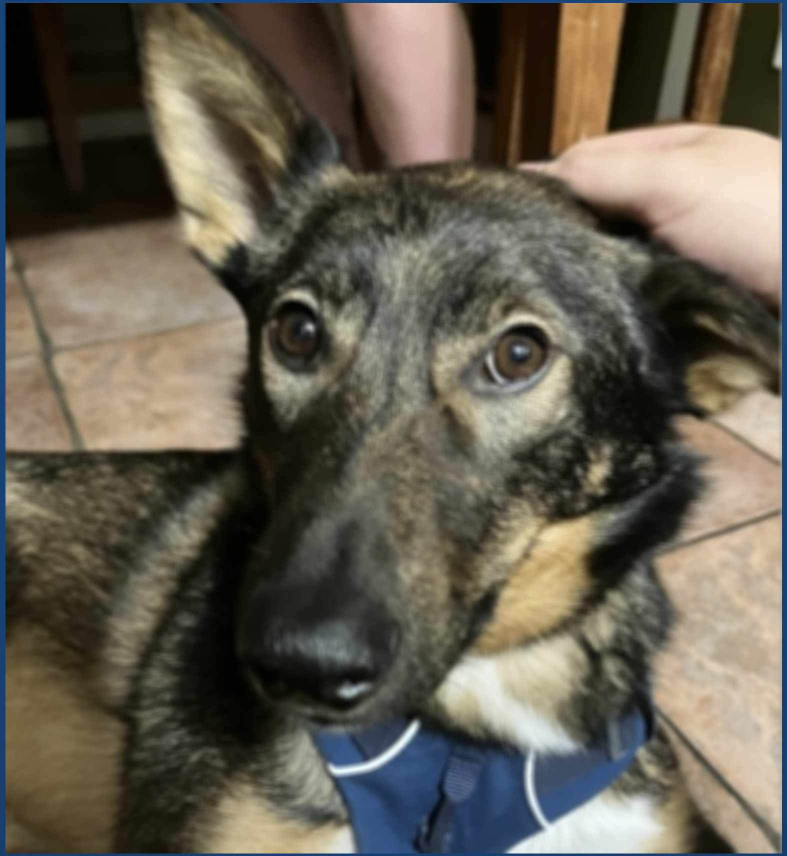
Interactivity

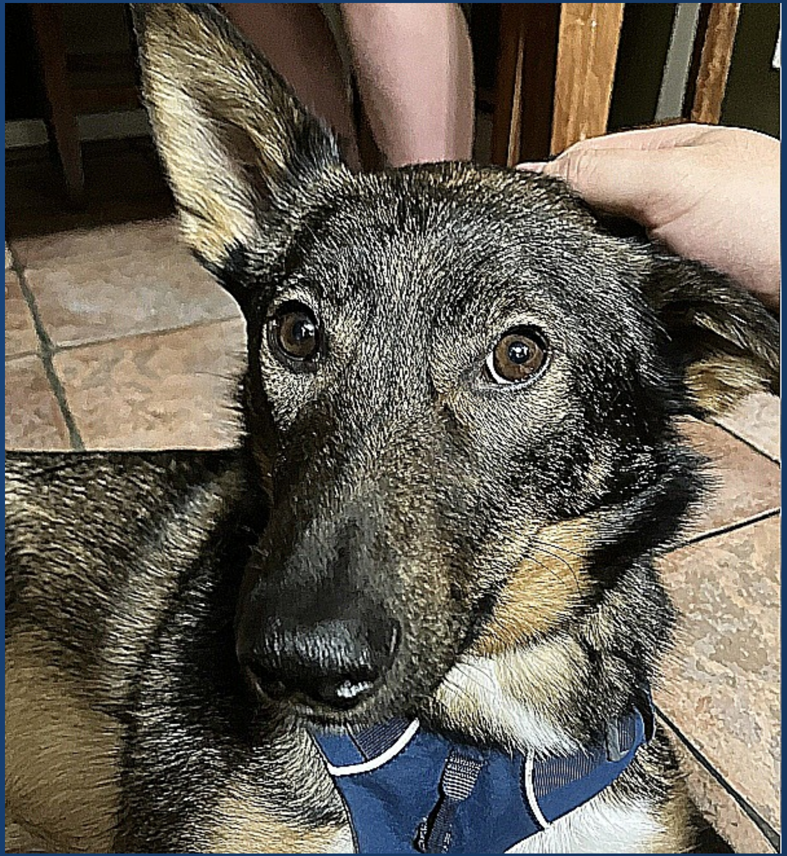
Last time ~~on Dragonball Z~~...

Image kernels can be used to process local information on images.

Applied via a process called *convolution*.

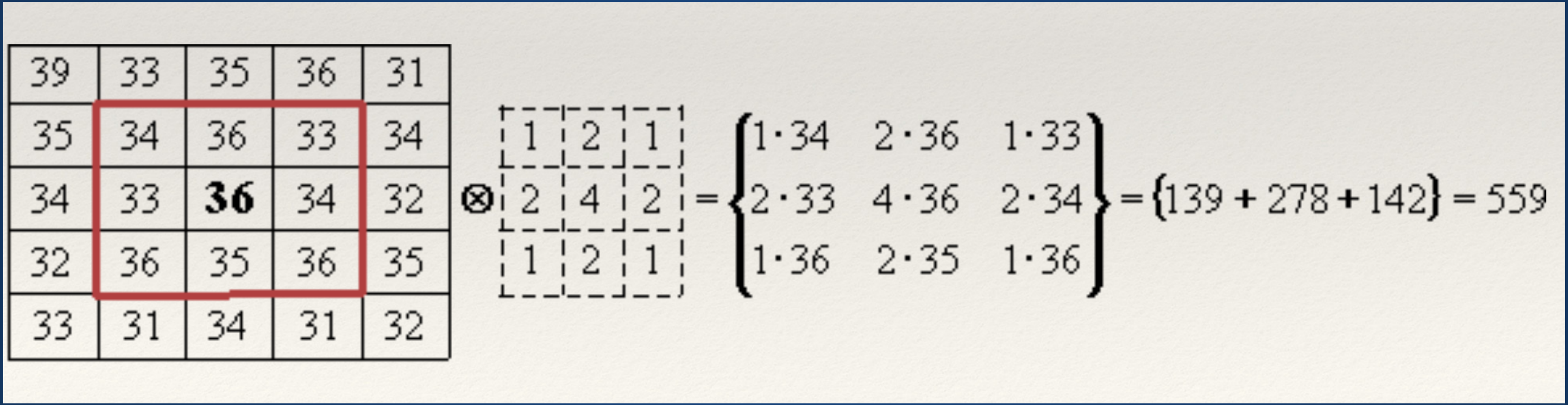






Convolution

Individual outputs are computed by a stack-multiply-add operation.



39	33	35	36	31
35	34	36	33	34
34	33	**36**	34	32
32	36	35	36	35
33	31	34	31	32

$$\otimes \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{Bmatrix} 1\cdot 34 & 2\cdot 36 & 1\cdot 33 \\ 2\cdot 33 & 4\cdot 36 & 2\cdot 34 \\ 1\cdot 36 & 2\cdot 35 & 1\cdot 36 \end{Bmatrix} = \{139 + 278 + 142\} = 559$$

Questions

Are projects always going to be based on material before the end of the week?

Do kernels have to be 3x3?

Nope.

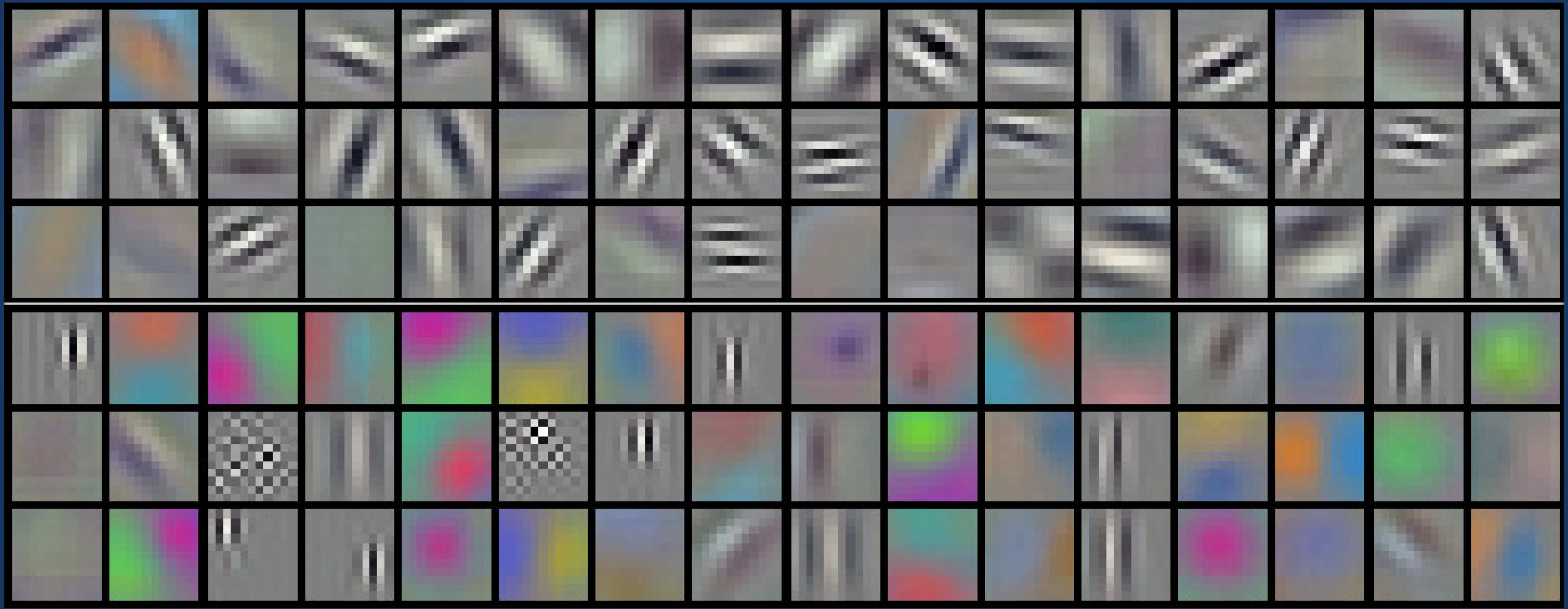
Do kernels have to be square?

Nope.

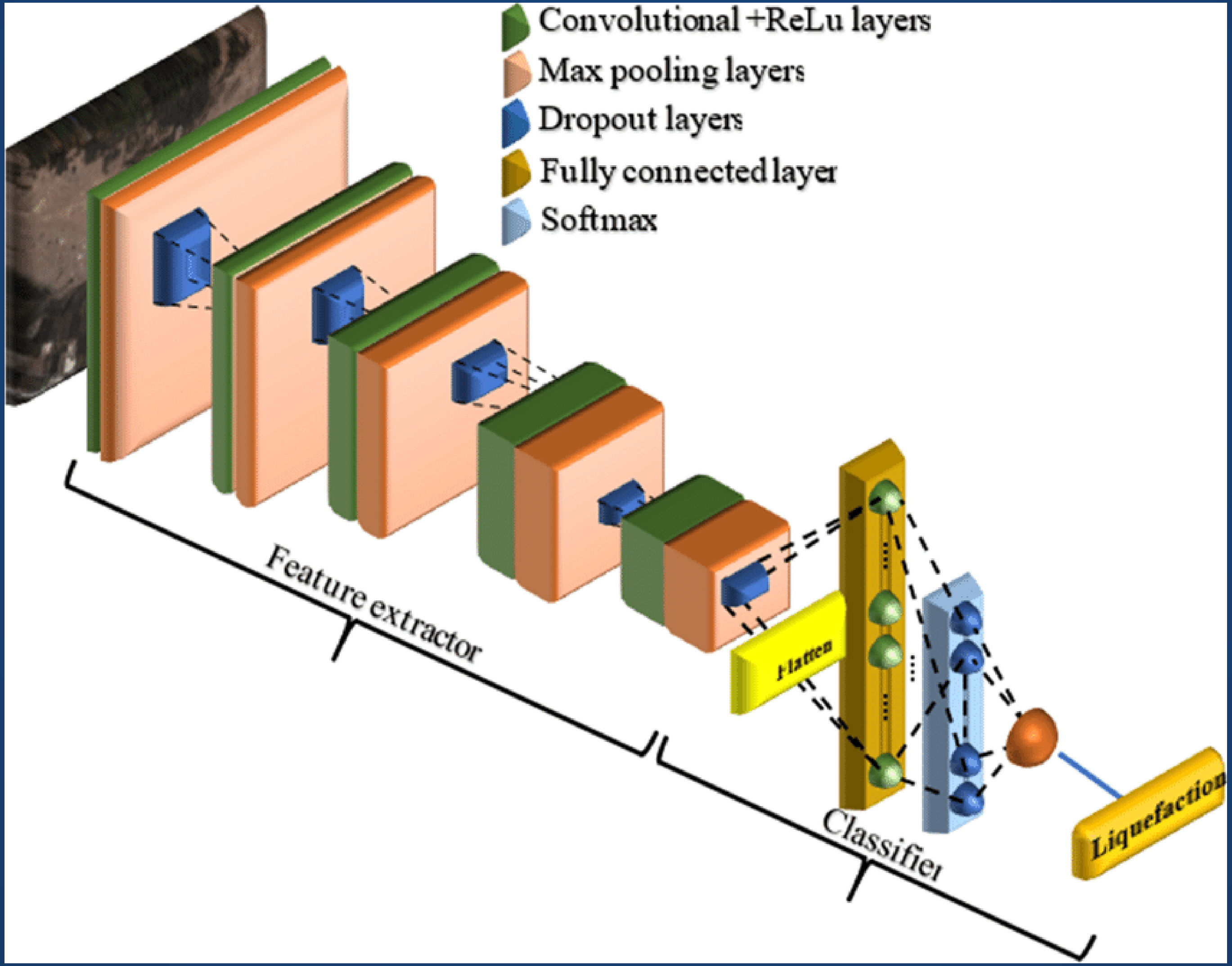
What are common kernel functions?

For human-designed kernel functions, we've actually covered most of them!

But most convolutional kernels aren't human-made anymore!



Almost all modern image/video processing is done with convolution as an operation.



Can we scale up images by undoing a convolution? By modifying pixels?

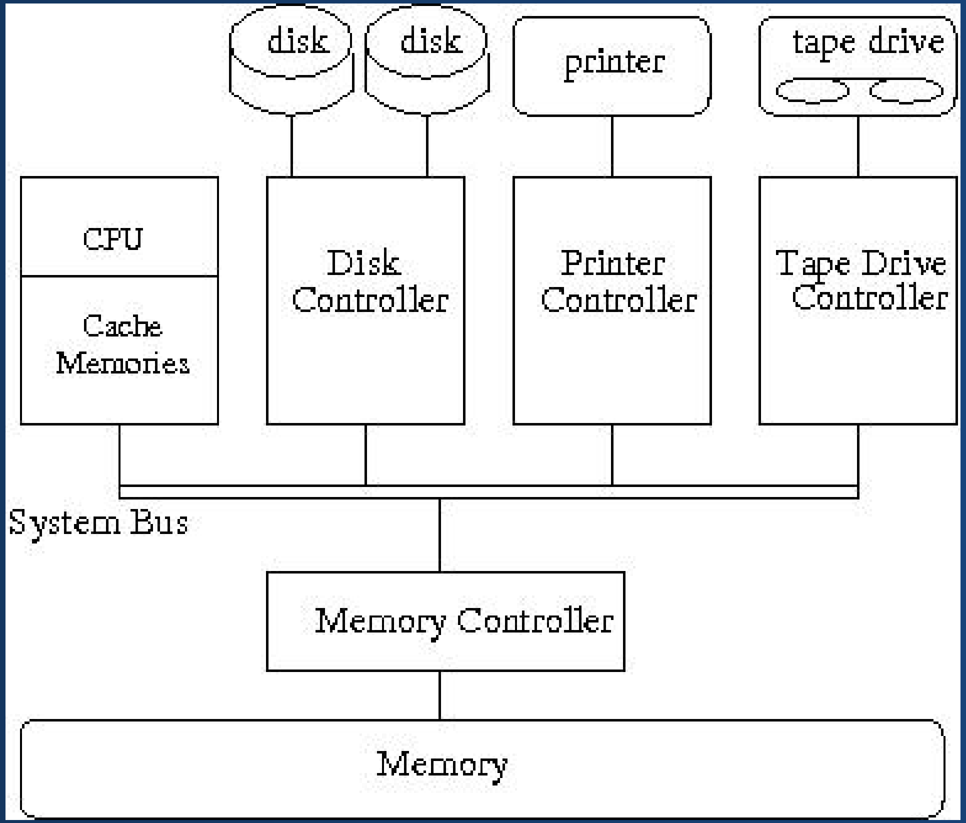
Input Devices

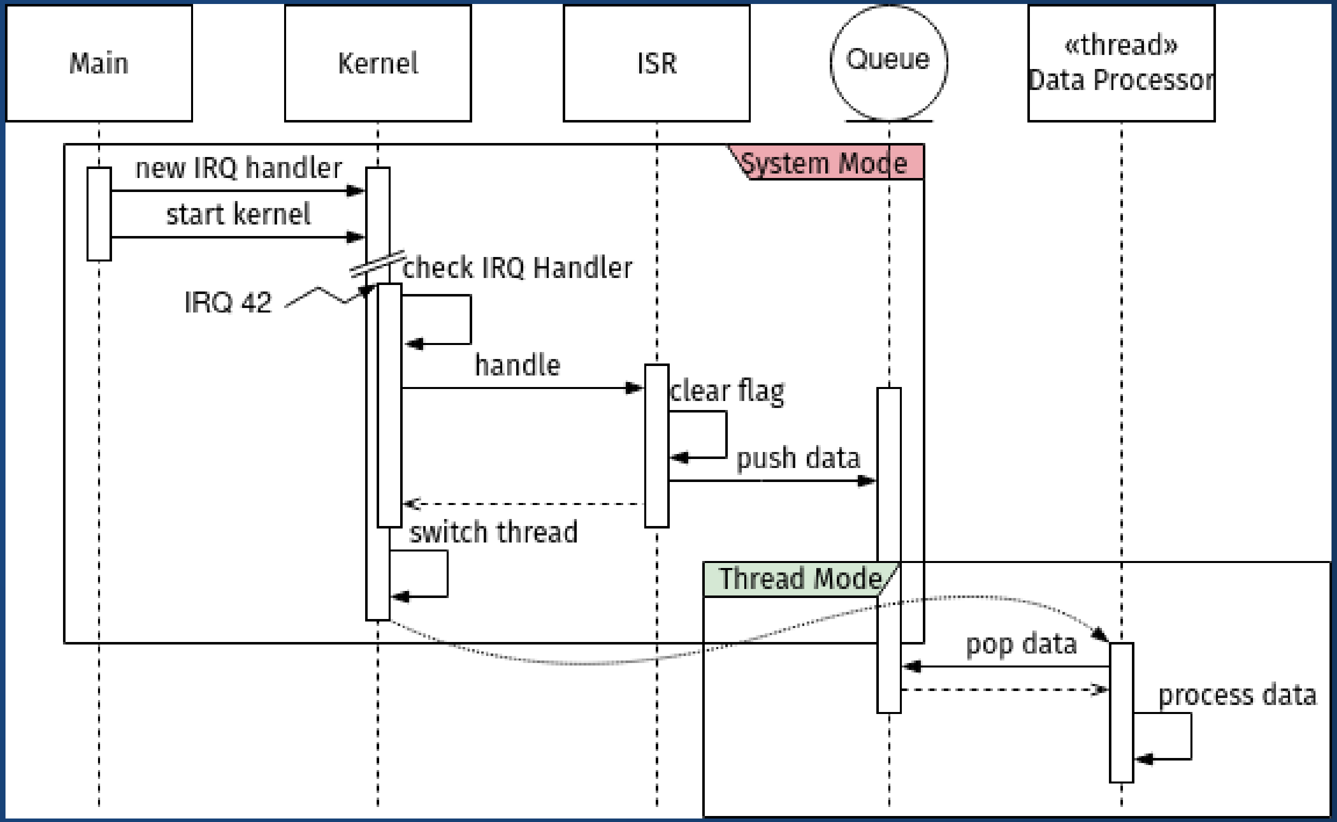
Devices which allow us to interact with computers.

Examples:

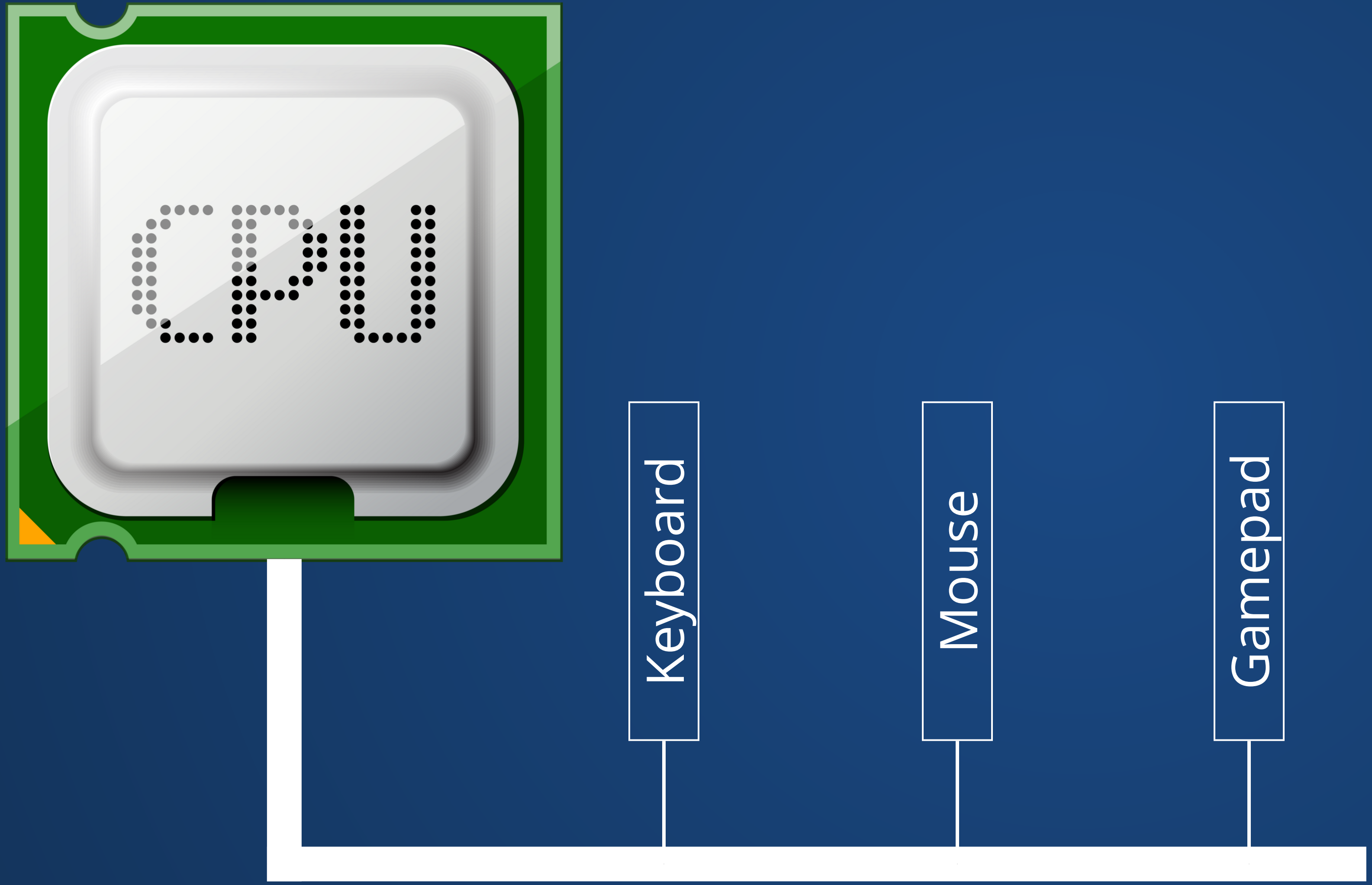
- Mouse
- Keyboard
- Scanner
- Microphone
- Webcam
- Touchscreen
- Controller

The full picture is somewhat complicated...





Simplified Picture



- CPU communicates with external devices on a *bus* (fancy name for a single big wire that everyone shares)
- Devices receive commands from CPU
- Devices can issue *interrupts* to let the CPU know about events
- Software converts *interrupt* to *event*

Events

Typically, when an event is triggered, the program will call a function. In most general-purpose languages, it is up to us to specify which functions are called on which events.

```c
void handle_sigint(int signum) {
    printf("Received SIGINT signal. Exiting...\n");
    exit(0);
}

int main() {
    struct sigaction sa;
    sa.sa_handler = handle_sigint;

    sigaction(SIGINT, &sa, NULL) == -1;
    // Rest of the program
}
```

Fortunately, Processing handles most of this boilerplate for us!

Events in Processing

Events

When Processing receives an event, it sets several special variables and tries to call an event handler.

If no event handler exists, then the handler is not called.

Examples of event handlers:

- `mousePressed()`
- `mouseReleased()`
- `mouseMoved()`
- `mouseDragged()`
- `keyPressed()`
- `keyReleased()`

Mouse Variables

- `mousePressed`: stores whether mouse button is pressed or not
- `mouseButton`: stores the most recently pressed button, one of `LEFT`, `CENTER`, or `RIGHT`
- `mouseX,mouseY`: The coordinates of the mouse cursor in the current window
- `pmouseX`, `pmouseY`: The coordinate of the mouse cursor in the window on the previous frame (useful for determining the direction of mouse motion, e.g. for dragging).

```
void setup(){
  size(800,800);
}

void mousePressed(){
  if (mouseButton == LEFT){
    textSize(128);
    text("LEFT", mouseX, mouseY);
  } else {
    textSize(168);
    text("RIGHT",mouseX, mouseY);
  }

}

void draw(){}
```

Keyboard Variables

- `keyPressed`: stores whether a key has is pressed or not
- key: stores the most recently pressed key. Can take the value of an ASCII character or one of `BACKSPACE`, `TAB`, `ENTER`, `RETURN`, `ESC`, `DELETE`
- `keyCode:` stores keypresses for non-ASCII characters. Useful values include `ALT`, `CONTROL`, `SHIFT`, `UP`, `DOWN`, `LEFT`, `RIGHT`

```
int x = 0;
int y = 0;
void setup(){
  size(800,800);
}

void keyPressed(){
  if(keyCode == DOWN){ y += 5; }
  else if (keyCode == UP){ y -= 5; }
  else if (keyCode == LEFT){ x -= 5; }
  // Why not else?
  else if (keyCode == RIGHT){ x += 5; }
}

void draw(){
  fill(0);
  background(200);  // Reset the screen
  ellipse(x, y, 30, 30);
}
```

Draw Loop

A kind of system-generated event which is triggered every 16.66 ms (60 times per second).

When event triggers, calls the draw() function.

```
public static void main(String[] args){
  // Create a timer which fires every 16ms
  Trigger draw_tr = create_timer(16);
  // Tell the program that when the event
  // occurs, it should respond by calling draw()
  register_event_handler(draw_tr, draw());
  setup();
  wait_for_exit();
}
```

But we as the programmer can control when this happens!

Draw Loop Modification

- `noLoop()` stops the `draw()` command
- `loop()` resumes the `draw()` command
- `redraw()` executes the `draw()` command exactly once

```
int x = 0;
int y = 0;
void setup(){
  size(200,200);
}

void keyPressed(){
  if(keyCode == DOWN){ y += 5; }
  else if (keyCode == UP){ y -= 5; }
  else if (keyCode == LEFT){ x -= 5; }
  // Why not else?
  else if (keyCode == RIGHT){ x += 5; }
}

void draw(){
  fill(0);
  background(200);
  // Prime-factorize 2000 random numbers
  expensive_operation();
  ellipse(x, y, 30, 30);
}
```

CPU Temp: 67 C

```
int x = 0;
int y = 0;
void setup(){
  size(200,200);
  noLoop();
  redraw();
}

void keyPressed(){
  if(keyCode == DOWN){ y += 5; }
  else if (keyCode == UP){ y -= 5; }
  else if (keyCode == LEFT){ x -= 5; }
  // Why not else?
  else if (keyCode == RIGHT){ x += 5; }
  redraw();
}

void draw(){
  fill(0);
  background(200);  // Reset the screen
  expensive_operation();
  ellipse(x, y, 30, 30);
}
```

CPU Temp: 39 C

Hands-On: Interactivity

1. Use the variables `mousePressed` and `mouseButton` in the draw() loop to control the background color of the sketch.
2. Comment out the code in `draw()` and reimplement this in the `mousePressed()` function.
3. Use `mouseX` and `mouseY` in the `mouseMoved()` function to draw a circle that follows the mouse.
4. Display an object to the screen when the 'f' key is pressed using the `keyPressed()` function.
5. Remove the object once the key is released using `keyReleased()` but keep the background color changes and color.

This is the only hands-on for today. Once you're done, you may submit your index card and leave, or stick around to work on the project until the end of class.

Project Time

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.