CS3483 p5js Notes

Introduction to p5.js

- **Purpose**: JavaScript library for creative coding, treating the browser as a sketchpad.
- **Core Features**:
  - Drawing, text, image, video, and sound via HTML5.
- **Core Workflow**:
  - `setup()` → `draw()` loop + event handlers.

Program Structure

1. **Global Variables**: Retain values outside function scope.
2. **Setup Function**: Runs once at program start.

```
function setup() {
  createCanvas(400, 400);
}
```

3. **Draw Function**: Continuously loops until stopped.

```
function draw() {
  background(220);
  // things to draw
}
```

2D Drawing Basics

- **Coordinate System**: Origin `(0,0)` at top-left; `(x,y)` pixel addressing.
- **Key Functions**:
  - `createCanvas(w, h)`: Sets canvas dimensions.
  - `background(r, g, b)`: Clears canvas with color.
  - `stroke(r, g, b)`: Sets outline color.

- `fill(r, g, b)`: Sets fill color.
- `noFill()` / `noStroke()`: Disables fill/outline.

- **Shape Drawing**:

```
point(x, y);              // Single pixel
line(x1, y1, x2, y2);     // Line between points
rect(x, y, w, h);         // Rectangle (top-left anchor)
ellipse(x, y, w, h);      // Ellipse (center anchor)
triangle(x1, y1, x2, y2, x3, y3);  // Triangle
quad(x1, y1, x2, y2, x3, y3, x4, y4); // Quadrilateral
```

Interaction Handling

- **Mouse Events**:

```
function mousePressed() { /* code */ }  // Trigger on click
function mouseReleased() { /* code */ } // Trigger on release
let mouseIsPressed;                     // Boolean state
```

- **Keyboard Events**:

```
function keyPressed() { /* code */ }    // Trigger on key press
let keyIsPressed;                       // Boolean state
```

- **Touch Events**:

```
function touchStarted() { /* code */ }  // Trigger on touch
```

Image Operations

- **Loading/Displaying Images**:

```
let img;
function setup() {
  img = loadImage('path/image.jpg');  // Load image
}
function draw() {
```

```
  image(img, x, y, width, height);          // Display image
}
```

- **Pixel Manipulation**:

```
let c = get(x, y);     // Get pixel color
set(x, y, c);          // Set pixel color
updatePixels();        // Apply changes
```

3D Drawing

- **Setup**: Use `WEBGL` mode in `createCanvas`.

```
createCanvas(400, 400, WEBGL);
```

- **3D Primitives & Transformations**:

```
box(size);             // Draw a cube
sphere(radius);        // Draw a sphere
translate(x, y, z);    // Move object
rotateX(angle);        // Rotate around X-axis
lights();              // Enable shading
```

Face Processing

Video Capture in p5.js

- **Initialization**:

```
let video;
function setup() {
  createCanvas(625, 437);
  video = createCapture(VIDEO);
  video.size(width, height);
  video.hide();  // Hides the default video element
}
```

- **Display Frames**:

```
function draw() {
  image(video, 0, 0);  // Renders video frames on canvas
}
```

Face Detection with ml5.js

- **Library Setup**: Include ml5.js in HTML:

```
<script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
```

- **Model Initialization**:

```
let faceMesh, detections = [];
let options = { maxFaces: 1, refineLandmarks: false, flipped: false };

function preload() {
  faceMesh = ml5.faceMesh(options);
}
```

- **Detection Start**:

```
function setup() {
  faceMesh.detectStart(video, gotResults);
}
function gotResults(results) { detections = results; }
```

Drawing Bounding Boxes

- **Function**: Highlights detected faces with rectangles.

```
function drawBox(detections) {
  noFill();
  stroke(0, 255, 0);  // Green outline
  strokeWeight(2);
  for (let i = 0; i < detections.length; i++) {
    let box = detections[i].box;
    rect(box.xMin, box.yMin, box.width, box.height);
```

```
  }
}
```

Face Keypoints & Landmarks

- **Keypoints (e.g., eyes, lips)**:

```
function drawKeypoints(detections) {
  noStroke();
  fill(0, 255, 0);  // Green dots
  for (let i = 0; i < detections.length; i++) {
    let detection = detections[i];
    for (let j = 0; j < detection.keypoints.length; j++) {
      let keypoint = detection.keypoints[j];
      circle(keypoint.x, keypoint.y, 5);  // Draw keypoints
    }
  }
}
```

- **Landmarks (e.g., face oval, eyebrows)**:

```
function drawLandmarks(detections) {
  for (let i = 0; i < detections.length; i++) {
    let detection = detections[i];
    let faceOval = detection.faceOval.keypoints;
    let lips = detection.lips.keypoints;
    // ... (other features)
    drawPart(faceOval, true);  // Closed shape
    drawPart(lips, true);
  }
}
function drawPart(features, closed) {
  beginShape();
  for (let i = 0; i < features.length; i++) {
    vertex(features[i].x, features[i].y);
  }
  if (closed) endShape(CLOSE); else endShape();
}
```

Image Processing

- **Tinting**: Applies color overlay.

```
tint(255, 0, 0);  // Red tint
image(img, 0, 0);
```

- **Filters**:

```
filter(THRESHOLD);  // Binary threshold
filter(BLUR, 10);   // Blur with radius 10
filter(POSTERIZE, 3);  // Reduces color levels to 3
```

Interaction on p5.js

Mouse Interaction

1. **Mouse Position Tracking**
   - **Variables**:
     - `mouseX`, `mouseY`: Current cursor coordinates.
     - `pmouseX`, `pmouseY`: Previous frame's cursor coordinates.
   - **Transformations**: Use mouse coordinates for dynamic transformations.

```
translate(mouseX, mouseY);  // Move object with cursor
```

   - **Rotation Mapping**: Convert `mouseX` to angles (0 to `TWO_PI`).

```
let angle = map(mouseX, 0, width, 0, TWO_PI);
rotate(angle);
```

2. **Mouse Buttons & Events**
   - **Button Detection**:

```
if (mouseButton === LEFT) { /* code */ }  // Check left/right/CENTER
```

   - **Event Handlers**:

```
function mouseMoved() { /* code */ }   // Triggered on cursor move
function mouseDragged() { /* code */ } // Triggered on drag
```

Constraining Values

- `constrain(value, min, max)`:
  - Clips `value` to stay within `[min, max]`.

```
let mx = constrain(mouseX, 35, 65);  // Limit mouseX between 35–65
```

Distance Calculation

- `dist(x1, y1, x2, y2)`:
  - Computes Euclidean distance between two points.

```
let d = dist(width/2, height/2, mouseX, mouseY);  // Distance from cen
```

Keyboard Interaction

- **Key Detection**:

```
if (keyIsPressed && key === 'a') { /* code */ }  // Check 'a' key press
```