Die Idee des Projektes ist es, ein Pong Spiel zu entwickeln, bei dem man die Pong-Schläger mit Hilfe von Color Tracking steuert. Genauer heißt dass, man wählt eine Farbe aus, bei der die Durchschnittsposition in dem aktuellen Webcam Bild berechnet werden soll. Diese Position wird auf dem Pongschläger übertragen.

Das Color Tracking wird in der Klasse „colorTracking.pde“ umgesetzt. Die Methode „Initialize“ macht das Webcam Bild zugänglich. Generell wird das Webcam Bild in zwei Hälften aufgeteilt, um in jeder Hälfte eine Farbe unabhängig verfolgen zu können. Die Methode „updateVideo“ wird für jeden Frame einmal aufgerufen. Sie macht die Pixeldaten des aktuellen Bilds zugänglich und ruft die Methoden „calculateAvrgRight“ und „calcualteAvrgLeft“ auf. Diese Methoden machen das selbe einmal für die linke und einmal für die rechte Bildhälfte. Sie iterieren über jeden Pixel der entsprechenden Hälfte und überprüfen ob der Farbwert innerhalb der erlaubten Differenz zu dem vorgegebenen Farbwert liegt. Diese erlaubte Differenz nennen wir „acceptedColorRange“. Es wird dazu jeweils der Rot-, Grün- und Blau Wert der Farbe des Pixels von den Rot-, Grün- und Blauwerten der zu verfolgenden Farbe abgezogen und quadriert um keine negativen Werte zu erhalten. Ist das Ergebnis dieser Rechnung kleiner als „acceptedColorRange“ ins Quadrat, wissen wir, dass es sich hier um ein Pixel handelt, den wir als gleiche Farbe zur vorgegeben Farbe akzeptieren. Wir addieren dann den X-Wert des Pixels zu einer Summe aller X-Werte von akzeptieren Pixeln und erhöhen den Zähler der erkannten Pixel um eins. Das selbe geschieht für die Y-Wert. Hat man dies für alle Pixel in einer Hälfte gemacht, kann man die Summe der X-Werte durch die Anzahl der akzeptierten Pixel teilen und erhält die durchschnittliche X-Position der Farbe im Bild. Diese Position machen wir mit den Variable „avrgXLeft“ und „avrgXRight“ für alle Klassen zugänglich.

Die aktuelle Farbe, also die Farbe, die wir „tracken“ wollen, kann man auswählen, in dem man mit der Maus auf einen Pixel klickt. Das kann man für beide Bildschirmhälften machen. Der Farbwert, der damit ausgewählt wurde, wird in „colorToTrackLeft“ bzw. „colorToTrackRight“ gespeichert.

Mit „+“ und „-“ Tasten kann man die „AcceptedColorRange“ in 10-er Schritten ändern.

Dieser simple Color Tracking Algorithmus funktioniert für unser Vorhaben recht gut, wenn man auf zwei Bedingungen achtet. Der Gegenstand, den man verfolgen möchte, sollte eine Farbe haben, die nicht im Hintergrund auftritt. Ist im Bild noch ein anderer Gegenstand mit derselben Farbe, so wir als Durchschnittsposition die mittlere Position der zwei Gegenstände berechnet.

Die zweite Schwäche tritt auf, wenn ungleichmäßige Lichtverhältnisse herrschen. Wird die Farbe des verfolgenden Gegenstands in einem dunkleren Bildbereich ausgewählt, so ändert sich der Farbwert des Gegenstands im Bild, wenn man ihn in einen helleren Bereich bewegt. Jedoch ist die eingestellte Farbe immer noch die Farbe des Gegenstands im dunklen Bereich.

Die PONG-Implementierung findet in der Main-Klasse statt. Zunächst werden in der „setup“-Methode die Eigenschaften vorbereitet. (Größe des Feldes/Balles, oder der Schläger).

Das eigentliche Spiel läuft dann über die „draw“-Methode. Diese setzt sich aus mehreren kleineren Methoden zusammen.

Das Bild wird durch die Methode „clear“ immer wieder geleert und mit „updateVideo“ das Video aktualisiert. „drawGame“ kümmert sich um das Befüllen des Spielfelds mit den benötigten Formen (Schläger, Ball, Spielfeld, Spielstand). Sobald man für jede Spielhälfte eine Farbe (durch Mausklick) zum Steuern ausgesucht hat, kann das Spiel mit der Taste „p“ gestartet werden.  
Der Ball bewegt sich dann mit Hilfe der Variablen „ballSpeedX“ & „ballSpeedY“.

Die Methode „calcPlayerMovement“ ist für die Bewegung des Schlägers des jeweiligen Spielers zuständig. Wenn die oben errechnet und beschriebene Variable „avrgYRight“, bzw. „avrgYLeft“ kleiner/größer ist als die Y-Position des Schlägers bewegt sich der Schläger mit Hilfe der „playerSpeed“ Variable nach oben/unten.

Des Weiteren überprüft „checkCollision“ ob der Ball auf einen Spieler, eine Wand oder über den Spielfeldrand hinausgeht. Trifft er auf einen Spieler, wird die X-Richtung des Balles invertiert und die Geschwindigkeitsvariablen erhöht. Bei einer Kollision mit der Wand wird die Y-Richtung invertiert und würde der Ball das Spielfeld verlassen bekommt der Gegenüberliegende Spieler einen Punkt und die jeweilige „scoreL“/“scoreR“-Variable wird erhört.

Zuletzt übernimmt noch die Methode „restart“ die Repositionierung des Balles nach einem Punkt, bzw. das Zurücksetzen der Ballgeschwindigkeit.