

# Rock Paper Scissors

---

## Rock Paper Scissors

[Routine Experiment Effect](#)

[Code Explanation](#)

[Code detailed structure](#)

[Key Code](#)

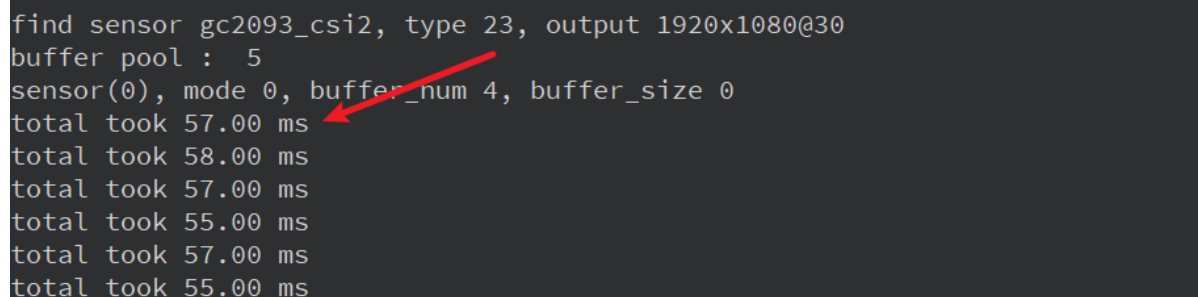
## Routine Experiment Effect

---

In this section, we will learn about gesture recognition based on K230 and make a game of rock-paper-scissors.

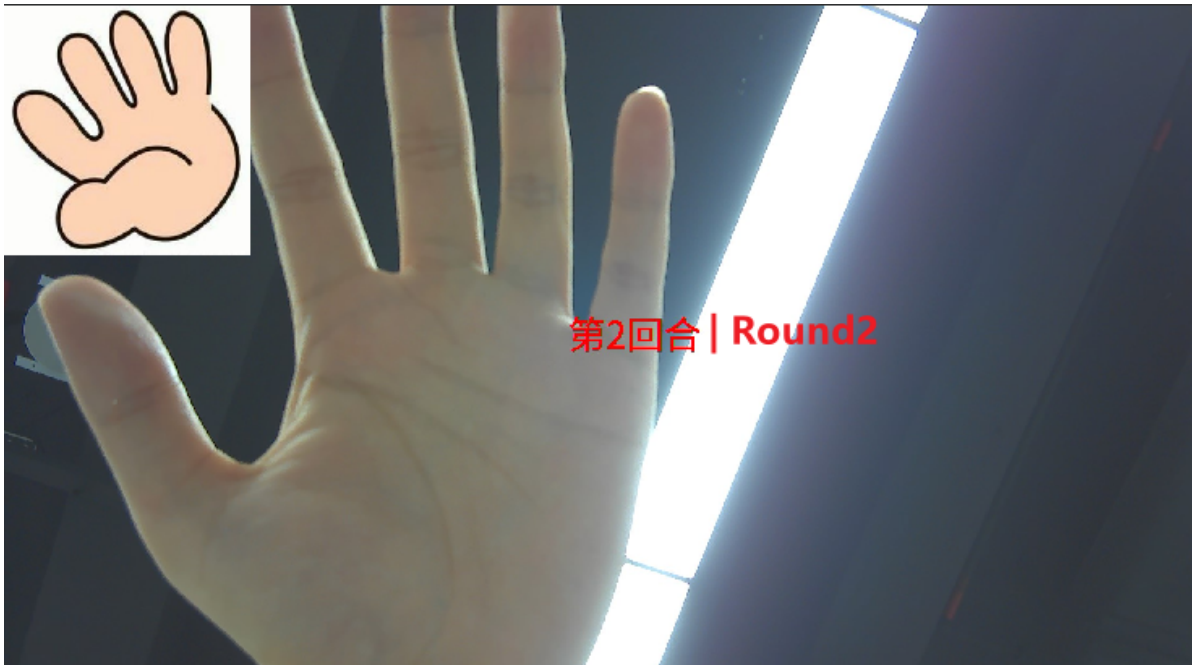
The game execution process is as follows:

1. Copy the sample code into CanMV IDE and click the Run button in the lower left corner.  
Please make sure that there is no "hand" in the camera at this time.
2. Wait for "Total took xxxxx" to appear in the lower left corner to indicate the game has started.



```
find sensor gc2093_csi2, type 23, output 1920x1080@30
buffer pool : 5
sensor(0), mode 0, buffer_num 4, buffer_size 0
total took 57.00 ms
total took 58.00 ms
total took 57.00 ms
total took 55.00 ms
total took 57.00 ms
total took 55.00 ms
```

1. After the game starts, we make the shape of "rock", "scissors" or "paper" with our hands, and it will appear in the camera. At this time, the screen will show the punches of the program and the result of winning or losing. After getting the result, it means that this round is over, please remove your hands from the camera range.



1. The game lasts for three rounds in total. After three rounds, the program will give the final result according to the best-of-three rule. [There is a small bug here. If the program does not give the final result immediately after the third round, you can let your hand enter the camera again and the program will give the final result.]



## Code Explanation

---

This is a rock-paper-scissors game program based on gesture recognition. It mainly includes the following parts:

1. Hand detection class (HandDetApp) and gesture key point classification class (HandKPClassApp)
  - These two classes implement the functions of palm detection and gesture key point recognition respectively, providing underlying support for the rock-paper-scissors game.

- They encapsulate operations such as image preprocessing, model inference, and post-processing, providing a unified interface for upper-level logic.

## 2. FingerGuess

- This class is responsible for the execution logic of the entire rock-paper-scissors game.
- It will first call the palm detection and key point classification model to obtain the user's current gesture.
- Then, according to different game modes (must win, must lose, multiple rounds), the corresponding computer punches are generated and the winner is determined.
- Finally, the game process and results are drawn to the screen for display.

## 3. Game execution function (exce\_demo)

- This function is responsible for initializing the game-related environment and resources and entering the main loop of the game.
- In the main loop, it continuously obtains the current frame image, calls the run method of the FingerGuess class to process the image, and finally displays the game screen.

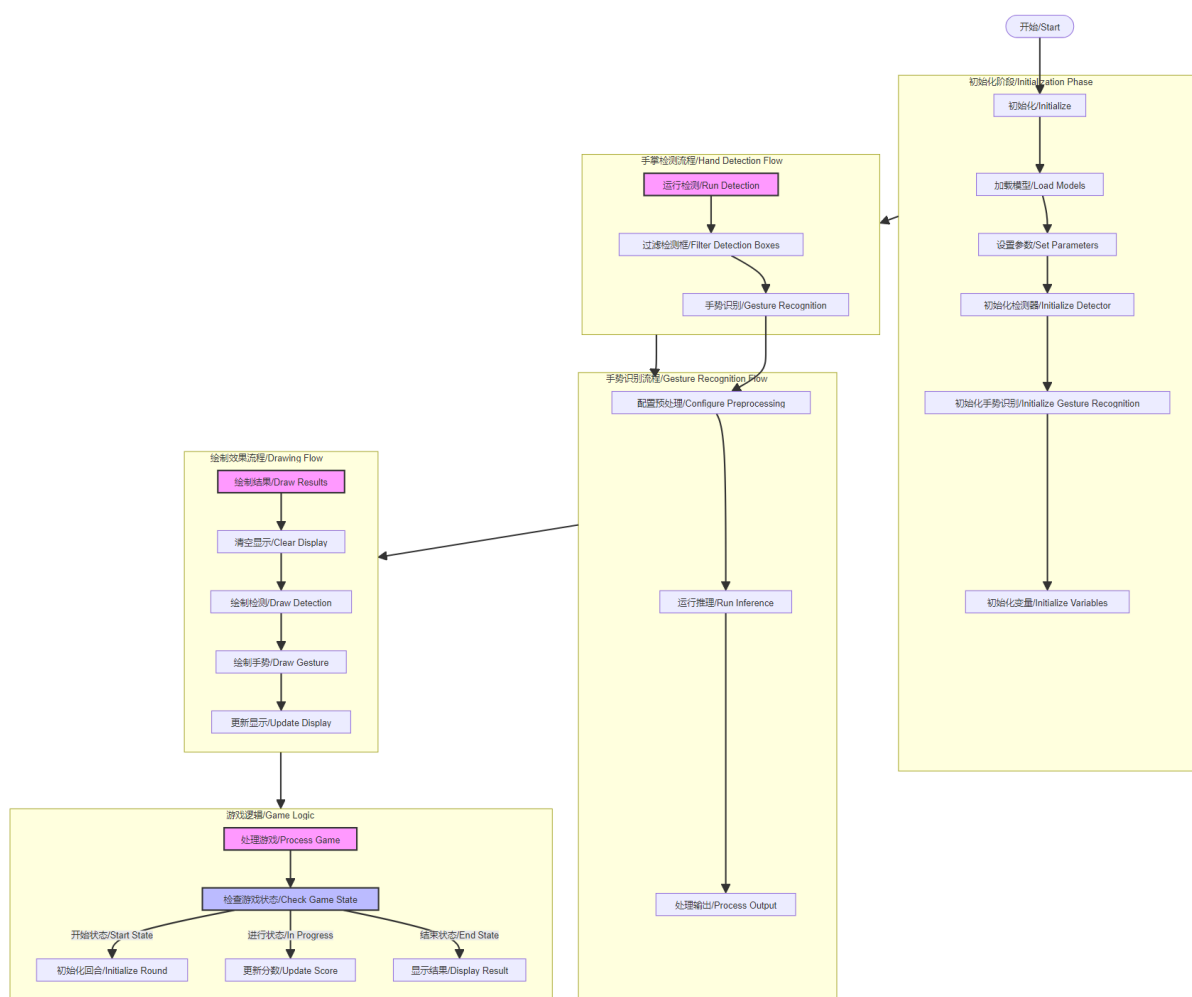
The execution flow of the entire game is:

### 1. Initialize the game environment and resources

### 2. Enter the main game loop

### 3. In each frame:

- Get the current image
- Detect palms and recognize gestures
- Generate computer punches based on game logic to determine the winner
- Draw the game screen and display it



## Code detailed structure

1. Initialization Phase:
  - Load detection and recognition models
  - Set various parameters/Set parameters
  - Initialize detector and gesture recognition module
  - Initialize game-related variables
2. Hand Detection Flow:
  - Run hand detection
  - Filter detection boxes
  - Pass to gesture recognition module
3. Gesture Recognition Flow:
  - Configure preprocessing
  - Run model inference
  - Process output
4. Drawing Flow:
  - Clear display
  - Draw detection
  - Draw gesture recognition results
  - Update display
5. Game Logic:
  - Process game logic
  - Check game state
  - Initialize new round
  - Update player and AI scores
  - Display final result

## Key Code

```
# 猜拳游戏任务类 / Rock-paper-scissors game task class
class FingerGuess:
    def __init__(self, hand_det_kmodel, hand_kp_kmodel, det_input_size,
kp_input_size, labels, anchors, confidence_threshold=0.25, nms_threshold=0.3,
nms_option=False, strides=[8, 16, 32], guess_mode=3, rgb888p_size=[1280, 720],
display_size=[1920, 1080], debug_mode=0):
    # 手掌检测模型路径 / Hand detection model path
    self.hand_det_kmodel = hand_det_kmodel
    # 手掌关键点模型路径 / Hand keypoint model path
    self.hand_kp_kmodel = hand_kp_kmodel
    # 手掌检测模型输入分辨率 / Hand detection model input resolution
    self.det_input_size = det_input_size
    # 手掌关键点模型输入分辨率 / Hand keypoint model input resolution
    self.kp_input_size = kp_input_size
    self.labels = labels
    # anchors / Anchor boxes
    self.anchors = anchors
    # 置信度阈值 / Confidence threshold
    self.confidence_threshold = confidence_threshold
    # nms阈值 / Non-Maximum suppression threshold
    self.nms_threshold = nms_threshold
    # nms选项 / NMS option
    self.nms_option = nms_option
```

```

# 特征图针对输入的下采样倍数 / Feature map downsampling multipliers relative
to input
self.strides = strides
# sensor给到AI的图像分辨率, 宽16字节对齐 / Image resolution from sensor to
AI, width aligned to 16 bytes
self.rgb888p_size = [ALIGN_UP(rgb888p_size[0], 16), rgb888p_size[1]]
# 视频输出VO分辨率, 宽16字节对齐 / Video output VO resolution, width aligned
to 16 bytes
self.display_size = [ALIGN_UP(display_size[0], 16), display_size[1]]
# debug_mode模式 / Debug mode
self.debug_mode = debug_mode
self.guess_mode = guess_mode
# 石头剪刀布的贴图array / Rock-paper-scissors image arrays
self.five_image = self.read_file("/sdcard/utils/five.bin")
self.fist_image = self.read_file("/sdcard/utils/fist.bin")
self.shear_image = self.read_file("/sdcard/utils/shear.bin")
self.counts_guess = -1
# 猜拳次数 计数 / Count of rock-paper-scissors rounds
self.player_win = 0
# 玩家 赢次计数 / Player win count
self.k230_win = 0
# k230 赢次计数 / K230 win count
self.sleep_end = False
# 是否 停顿 / Whether to pause
self.set_stop_id = True
# 是否 暂停猜拳 / Whether to suspend the game
self.LIBRARY = ["fist", "yeah", "five"]
# 猜拳 石头剪刀布 三种方案的dict / Dictionary of three rock-paper-
scissors options
self.hand_det = HandDetApp(self.hand_det_kmodel, self.labels,
model_input_size=self.det_input_size, anchors=self.anchors,
confidence_threshold=self.confidence_threshold,
nms_threshold=self.nms_threshold, nms_option=self.nms_option,
strides=self.strides, rgb888p_size=self.rgb888p_size,
display_size=self.display_size, debug_mode=0)
self.hand_kp = HandKPClassApp(self.hand_kp_kmodel,
model_input_size=self.kp_input_size, rgb888p_size=self.rgb888p_size,
display_size=self.display_size)
self.hand_det.config_preprocess()

# run函数 / Run function
def run(self, input_np):
# 先进行手掌检测 / First perform hand detection
det_boxes = self.hand_det.run(input_np)
boxes = []
gesture_res = []
for det_box in det_boxes:
# 对检测的手做手势识别 / Perform gesture recognition on detected hands
x1, y1, x2, y2 = det_box[2], det_box[3], det_box[4], det_box[5]
w, h = int(x2 - x1), int(y2 - y1)
if (h < (0.1 * self.rgb888p_size[1])):
continue
if (w < (0.25 * self.rgb888p_size[0]) and ((x1 < (0.03 *
self.rgb888p_size[0])) or (x2 > (0.97 * self.rgb888p_size[0])))):
continue
if (w < (0.15 * self.rgb888p_size[0]) and ((x1 < (0.01 *
self.rgb888p_size[0])) or (x2 > (0.99 * self.rgb888p_size[0])))):
continue

```

```

        self.hand_kp.config_preprocess(det_box)
        results_show, gesture = self.hand_kp.run(input_np)
        boxes.append(det_box)
        gesture_res.append(gesture)
    return boxes, gesture_res

# 绘制效果 / Draw results
def draw_result(self, pl, dets, gesture_res):
    pl.osd_img.clear()
    # 手掌的手势分类得到用户的出拳，根据不同模式给出开发板的出拳，并将对应的贴图放到屏幕上显示

    # Get user's gesture from hand gesture classification, determine the
    board's gesture based on different modes, and display corresponding images on
    screen
    if (len(dets) >= 2):
        pl.osd_img.draw_string_advanced(self.display_size[0] // 2 - 50,
        self.display_size[1] // 2 - 50, 30, "请保证只有一只手入镜 Make sure only one hand on
        the screen", color=(255, 255, 0, 0))
        elif (self.guess_mode == 0):
            draw_img_np = np.zeros((self.display_size[1], self.display_size[0],
            4), dtype=np.uint8)
            draw_img = image.Image(self.display_size[0], self.display_size[1],
            image.ARGB8888, alloc=image.ALLOC_REF, data=draw_img_np)
            if (gesture_res[0] == "fist"):
                draw_img_np[:, :400, :] = self.shear_image
            elif (gesture_res[0] == "five"):
                draw_img_np[:, :400, :] = self.fist_image
            elif (gesture_res[0] == "yeah"):
                draw_img_np[:, :400, :] = self.five_image
            pl.osd_img.copy_from(draw_img)
        elif (self.guess_mode == 1):
            draw_img_np = np.zeros((self.display_size[1], self.display_size[0],
            4), dtype=np.uint8)
            draw_img = image.Image(self.display_size[0], self.display_size[1],
            image.ARGB8888, alloc=image.ALLOC_REF, data=draw_img_np)
            if (gesture_res[0] == "fist"):
                draw_img_np[:, :400, :] = self.five_image
            elif (gesture_res[0] == "five"):
                draw_img_np[:, :400, :] = self.shear_image
            elif (gesture_res[0] == "yeah"):
                draw_img_np[:, :400, :] = self.fist_image
            pl.osd_img.copy_from(draw_img)
        else:
            draw_img_np = np.zeros((self.display_size[1], self.display_size[0],
            4), dtype=np.uint8)
            draw_img = image.Image(self.display_size[0], self.display_size[1],
            image.ARGB8888, alloc=image.ALLOC_REF, data=draw_img_np)
            if (self.sleep_end):
                time.sleep_ms(2000)
                self.sleep_end = False
            if (len(dets) == 0):
                self.set_stop_id = True
                return
            if (self.counts_guess == -1 and gesture_res[0] != "fist" and
            gesture_res[0] != "yeah" and gesture_res[0] != "five"):
                draw_img.draw_string_advanced(self.display_size[0] // 2 - 50,
                self.display_size[1] // 2 - 50, 30, "游戏开始 GAME START", color=(255, 255, 0, 0))
                time.sleep_ms(500)

```

```

        elif (self.counts_guess == self.guess_mode):
            draw_img.clear()
            if (self.k230_win > self.player_win):
                draw_img.draw_string_advanced(self.display_size[0] // 2 -
50, self.display_size[1] // 2 - 50, 30, "你输了 You lose", color=(255, 255, 0,
0))

                elif (self.k230_win < self.player_win):
                    draw_img.draw_string_advanced(self.display_size[0] // 2 -
50, self.display_size[1] // 2 - 50, 30, "你赢了 You win", color=(255, 255, 0, 0))
                else:
                    draw_img.draw_string_advanced(self.display_size[0] // 2 -
50, self.display_size[1] // 2 - 50, 30, "平局 | tie", color=(255, 255, 0, 0))
                    self.counts_guess = -1
                    self.player_win = 0
                    self.k230_win = 0
                    self.sleep_end = True
            else:
                if (self.set_stop_id):
                    if (self.counts_guess == -1 and (gesture_res[0] == "fist" or
gesture_res[0] == "yeah" or gesture_res[0] == "five")):
                        self.counts_guess = 0
                    if (self.counts_guess != -1 and (gesture_res[0] == "fist" or
gesture_res[0] == "yeah" or gesture_res[0] == "five")):
                        k230_guess = randint(1, 10000) % 3
                        if (gesture_res[0] == "fist" and
self.LIBRARY[k230_guess] == "yeah"):
                            self.player_win += 1
                        elif (gesture_res[0] == "fist" and
self.LIBRARY[k230_guess] == "five"):
                            self.k230_win += 1
                        if (gesture_res[0] == "yeah" and
self.LIBRARY[k230_guess] == "fist"):
                            self.k230_win += 1
                        elif (gesture_res[0] == "yeah" and
self.LIBRARY[k230_guess] == "five"):
                            self.player_win += 1
                        if (gesture_res[0] == "five" and
self.LIBRARY[k230_guess] == "fist"):
                            self.player_win += 1
                        elif (gesture_res[0] == "five" and
self.LIBRARY[k230_guess] == "yeah"):
                            self.k230_win += 1
                        if (self.LIBRARY[k230_guess] == "fist"):
                            draw_img_np[:400, :400, :] = self.fist_image
                        elif (self.LIBRARY[k230_guess] == "five"):
                            draw_img_np[:400, :400, :] = self.five_image
                        elif (self.LIBRARY[k230_guess] == "yeah"):
                            draw_img_np[:400, :400, :] = self.shear_image
                        self.counts_guess += 1
                        draw_img.draw_string_advanced(self.display_size[0] // 2
- 50, self.display_size[1] // 2 - 50, 30, "第" + str(self.counts_guess) + "回合 |
Round " + str(self.counts_guess), color=(255, 255, 0, 0))
                        self.set_stop_id = False
                        self.sleep_end = True
                    else:
                        draw_img.draw_string_advanced(self.display_size[0] // 2
- 50, self.display_size[1] // 2 - 50, 30, "第" + str(self.counts_guess + 1) + "回
合 | Round " + str(self.counts_guess), color=(255, 255, 0, 0))

```

```
pl.osd_img.copy_from(draw_img)
```

```
# 读取石头剪刀布的bin文件方法 / Method to read rock-paper-scissors bin files
```

```
def read_file(self, file_name):
```

```
    image_arr = np.fromfile(file_name, dtype=np.uint8)
```

```
    image_arr = image_arr.reshape((400, 400, 4))
```

```
    return image_arr
```