南開大學
Nankai University

《Python综合课程设计----趣味工具箱》

Python综合课程设计

题　　目：	Python综合课程设计----趣味工具箱
上课时间：	周三下午
授课教师：	王斌辉
姓　　名：	张怡桢
学　　号：	2013747
年　　级：	2020级本科生
日　　期：	2023/1/8

Python综合课程设计----趣味工具箱

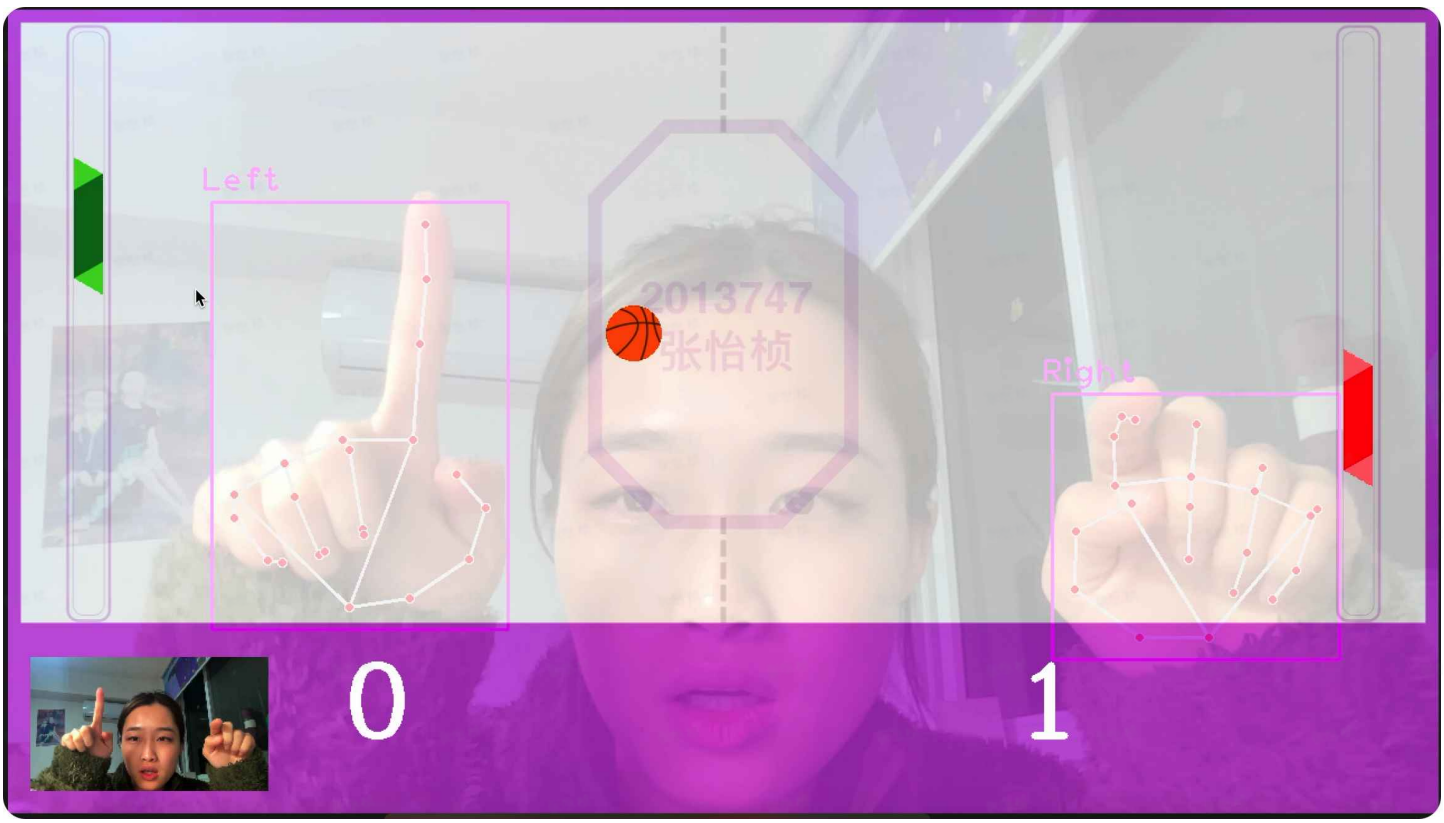
项目负责人：2013747张怡桢

1. 项目的功能

在这次的python综合课程设计中，我使用mediapipe开发了一个趣味工具箱，主要包含三个功能模块：

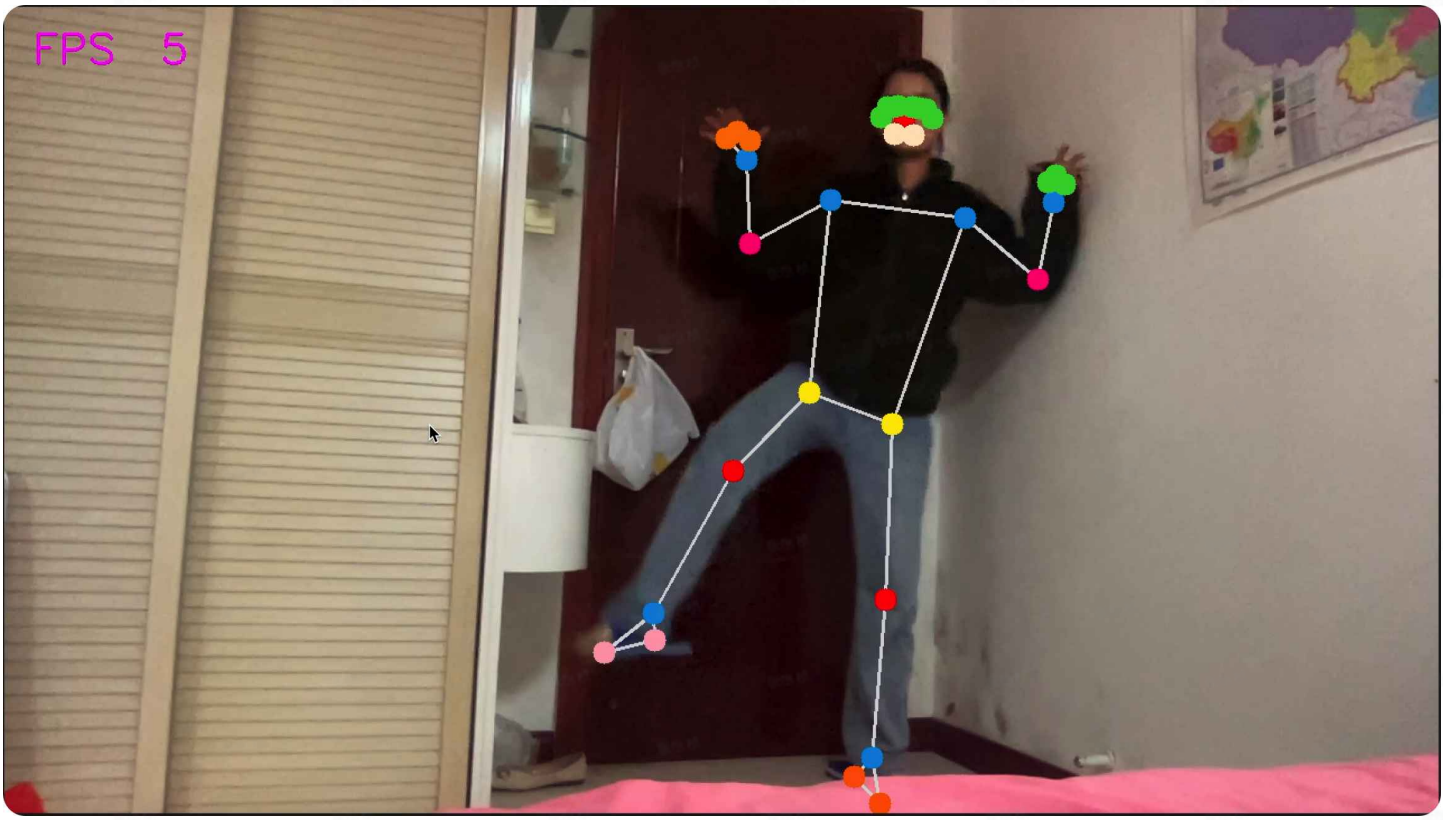
功能模块	功能定位	对应的技术栈	意义
手指乒乓	趣味游戏	cvzone=mediapipe+opencv	有趣的交互游戏，主要是手势识别
骨架检测	cv识别	Mediapipe	美观的骨架绘制，彩色的交互展示了mediap
俯卧撑计数	健身运动	Mediapipe	有趣的交互功能，主要是骨架识别以及对于姿

1.1 手指乒乓



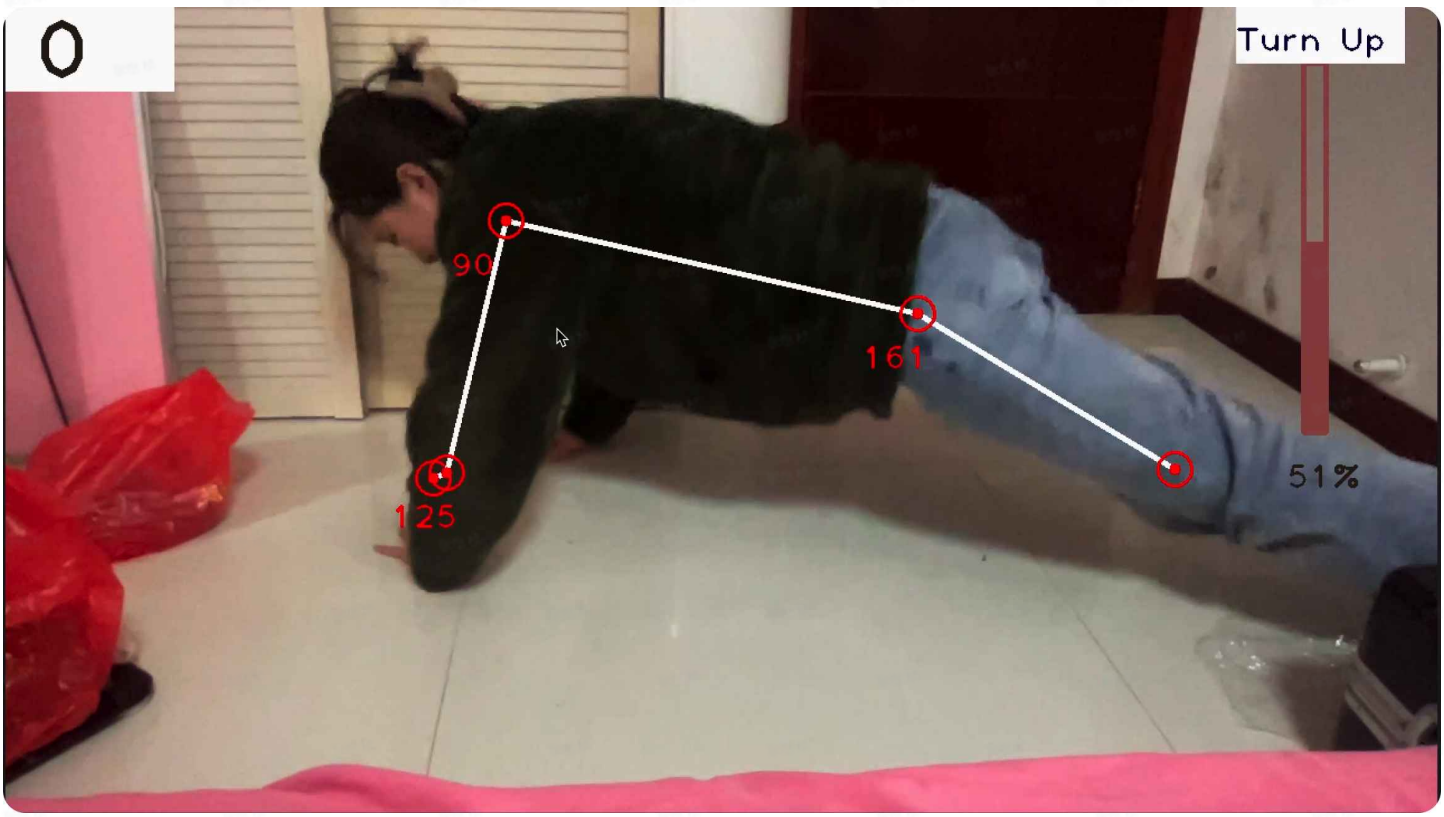
一个有趣的手势交互游戏，使用摄像头识别左右手的位置从而生成反弹条，小球在白色区域进行自动弹射，反弹条使小球往回运动，进行累分，最后的总得分为左手+右手的总得分，随着反弹次数的增大，小球的运动速度也一直在增大，游戏难度提高，增加游戏的耐玩度以及趣味性。

1.2 骨架检测



检测出人体的每一个部位点，并进行标记，使用彩色的图像进行展示，美观骨架模型。

1.3 俯卧撑计数



使用mediapipe进行健身功能的开发，使用mediapipe自动对健身的姿态进行检测与计数，使得运动可以得到监督，增添趣味。

2. 技术栈

2.1 UI开发以及音效

功能	技术栈（python库）
UI界面	PyQt5（QtCore, QtGui, QtWidgets）
音效	PyQt5.QtMultimedia playsound

2.2 功能

功能模块	对应的技术栈
手指乒乓	cvzone=mediapipe+opencv
骨架检测	Mediapipe opencv
俯卧撑计数	Mediapipe opencv

1. 手指乒乓

在开发趣味工具箱时，主要使用到的计数栈为mediapipe，opencv以及cvzone，**CVzone**是一个计算机视觉包，可以让我们轻松运行像人脸检测、手部跟踪、姿势估计等，以及图像处理和其他 AI 功能。它的核心是使用 OpenCV 和 **MediaPipe** 库。在手指乒乓中，使用的手部跟踪绘制出反弹条就是用的cvzone.HandTrackingModule模块，其主要是使用了mediapipe进行二次python库开发，使得我们对于mediapipe的应用更加快速便捷。

2. 骨架检测

在骨架检测部分，我直接使用了mediapipe的Pose检测身体的landmarks，并对于mediepipe绘制的骨架进行丰富，进行骨架的美化。

3. 俯卧撑计数

这一部分也是使用了mediapipe检测出身体的姿势之后，自己绘制判断函数，对于姿势进行判断，从而进行俯卧撑的标准判断以及计数。

3. 实现思路

对于图像识别，我将上述三种功能进行封装成类，并对类函数中的功能函数传入摄像头读取到的image，而后输出处理后的image，在页面上进行显示。

模块	类文件	类名	处理函数
手指乒乓	Game.Model.finger_game.py	finger_game_model	finger_game_process
骨架检测	Game.Model.person_draw.py	person_draw_model	person_draw_process
俯卧撑计数	Game.Model.push_up.py	push_up_model	push_up_process

3.1 手指乒乓

如下述的代码片段，实现手指乒乓的思路是：

1. 使用cvzone的HandDetector()检测出手的位置，并根据手的高度绘制出反弹条
2. 给小球限制运动范围并指定小球在规定的范围内进行弹射运动，碰到边界时发出音效同时反向运动
3. 反弹条对小球进行拦截，拦截成功分数+1，拦截失败游戏结束

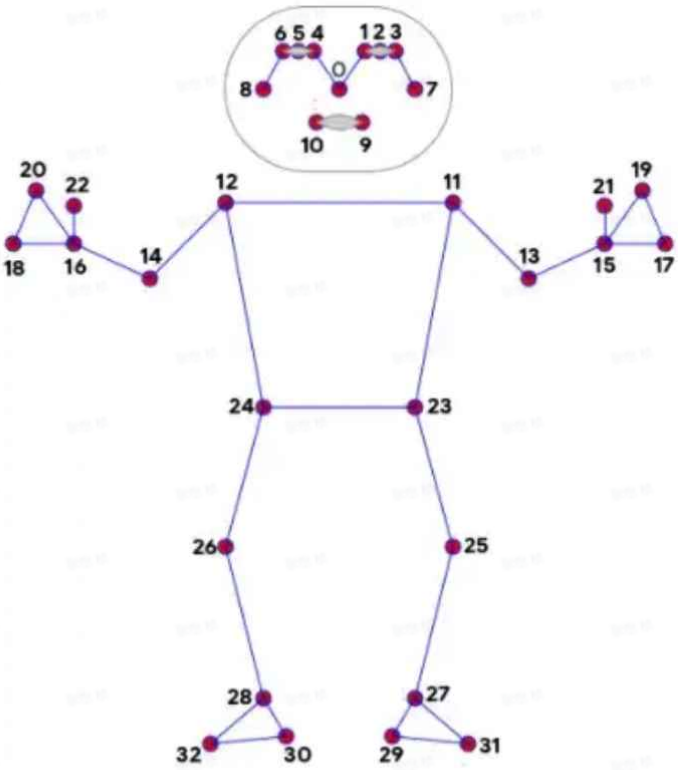
```
def finger_game_process(self,image):
    img = image
    img = cv2.flip(img, 1)
    imgRaw = img.copy()

    # Find the hand and its landmarks
    hands, img = detector.findHands(img, flipType=False)  # with draw
    img = cv2.addWeighted(img, 0.2, imgBackground, 0.8, 0)

    # Check for hands
    if hands:
        for hand in hands:
            x, y, w, h = hand['bbox']
            h1, w1, _ = imgBat1.shape
            y1 = y - h1 // 2
            y1 = np.clip(y1, 20, 415)

            if hand['type'] == "Left":
                img = cvzone.overlayPNG(img, imgBat1, (59, y1))
                if 59 < self.ballPos[0] < 59 + w1 and y1 < self.ballPos[1] < y1
                    self.speedX = -self.speedX
                    playsound("../Resources/ball_collision.wav")
                    self.ballPos[0] += 30
                    self.score[0] += 1

            if hand['type'] == "Right":
                img = cvzone.overlayPNG(img, imgBat2, (1195, y1))
                if 1195 - 50 < self.ballPos[0] < 1195 and y1 < self.ballPos[1] <
                    self.speedX = -self.speedX
                    playsound("../Resources/ball_collision.wav")
```

```
                    self.ballPos[0] -= 30
                    self.score[1] += 1

    # Game Over
    if self.ballPos[0] < 40 or self.ballPos[0] > 1200:
        self.gameOver = True

    if self.gameOver:
        img = self.imgGameOver
        cv2.putText(img, str(self.score[1] + self.score[0]).zfill(2), (585, 360)
                    2.5, (200, 0, 200), 5)
        if not self.played:
            playsound("../Resources/fail.mp3")
            self.played = True

    # If game not over move the ball
    else:

        # Move the Ball
        if self.ballPos[1] >= 500 or self.ballPos[1] <= 10:
            self.speedY = -self.speedY
            playsound("../Resources/success.wav")


        self.ballPos[0] += self.speedX
        self.ballPos[1] += self.speedY

        # Draw the ball
        img = cvzone.overlayPNG(img, imgBall, self.ballPos)

        cv2.putText(img, str(self.score[0]), (300, 650), cv2.FONT_HERSHEY_COMPLE
        cv2.putText(img, str(self.score[1]), (900, 650), cv2.FONT_HERSHEY_COMPLE

    img[580:700, 20:233] = cv2.resize(imgRaw, (213, 120))

    return img
```

3.2 骨架检测

骨架检测的实现思路：

1. 使用mediapipe.solutions.pose.Pose().process(image)检测得到的身体的pose_landmarks，可以看到下图为mediapipe官方给出的pose的landmarks可以检测到的器官点



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Landmarks detected by the mediapipe Pose Module

2. 使用mediapipe.solutions.drawing_utils.draw_landmarks(image,pose_landmarks)工具可视化关键点及骨架连线，同时对于所有的关节进行彩色汇点，丰富mediapipe原来的可视化骨架。
3. 同时使用time函数在页面左上角显示视频处理的帧率。

如下的代码就是主要实现该功能的部分，对于特定的关节点，使用特定的颜色进行绘制，美化mediapipe原有的骨架模型绘图工具。

```
def person_draw_process(self, img):

    # 记录该帧开始处理的时间
    start_time = time.time()

    # 获取图像宽高
    h, w = img.shape[0], img.shape[1]

    # BGR转RGB
    img_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # 将RGB图像输入模型，获取预测结果
    results = self.pose.process(img_RGB)

    if results.pose_landmarks:  # 若检测出人体关键点

        # 可视化关键点及骨架连线
        self.mp_drawing.draw_landmarks(img, results.pose_landmarks, self.mp_pose
```

```
18
19          for i in range(33):  # 遍历所有33个关键点，可视化
20
21              # 获取该关键点的三维坐标
22              cx = int(results.pose_landmarks.landmark[i].x * w)
23              cy = int(results.pose_landmarks.landmark[i].y * h)
24              cz = results.pose_landmarks.landmark[i].z
25
26              radius = 10
27
28              if i == 0:  # 鼻尖
29                  img = cv2.circle(img, (cx, cy), radius, (0, 0, 255), -1)
30              elif i in [11, 12]:  # 肩膀
31                  img = cv2.circle(img, (cx, cy), radius, (223, 155, 6), -1)
32              elif i in [23, 24]:  # 髋关节
33                  img = cv2.circle(img, (cx, cy), radius, (1, 240, 255), -1)
34              elif i in [13, 14]:  # 胳膊肘
35                  img = cv2.circle(img, (cx, cy), radius, (140, 47, 240), -1)
36              elif i in [25, 26]:  # 膝盖
37                  img = cv2.circle(img, (cx, cy), radius, (0, 0, 255), -1)
38              elif i in [15, 16, 27, 28]:  # 手腕和脚腕
39                  img = cv2.circle(img, (cx, cy), radius, (223, 155, 60), -1)
40              elif i in [17, 19, 21]:  # 左手
41                  img = cv2.circle(img, (cx, cy), radius, (94, 218, 121), -1)
42              elif i in [18, 20, 22]:  # 右手
43                  img = cv2.circle(img, (cx, cy), radius, (16, 144, 247), -1)
44              elif i in [27, 29, 31]:  # 左脚
45                  img = cv2.circle(img, (cx, cy), radius, (29, 123, 243), -1)
46              elif i in [28, 30, 32]:  # 右脚
47                  img = cv2.circle(img, (cx, cy), radius, (193, 182, 255), -1)
48              elif i in [9, 10]:  # 嘴
49                  img = cv2.circle(img, (cx, cy), radius, (205, 235, 255), -1)
50              elif i in [1, 2, 3, 4, 5, 6, 7, 8]:  # 眼及脸颊
51                  img = cv2.circle(img, (cx, cy), radius, (94, 218, 121), -1)
52              else:  # 其它关键点
53                  img = cv2.circle(img, (cx, cy), radius, (0, 255, 0), -1)
54
55          # 展示图片
56          # look_img(img)
57
58      else:
59          scaler = 1
60          failure_str = 'No Person'
61          img = cv2.putText(img, failure_str, (25 * scaler, 100 * scaler), cv2.FON
62                            (255, 0, 255), 2 * scaler)
63          # print('从图像中未检测出人体关键点，报错。')
64
```

```
    # 记录该帧处理完毕的时间
    end_time = time.time()
    # 计算每秒处理图像帧数FPS
    FPS = 1 / (end_time - start_time)

    scaler = 1
    # 在图像上写FPS数值，参数依次为：图片，添加的文字，左上角坐标，字体，字体大小，颜色，
    img = cv2.putText(img, 'FPS  ' + str(int(FPS)), (25 * scaler, 50 * scaler),
                      1.25 * scaler,
                      (255, 0, 255), 2 * scaler)
    return img
```

3.3 俯卧撑计数

俯卧撑计数的实现思路是在上一个功能的基础上，对于某些特定的关节进行检测，对于特定的关节连线形成的夹角进行计算与分析，进而进行判断是否符合标准，然后进行计数。

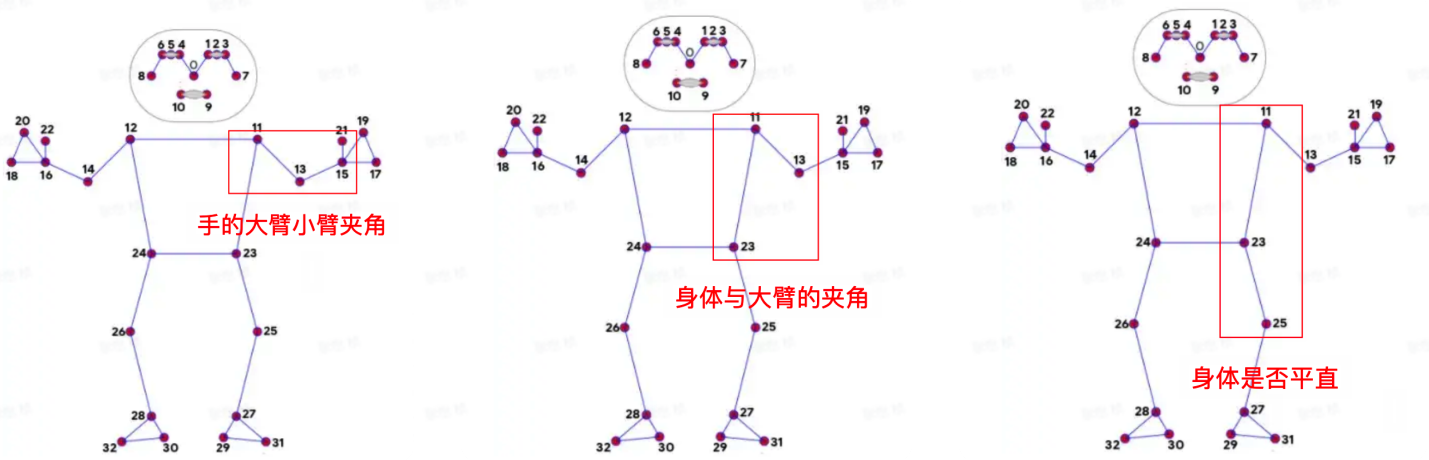
```
def findAngle(self, img, p1, p2, p3, draw=True):
    # Get the landmarks
    x1, y1 = self.lmList[p1][1:]
    x2, y2 = self.lmList[p2][1:]
    x3, y3 = self.lmList[p3][1:]

    # Calculate Angle
    angle = math.degrees(math.atan2(y3 - y2, x3 - x2) -
                         math.atan2(y1 - y2, x1 - x2))
    if angle < 0:
        angle += 360
        if angle > 180:
            angle = 360 - angle
    elif angle > 180:
        angle = 360 - angle
    # print(angle)

    # Draw
    if draw:
        cv2.line(img, (x1, y1), (x2, y2), (255, 255, 255), 3)
        cv2.line(img, (x3, y3), (x2, y2), (255, 255, 255), 3)

        cv2.circle(img, (x1, y1), 5, (0, 0, 255), cv2.FILLED)
        cv2.circle(img, (x1, y1), 15, (0, 0, 255), 2)
        cv2.circle(img, (x2, y2), 5, (0, 0, 255), cv2.FILLED)
        cv2.circle(img, (x2, y2), 15, (0, 0, 255), 2)
        cv2.circle(img, (x3, y3), 5, (0, 0, 255), cv2.FILLED)
        cv2.circle(img, (x3, y3), 15, (0, 0, 255), 2)
```

```
29
30          cv2.putText(img, str(int(angle)), (x2 - 50, y2 + 50),
31                      cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 255), 2)
32      return angle
```

其中最为重要的是对于角度的计算函数，实现的思路如上，根据传入的三个器官点进行角度的计算，从而进一步实现俯卧撑标准的判断。

在俯卧撑判断中：

```
1  # 11: left shoulder
2          # 13: left elbow
3          # 15: left wrist
4          # 23: left hip
5          # 25: left knee
6          elbow = self.detector.findAngle(img, 11, 13, 15)
7          shoulder = self.detector.findAngle(img, 13, 11, 23)
8          hip = self.detector.findAngle(img, 11, 23, 25)
```



主要是计算这三个关节的夹角，手肘，肩膀以及臀部。

判断	elbow	shoulder	hip
开始计数	elbow > 160	shoulder > 40	hip > 160
up（到达顶部）	elbow > 150	shoulder > 40	hip > 150
down（到达底部）	elbow <= 95	--	hip > 150

上表是对于俯卧撑姿势的判断，在达到相应的条件时，会出现对应的音效音提示，在画面中也会实时显示每个部位的夹角大小以及形成了一个以elbow的大小进行线性插值的进度条，以及提示接下来该

turn down还是turn up的提示牌。

实现俯卧撑计数以及音效提示up and down的process代码如下

```
def push_up_process(self, img):

    img = self.detector.findPose(img, False)
    lmList = self.detector.findPosition(img, False)
    # print(lmList)
    if len(lmList) != 0:
        # 11: left shoulder
        # 13: left elbow
        # 15: left wrist
        # 23: left hip
        # 25: left knee
        elbow = self.detector.findAngle(img, 11, 13, 15)
        shoulder = self.detector.findAngle(img, 13, 11, 23)
        hip = self.detector.findAngle(img, 11, 23, 25)

        # Percentage of success of pushup
        per = np.interp(elbow, (90, 160), (0, 100))

        # Bar to show Pushup progress
        bar = np.interp(elbow, (90, 160), (380, 50))

        # Check to ensure right self.form before starting the program
        if elbow > 160 and shoulder > 40 and hip > 160:
            self.form = 1

        # Check for full range of motion for the pushup
        if self.form == 1:
            if per == 0:
                if elbow <= 95 and hip > 150:
                    self.feedback = "Turn Up"
                    if self.direction == 0:
                        self.count += 0.5
                        playsound("../Resources/success.wav")
                        self.direction = 1
                else:
                    self.feedback = "Fix Form"

            if per == 100:
                if elbow > 150 and shoulder > 40 and hip > 150:
                    self.feedback = "Turn Down"
                    if self.direction == 1:
                        self.count += 0.5
                        playsound("../Resources/success2.wav")
```

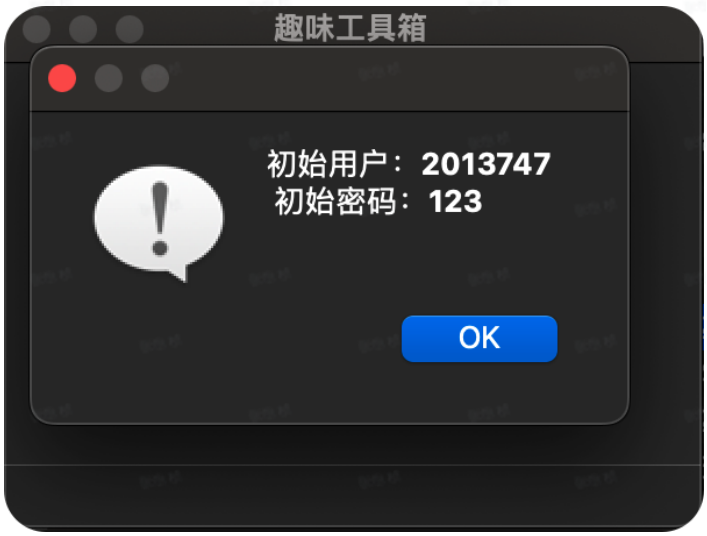
```
                    self.direction = 0
                else:
                    self.feedback = "Fix Form"
                    # self.form = 0

        # print(self.count)

        # Draw Bar
        if self.form == 1:
            cv2.rectangle(img, (1160, 50), (1180, 380), (97, 100, 159), 3)
            cv2.rectangle(img, (1160, int(bar)), (1180, 380), (97, 100, 159), cv
            cv2.putText(img, f'{int(per)}%', (1145, 430), cv2.FONT_HERSHEY_PLAIN
                        (43, 51, 62), 2)

        # Pushup counter
        cv2.rectangle(img, (0, 0), (150, 75), (255, 255, 255), cv2.FILLED)
        cv2.putText(img, str(int(self.count)), (25, 65), cv2.FONT_HERSHEY_PLAIN,
                    (43, 51, 62), 5)

        # Feedback
        cv2.rectangle(img, (1100, 0), (1250, 50), (255, 255, 255), cv2.FILLED)
        cv2.putText(img, self.feedback, (1100, 40), cv2.FONT_HERSHEY_PLAIN, 2,
                    (77, 16, 24), 2)

    return img
```

4. 交互

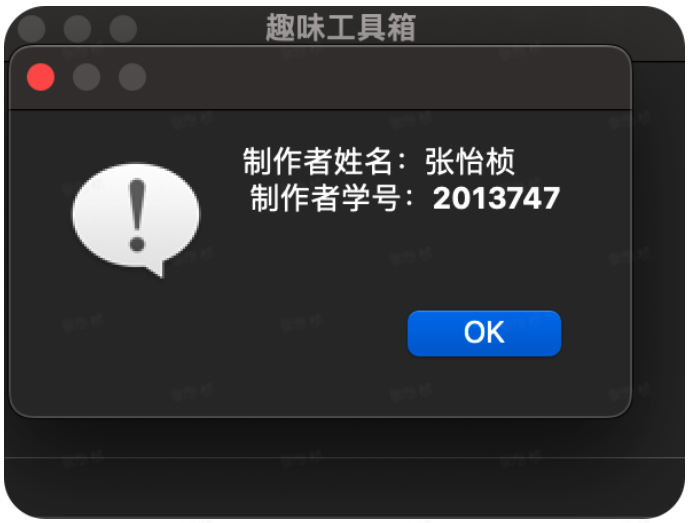
交互使用了pyqt5进行实现，同时辅以音频进行交互实现。

4.1 登陆页面

4.2 帮助页面



4.3 其他页面



4.4 登陆提示

4.5 主菜单

交互部分主要使用了Pyqt5进行开发，开发了对应的UI系统，实现完整的趣味工具箱的应用。

5. 测试

该视频为全部功能的测试与说明分析：

点击链接进行观看

2013747张怡桢——python大作业功能介绍.mp4

6. 环境

我的代码环境为Mac+Python3.9虚拟环境。

Python环境为使用Python虚拟环境创建的Python3.9的编程环境，生成的requireents.txt文件如下，为具体的虚拟环境

```
absl-py==1.3.0
attrs==22.2.0
certifi==2022.12.7
charset-normalizer==2.1.1
click==7.1.2
contourpy==1.0.6
cvzone==1.5.6
cycler==0.11.0
flatbuffers==23.1.4
fonttools==4.38.0
idna==3.4
kiwisolver==1.4.4
matplotlib==3.6.2
mediapipe==0.9.0.1
numpy==1.24.1
```

```
opencv-contrib-python==4.7.0.68
opencv-python==4.7.0.68
packaging==22.0
Pillow==9.4.0
playsound==1.3.0
protobuf==3.20.3
pyobjc==9.0.1
pyobjc-core==9.0.1
pyobjc-framework-Accessibility==9.0.1
pyobjc-framework-Accounts==9.0.1
pyobjc-framework-AddressBook==9.0.1
pyobjc-framework-AdServices==9.0.1
pyobjc-framework-AdSupport==9.0.1
pyobjc-framework-AppleScriptKit==9.0.1
pyobjc-framework-AppleScriptObjC==9.0.1
pyobjc-framework-ApplicationServices==9.0.1
pyobjc-framework-AppTrackingTransparency==9.0.1
pyobjc-framework-AudioVideoBridging==9.0.1
pyobjc-framework-AuthenticationServices==9.0.1
pyobjc-framework-AutomaticAssessmentConfiguration==9.0.1
pyobjc-framework-Automator==9.0.1
pyobjc-framework-AVFoundation==9.0.1
pyobjc-framework-AVKit==9.0.1
pyobjc-framework-AVRouting==9.0.1
pyobjc-framework-BackgroundAssets==9.0.1
pyobjc-framework-BusinessChat==9.0.1
pyobjc-framework-CalendarStore==9.0.1
pyobjc-framework-CallKit==9.0.1
pyobjc-framework-CFNetwork==9.0.1
pyobjc-framework-ClassKit==9.0.1
pyobjc-framework-CloudKit==9.0.1
pyobjc-framework-Cocoa==9.0.1
pyobjc-framework-Collaboration==9.0.1
pyobjc-framework-ColorSync==9.0.1
pyobjc-framework-Contacts==9.0.1
pyobjc-framework-ContactsUI==9.0.1
pyobjc-framework-CoreAudio==9.0.1
pyobjc-framework-CoreAudioKit==9.0.1
pyobjc-framework-CoreBluetooth==9.0.1
pyobjc-framework-CoreData==9.0.1
pyobjc-framework-CoreHaptics==9.0.1
pyobjc-framework-CoreLocation==9.0.1
pyobjc-framework-CoreMedia==9.0.1
pyobjc-framework-CoreMediaIO==9.0.1
pyobjc-framework-CoreMIDI==9.0.1
pyobjc-framework-CoreML==9.0.1
pyobjc-framework-CoreMotion==9.0.1
```

```
pyobjc-framework-CoreServices==9.0.1
pyobjc-framework-CoreSpotlight==9.0.1
pyobjc-framework-CoreText==9.0.1
pyobjc-framework-CoreWLAN==9.0.1
pyobjc-framework-CryptoTokenKit==9.0.1
pyobjc-framework-DataDetection==9.0.1
pyobjc-framework-DeviceCheck==9.0.1
pyobjc-framework-DictionaryServices==9.0.1
pyobjc-framework-DiscRecording==9.0.1
pyobjc-framework-DiscRecordingUI==9.0.1
pyobjc-framework-DiskArbitration==9.0.1
pyobjc-framework-DVDPlayback==9.0.1
pyobjc-framework-EventKit==9.0.1
pyobjc-framework-ExceptionHandling==9.0.1
pyobjc-framework-ExecutionPolicy==9.0.1
pyobjc-framework-ExtensionKit==9.0.1
pyobjc-framework-ExternalAccessory==9.0.1
pyobjc-framework-FileProvider==9.0.1
pyobjc-framework-FileProviderUI==9.0.1
pyobjc-framework-FinderSync==9.0.1
pyobjc-framework-FSEvents==9.0.1
pyobjc-framework-GameCenter==9.0.1
pyobjc-framework-GameController==9.0.1
pyobjc-framework-GameKit==9.0.1
pyobjc-framework-GameplayKit==9.0.1
pyobjc-framework-HealthKit==9.0.1
pyobjc-framework-ImageCaptureCore==9.0.1
pyobjc-framework-IMServicePlugIn==9.0.1
pyobjc-framework-InputMethodKit==9.0.1
pyobjc-framework-InstallerPlugins==9.0.1
pyobjc-framework-InstantMessage==9.0.1
pyobjc-framework-Intents==9.0.1
pyobjc-framework-IntentsUI==9.0.1
pyobjc-framework-IOSurface==9.0.1
pyobjc-framework-iTunesLibrary==9.0.1
pyobjc-framework-KernelManagement==9.0.1
pyobjc-framework-LatentSemanticMapping==9.0.1
pyobjc-framework-LaunchServices==9.0.1
pyobjc-framework-libdispatch==9.0.1
pyobjc-framework-LinkPresentation==9.0.1
pyobjc-framework-LocalAuthentication==9.0.1
pyobjc-framework-LocalAuthenticationEmbeddedUI==9.0.1
pyobjc-framework-MailKit==9.0.1
pyobjc-framework-MapKit==9.0.1
pyobjc-framework-MediaAccessibility==9.0.1
pyobjc-framework-MediaLibrary==9.0.1
pyobjc-framework-MediaPlayer==9.0.1
```

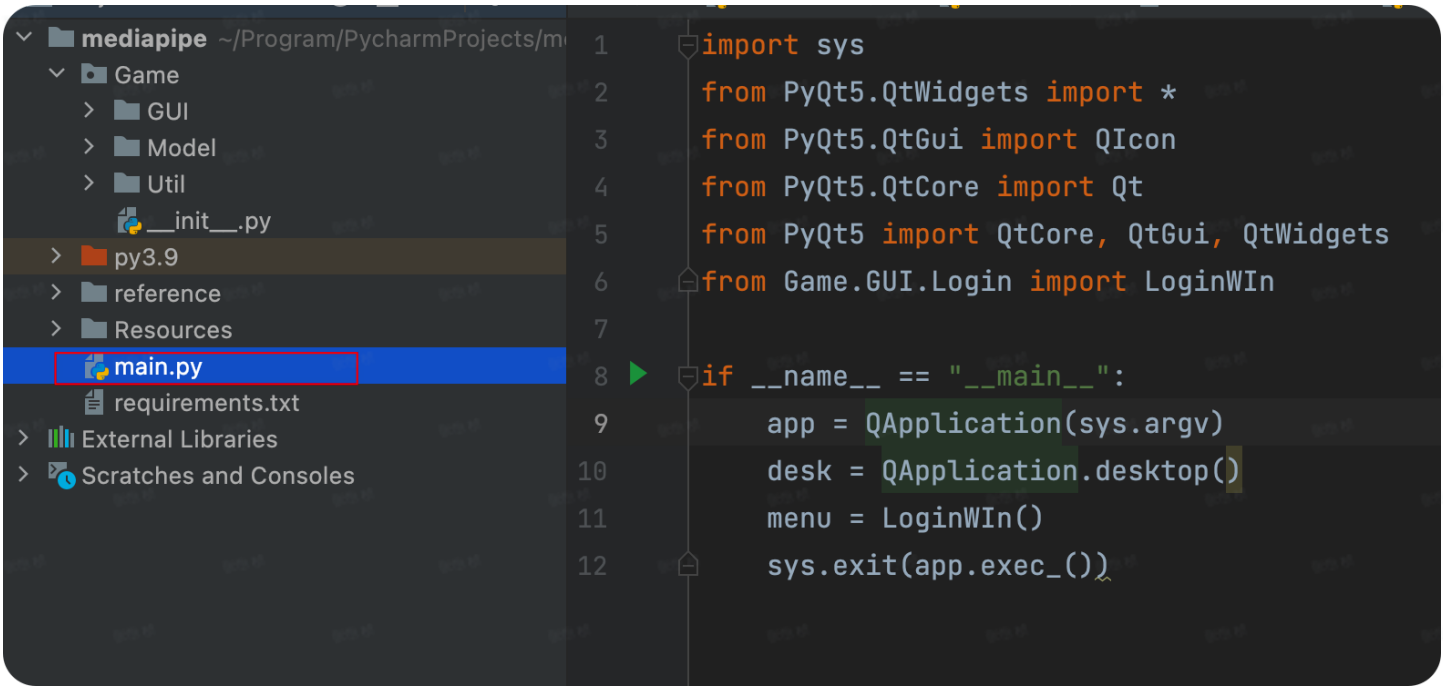
```
pyobjc-framework-MediaToolbox==9.0.1
pyobjc-framework-Metal==9.0.1
pyobjc-framework-MetalFX==9.0.1
pyobjc-framework-MetalKit==9.0.1
pyobjc-framework-MetalPerformanceShaders==9.0.1
pyobjc-framework-MetalPerformanceShadersGraph==9.0.1
pyobjc-framework-MetricKit==9.0.1
pyobjc-framework-MLCompute==9.0.1
pyobjc-framework-ModelIO==9.0.1
pyobjc-framework-MultipeerConnectivity==9.0.1
pyobjc-framework-NaturalLanguage==9.0.1
pyobjc-framework-NetFS==9.0.1
pyobjc-framework-Network==9.0.1
pyobjc-framework-NetworkExtension==9.0.1
pyobjc-framework-NotificationCenter==9.0.1
pyobjc-framework-OpenDirectory==9.0.1
pyobjc-framework-OSAKit==9.0.1
pyobjc-framework-OSLog==9.0.1
pyobjc-framework-PassKit==9.0.1
pyobjc-framework-PencilKit==9.0.1
pyobjc-framework-Photos==9.0.1
pyobjc-framework-PhotosUI==9.0.1
pyobjc-framework-PreferencePanes==9.0.1
pyobjc-framework-PushKit==9.0.1
pyobjc-framework-Quartz==9.0.1
pyobjc-framework-QuickLookThumbnailing==9.0.1
pyobjc-framework-ReplayKit==9.0.1
pyobjc-framework-SafariServices==9.0.1
pyobjc-framework-SafetyKit==9.0.1
pyobjc-framework-SceneKit==9.0.1
pyobjc-framework-ScreenCaptureKit==9.0.1
pyobjc-framework-ScreenSaver==9.0.1
pyobjc-framework-ScreenTime==9.0.1
pyobjc-framework-ScriptingBridge==9.0.1
pyobjc-framework-SearchKit==9.0.1
pyobjc-framework-Security==9.0.1
pyobjc-framework-SecurityFoundation==9.0.1
pyobjc-framework-SecurityInterface==9.0.1
pyobjc-framework-ServiceManagement==9.0.1
pyobjc-framework-SharedWithYou==9.0.1
pyobjc-framework-SharedWithYouCore==9.0.1
pyobjc-framework-ShazamKit==9.0.1
pyobjc-framework-Social==9.0.1
pyobjc-framework-SoundAnalysis==9.0.1
pyobjc-framework-Speech==9.0.1
pyobjc-framework-SpriteKit==9.0.1
pyobjc-framework-StoreKit==9.0.1
```

```
pyobjc-framework-SyncServices==9.0.1
pyobjc-framework-SystemConfiguration==9.0.1
pyobjc-framework-SystemExtensions==9.0.1
pyobjc-framework-ThreadNetwork==9.0.1
pyobjc-framework-UniformTypeIdentifiers==9.0.1
pyobjc-framework-UserNotifications==9.0.1
pyobjc-framework-UserNotificationsUI==9.0.1
pyobjc-framework-VideoSubscriberAccount==9.0.1
pyobjc-framework-VideoToolbox==9.0.1
pyobjc-framework-Virtualization==9.0.1
pyobjc-framework-Vision==9.0.1
pyobjc-framework-WebKit==9.0.1
pyparsing==3.0.9
PyQt5==5.15.4
pyqt5-plugins==5.15.4.2.2
PyQt5-Qt5==5.15.2
PyQt5-sip==12.11.0
PyQt5-stubs==5.15.6.0
pyqt5-tools==5.15.4.3.2
python-dateutil==2.8.2
python-dotenv==0.21.0
qt5-applications==5.15.2.2.2
qt5-tools==5.15.2.1.2
requests==2.28.1
six==1.16.0
urllib3==1.26.13
```

7. 部署文档

创建一个Python3.9的虚拟环境，cd到项目文件夹中，使用下面的语句批量安装包

```
pip install -r requirements.txt
```



运行文件夹中的main文件，即可运行该app。

8. 附录

该实验报告的飞书云文档地址（Python综合课程设计----趣味工具箱 ）

代码的github地址【https://github.com/nkuzyz/2013747zyz_python】