

國立虎尾科技大學資訊工程系

說『畫』

參與成員：高家祥
張壹智
韓博丞

指導教授：黃世昌教授

中華民國 109 年 6 月 14 日

目錄

壹、摘要	3
貳、專題緣由與目的	5
參、作品架構	6
一、系統整體架構	6
二、JavaFx 使用者介面與 Python 語音接收架構	6
三、JavaFx 端的系統架構	7
四、JavaFx 端的指令架構	8
五、JavaFx 端的指令執行架構	9
六、Python 語音分析架構	10
七、Python 語音分析與接收端的指令執行架構	10
八、Python 語音分析程式碼架構	11
肆、作品原理	12
一、作品提供	12
A. 解析語音詞彙	12
B. 執行語音/文字的指令動作	12
C. 對使用者的輸入除錯	12
二、程式技術	13
A. 使用語言	13
B. 軟體應用之演算法	14
伍、系統需求	15
一、系統功能需求	15
二、系統非功能需求	16
三、使用者案例	17
陸、作品特色	20
一、作品應用類型與範圍	20
A. 使用軟體年齡範圍 (2+)	20
B. 作品應用	20

二、特性與設計重點	21
A. 特性	21
B. 設計重點	21
三、創意及原創性說明	22
A. 創意：	22
B. 原創性	22
C. 原創 LOGO 設計	22
四、市場潛力說明	23
A. 以一般應用為主	23
B. 以小朋友為主	23
C. 以打擊犯罪為主	23
D. 以資訊應用為主	23
柒、操作手冊	24
一、環境建置	24
二、使用者介面	25
三、指令功能表	28
捌、系統環境需求與限制	29
一、最低硬體配置需求	29
二、環境配置需求	29
玖、結論	30
壹拾、團隊分工	31
壹拾壹、參考文獻	32
壹拾貳、心得	34

國立虎尾科技大學資訊工程系專題

題目：說『畫』

指導老師：黃世昌教授

專題參與人員：張壹智、高家祥、韓博丞

班級：資訊工程系三年甲班

壹、摘要

你可曾想過只使用口說來繪製出一幅畫？就好比我對著一片空白的紙說：「這幅畫有個綠意盎然山丘。」而這時這張白紙上就會神奇地出現一個綠意盎然的山丘，這時我再說：「在山丘下有個河流。」這時就會在山丘下出現一個河流，依此類推將一幅畫利用口語描述，繪製出一幅自己理想的畫作。關於以上的形容就是我們所要達成的，利用聲控來繪製出一幅屬於自己的畫作。

關鍵字：肢體障礙者、語音、聲控、繪圖、便利性、快速創作

Summary

Have you ever been wondered could we can just speak to draw picture? For example, we said : 「 that picture have a mountain 」 . magically, that will appear a mountain. and, We go on said : 「 A river is under the mountain 」 .The picture will appear a river under the mountain. And so on. Finally you can only describe that your imagination to create a picture by speech. so, Above the describe. That is what we want to make it. Using speech to draw a picture.

Keywords: voice 、 drawing 、 convenience 、 fast create 、 voice control

貳、專題緣由與目的

隨著社會進步、醫藥發達，雖然現在許多身心障礙者或肢體不便者在生活上已經獲得了改善，但是在某些部分對於這些族群還是很不友善。所以我們就想，假如這些身心障礙者或肢體不便者想要描述一些事物，而欲描述的事物又過於複雜時，他們該怎麼表達？依照一般人基本上都是拿筆與紙在上面畫出他們想要表達的事物，好讓他人去理解，但是這些身心障礙人士中手部行動不方便者又要如何去使用紙筆來寫下或畫下他們所要表達的事物。

在現今社會中，許多人的工作複雜且繁忙，並且許多工作都必須使用兩手並用來完成工作。假如這時主管要求自己，需要做個對公司的作品簡單概述的插圖，而這時就可以使用我們的作品，簡單的用聲控描述下想要展示出來的結果，就能做出一些初步的構圖，雖然這樣使用者需要一心多用，不過能解省工作量，也能增加工作中的效率。

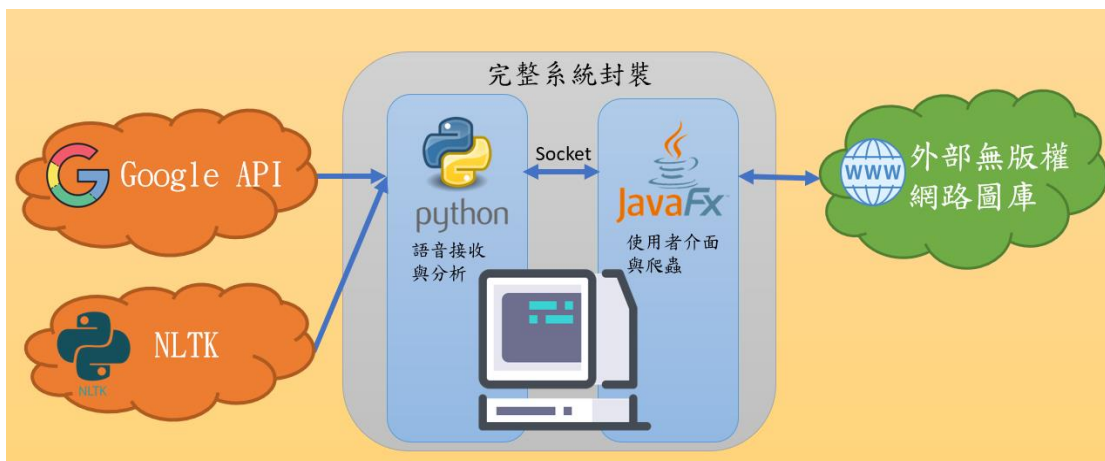
除了上面提到的之外，在一般人中，繪畫是一個需要天分且大量時間練習的事情，所以絕大數人在繪畫上都有一些困難，而當他們需要設計一些草圖時，難免會遇到一些困難。而本人也是屬於不太會繪圖的人，當自己需要為專題設計一些美術圖或草圖時，都難免遇到一些問題，像是需要設計帶有科技感的海報或具有樸素風格的書面等等。而在我製作過程中發現，我所使用的素材都源自於網路上的無版權素材進行拼貼，進而達到自己所理想的設計。其間給我們了對此專題許多的靈感，讓我們有了初步的構想。

根據上述的需求，本計劃希望能透過這些需求，結合資訊科技的技術，開發出一套對這些族群有幫助的科技作品，使用聲控來繪製或設計出一張屬於自己的圖，而本作品的宗旨就是“簡單快速”減少手部操作的次數，能完全使用聲控創造一張無瑕的畫作，並且能讓語音辨識能非常的人性化，不需要讓使用者去熟記有哪些命令，以減少操作上的複雜度。除此之外能提供給各個年齡層以及族群使用，增加他們的工作與操作的靈活度，能讓每個人都能“說”出自己心目中的畫作。

參、作品架構

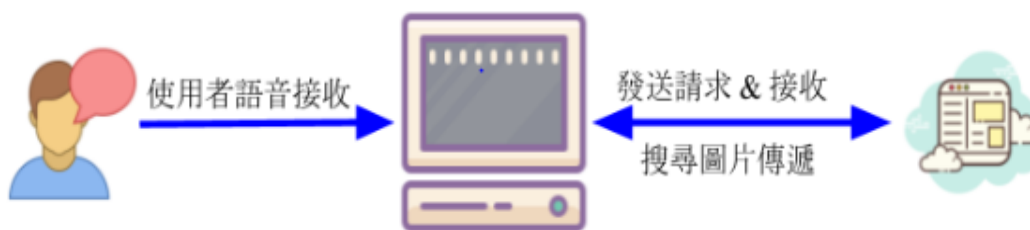
一、系統整體架構

由 JavaFx 當使用者介面的基底，Python 提供語音接收與分析，再由 Socket 傳遞訊息。並且 JavaF 會連上網站抓取圖片資訊，並儲存到程式裡提供使用者使用。



圖一、系統整體架構

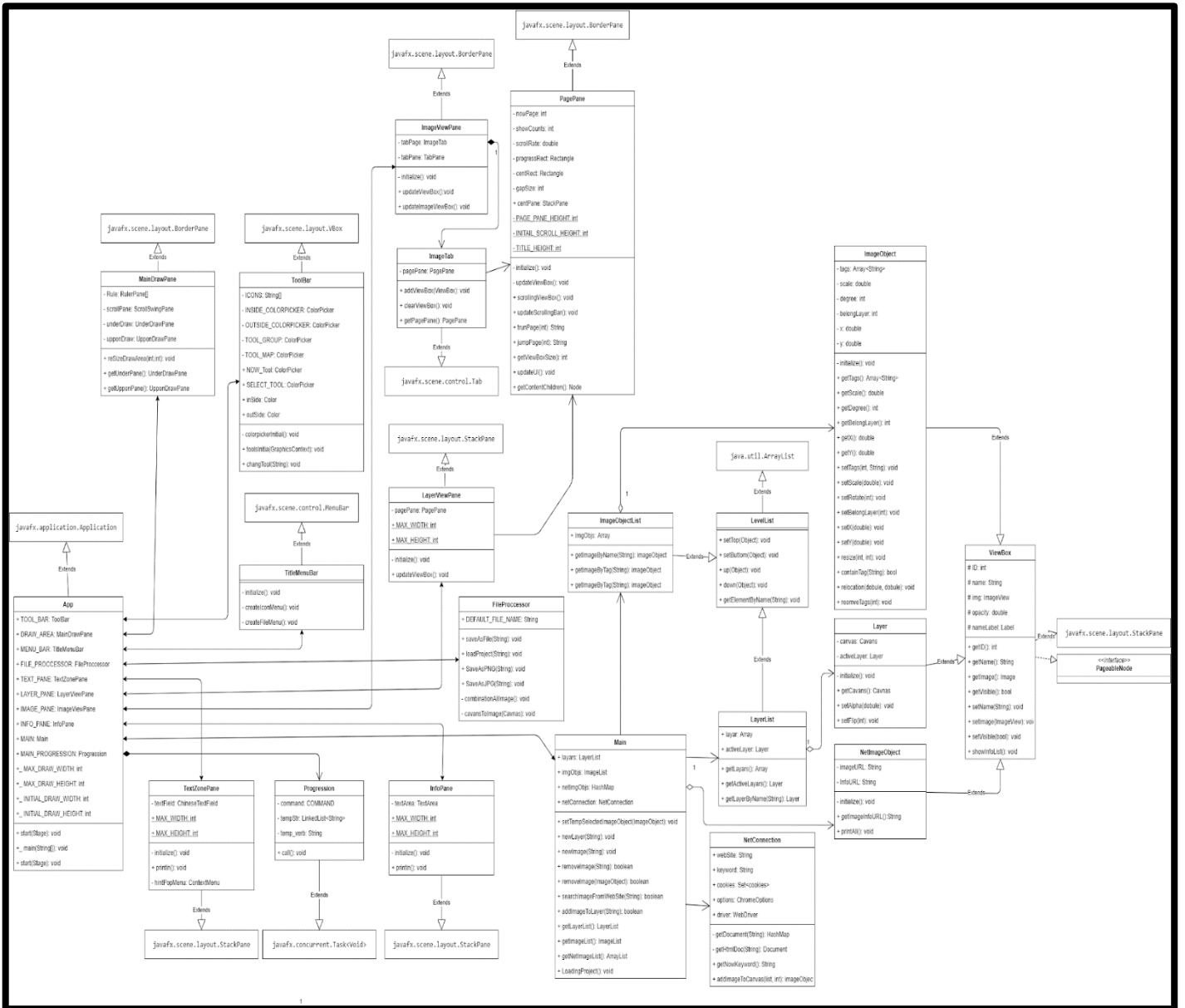
二、JavaFx 使用者介面與 Python 語音接收架構



圖二、JavaFx 使用者介面與 Python 語音接收架構圖

三、JavaFx 端的系統架構

我們使用多層架構，並且將各個物件分成細小的物件，並加以組合。這樣不但當上層修改時，底層也不會受到較大的影響。除此之外，我們將裡面的物件使用完好的封裝，可以方便的取用數值，也不會使的物件的數值發生異常修改。



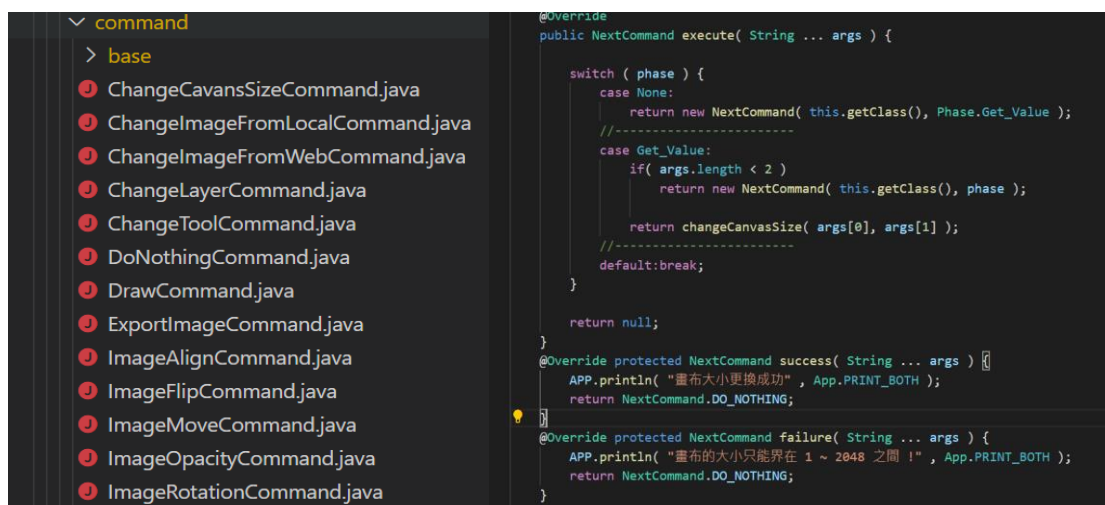
圖三、JavaFx UI 的系統架構圖

四、JavaFx 端的指令架構

我這裡使用 .yaml 檔案系統來管理我們的指令，這種寫法有許多的優點。首先，這能使得我們的指令架構能良好的管理，在來能動態的新增語移除指令，而且也可以依照我們的需求而修改裡面的參數與數值等情況。並且我們將每一個指令分別分成一個物件，利用了 OOP 的泛型的特性，可以在不同的指令輸入進而做不同的動作，並且也不會與其他指令互相干擾，也能根據各個指令的需求而做相對應的修改。

```
1  ---
2  name: DoNothingCommand
3  date: 2020/05/02
4  attribute:
5    description: "沒有任何功能，只用來等待指令"
6    englishName: "Do_Nothing"
7    chineseName: "不做事"
8    useTemporary: false
9    usage: "DoNothing"
10   args: [""]
11
12
13  ---
14  name: ChangeLayerCommand
15  date: 2020/05/02
16  attribute:
17    description: "用來改變目前操作圖層的指令"
18    englishName: "Change_Layer"
19    chineseName: "改變操作圖層"
20    useTemporary: false
21    usage: "ChangeLayer [name]"
22    args: [""]
23
24
25  ---
26  name: ChangeCavansSizeCommand
27  date: 2020/05/02
28  attribute:
29    description: "用來改變目前畫布大小的指令"
30    englishName: "Change_Cavans_Size"
31    chineseName: "改變畫布大小"
32    useTemporary: false
33    usage: "CavansSize [width] [height]"
34    args: [""]
35
36  ---
37  name: NewImageFromLocalCommand
38  date: 2020/06/02
39  attribute:
40    description: "取得本地上的圖片"
41    englishName: "Add_Image_From_Local"
42    chineseName: "取得本地圖片"
43    useTemporary: false
44    usage: "newLocalImage [keyword]"
45    args: [""]
46
```

圖四、JavaFx 端的指令集架構圖



```
command
├── base
│   ├── ChangeCavansSizeCommand.java
│   ├── ChangeImageFromLocalCommand.java
│   ├── ChangeImageFromWebCommand.java
│   ├── ChangeLayerCommand.java
│   ├── ChangeToolCommand.java
│   ├── DoNothingCommand.java
│   ├── DrawCommand.java
│   ├── ExportImageCommand.java
│   ├── ImageAlignCommand.java
│   ├── ImageFlipCommand.java
│   ├── ImageMoveCommand.java
│   ├── ImageOpacityCommand.java
│   └── ImageRotationCommand.java
└── ...

@Override
public NextCommand execute( String ... args ) {

    switch ( phase ) {
        case None:
            return new NextCommand( this.getClass(), Phase.Get_Value );
            //-----
        case Get_Value:
            if( args.length < 2 )
                return new NextCommand( this.getClass(), phase );

            return changeCanvasSize( args[0], args[1] );
            //-----
        default:break;
    }

    return null;
}

@Override protected NextCommand success( String ... args ) {
    APP.println( "畫布大小更換成功" , App.PRINT_BOTH );
    return NextCommand.DO_NOTHING;
}

@Override protected NextCommand failure( String ... args ) {
    APP.println( "畫布的大小只能界在 1 ~ 2048 之間 !" , App.PRINT_BOTH );
    return NextCommand.DO_NOTHING;
}
```

圖五、JavaFx 端的指令程式碼架構圖

五、JavaFx 端的指令執行架構

這裡使用有現狀態機的指令架構，當 Python 語音端傳遞指令過來時，程式就會進入相對應的狀態，進而執行相對應的指令。

```
while(true){

    //Background work
    //final CountdownLatch latch = new CountdownLatch(1);

    System.out.println( ConsoleColors.INFO + "Waiting Command Voice Singal " + System.currentTimeMillis() );

    //判斷語音輸入 or 文字輸入 是否為空
    if( IAPP.SOCKET_SEVER.isReceived() && !APP.TEXT_PANE.hasInput() && commandStack.size() <= 0 ){
        Thread.sleep( 500 );
        APP.STATUS_PANE.animation();
        APP.SOCKET_SEVER.getFloatingPane().animation();
        continue;
    }

    //假如指令 Stack 有指令
    if( commandStack.size() > 0 ){
        nowCommand = commandStack.pop().execute( "" ).toCommand( commandMap );
        APP.STATUS_PANE.changeBothText( nowCommand.getChineseName(), nowCommand.getPhase().getName() );
    }

    //-----
    //假如接收到 語音資訊 or 文字輸入
    else if( APP.SOCKET_SEVER.isReceived() || APP.TEXT_PANE.hasInput() ){

        if( APP.SOCKET_SEVER.isReceived() ){

            //假如他是垃圾訊息就印出所有字串 (不執行)
            if( APP.SOCKET_SEVER.isSpamMsg() ){
                String temp = APP.SOCKET_SEVER.getFormatWholeMsg();
                APP.INFO_PANE.println( temp );
                continue;
            }

            //判斷 Python 讀入的字串是否有多個，有多個就直接存到 List 裡
            argList.addAll( APP.SOCKET_SEVER.getVoiceMsg() );
        }
        else if( APP.TEXT_PANE.hasInput() ){
            argList.addAll( APP.TEXT_PANE.getTextInput() );
        }

        //判斷使用者是不是要翻页
        if( checkTurnPage( argList.getFirst() ) ){
            argList.removeFirst();
            continue;
        }
    }

    //-----
    //假如現在是 DO_NOTHING 就判斷 command
    if( nowCommand instanceof DoNothingCommand ){
        //判斷是否有包含這個指令
        Command tempCmd = commandMap.get( argList.getFirst() );
        if( tempCmd != null ){
            //移除第一個指令，只剩下數值
            argList.removeFirst();

            //指令階段初始化
            tempCmd.initial();
            NextCommand nextCommand = tempCmd.execute( "" );

            //假如 NextCommand 的 isStack 為 true，代表要將上個指令推入堆疊裡，以方便下次執行
            if( nextCommand.isStack() ){
                commandStack.push( tempCmd );
            }

            //取得下個指令
            nowCommand = nextCommand.toCommand( commandMap );
        }

        //傳送目前狀態給 Python Socket
        APP.SOCKET_SEVER.sendStatus( nowCommand.getClass().getSimpleName() );

        //更改訊息狀態顯示
        APP.STATUS_PANE.changeBothText( nowCommand.getChineseName(), nowCommand.getPhase().getName() );
    }
    else {
        //將 LinkedList 轉成 ArrayList 陣列
        String args[] = new String [ argList.size() ];
        runTemporaryCommand( argList.toArray( args ) );

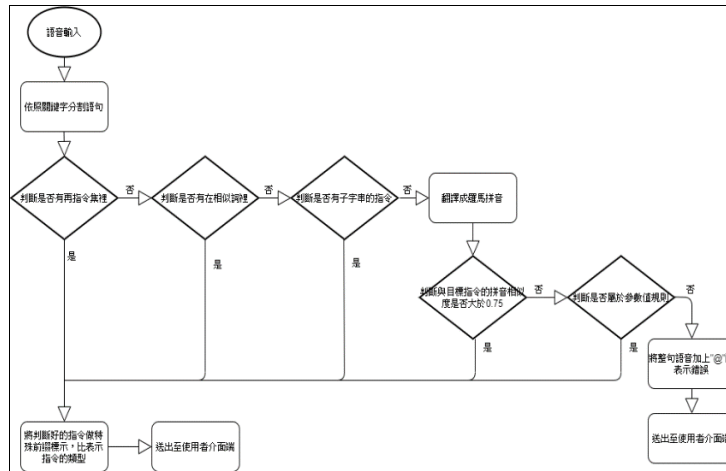
        System.out.println( ConsoleColors.WARNING + "Command Not Found!" );
        argList.clear();
    }
}

//執行指令的地方
runCommand();
```

圖六、JavaFx 端的指令執行程式碼架構圖

六、Python 語音分析架構

由語音接收進來後，將使用者說的話進一步分析。假如分析的指令存在，就會直接發送至 JavaFx 端做執行，沒有的話則會加前綴符號，讓 JavaFx 端可以忽略此輸入



圖七、Python 端的指令分析架構圖

七、Python 語音分析與接收端的指令執行架構

Python 端也會也一個指令集的架構，這裡我們使用 Json 格式來儲存此架構。但是與 JavaFx 端較為不一樣的，我們多了一些屬性來幫助我們可以更快的分析使用者所講的指令，並且可以減少指令誤差值，也能方便的管理指令。而參數需求是根據指令進而變動的，可由此架構進行快速的更改。

```
"RedoCommand": {
  "英文名稱": "Redo",
  "中文名稱": [
    "重作",
    "下一步"
  ],
  "相似詞": [
    "銅作",
    "桶座",
    "重做"
  ],
  "參數需求": []
},
"ImageAlignCommand": {
  "英文名稱": "Image_Align",
  "中文名稱": [
    "圖片對齊"
  ],
  "相似詞": [],
  "參數需求": [
    "對齊"
  ]
},
"ImageRotationCommand": {
  "英文名稱": "Image_Rotation",
  "中文名稱": [
    "圖片旋轉"
  ],
  "相似詞": [],
  "參數需求": [
    "數值"
  ]
},
"ImageMoveCommand": {
  "英文名稱": "Image_Move",
  "中文名稱": [
    "圖片移動"
  ],
  "相似詞": [],
  "參數需求": [
    "方向",
    "數值"
  ]
},
},
```

圖八、Python 端的指令集

八、Python 語音分析程式碼架構

```
# =====
# 判斷使用者的輸入是否有指令
inputMsg_split = inputMsg.split(" ", -1)
firstText = getKeyWord(inputMsg_split[0], dict.keys(COMMAND_MAP))
print("使用者輸入：", inputMsg )
# =====
# 開始判斷 inputMsg 的 "字詞" 和 "相似詞" 是否包含指令
if firstText in dict.keys(COMMAND_MAP):
    print(inputMsg + " 找到字串 " + firstText )
    # 送出訊息
    SendingMsg(clientSocket, firstText, inputMsg_split)
    raise CommandFound('CommandFound occured')
# =====
# 開始判斷 inputMsg 的羅馬拼音
else:
    #翻譯羅馬拼音
    word_Romma = extend_pinyin( firstText )
    for key in dict.keys(COMMAND_MAP):
        cmd = COMMAND_MAP[key]
        if( cmd.judgege_Romma( firstText, word_Romma ) == True):
            print(inputMsg + " 找到相似詞 " + firstText )
            # 送出訊息
            SendingMsg( clientSocket, cmd.key, inputMsg_split )
            raise CommandFound('CommandFound occured')

#===== 垃圾回收中心 =====
# 假如都沒有執行就去判斷垃圾
if( doCommandAction(clientSocket, __DO_NOTHING__, inputMsg) == False ):
    msg = get_Natural_Command( inputMsg )
    if( msg != "" ):
        SendingNaturalMsg( clientSocket, msg )
        # 假如連垃圾回收中心都救不了他的話就直接送出
    else:
        temp_SendMsg = '#' + inputMsg + " ----- 無法辨識" + '\n'
        temp = clientSocket.send( temp_SendMsg.encode() )

# =====
# 找到"預設字串" "相似詞" 裡有包含 inputMsg 此字串
except CommandFound as e:
    print("使用 ", inputMsg, " 找到指令")

# =====
# "判斷不到使用者說的話" 或 "使用者沒有說說話"
except NoSpeech as e:
    print("")
```

圖九、Pyhton 端的指令執行架構程式碼

肆、作品原理

一、作品提供

A. 解析語音詞彙

為了能在語音控制時更接近人的講話習慣，需要做剖析語音詞彙，以達到更人性化的語音控制輸入。因為我們的程式要做執行動作都是透過口語輸入指令，這時使用者並不會把每一個指令都記得非常熟，所以需要根據我們預設的指令庫與使用者輸入去比對與修正，判斷使用者想要正確執行的動作是哪一個。

B. 執行語音/文字的指令動作

處理當使用者語音輸入而需要執行的動作。程式會根據使用者輸入的指令而做下一個指令的執行或接收。

C. 對使用者的輸入除錯

因我們程式有提供語音輸入，而在人講話的時候難免會有講錯的的時候，這時就要去處理使用者所輸入錯誤的情況，像是數值的範圍超過、未找到相對應的名稱等等，這些訊息除了修正以外還要回饋給使用者，讓使用者知道是哪裡輸入錯誤。

二、程式技術

A. 使用語言

程式語言	功能名稱	功能說明
Python	Speech_Recognition	Python 提供的套件，能辨識使用者的語音，轉換成正確的詞彙。
	Socket	用來與 Java 端進行資料傳輸。
	Pinyin	根據 Mandarin.dat 將中文字符翻譯為羅馬拼音。
	GTTS	gTTS (Google 文本到語音)，一個 Python 庫和 CLI 工具，可與 Google Translate 的文本到語音 API 交互。將語音 mp3 數據寫文件，以進行進一步的處理。或者只是生成 Google Translate TTS 請求 URL，以將其提供給外部程序。
	NLTK	自然語言工具包，提供許多解析自然語言的函式庫。
Java	JavaFX	最主要的 UI 設計，並且可以移植到手機平台上的跨平台語言。
	CSS	Javafx 支援的樣式表，可以簡單的修改 UI 顯示樣貌。
	Selenium	提供無 UI 瀏覽器，能抓取並執行網頁的 JavaScript，好讓我們去剖析網頁資料。
	Jsoup	用於解析、提取 HTML 的開源 Java 函式。
	Socket	用來與 Python 端進行資料傳輸。

B. 軟體應用之演算法

演算法	功能說明
<p>布雷森漢姆直線演算法 (英語：Bresenham's line algorithm)</p>	<p>是用來描繪由兩點所決定的直線的演算法，它會算出一條線段在 n 維點陣圖上最接近的點。這個演算法只會用到較為快速的整數加法、減法和位元移位，常用於繪製電腦畫面中的直線。是計算機圖形學中最先發展出來的演算法。</p>
<p>洪水填充演算法 (英語：Flood fill)</p>	<p>Flood fill 算法是從一個區域中提取若干個連通的點與其他相鄰區域區分開（或分別染成不同顏色）的經典算法。因為其思路類似洪水從一個區域擴散到所有能到達的區域而得名。在 GNU Go 和掃雷中，Flood Fill 算法被用來計算需要被清除的區域。</p>
<p>最長公共子序列 (英語：Longest Common Subsequence; LCS)</p>	<p>最長公共子序列 (LCS) 是一個在一個序列集合中（通常為兩個序列）用來尋找所有序列中最長子序列的問題。這與尋找最長公共子串的問題不同的地方是：子序列不需要在原序列中占用連續的位置。在此軟體中用來判斷句子中使用者可能想說的指令。</p>
<p>編輯距離 (Levenshtein Distance)</p>	<p>是指兩個字串之間，由一個轉成另一個所需的最少編輯操作次數。許可的編輯操作包括將一個字元替換成另一個字元，插入一個字元，刪除一個字元。</p>

伍、系統需求

一、系統功能需求

功能需求編號	功能需求	描述
DBS-F-001	基本繪圖軟體工具	畫筆 / 橡皮擦 / 直線 / 矩形 / 圓形 / 橢圓形 / 填滿 / 文字
DBS-F-002	基本繪圖軟體功能	復原 / 重做
		存檔 / 讀檔
		輸出圖檔 PNG / JPG
		畫框大小設定 / 工具顯示或隱藏
DBS-F-003	基本圖片動作功能	透明度 / 旋轉 / 翻轉 / 刪除 / 新增 / 顯示 / 移動 / 長寬設定
DBS-F-004	圖層管理功能	移動 / 新增 / 刪除 / 透明度 / 長寬設定
		畫布預覽，當使用者對圖層進行操作時，可以預覽此圖層的改變。
DBS-F-005	圖片物件管理功能	重新命名圖片物件，可以使用語音來選擇此圖片。
		圖片標籤，對此圖片的分類。比說樹木是屬於大自然的，這時候使用者只要說『大自然』就可以選擇到有關於這個描述的圖片。
DBS-F-006	網路擷取無版權圖片	使用者可以使用語音從網路上新增圖片至此軟體，並且不受版權限制。
DBS-F-007	本地圖片新增	本地圖片新增 使用者可以選擇從電腦裡的本地圖片，新增至畫布裡。
DBS-F-008	基本語音辨識功能	當使用者說出語音指令時要做相對應的動作。
DBS-F-009	進階語音辨識功能	使用者不一定需要將指令完整的講出，程式也能正確的執行。
DBS-F-010	同音字指令檢查	同音字指令檢查 當使用者說出發音很像，但是字不正確時就可以正確的判斷到。例如使用者說『復原』時，假如語音辨識判斷出『復元』，即可找出正確的指令。

DBS-F-011	語音回饋系統	語音回饋系統將使用者語音輸入時的訊息或錯誤回饋給予使用者，好讓他們做修正。
DBS-F-012	開發人員除錯欄	開發人員除錯欄提供以文字代替語音的方式輸入，以方便測試人員做語音動作測試，並且也能提供給一般使用者做使用。
DBS-F-013	自然語言分析	自然語言分析當使用者講出『樹下有一隻貓與一隻狗』時，軟體會產生一顆樹、一隻貓與一隻狗，並且會以描述的樣子放置在畫布上。
DBS-F-014	人臉模式	當使用者選擇此模式的時候，此軟體會提供臉部定位，供使用者去擺放五官。

二、系統非功能需求

非功能需求編號	非功能需求描述
DBS-NF-001	軟體使用的配色須以暗色系為主，以減少長時間使用此軟體造成的眼睛不適。
DBS-NF-002	軟體介面布局需與大部分軟體相似，以減少使用者在熟悉軟體上所耗費的時間。
DBS-NF-003	在更動視窗大小時，不會造成版面混亂與變形。
DBS-NF-003	在以像素為 720x640 時圖層數小於 10 個，並且圖片物件也小於 10 個時，程式之使用記憶體不超過 512MB。
DBS-NF-004	在使用者以非常快速的在畫布上進行更動時，使用 GPU 做運算並且平均不高過 30%(GTX 1050 Ti)，並且不會卡頓。
DBS -NF-005	在使用者以非常快速的在畫布上進行繪圖時，滑鼠與畫筆的延遲不超過 100ms，並且不會造成快速繪圖時所造成的空洞。
DBS-NF-006	在使用者在網路上請求圖片時，每次與網站請求平均時間不超過 5 秒，並且再次請求同一圖片時必須在 2 秒內回應給使用者。
DBS-NF-007	在使用者講出語音指令後，判斷指令並且執行結束的時間必須在 1 秒內結束。
DBS-NF-008	在使用者重複講述同一指令時，不必再講一次指令。例如：『圖片旋轉 50』接下來使用者只要講『180』當前選擇的圖片就會順時針旋轉至 180 度。

DBS-NF-009	在語音辨識到有同音或發音相似的字詞時，必須要能辨識得到，並且再次講出時，不必再重新做辨識。
DBS-NF-010	以 1080x720 的畫布時，在輸出成圖片後，不會造成嚴重且明顯的失真。
DBS-NF-011	在專案存檔與開檔的過程中，圖片不會有嚴重且明顯的失真情況。
DBS-NF-012	使用者端的每一指令架構採取獨立性的類別，以減少每個指令的耦合，並且要具有良好的可擴充性、可維護性。
DBS-NF-013	在 Python 端的語音辨識系統的語音辭庫，要使用 Json 或 YMAL 來做為指令集的語詞辭典，以增加可維護性。

三、使用者案例

使用案例編號：DBS-UC001		使用案例名稱：語音控制圖片選轉	
動作描述	使用者使用語音控制圖片旋轉 90 度		
系統目前狀態	監聽指令中		
先前使用者動作	已經選擇了一張圖片，但是沒有做任何動作過		
系統反應動作		使用者操作動作	
a. 等待語音指令			
		b. 使用者說：『圖片旋轉 90 度』	
c. 接收語音			
d. 語音轉換成文字字串			
e. 判斷是否是指令			
f. 送出編碼後的指令至使用者端			
g. 根據傳送過來指令將狀態切換至〔旋轉圖片〕			
h. 再接收到後面的 90 數值進而將圖片選轉 90 度			
i. 狀態切回等待語音指令			
		j. 使用者看到圖片以旋轉成功	

使用案例編號：DBS-UC002		使用案例名稱：替換圖片動作
動作描述	使用者使用語音替換圖片	
系統目前狀態	監聽指令中	
先前使用者動作	已經選擇了一張圖片，圖片為一棵『樹』，但沒有做任何動作。	
系統反應動作		使用者操作動作
a. 等待語音指令		
		b. 使用者說：『替換圖片為貓』
c. 接收語音		
d. 語音轉換成文字字串		
e. 判斷是否是指令		
f. 送出編碼後的指令至使用者介面端		
g. 去抓取網路圖片庫有關於「貓」的圖片，並加上編號於每一張圖片。		
h. 顯示關於〔貓〕的圖片		
i. 並將目前系統狀態切換至〔替換圖片-取得編號〕		
		j. 使用者在介面右側看到每一張關於「貓」的圖片
		k. 使用者說：『A5』
l. 語音接收到編號 A5		
m. 傳至使用者介面端		
n. 從網頁裡抓取有關於編號 A5 的圖片標籤		
o. 新增至畫布		
p. 狀態切回等待語音指令		
		q. 使用者看到圖片已替換成功

使用案例編號：DBS-UC003		使用案例名稱：使用者新增組合圖形
動作描述	使用者使用語音新增複雜圖形	
系統目前狀態	監聽指令中	
先前使用者動作	畫布為空，沒有做任何動作	
系統反應動作		使用者操作動作
a. 等待語音指令		
		b. 使用者說：『樹下有一隻貓』
c. 接收語音		
d. 語音轉換成文字字串		
e. 判斷是否是指令		
f. 分割文字字串，並且定義出名詞與物件位置		
g. 送出編碼後的指令至使用者介面端		
h. 去抓取網路圖片庫有關於「樹」的圖片。		
i. 隨機選擇一張關於「樹」的圖片。		
j. 取得關於此圖片的標籤		
k. 新增「樹」圖片至畫布		
		l. 使用者在畫布看到「樹」的圖片
m. 再去抓取網路圖片庫有關於「貓」的圖片		
n. 隨機選擇一張關於「貓」的圖片。		
o. 取得關於此圖片的標籤		
p. 由分割過後的指令，判斷出「貓」的圖片該擺放的位置。		
q. 新增「貓」圖片至畫布		
r. 狀態切回等待語音指令		
		s. 使用者看到『樹下有一隻貓』的圖片

陸、作品特色

一、作品應用類型與範圍

A. 使用軟體年齡範圍（2+）

1. 幼年時期在牙牙學語的階段，可以利用這軟體練習母語，且發揮創造力與想像力。
2. 中年人在於一些職業需求，總會需要臨時設計一些插圖，而這就可以利用我們這軟體快速的製作出插圖或一些設計圖。
3. 老年人也能使用這軟體，在這些族群中，有些老人可在手部操作滑鼠會不方便，想要表達自己的想法但是又難以繪製，這時就可以利用我們的軟體來幫助他們來達到目的。

B. 作品應用

1. 提供某些職業在於設計與繪圖的方便性
2. 提供對於某些軟體的延伸應用
3. 提供一般民眾的使用
4. 提供某些硬體的功能延伸功能

二、特性與設計重點

A. 特性

由於最底層技術是使用 JavaFX，而 JavaFX 的特性就是易於擴充，容易將程式於其他平台上執行，而且我們的軟體架構有將他分的很開，盡量能方便使用者去擴充他們的功能。尤其是指令的架構，我們採用一種特殊的架構能使我們的指令易於擴充，且能接收其他的命令。

除此之外，我們設計的系統，使用了多層次的架構，當上層被更動時，底層的物件也不會受到影響。並且我們也使用了指令集架構，能快速且方便的修改指令功能，也不會與其他指令互相的干擾，而指令的狀態是使用有限狀態機的方式，當在甚麼樣的狀態就只能傳入指定的參數，而其他參數將會被遺棄，這樣不但不會與其他指令屬性互相干擾，也能增加指令屬性在判斷時候的速度

B. 設計重點

由於我們的專題是純軟體，並且擁有使用者介面，所以我們有參考許多的版面配置，當我們的使用者界面擁有良好的版面配置，也能讓使用者能輕鬆駕馭與獲得良好的使用體驗。而版面的配色我們也有經過一些研究與測試，看哪樣的配色比較不會使眼睛疲勞或太過於花俏，而影響使用者的體驗。除此之外，我們也有著重於減少記憶體與 CPU 的消耗，為了能使這個應用程式能相容在許多平台上，也不會造成硬體上的負擔。

我們致力於將指令變得好使用，且人性化，當使用者執行指令的時候，會將目前指令移入暫存，這樣當使用者下次再講類似的參數時，就可以直接備暫存的指令執行，不用再說指令的功能。

三、創意及原創性說明

A. 創意：

1. 繪圖軟體結合語音辨識，提高使用族群範圍，增加繪圖軟體使用的便利性。
2. 使用開源網站的圖庫，讓不擅長畫畫的人或急需快速構圖的人，不需要冒著可能會侵犯著作財產權的風險上網抓圖片來使用。能夠安全並且快速地使用這些開源圖片，並且不會侵犯著作權。
3. 繪圖軟體結合文字指令控制台，輸入指令就可以讓繪圖軟體執行某部分的功能，讓中小學生在繪圖的同時體驗到程式設計的樂趣，甚至可以自行DIY創意繪本，達到簡易創作的目的。
4. 易於擴充的軟體架構，能根據不同使用者的需求，而將軟體進行修改，也因為我們採用多層次架構，所以當進行前端改動時，也不會影響底層架構。

B. 原創性

當前市面上的繪圖軟體 PhotoshopCC、PaintTool SAI、Paintstorm Studio、Corel Painter2020.....以及 windows 的小畫家，都沒有支援聲控的方式來編輯圖片以及繪圖。並且，繪圖軟體所使用的圖片物件是自身圖庫中的圖片，要等到開發者更新圖庫的物件才能夠使用新的圖片物件，而我們是利用 Java 提供的 Selenium 及 Jsoup 去剖析提取 OpenData 的網頁資料，OpenData 的圖庫會隨著不同人提供圖片，隨時更新並增加能夠使用的圖片物件，因為圖片的來源是開源 OpenData 的關係，所以不會有侵犯著作權的疑慮。

C. 原創 LOGO 設計

由創作者的姓名英文字首組合，並融入畫筆代表程式的主要功能，貓元素的組合是神秘的象徵，為了達到勤奮學習的概念目標，將擺放設置為類似書架的外觀，展現出精簡又明確的主題。

四、市場潛力說明

A. 以一般應用為主

快速 UML 製作 - 像是我們的科系，經常需要畫些系統架構圖、實體關係圖、活動圖、狀態圖等等之類的 UML 圖形，當我們在使用其他軟體時，往往需要使用滑鼠慢慢的牽線與拉圖形出來。但是這時候只要將我們的軟體的圖庫與指令改一下，就可以使用語音來快速創建 UML 圖，只要簡單的描述實或物件之間的關係就可以將它們聯繫在一起，以節省使用者更多的時間。

一般軟體使用 - 當手邊的工作忙不過來時，此軟體提供線上撈取 OpenData 的免費開源圖庫，並且透過語音辨識的功能，讓使用者能夠用"說"的方式快速建構出理想中的圖片。

B. 以小朋友為主

圖書繪本製作 - 小朋友可以利用聲控繪圖的方式為他們製作屬於自己的繪本，也可以繪製出自己的圖畫，不但能增加他們的創作度也能讓他們學習語言。對於語言能力有一定基礎的小朋友，他們可以透過講出物件的名稱，來快速搜尋該物件的圖片。

C. 以打擊犯罪為主

犯人臉部建置系統 - 當發生犯罪的時候，攝影機可能並沒有清楚的補捉到犯人的面部，但是有目擊者目擊到犯人的長相。這時候只需要將我們的圖庫置換成人臉的圖庫，讓這些目擊者可以描述出這個犯人長的大概樣子，這不但可以使得判斷更為精準，假如這個目擊者描述得非常詳細，甚至可以用來做人臉 AI 分析，而近一步達到更有效率的抓捕犯罪嫌疑人。

D. 以資訊應用為主

商品搜尋系統 - 我們都知道一間大賣場物品玲瓏滿目，假設我們要找一件物品，自己知道這物品的形狀，但是又不知道名子。這時廠商只需要在他們的系統裡安裝我們的程式，使用者就可以簡單的描述出這個物品的大概，然後再去分析這個物品可能像是什麼，在指引使用者到對應的區塊購買此物品。

柒、操作手冊

一、環境建置

先去 https://www.java.com/zh_TW/download/ 下載 JRE
版本：Java(TM) SE Runtime Environment (build 13.0.2)+ 以上

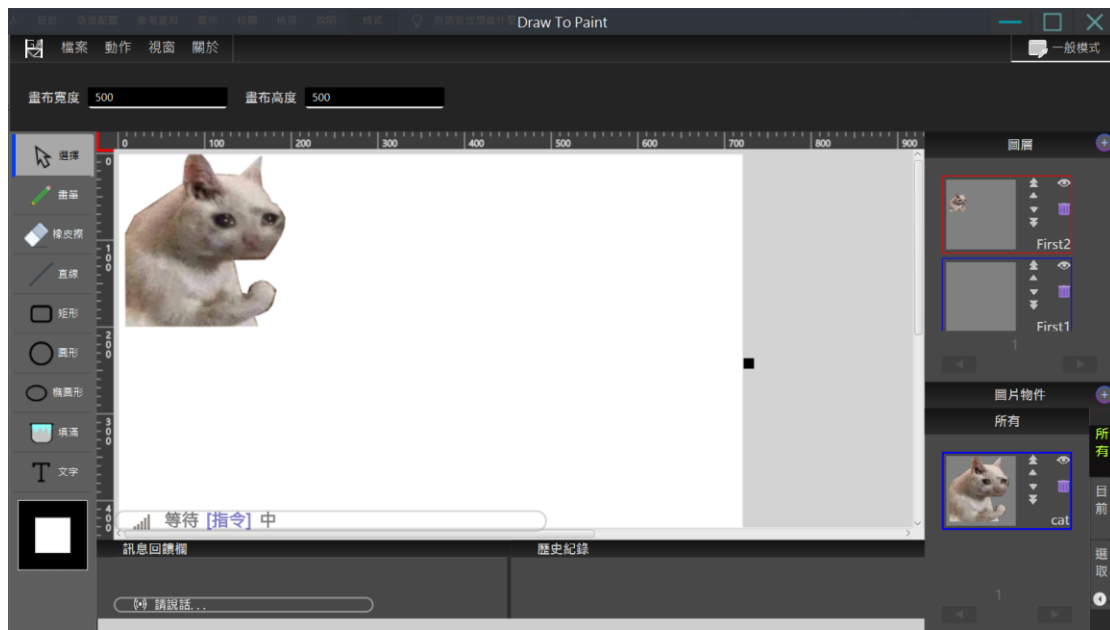


圖十、JRE 環境安裝視窗圖片



圖十一、JRE 環境安裝完成圖片

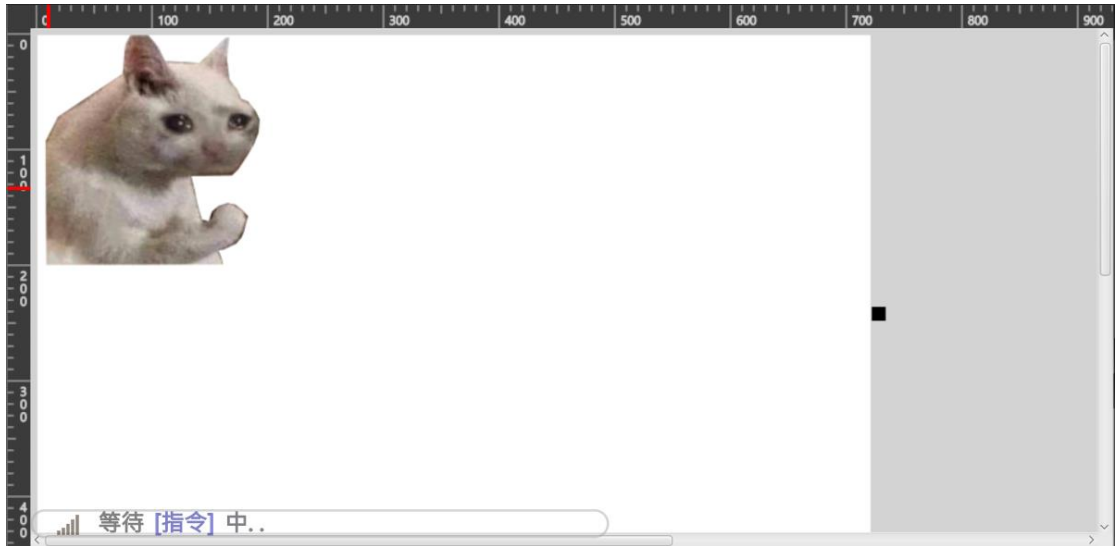
二、使用者介面



圖十二、使用者的 UI 介面



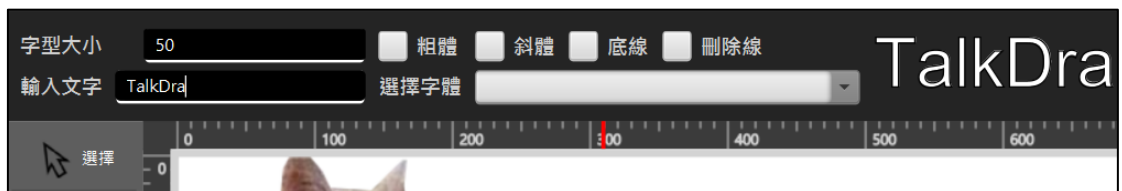
圖十三、圖層與圖片物件介面



圖十四、中間畫部區



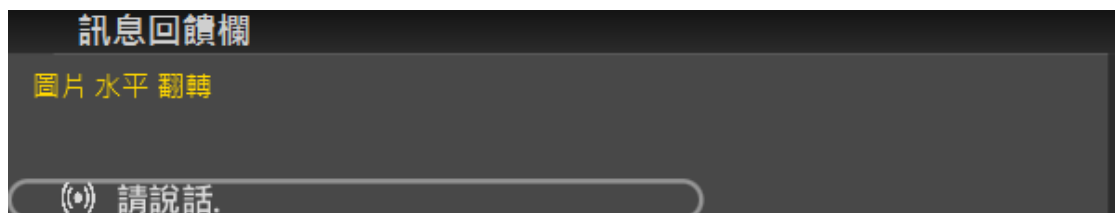
圖十五、工具介面



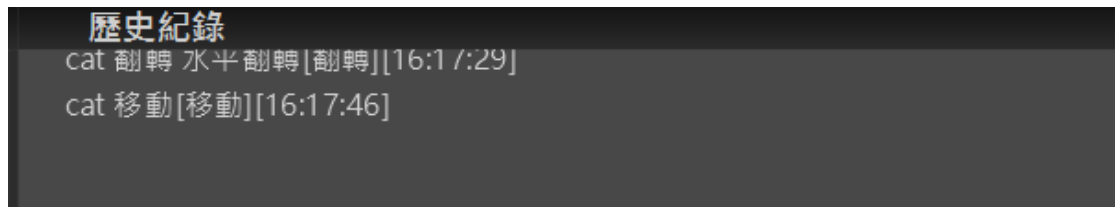
圖十六、工具屬性



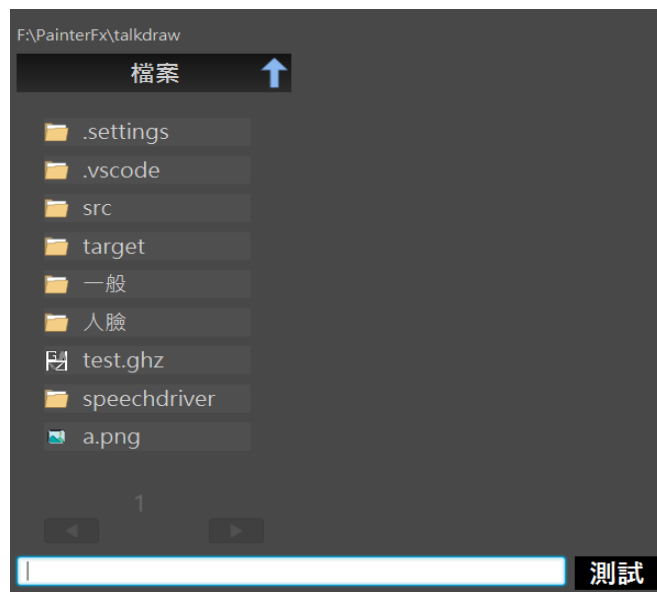
圖十七、工具欄介面



圖十八、訊息回饋欄



圖十九、歷史紀錄欄



圖二十、檔案開啟與儲存欄

三、指令功能表

功能	使用方式	其他參數
改變操作圖層	改變操作圖層 [圖層名稱]	無
改變畫布大小	畫布大小 [長] [寬]	無
選擇圖片	選擇圖片 [圖片名稱]	無
新增圖片	新增圖片 [關鍵字]	無
更換圖片	更換圖片 [關鍵字 編號]	無
從本地新增圖片	本地圖片 [關鍵字]	無
更換本地圖片	更換本地圖片 [關鍵字 編號]	無
更換工具	工具 [工具名稱]	選擇、畫筆、橡皮擦、圓形、橢圓形、矩形、填滿、文字、直線
設定工具細節	工具細節 [細節名稱] [數值]	尺吋、間隔、移動量、圓框、方框、虛線、角落、圓心、誤差值、輸入文字、粗體、斜體、字體
語音繪圖	繪圖 [方向 座標] [數值]	往上、往下、往左、往右
復原	復原	無
重做	重做	無
設定圖片對齊	圖片對齊 [位置]	置中、靠左、靠右、靠下、靠上
更改圖片透明度	圖片透明度 [數值]	無
設定圖片旋轉	圖片旋轉 [數值]	無
設定圖片大小	圖片大小 [數值1] [數值2]	無
設定圖片移動	圖片移動 [方向 數值]	往上、往下、往左、往右
設定圖片翻轉	圖片翻轉 [動作]	垂直、水平
設定圖片順序	圖片順序 [動作]	至頂、至底、往上、往下
查看圖片資訊	圖片資訊	無
設定圖層順序	圖片順序 [動作]	至頂、至底、往上、往下
輸出圖片檔	輸出圖檔	無
儲存專案	儲存專案	無
清空圖層	清空圖層 [確定]	無
設定外框顏色	外框顏色 [顏色]	無
設定內框顏色	內框顏色 [顏色]	無
設定人臉模式	人臉模式	無
設定一般模式	一般模式	無
設定混合模式	混合模式	無
自然語言新增圖片	自然語言新增圖片 [關鍵字] [關鍵字1] [位置1]...	無

圖二十一、功能指令表

捌、系統環境需求與限制

一、最低硬體配置需求

1. 作業系統： windows 10
2. 處理器： Intel Core i3-3210、AMD A8-7600 APU
3. 記憶體： 2GB
4. 顯示卡：
內顯： Intel HD Graphics 4000 (Ivy Bridge) 或
AMD Radeon R5 系列 (Kaveri line)
獨顯： Nvidia GeForce 400 系列 或
AMD Radeon HD 7000 系列
5. 麥克風： 語音輸入裝置

二、環境配置需求

1. 套件： Java(TM) SE Runtime Environment (build 13.0.2)+
2. 網路： 寬頻網際網路連線
3. 儲存空間： 200 MB 可用空間

玖、結論

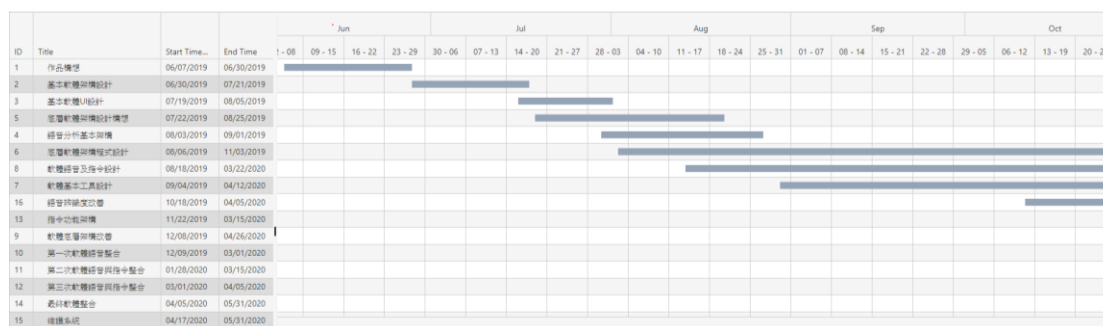
說「畫」在繪圖方面的工具擁有畫筆、橡皮擦、直線、矩形、圓形、橢圓形、填滿、文字方塊等等，還可以更改工具的屬性以及內外框的顏色。相較於 windows 內建的小畫家，我們還擁有旋轉圖片、更改圖片圖層以及透明度的功能。語音方面，我們可以透過語音輸入執行軟體幾乎所有的功能，並且透過 socket 傳輸，甚至可以達到由 A 電腦語音輸入指令至 B 電腦。但語音方面仍然是很容易受到環境聲音的影響，導致偵測容易無法辨識。

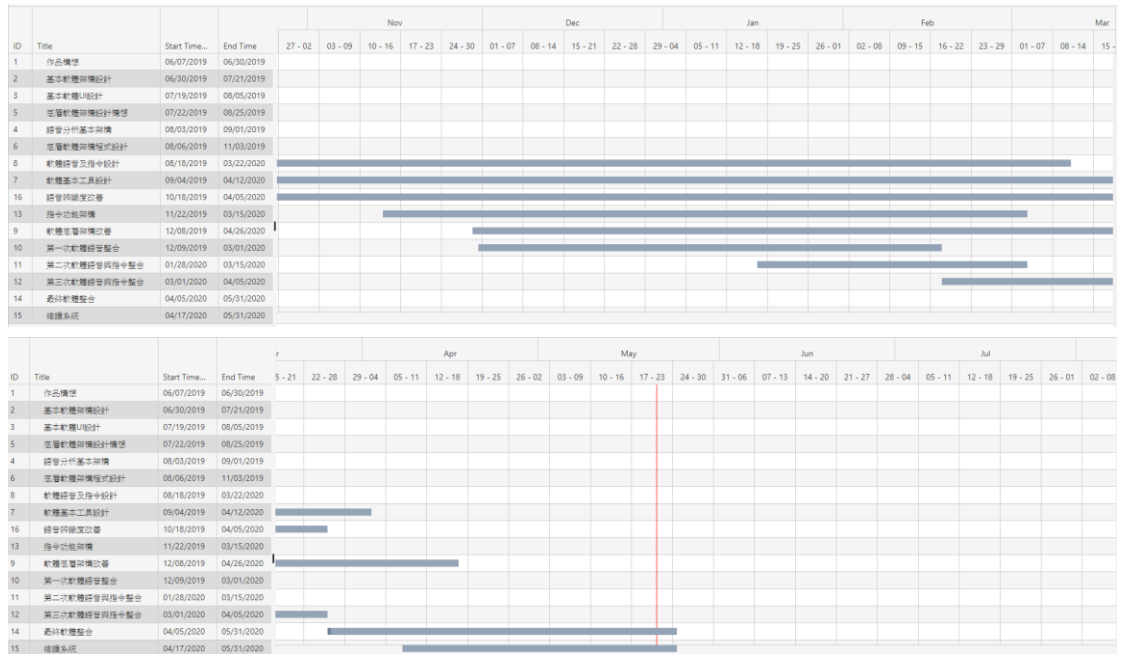
根據我們製作的軟體，雖然目前成品還沒有到非常的完善，但是我們覺得此軟體假如擁有更多樣的指令功能，就能使此軟體的應用達到非常的廣。但是此軟體最需要克服的就是自然語言的處理以及與指令的協調性，當我們在描述一件事物時，都以非常的直觀去描述，但是電腦要處理我們說的這些語言時，電腦難以理解我們所描述的事物，並且須要加以解析，除此之外，解析完之後還要能讓此命令能正常執行，光是這些就足以讓我們感到困難，而這些就是我們未來將要處理的問題。

壹拾、團隊分工

工作內容	描述	負責人
UI/UX 設計	使用者介面設計。	高家祥
語音分析與接收系統語	做自然語言處理以及剖析，將語言作相似度判斷等等。	
軟體底層架構	軟體的核心架構設計與軟體的規劃。	
軟體基本功能	繪圖軟體的基本功能，繪圖工具、檔案讀寫、繪圖操作等等功能。	張壹智
音通訊協定	作使用者端與語音接收端的通訊協定。	
繪圖功能測試	測試基本繪圖軟體功能是否正常，並且是否符合系統要求。	
技術文件撰寫與規劃	撰寫技術文件並且稽核系統是否有達到需求。	韓博丞
語音功能測試	測試語音輸入後在執行使用者介面端上的執行。	
軟體指令動作系統	設計接收語音指令的狀態圖以及動作順序。	

甘特圖：





圖二十二、工作甘特圖

壹拾壹、參考文獻

1. 中文句子相似度之計算與應用 (Chinese Sentence Similarity Computing and Appling) [In Chinese] 鄭守益 梁婷 國立交通大學資訊科學系
<https://www.aclweb.org/anthology/O05-1008.pdf>
2. 基于语义依存分析的句子相似性度量算法及应用研究
https://www.lunwen66.com/papers_detail_7003.html
3. 運用 Python 結合語音辨識及合成技術於自動化音文同步之實作(A Python Implementation of Automatic Speech-text Synchronization Using Speech Recognition and Text-to-Speech Technology)[In Chinese]，賴俊翰 Chun-Han Lai、張朝凱 Chao-Kai Chang、呂仁園 Renyuan Lyu 長庚大學資訊工程學系
<https://www.aclweb.org/anthology/O15-1026.pdf>
4. AiLearning: 机器学习 - MachineLearning - ML、深度学习 - DeepLearning - DL、自然语言处理 NLP
<https://github.com/apache/AiLearning>
5. Python 中拼音库 PyPinyin 的用法
<https://pypi.org/project/pinyin/>
6. 最長共同子序列 (Longest Common Subsequence; LCS)

<https://yungshenglu.github.io/2018/05/15/LongestCommonSubsequence1/>

7. An Evaluation of the Accuracy of Online Translation Systems , Milam Aiken 、 Kaushik Ghosh 、 John Wee 、 Mahesh Vanjani , University of Mississippi
<https://pdfs.semanticscholar.org/9ecd/ae40e20ee31a33f12ab99e1cc67db50534d0.pdf>
8. Selenium Documentation Release 1.0.pdf
http://oss.infoscience.co.jp/seleniumhq/docs/book/Selenium_Documentation.pdf
9. The Selenium official website
<https://selenium.dev/documentation/en/>
10. HtmlUnit official website
<http://htmlunit.sourceforge.net/>
11. jsoup: Java HTML Parser official website
<https://jsoup.org/>
12. Socket Python & Java
https://en.wikipedia.org/wiki/Network_socket
13. JavaFX official website
<https://openjfx.io/>
14. Package javax.swing website
<https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>
15. Json
<https://www.json.org/json-en.html>
16. Natural Language Toolkit
<https://www.nltk.org/>
17. JavaFx
<https://openjfx.io/>
18. FXML
<https://en.wikipedia.org/wiki/FXML>
19. YMAL
<https://en.wikipedia.org/wiki/YAML>
<https://yaml.org/>
20. 布雷森漢姆 Bresenham's line algorithm 直線演算法
https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm

壹拾貳、心得

學生 張壹智 的專題製作心得：

當我們討論出這個專題的題目時，我認為只要將現成的語音套件與繪圖軟體作簡單的結合就好了，但是在開始的時候就碰到了很重要的問題，要用什麼語言來製作？在接觸了語音套件的相關訊息後，我們得出了結論，使用 python 作為處理語音的程式語言，在以 java 作為 UI 設計的主體，兩者之間在以 socket 傳輸資料，這就是我們的主體架構了。

整個專題中，我主要負責繪圖工具以及 UI 介面設計，首先的難點就是圖層的管理，以不同的圖層來作疊加，這是電腦繪圖最好用的地方，實作起來卻很有挑戰，再來就是【油漆桶】的工具，我使用了【Flood fill Algorithm】來提高效能，這個演算法是為了找到同一個顏色的實際面積位置，除此之外，我增加了誤差值來更人性化的填滿顏色，另外，繪圖的基礎功能我採用 javaFx 的現有函式，甚至對他作了一些修改，以符合我們的需求。

接下來就是 UI 介面的設計，在 java 程式中，預設的 UI 介面總是沒有辦法更有個性的表現出這個軟體的風格，所以後來我們使用 javaFx 來優化它，甚至撰寫專屬於 javaFx 的 css 來調整顯示的樣式，當然，這也提高了製作的難度，不過也大大的提升了視覺完整度，讓使用者能夠更快的適應編輯環境，不會因為初次使用而找不到需要的功能在那裡，這也讓語音控制可以更好的發揮，以簡易的指令就能達到精確的功能，因為人性化的名稱與指令，使用者甚至不用預先知道指令有甚麼就可以正確的操作程式達到想要的功能，這樣就不用為了使用它而另外花時間學習，可以說使用入門沒有門檻，只要打開它就可以很直覺的使用。

完成這個專題，我學到了非常多上課沒有教的東西，包含新的程式語言、演算法的應用與修改、監聽器與執行緒的各項運用等等，雖然上課時有提到，但果然還是要自己摸索才會更加瞭解可以運用的效果，還有團隊合作時的程式撰寫，若沒有好的註解與溝通，多人撰寫一個程式或許根本做不到吧。最後，參考其他人的意見也非常重要，當自己沒有好的方法時，多問一些人來找到最符合需求的方法，或許本來覺得很難做的功能也可以輕鬆的完成呢。

學生 韓博丞 的專題製作心得：

一開始在想專題要做什麼題目時我們這組是最後一組才想出來要做什麼的。原本的構想是可以透過使用者的描述來畫出一個情境，例如：「山下有一條河」、「樹上有一顆蘋果」、「有一個人臉型是瓜子臉、頭髮是光頭、眼睛是單眼皮、耳朵普通、鼻子小小的、嘴巴大大的」程式就會將我們所講的情境畫出來。但在我們進行可執行性的評估時發現要實現上述所講的功能，必須要剖析自然語言，而自然語言又屬於人工智慧的部分，這使得我們要實現這個功能的難度大幅度的提升了，因此雖然無法達到所期望的功能，但我們有將一部份自然語言的功能實現，例如：「樹下有一隻貓和一隻狗，樹上有一隻鳥」，我們可以確實的將物件擺在正確的位置。

我所負責的部分是語音辨識、語音回饋與自然語言剖析，當初我所使用的是 C++ 來撰寫程式，使用的套件是 Minecraft Azure 的語音辨識 API，但這個套件並不是免費的，所以後來就去學習 Python，使用他們的套件 Speech Recognition，在剛使用 Python 時有許多不習慣的地方，因為 Python 是利用縮排來區分程式段落，而其他的程式語言大多都是用括號來區分程式段落；Python 的變數不需要宣告資料型態，而其他的也大多數需要。但 Python 在免費的模組套件上資源非常的豐富，因此來程式就改用 Python 來撰寫。

在製作此專題的過程我從中學到很多東西，在一個團隊中每個人的工作都不盡相同，如何將彼此所撰寫的程式結合在一起，並且也要思考在撰寫程式時的架構，像是寫出模組化的架構，降低程式之間的耦合、提高程式的獨立性。並且在設計某個地方的架構時可以與隊友一起討論，以及碰壁時也可以和隊友思考解決問題的方法，Debug 時自己一個人可能想很久也解決不了，但換一個人看的話說不定一下就看出 Bug 出在哪裡，而且也可以向隊友學習程式撰寫的風格、方式以及技巧，藉此來提升自身的程式撰寫能力、團隊溝通能力、除錯能力、解決問題的能力。

學生 高家祥 的專題製作心得：

當我們在構想此專題時，我們以為會非常的簡單，認為就只是接收使用者的語音然後做相對應的動作。結果，我們在第一版的時候是使用 Java Swing 當底層架構，發現再編寫程式碼時，非常的艱苦，因為許多的功在 Swing 中非常難達到。而且一開始我們也沒有使用多層次的系統架構管理，所有的物件、類別等等都全部塞在一起，變數也沒有別的封裝，這導致了我們在第一版時，感到非常的挫折。

後來我們意外發現了 JavaFx，一開始我們還很猶豫要不要將底層轉過去，因為轉過去之後，勢必又是個大改。我們大約做了兩個禮拜的評估，決定轉過去了，順便將原本的系統架構統一整理，並且將物件都進行了封裝，以減少系統的耦合。我們也特別去研究一些關於註解的指令以及一些建立大型系統的架構設計及管理，我們連系統的效率與記憶體有考量到，以增加使用者的體驗。

而且我們在設計的時候，「指令」是一個非常難設計得架構，我們前前後後改了3次以上，就為了能讓指令靈活好用且易修改。最後選用了指令集架構，將指令統一管理，並且能快速的修改指令。

藉由此專題，我學到了很多東西，像是如何將系統做分類，並且有良好的封裝、與其他隊友相互修改系統程式碼而建立修改日誌、學習註解的方法、減少 Dirty Code 的發生、如何建立易於擴充的架構和減少系統的耦合性等等。這些都是學校學不太到的知識，都是需要經過撰寫中型或大型系統才會比較會遇到的問題。而這些問題都是以後出去工作時常會遇到的問題，所以我覺得這個專題真的讓我受益良多，讓我知道了我哪個部分需要改進，以增進自己的程式撰寫能力。

國立虎尾科技大學
資訊工程系

專題製作報告

說「畫」

109

學年