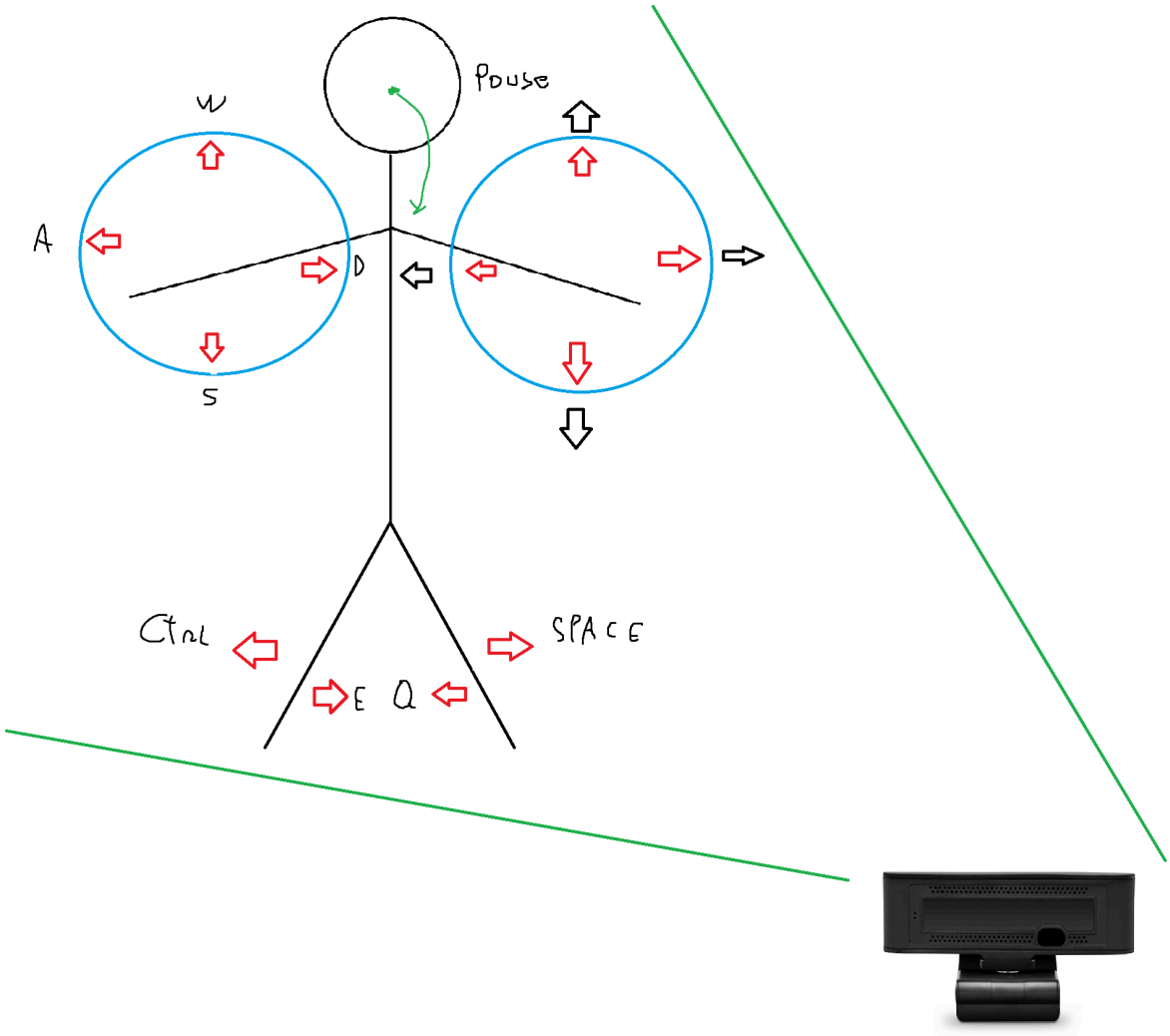
Interfaz Instintiva

Controlador especializado en Aplicación



Mario Pérez Carmona

04/01/2025

Índice

1. Introducción

Este proyecto busca desarrollar un pequeño ejemplo de aplicación que aproveche la cámara para ofrecer un control adaptado a contextos específicos, en lugar de proporcionar una solución generalista.

Uno de los motivos más interesantes para el desarrollo de esta aplicación es tener la capacidad de interactuar con las aplicaciones de manera natural y eficiente.

En otras palabras, crear un enfoque en la interacción humano-computadora, mucho más específica y personalizada del software y la adaptabilidad a diferentes entornos.

La tecnología actual a menudo no está completamente adaptada para personas con ciertas discapacidades físicas o limitaciones, este proyecto propone una idea muy generalista que puede ser explorada incluso más en profundidad para abarcar aún más situaciones.

En resumen, crear una herramienta que no solo sea funcional sino también intuitiva, permitiendo un manejo natural mediante movimientos simples.

2. Objetivo de la Propuesta

El proyecto tiene como principales objetivos:

1. Crear una propuesta simple de control específico sobre una aplicación compleja
2. Esto permite sobre el tiempo revisitar estos mismos casos para situaciones más complejas. Como ciertas discapacidades específicas que impidan el movimiento de las piernas
3. A la hora del funcionamiento del proyecto, crear una aplicación que permita abordar todos los casos posibles de la aplicación y que demuestre flexibilidad para casos más complejos.

3. Herramientas Utilizadas

Dentro de las herramientas usadas en el proyecto se encuentra varias librerías, las cuales se pueden agrupar en 4 grupos:

1. Librerías del Procesamiento de Imágenes
    - **cv2 (OpenCV)**: Librería fundamental para el procesamiento de imágenes. Su trabajo principal será tomar la cámara
    - **numpy (NumPy)**: Esencial para el manejo eficiente de arrays y matrices. En este caso se utiliza para la creación de una opción de debugging.
    - **ultralytics YOLO**: Librería principal en la que se sustenta el proyecto, YOLO (You Only Look Once), modelos usados para la detección de objetos en tiempo real. Servirá para detectar el esqueleto del usuario
2. Interacción con el Sistema Operativo y Gestión de Ventanas
    - **pygetwindow (gw)**: Permite interactuar con las ventanas del sistema operativo, obteniendo información sobre las ventanas abiertas y permitiendo manipular su estado o propiedades.
    - **win32gui, win32con**: Parte del paquete PyWin32, estas librerías permiten la interacción a bajo nivel con la API de Windows.
    - **tkinter (tk)**: Librería para la creación de interfaces gráficas de usuario. Se utiliza para crear un pequeño menú de ejemplo
    - **ctypes**: Biblioteca que permite interactuar con código de bajo nivel en operaciones de sistema.
3. Utilidades Matemáticas
    - **time**: Utilizada para manejar operaciones relacionadas con el tiempo.
    - **math**: Provee funciones matemáticas básicas necesarias para cálculos geométricos y de ángulos, usado bastante.
4. Programación Concurrente y Multihilo
    - **threading, Thread**: Utilizada para manejar la ejecución concurrente de múltiples procesos o tareas. Se explicará en más detalle después

Cabe destacar que el codigo esta hecho sobre python

4. Planteamiento

Está sería la parte más importante de todo el proyecto ya que se debe entender que se va a hacer, ya que incluso si se piensa que esto es algo pequeño realmente se necesita bastantes cosas. Se dividirán en grupos que se desarrollarán poco a poco.

1. Cálculo de puntos establecidos
2. Control de aplicación
3. Cálculo de dirección en base a puntos

4.1.- Cálculo de puntos establecidos

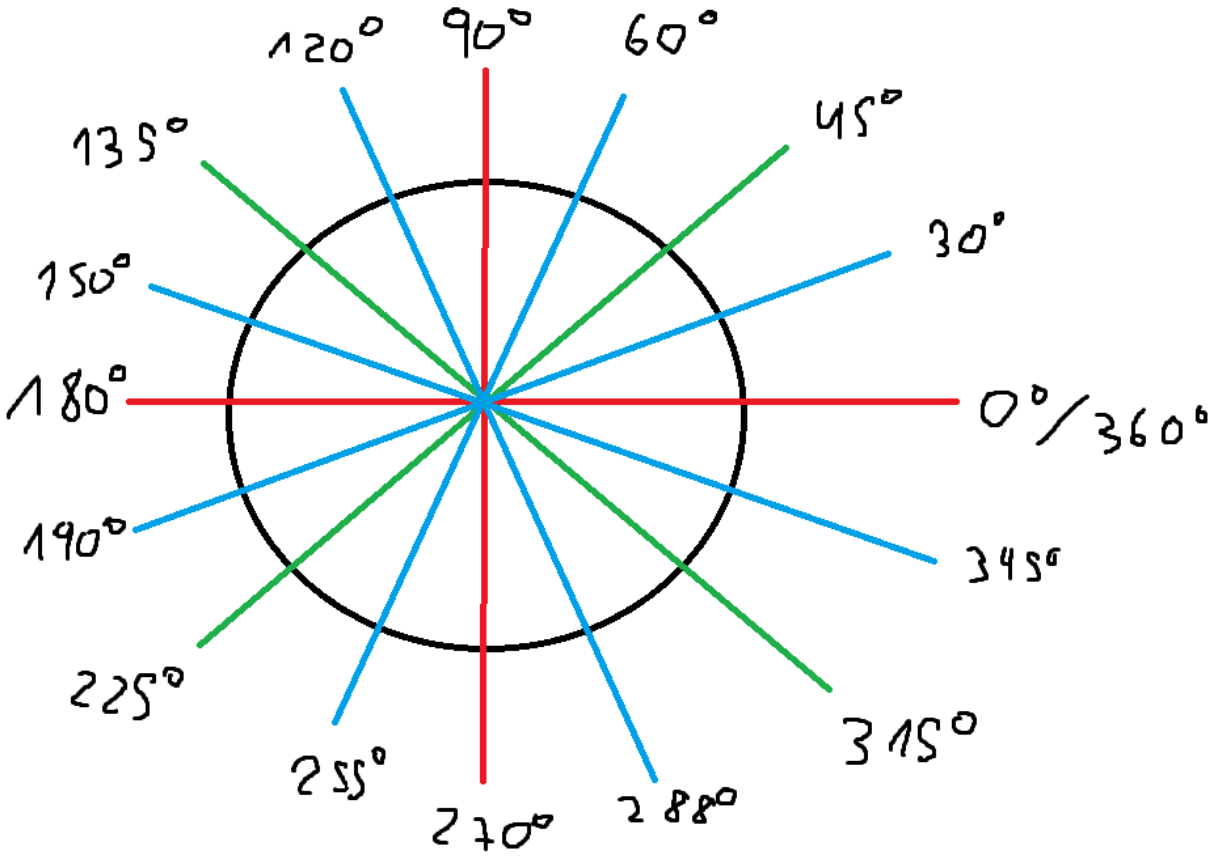
Lo primero es entender que se necesita realmente hacer. Para esta aplicación se necesitan 13 movimientos posibles, que si se categorizan sale:

1. **w, a, s, d** para movimientos básicos.
2. Flechas **arriba, abajo, izquierda, derecha** para navegación direccional.
3. **Ctrl, q, e, Space** para acciones especiales.
4. **esc** para pausar.

Cada grupo debe desarrollarse de manera independiente.

Para poder abordar los dos primeros grupos se puede utilizar ambas manos para ello, utilizando un cálculo de ángulos entre los hombros, los codos y las muñecas determinando el ángulo entre dichos puntos y ajustándose en un eje de coordenadas.

1. En el caso del primer grupo el cálculo iría los puntos situados en la mano izquierda del hombro al codo, esto forma una recta de la cual se puede primero encontrar el ángulo en radianes y luego convertirlo a grados para que sea más fácil de trabajar. Luego se aplica el mismo método para la distancia entre el hombro y la muñeca lo cual otorga una subdirección, esto permite generar ángulos como “Arriba-Izquierda”
2. El segundo grupo reutiliza el código del primer grupo para generar direcciones pero con el brazo derecho, la diferencia es que aquí no es realmente necesario ya que mientras que w, a, s, d debe poder elegirse en diagonales, las flechas son siempre rectas, su reutilización son mero ahorro de código



Este sería una representación en ángulos, de las posibles opciones que disponen para usar en cada grupo.

El color Rojo muestra las posiciones cardinales, y muestra las posiciones cardinales, utilizadas sobre todo por los brazos, para determinar las direcciones rectas.

El color Verde muestra las posiciones que forman los límites entre ser una dirección u otra.

El color Azul muestra las posiciones de las regiones de las subdirecciones

Dicho todo junto sería la roja marca que dirección es, la verde marca las bandas los límites donde se considera ese valor, mientras que la azul determina dentro de ese rango para su subdirección, por ejemplo Arriba-derecha.

Existe un problema de usar las manos para ambos cálculos y es que no siempre se necesita la utilización de estás teclas, esto provoca necesitar un quinto movimiento, un valor neutro que no mueva nada.

Para sacar ese valor, se saca la distancia que existe entre el hombro y la muñeca y la distancia entre el hombro y la cadera, esto genera una comparación simple en el que si la distancia hom-muñ es menor que la distancia (muñ-cad * 0.8), lo considera un valor neutro. El 0.8 se aplica ya que YOLO detecta las caderas de una manera que las hace poner un poco más bajas de lo que debería.

3. En el tercer grupo se utilizará el cálculo de las piernas, donde cada pierna puede dar 3 valores, neutro que sería pierna recta y no haría nada, izquierda y derecha, lo que deja 4 combinaciones
    a. Pierna Izquierda - Izquierda: Ctrl
    b. Pierna Izquierda - Derecha: E
    c. Pierna Derecha - Izquierda: Q
    d. Pierna Derecha - Derecha: Space
4. Para el esc se necesita algo que no utilice lo ya usado y vaya independiente y será el pause. Para ello se usará la nariz, la cual se calculará su posición en el eje Y con los hombros izquierdo y derecho, si la nariz está por debajo en el plano vertical de ambos hombros, se considera como parado, mientras esté en ese estado no se registrará ninguna otra acción.

4.2.- Control de aplicación

Ahora toca tener la capacidad de controlar la aplicación deseada, para ello se utiliza varias funciones, las cuales permiten tomar el control del programa designado, estás serían:

1. **allow_set_foreground()**: Esta función habilita que el proceso actual pueda poner ventanas al frente del escritorio.
2. **press_key(hexKeyCode, delay_time)**: Simula la pulsación de una tecla en el teclado usando su código hexadecimal. La tecla se mantiene presionada por un tiempo definido y luego se suelta.
3. **focus_and_fullscreen(window_keyword)**: Localiza y enfoca una ventana específica basándose en una palabra clave del título de la ventana..
4. **run_menu()**: Crea una interfaz gráfica de usuario con botones para iniciar y cerrar la aplicación.
5. **run_camara()**: Inicia la cámara en un hilo separado, permitiendo que el proceso de captura de imágenes o video funcione de forma independiente.

La interfaz gráfica es realmente innecesaria, es solo una manera sencilla de ver algo bonito para poder controlar la acción del código de manera fluida.

4.3.- Cálculo de dirección en base a puntos

Una vez que se tienen las direcciones y el control sobre la aplicación, se usará unas nuevas funciones dedicadas solamente a tomar la información de control y utilizarla para llamar a la función press_key() antes mencionada para simular el movimiento, para ello se usará los siguientes valores:

- **W**: 0x57
- **A**: 0x41
- **S**: 0x53
- **D**: 0x44
- **Flecha Arriba**: 0x26
- **Flecha Abajo**: 0x28
- **Flecha Izquierda**: 0x25
- **Flecha Derecha**: 0x27
- **Ctrl**: 0x11
- **Esc**: 0x1B
- **Q**: 0x51
- **E**: 0x45
- **Space**: 0x20

5. Justificación del Planteamiento

Ahora que se sabe que hace el código para ejecutar, toca preguntar porque este sistema, y cómo se podría llegar a mejorar en otros contextos.

Para entender el contexto del planteamiento hay que entender el funcionamiento de la aplicación a desarrollar

La aplicación a desarrollar es un juego llamado The Binding of Isaac, el motivo de usar este juego es por tener un sistema de controles bastante complejos, pudiendo llegar a tener 13 controles mínimos mapeados.

Dentro del juego se dispone de el movimiento que requiere de las teclas W, A, S, D. Al mismo tiempo se requiere de las flechas de dirección para poder disparar en la dirección designada. Teniendo en cuenta el requerimiento, se disponen 3 posibles opciones, una primera serían los brazos, y marca directamente la dirección a la que caminar o disparar, otra sería parecida pero con las manos, la última opción sigue el mismo esquema podría ser los ojos con el uso de mediapipe.

En este caso, por ser una prueba sencilla se utilizó los brazos, ya que las otras requeriría un desarrollo algo más complejo pero perfectamente accesible.

Los otros controles incluirán valores por separados, que no tienen conexión entre sí tales como:

- **Q:** usar una carta
- **E:** usar una bomba
- **Mantener el Ctrl:** Dejar la carta en el suelo
- **Space:** Utilizar la activa
- **Esc:** Pausar el juego

Hay muchas opciones para todo ello, existen 3 opciones bastante obvias:

1. Utilizar mediapipe para sacar cálculos con la cara, esto permite jugar sin tener en cuenta las piernas
2. Utilizar Aprendizaje profundo para determinar ciertas muecas y que detecte cada mueca como una opción
3. Utilizar nuevamente cálculos en base a YOLO, que nos deja disponible solo el uso de la cabeza y las piernas.

En este caso se volverá a usar YOLO por varios motivos que no son beneficiarias de YOLO sino porque las otras opciones tienen problemas irresolubles.

Tanto el aprendizaje automático como mediapipe tiene el problema de que al necesitar usar los brazos hay que alejarse bastante de la cámara lo que dificulta la detección de la cara.

Otro problema sería tener dos sistemas totalmente distintos corriendo a la vez y actuando de manera simultánea, lo que puede generar problemas de rendimiento con sistemas no muy potentes.

Lo cual deja a YOLO como el más simple para desarrollar, tal vez en un contexto en el que se utilizará las manos como control principal, se decantaría hacia mediapipe.

Pero teniendo YOLO, cual es la mejor opción teniendo la información del esqueleto, las opciones son relativamente limitadas pero más amplias de las pensadas. Al ser un ejemplo sencillo, se designó por la opción más sencilla, hay 5 comandos a usar, entonces se pueden usar ambas piernas para denominar 4 opciones, y la nariz para la última.

El funcionamiento se explica de manera algo más detallada en el ReadME del proyecto que se puede ver en el repositorio que se encuentra en el Anexo

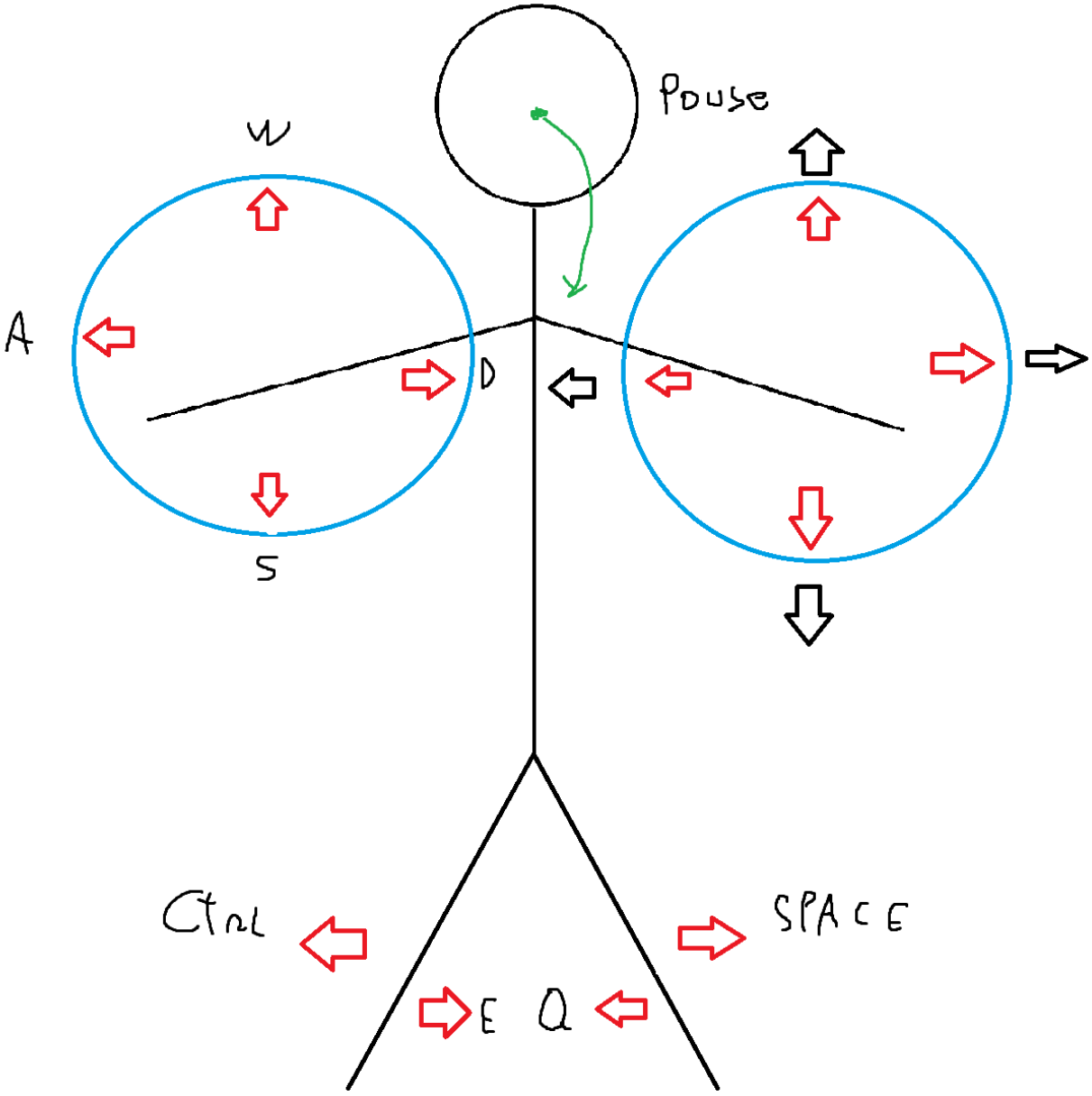
6. Evaluación de resultados

Para poder visualizar el resultado de una manera correcta existe un modo de debug con el que se puede visualizar que está ocurriendo en el código. Esto está hecho con una pequeña consola propia que registra todos los datos importantes a la vez. Aqui un pequeño ejemplo del código

```
       texto_consola.append(f"Resultado direccion brazo izquierda:
{resultado_mano_hombro_izquierda[0], resultado_mano_hombro_izquierda[1],
resultado_mano_hombro_izquierda[2]}\n----------------------------------")
        texto_consola.append(f"Resultado direccion brazo derecha:
{resultado_mano_hombro_derecha[0], resultado_mano_hombro_derecha[1],
resultado_mano_hombro_derecha[2]}\n----------------------------------")

        texto_consola.append(f"Resultado direccion pierna izquierda:
{resultado_cadera_pies_izquierda[0],
resultado_cadera_pies_izquierda[1]}\n----------------------------------")
        texto_consola.append(f"Resultado direccion pierna derecha:
{resultado_cadera_pies_derecha[0],
resultado_cadera_pies_derecha[1]}\n----------------------------------")

        texto_consola.append(f"FPS:
{fps_2_seconds}\n----------------------------------")
        texto_consola.append(f"Izquierda: Izquierda:
{contador_frame_izq['Izquierda']}, Derecha:
{contador_frame_izq['Derecha']}\n----------------------------------")
        texto_consola.append(f"Derecha: Izquierda:
{contador_frame_der['Izquierda']}, Derecha:
{contador_frame_der['Derecha']}\n----------------------------------")

        texto_consola.append(f"Resultado check Nariz:
{resultado_check_nariz}\n----------------------------------")
        texto_consola.append(f"Value: {Value}, Parado:
{Parado}\n----------------------------------")
```

La idea es solo ver cómo está funcionando el código a un nivel interno.

7. Controles de la aplicación



Esto sería la idea del mapeado de los controles de la aplicación de manera esquematizada y que es utilizado en la portada.

Importante los controles de las piernas requieren ejecutarlos durante 30 frames para poder ejecutarse

8. Dificultades durante el proyecto

Aquí habrá una pequeña recopilación de problemas surgidos durante el desarrollo y sus maneras de ser solucionados.

1. Utilización de mediapipe

Al principio se pretendía usar mediapipe junto con YOLO, llegó al punto de crear un código capaz generar dos acciones solo con el movimiento de la cara.

El código es funcional pero fue descartado después de su implementación debido al uso de los brazos con YOLO lo cual obliga a estar algo alejado de la cámara y provoca que deje de funcionar.

Se intentó también utilizar las manos de mediapipe, pero era muy inconsistente para poder ser usado.

2. Utilización de pyautogui

Al principio se pretendía usar la librería pyautogui, pero tras varias pruebas parecer no funcionar de manera consistentes entre todos los programas, al probar con un bloc de notas parece consistente pero no siempre en otras aplicaciones.

Para evitar problemas se optó por usar la librería ctypes

9. Anexo/Extras

Aquí hay una pequeña recopilación de apartados extras relacionados, así como el enlace al código final del código:

1. Enlace a código final

https://github.com/MarTonPerCar/Interfaz-Instintiva-

2. Código extra de mediapipe

Al final de código final existe una parte extra la cual es un código que se hizo en el mitad del desarrollo com mediapipe para generar controles con el movimiento de la cara.

3. Código extra

Así mismo hay otro pequeño código también funcional sobre otra aplicación llamada “Just Shapes and Beats”, es otro juego de funcionamiento mucho más sencillo con solo necesitar 5 acciones, de las 4 son movimientos y la otra sería el espacio.

Se usó como prueba técnica, pero esto demuestra lo reutilizable y modulable que puede ser el código para otras aplicaciones.

4. Video de presentación del proyecto

https://alumnosulpgc-my.sharepoint.com/:v:/g/personal/mario_perez114_alu_ulpgc_es/EetElXIoMx1EkWOhzBuSjckBJGsbBy_40pvkqxHQeyNarw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcmI2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=9GPUVc

10. Bibliografía

1. Página para ver los puntos de mediapipe y pagina para descargar los archivos para manejar el mediapipe así como tener una explicación de uso

https://medium.com/@hotakoma/mediapipe-landmark-face-hand-pose-sequence-number-list-view-778364d6c414
https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=es-419
https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker?hl=es-419#models

2. Geogebra para poder probar el funcionamiento de los puntos

https://www.geogebra.org/calculator

3. Varias páginas stack overflow para solucionar problemas en el desarrollo

https://es.stackoverflow.com/questions/352451/typeerror-nonetype-object-is-not-subscriptable

https://stackoverflow.com/questions/16770909/python-win32gui-setforegroundwindow

https://stackoverflow.com/questions/36680402/typeerror-only-length-1-arrays-can-be-converted-to-python-scalars-while-plot-sh

https://stackoverflow.com/questions/63668351/sympy-attribute-error-module-sympy-has-no-attribute-derive-by-array

https://stackoverflow.com/questions/64565909/attributeerror-partially-initialized-module-sympy-has-no-attribute-s-most

4. Documentación del comando usado para generar el ángulo

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math/atan2

5. Documentación de ctypes

https://docs.python.org/3/library/ctypes.html

6. Documentación de Threads

https://recursospython.com/guias-y-manuales/tareas-en-segundo-plano-con-tcl-tk-tkinter/