Cuando el Código Conversa: Sockets desde Python a Unity

Tracking en tiempo real con ArUco, Python y Unity

Nietsnie Alberto Perez Cruz

Desarrollador Fullstack Dsi

https://github.com/Nietsnie-beep

💬 Spoiler: sí, usé sockets. Y sí, también visión computacional.

🤔 ¿Qué problema quería resolver?

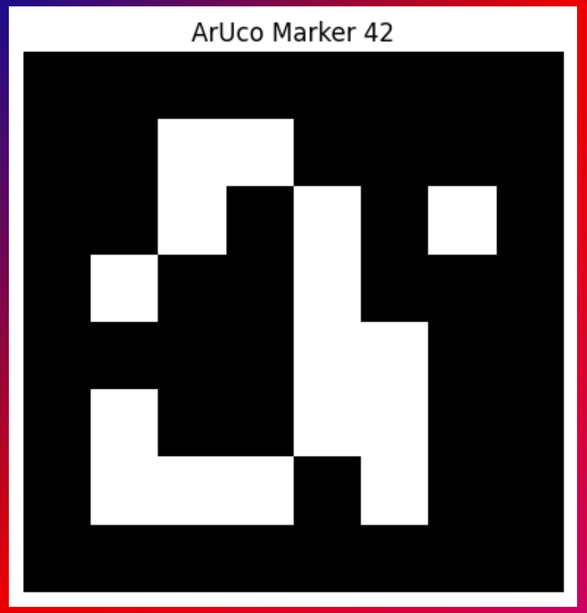
Me preguntaba:
"¿Cómo puedo combinar la capacidad de visión artificial de Python con el poder gráfico de Unity... y hacer que hablen entre sí?"

🔍 Python es genial para detectar cosas del mundo real

🎮 Unity es perfecto para mostrar cosas en 3D con estilo.

💡 **La idea:**
Usar Python para ver y Unity para mostrar.
Unirlos con WebSockets y lograr una conversación en tiempo real entre código y gráficos.

"Visión + Gráficos = Interacción"

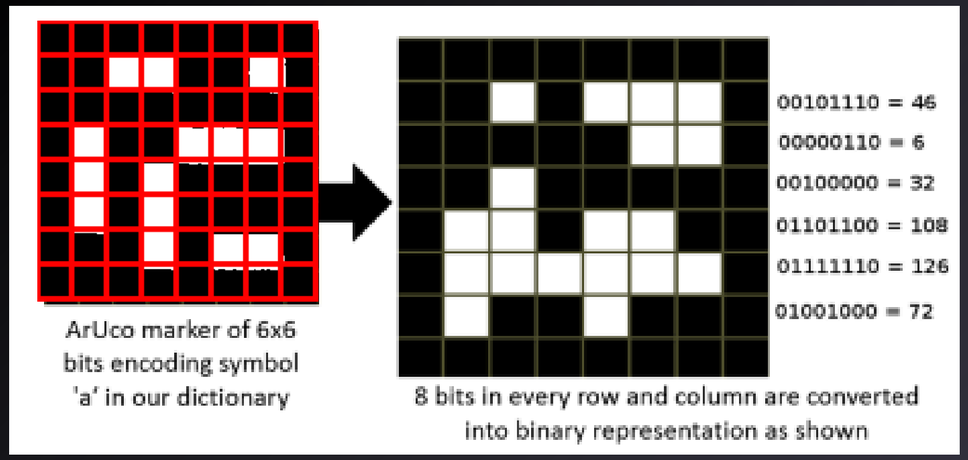


Son como QR pero más cool y para desarrolladores.

¿Qué es ArUco y por qué lo usé?

- Cuadrado con borde negro grueso
- Tiene un patrón binario que representa un ID único
- Es fácil de detectar incluso con poca luz 😅
- Permiten obtener posición y rotación en tiempo real.

Fáciles de detectar con OpenCV.
Te dicen: “Ey, estoy aquí y además estoy inclinado así 🤸‍♂️”



- 🔍 Detección de códigos ArUco con OpenCV
- OpenCV incluye un módulo específico para trabajar con ArUco, el cual permite:
- 🔲 Detectar automáticamente los marcadores en una imagen o video.
- 🆔 Leer el ID único del marcador.
- 📐 Calcular su posición y orientación (pose estimation) con respecto a la cámara.
- 📸 Usar la calibración de cámara para obtener datos precisos en 3D.

```python
import cv2
import cv2.aruco as aruco
```

Detección de marcador ArUco en tiempo real

```python
import cv2
import cv2.aruco as aruco
import numpy as np

def detectar_aruco():
    # Inicializar la cámara
    cap = cv2.VideoCapture(0)

    # Cargar el diccionario de ArUco
    diccionario = aruco.getPredefinedDictionary(aruco.DICT_4X4_50)

    # Crear parámetros para la detección
    parametros = aruco.DetectorParameters_create()

    while True:
        # Capturar frame por frame
        ret, frame = cap.read()

        if not ret:
            print("Error al capturar el frame")
            break

        # Convertir a escala de grises
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Detectar marcadores
        esquinas, ids, _ = aruco.detectMarkers(gray, diccionario, parameters=parametros)

        # Dibujar los marcadores detectados
        if ids is not None:
            aruco.drawDetectedMarkers(frame, esquinas, ids)

            # Mostrar los IDs de los marcadores
            for i in range(len(ids)):
                cv2.putText(frame, str(ids[i][0]),
                            (int(esquinas[i][0][0][0]), int(esquinas[i][0][0][1])),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        # Mostrar el frame resultante
        cv2.imshow('Detección de ArUco', frame)

        # Salir con la tecla 'q'
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Liberar la cámara y cerrar ventanas
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    detectar_aruco()
```

¿Y los sockets pa’ qué?

“REST es mandar cartas. WebSocket es una llamada en tiempo real 📞”

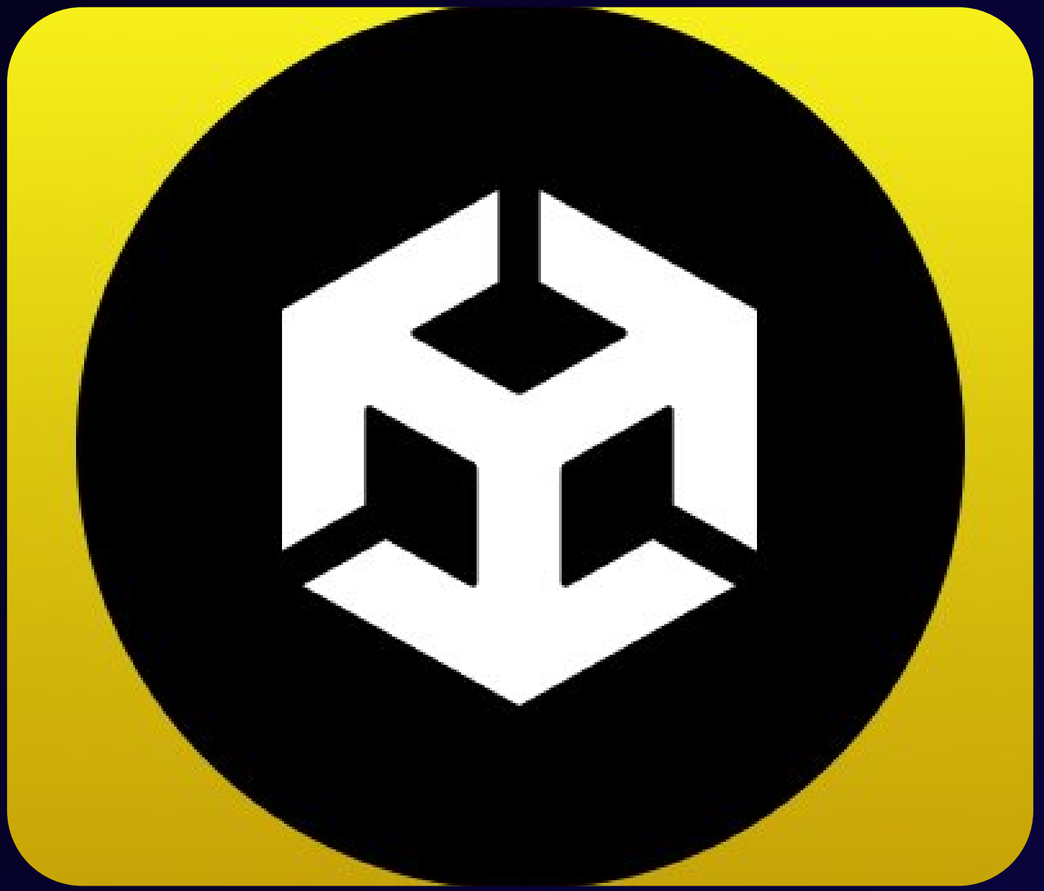
Mantiene una conexión abierta entre Python y Unity

Permite enviar datos constantemente, sin esperar respuesta

```
{ "id": 42, "position": [1.2, 0.5, -3.1], "rotation": [0, 45, 90] }
```

¿Y Unity qué hace?

- Escucha el socket
- Recibe las coordenadas
- Mueve un objeto 3D (como un cubo) en base a lo que ve Python
- ¡Listo! 🎯 Movimiento en tiempo real



¿Cómo se conecta Python con Unity?

"Básicamente, Python le está chismeando a Unity lo que ve la cámara, y Unity lo usa para mover algo."

- Python (el cerebro y los ojos)
- Detecta el marcador ArUco con OpenCV.
- Calcula su posición y orientación.
- Envía esos datos por un WebSocket en formato JSON
- WebSocket (el canal)
- Es una conexión constante (como una llamada en Zoom).
- Python manda datos todo el tiempo sin tener que esperar respuesta.
- Unity (el cuerpo)
- Se conecta como cliente al WebSocket de Python.
- Escucha los mensajes que llegan.
- Toma los datos (posición y rotación) y mueve un objeto 3D.

```python
import asyncio
import websockets
import json

# Función que maneja cada conexión
async def handler(websocket):
    print("Cliente conectado")
    while True:
        # Aquí puedes simular datos o recibir datos reales del ArUco
        data = {
            "id": 42,
            "position": [1.2, 0.5, -3.1],
            "rotation": [0, 45, 90]
        }
        await websocket.send(json.dumps(data))
        await asyncio.sleep(0.05)  # manda datos cada 50 ms

# Iniciar el servidor en localhost:3000
async def main():
    async with websockets.serve(handler, "0.0.0.0", 3000):
        print("Servidor WebSocket corriendo en el puerto 3000")
        await asyncio.Future()  # Mantiene el servidor corriendo

asyncio.run(main())
```

Ejemplo de socket

Y ahora_ ¡una demo!

Lo que me costó (y me hizo sudar 🥵)

🔁 **Coordenadas locas:**

- OpenCV y Unity usan sistemas de ejes distintos (izquierda vs derecha, arriba vs abajo...).

🐢 **Latencia**

- Si mandas muchos datos por segundo, se traba o se desincroniza. Tuve que controlar bien el ritmo.

🧾 **JSON en Unity**

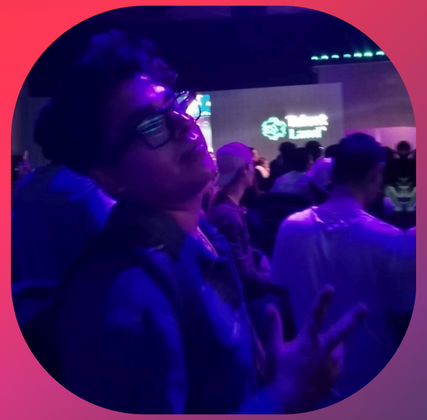
- Parsear los datos en C# fue más complejo de lo que pensaba. Toca convertirlos bien para no romper todo.

Porque todo parece fácil... hasta que 😅

¡Gracias por escuchar!

¿Preguntas, ideas, chismes técnicos?

El codigo aqui!!



Nietsnie Alberto Perez Cruz

Desarrollador Fullstack Dsi

https://github.com/Nietsnie-beep