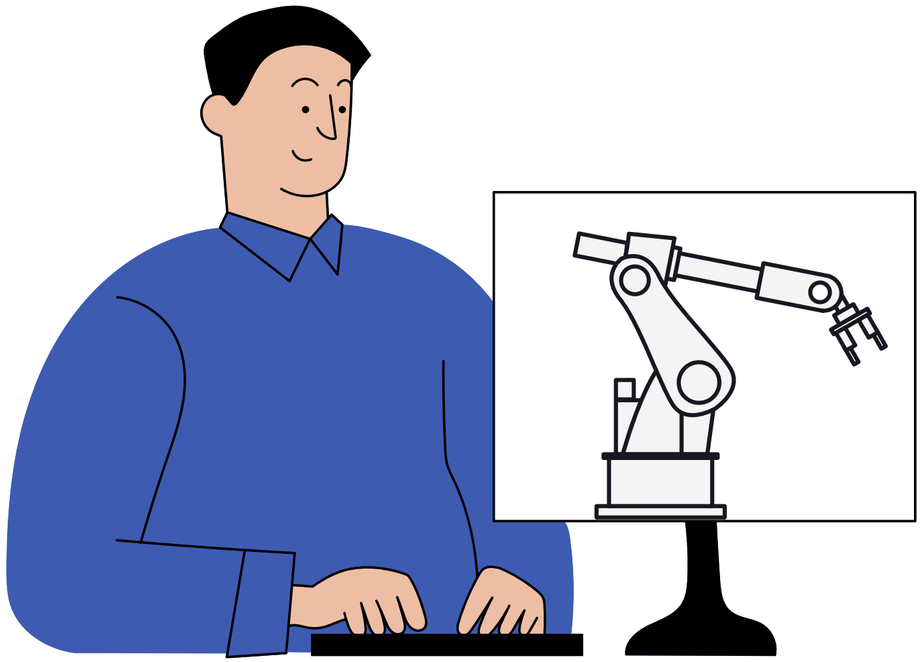
CONF42

Implementing a Virtual-Physical Environment Manipulation System based on ROS using Python and Three.js

Ivan Chiou

Today's Agenda



- What is ROS and Three.js
- Digitaltwins User Interface
- The Control of Physical Robotic Arm
- Three.js + Vite in the Virtual Environment
- WebRTC bind IP cam to display the Physical Environment
- Establish a connection to ROS cluster via Rosbridge
- Integrate with APIs
- Conclusion

Speaker Introduction

Ivan Chiou

My background in multimedia integration and cross-functional collaboration has honed my ability to mentor and inspire teams effectively. I take pride in guiding young engineers, offering them support and optimism for their future careers.

Tools & Skills

01

Design tool

- Figma

02

Framework

- Vite
- Vue
- Vue Router
- Pinia
- Tailwind
- SCSS

03

Skills

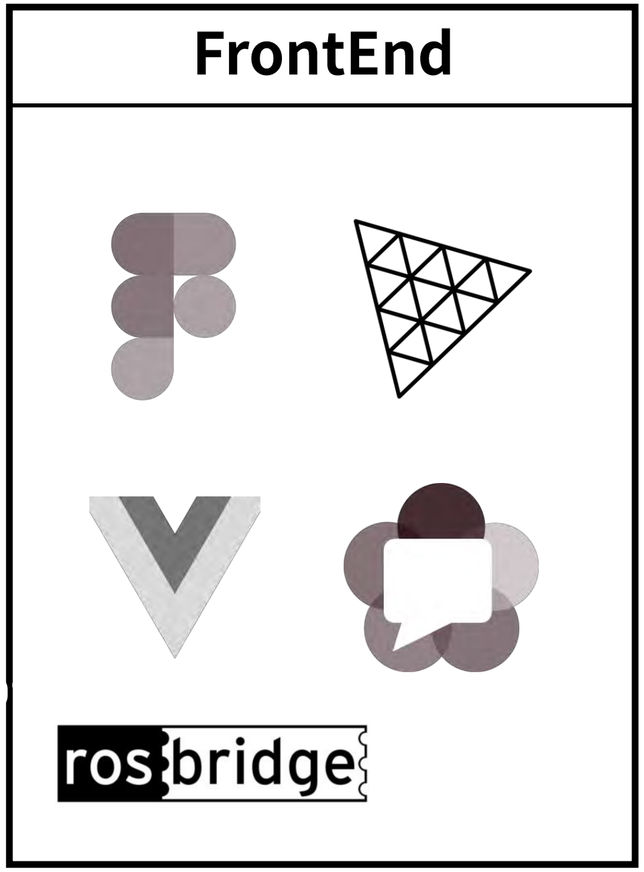
- **Three.js**
  - 3D Model of the Robotic Arm
- **ROS bridge**
  - Receiving real-time messages from the Robotic Arm
- **WebRTC**
  - Connect to the IP Cam to display the Physical Robotic Arm
- **ROS Cloud API**
  - Update and Record the Status of the Robotic Arm

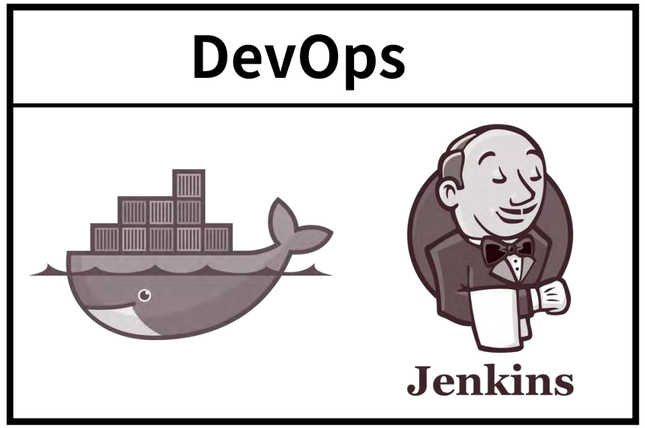
04

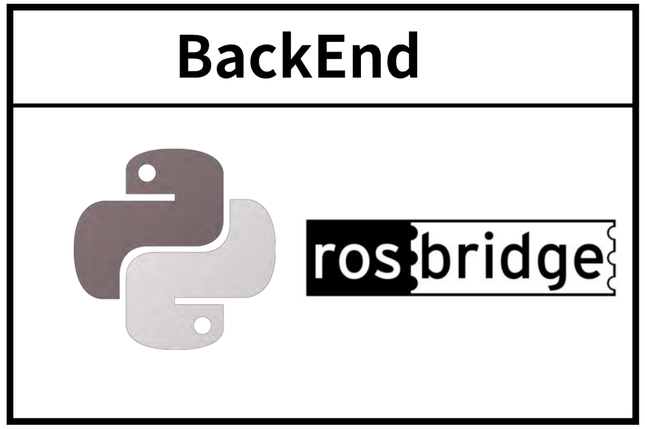
Others

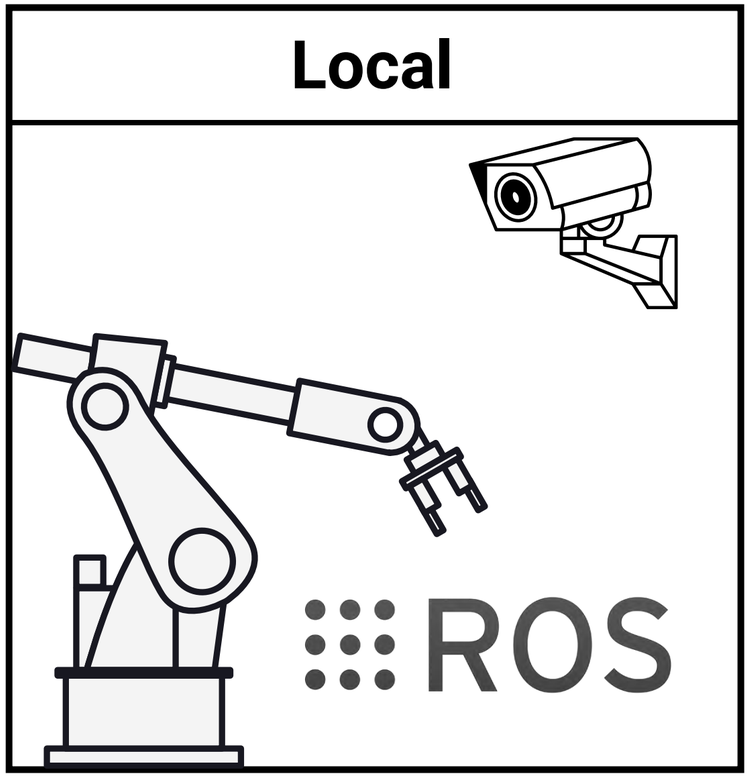
- Docker
- Jenkins

Architecture

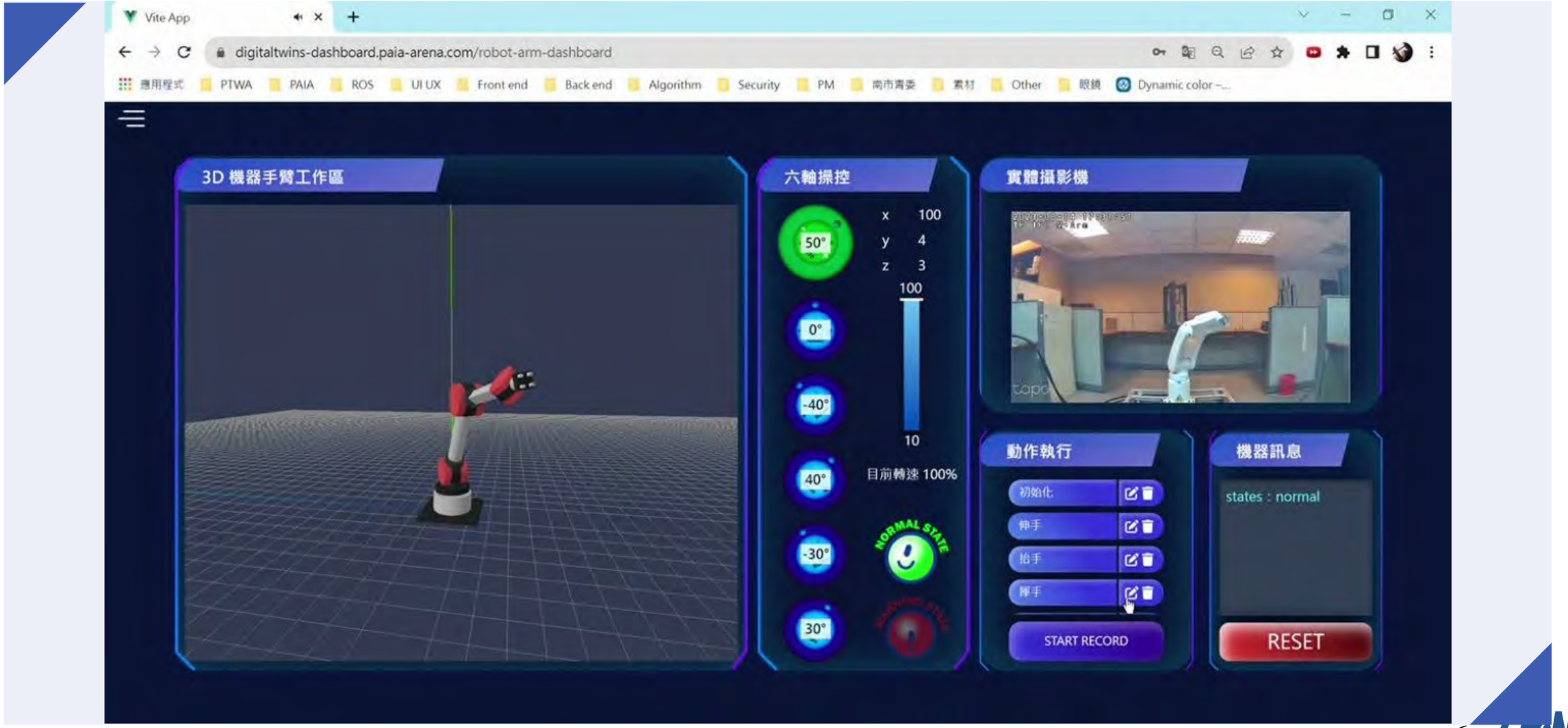








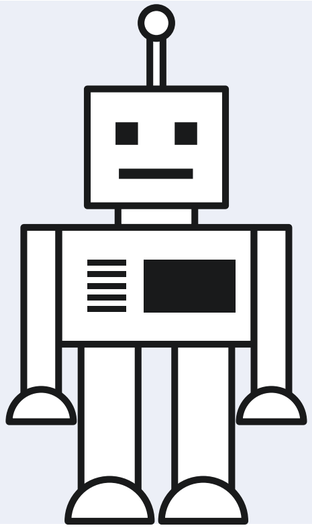
DEMO


Vite App
digitaltwins-dashboard.paia-arena.com/robot-arm-dashboard
3D 機器手臂工作區
六軸操控
x 100
y 4
z 3
100
10
50°
0°
-40°
40°
-30°
30°
目前轉速 100%
NORMAL STATE
實體攝影機
動作執行
初始化
伸手
抬手
揮手
START RECORD
機器訊息
states : normal
RESET

What is ROS and Three.js

ROS

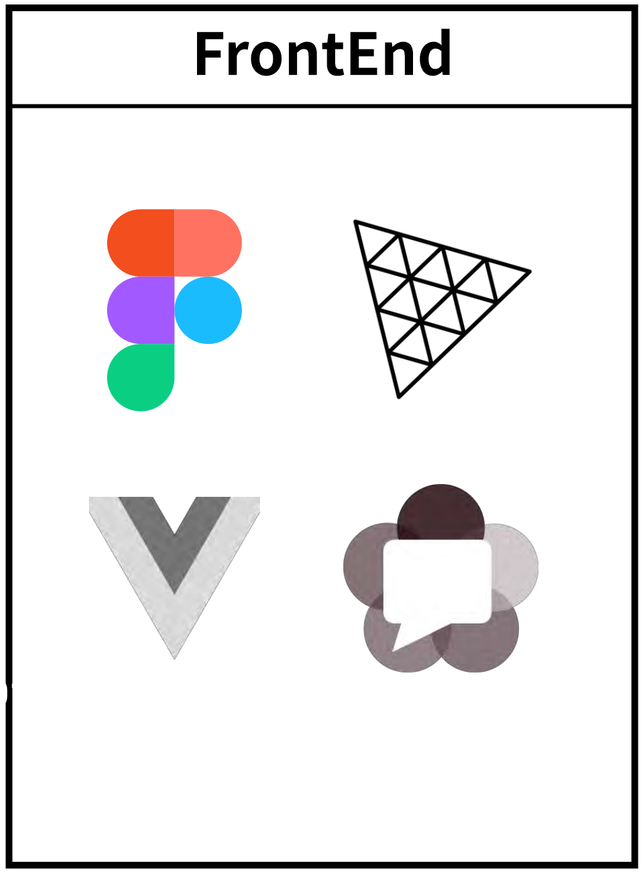
- Open-source robot development software system framework
- Modular characteristics: Stacks -> Packages -> Nodes
- Communication mechanism between ROS nodes: Topic, Service, Actionlib

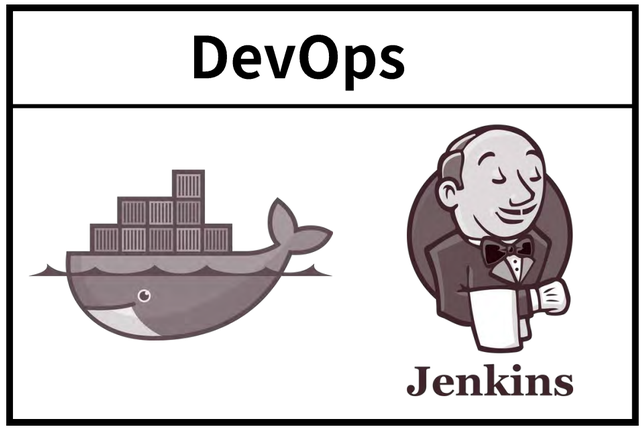


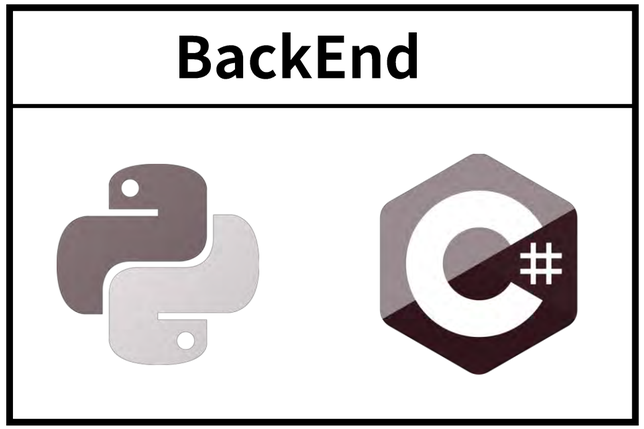
Three.js

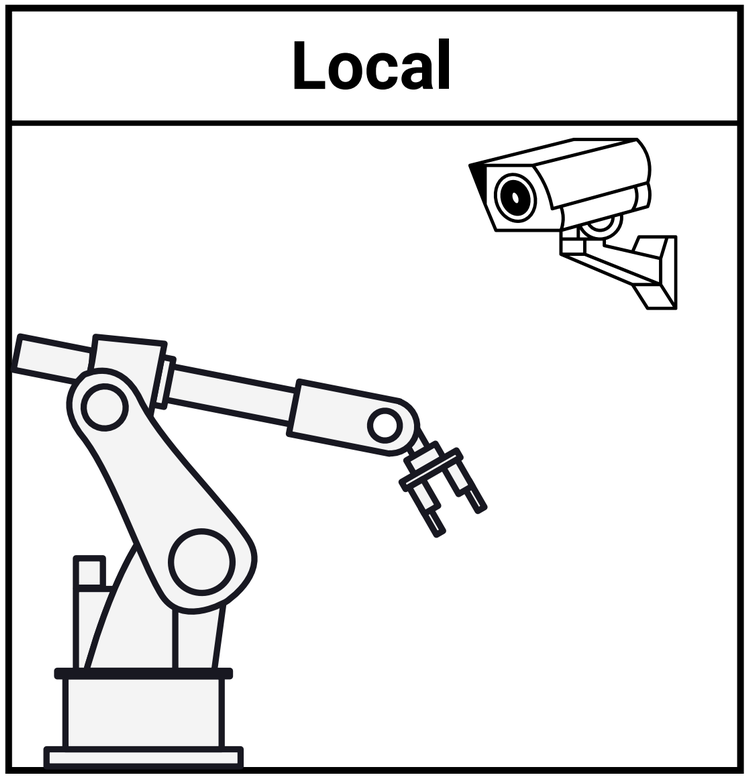
- APIs are developed based on WebGL, encapsulated and simplified.
- A 3D graphics library to easily create interactive 3D graphics.
- Low learning curve.

Figma

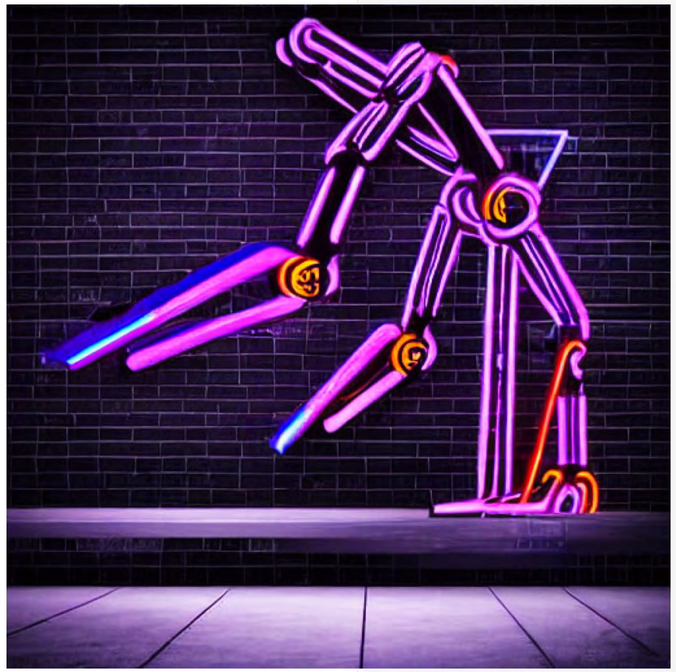








Design Mockup through Figma + AI



Drafts / Ando - AI Copilot for Designers (Community)

Share

100%

Layers Assets Page 1

Ando - AI Copilot for Designers

Render Process Lab

Prompt

neon lamp black background high contrast metal cool robotic arm machine cyberpunk future

Reference

None Select a frame as the reference.

Prompt Weight 80

Image Prompt

Number of Outputs 4

Output Size Medium

Render

Design Prototype

Page

F5F5F5 100%

Local variables

Local styles

Export

Design Mockup through Figma + AI

Cyberpunk style

01 neon lamp

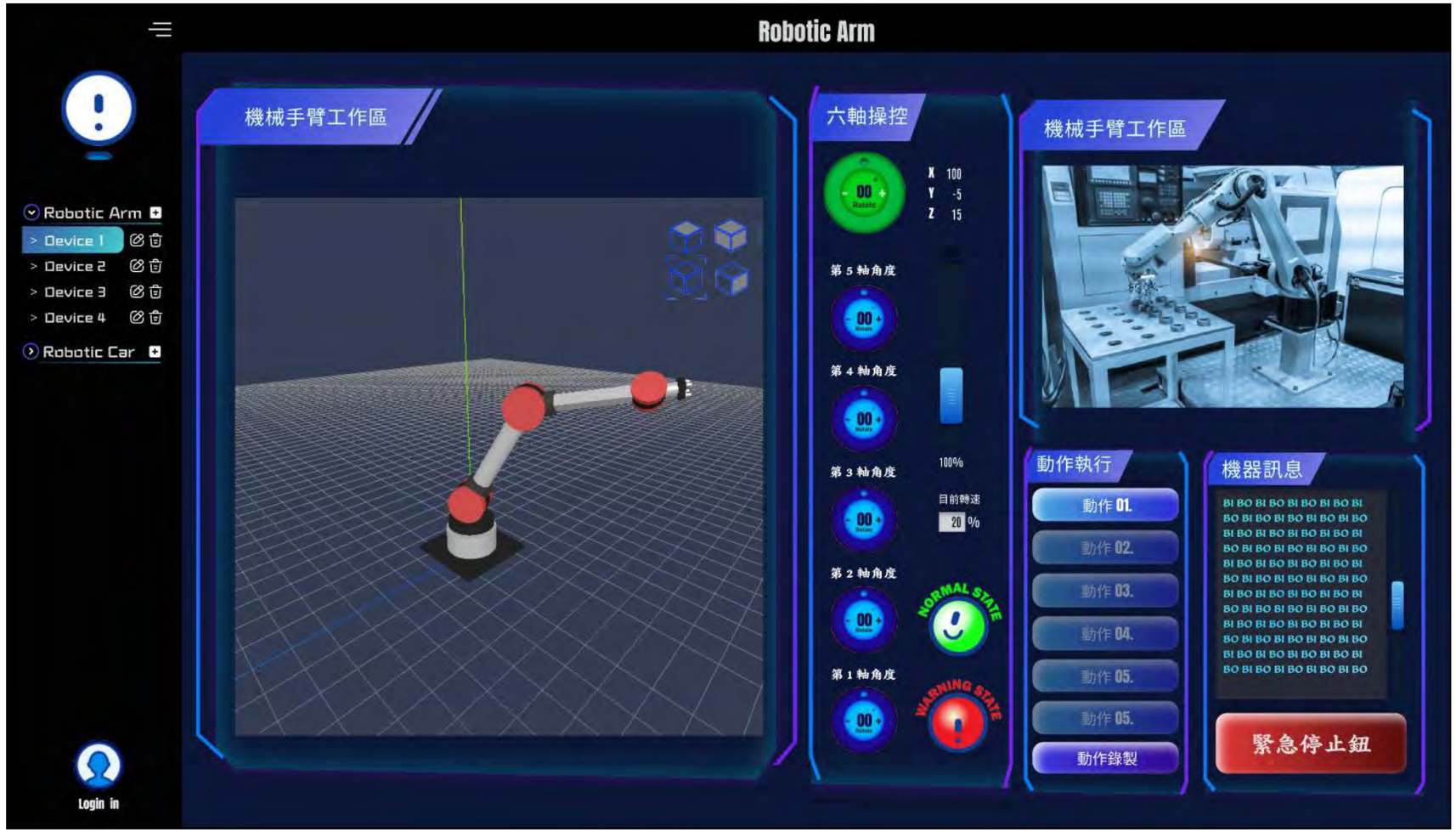
- Blue, purple, and pink
- Vibrant neon colors
- The night skyline of a futuristic city

02 black background / high contrast

- Dark background
- The brightness of neon
- Futurism and mystery vibe

03 Metal and Cool color

- Silver and chrome
- Cool blue and cool green
- High-tech and mechanized



Three Parts for the User Interface

Information Display 01

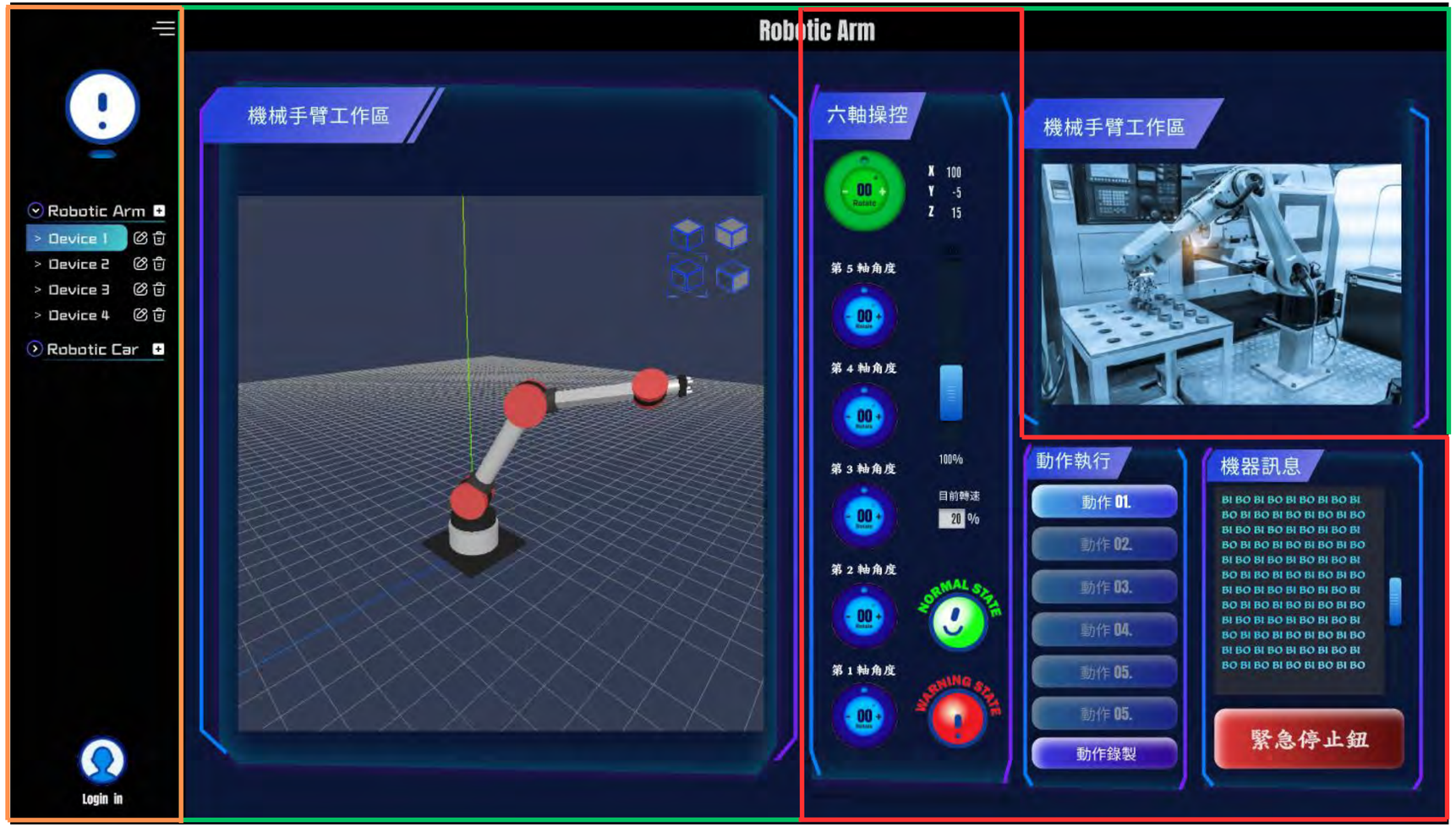
- Virtual - 3D robotic arm
- Physical – IP Camera
- Coordinates of the sixth axis
- Error messages

Control panel 02

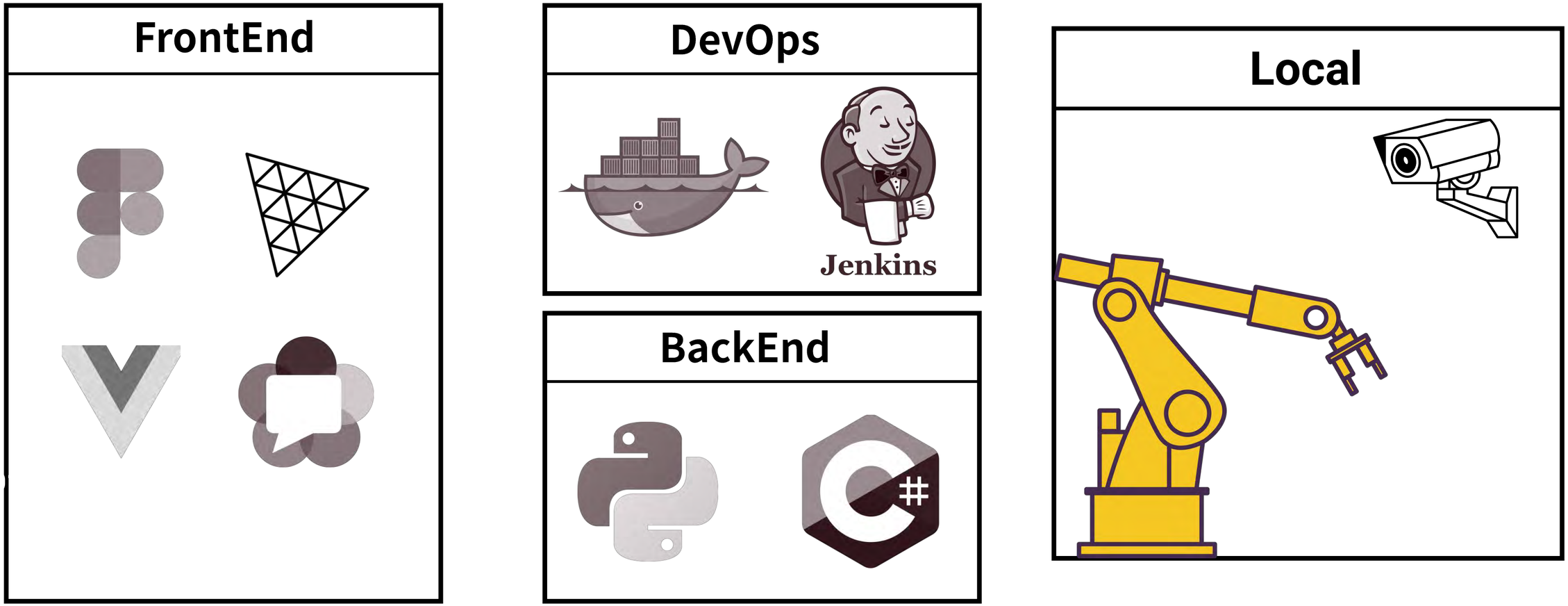
- Reset button
- Six-axis angle control gear
- Motor movement speed
- Action recording (editable)

Sidebar 03

- Logo
- Device List (editable)
- User login



Physical Robotic Arm

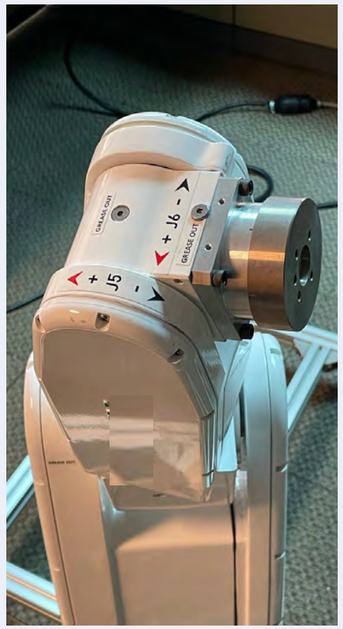


Physical robotic arm with six axes and control

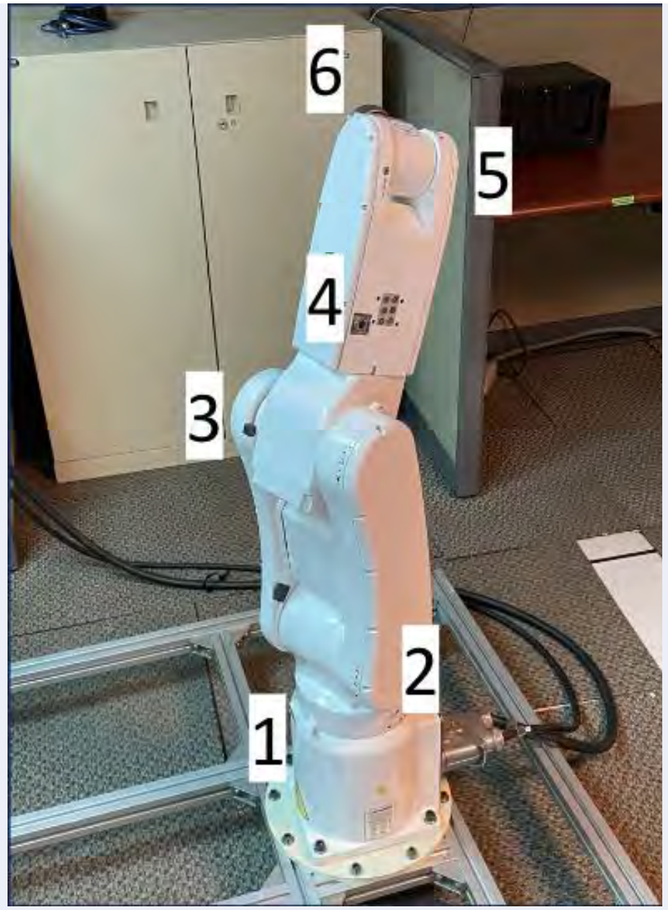
1st, 2nd axes

3rd axis

4th axis

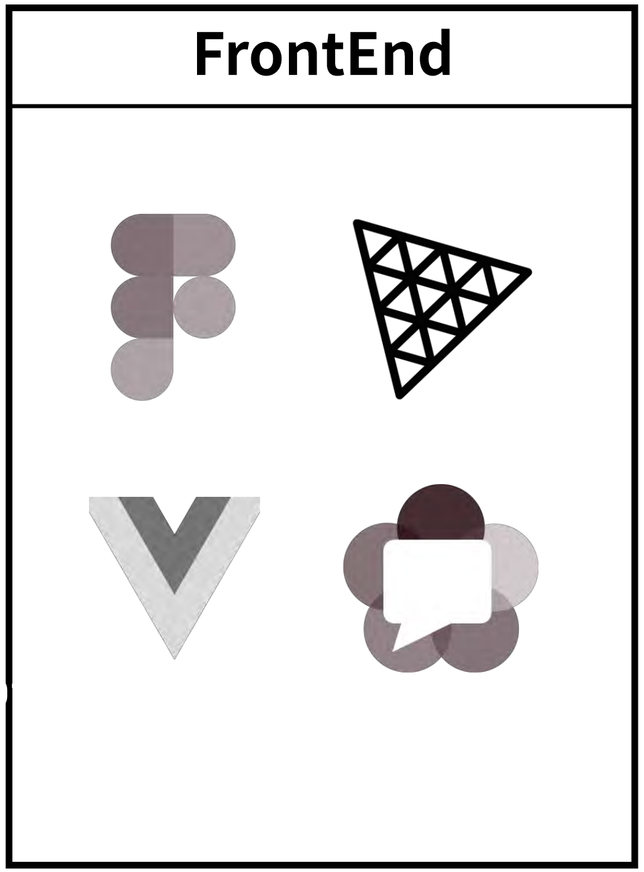


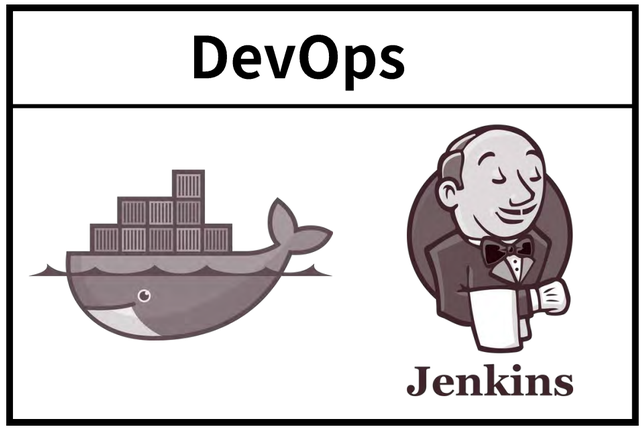
5th , 6th axes

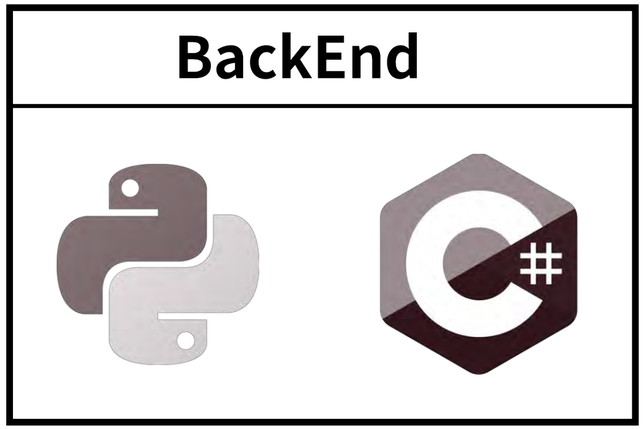


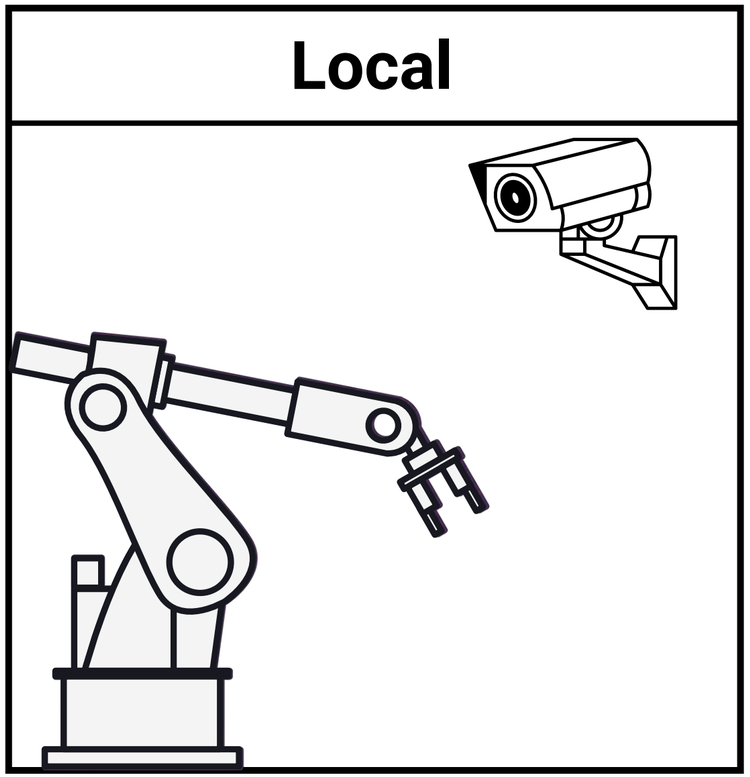
Overview

Three.js



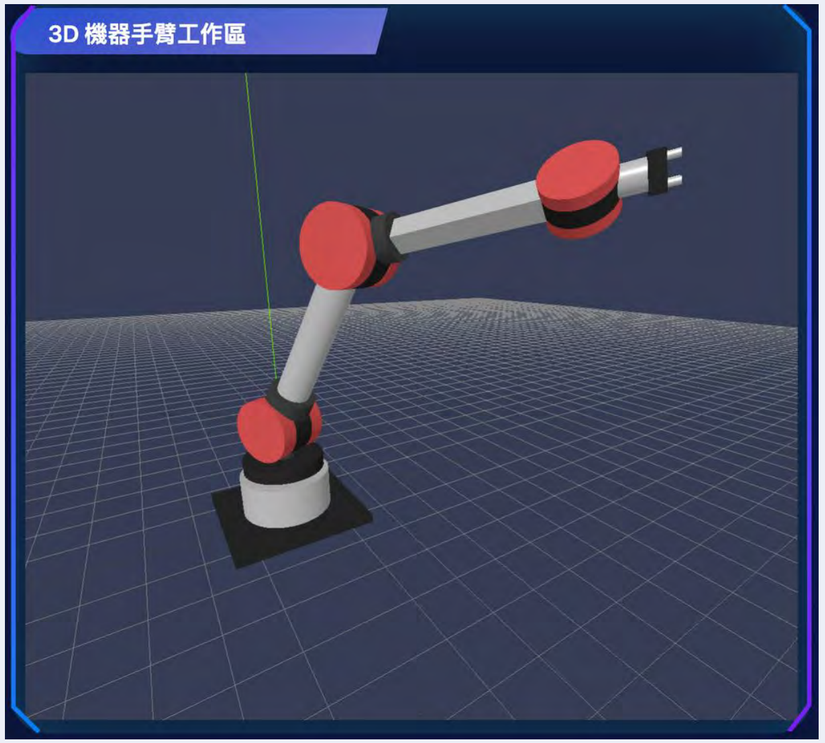






Three.js Fundamentals

- **Scene：**
  - A virtual 3D stage where cameras, objects, and light sources are all present.
- **Camera：**
  - Determining the position, perspective, and projection.
- **Objects：**
  - Operations such as rotation, scaling, and translation on objects like cubes, spheres, and models.
- **Light：**
  - The brightness and shadow effects of objects are determined by the position of light sources such as ambient light, directional light, point light, and spotlight.



- **Renderer：**
  - Converting 3D objects and lighting information into 2D, resulting in transformed images on the screen from the camera's perspective within the scene.

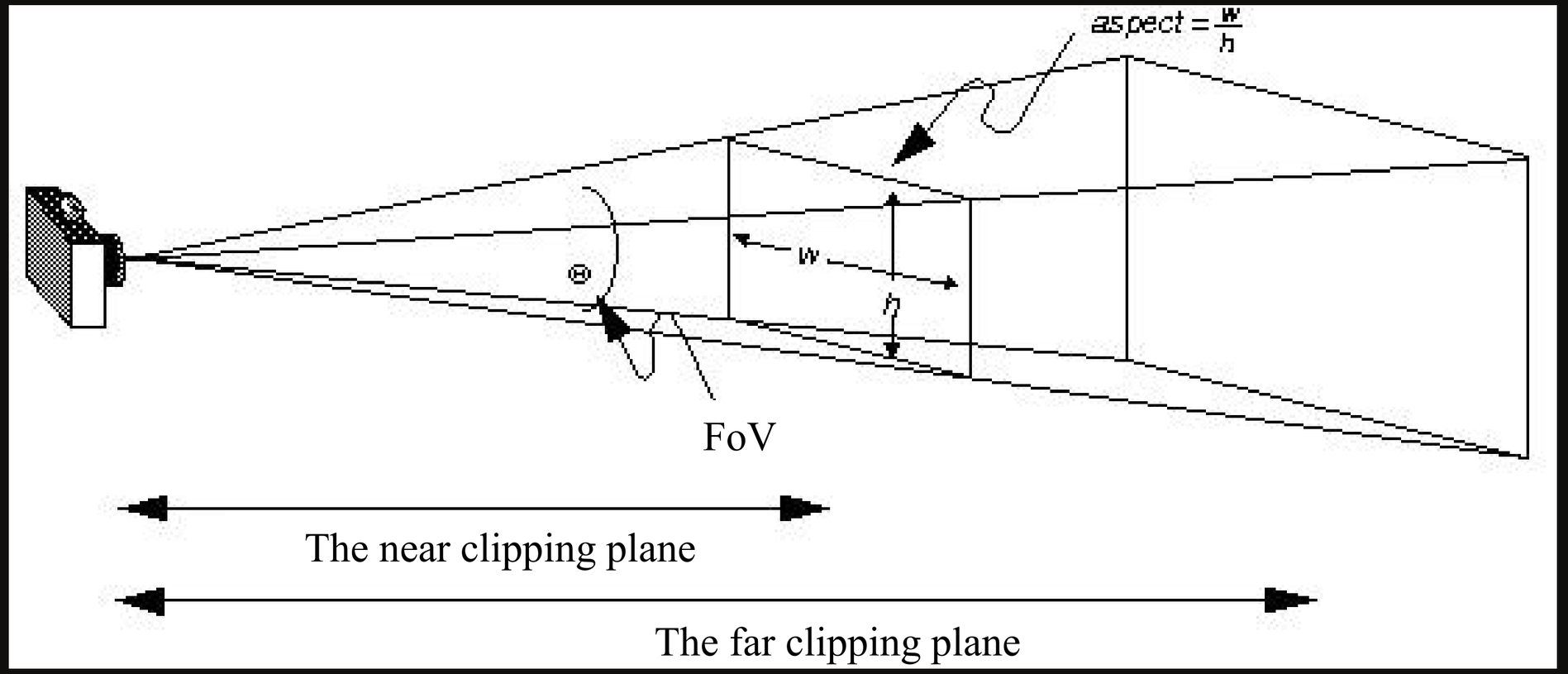
Three.js Scene

```
new THREE.Scene();
```

Three.js Camera

Scene

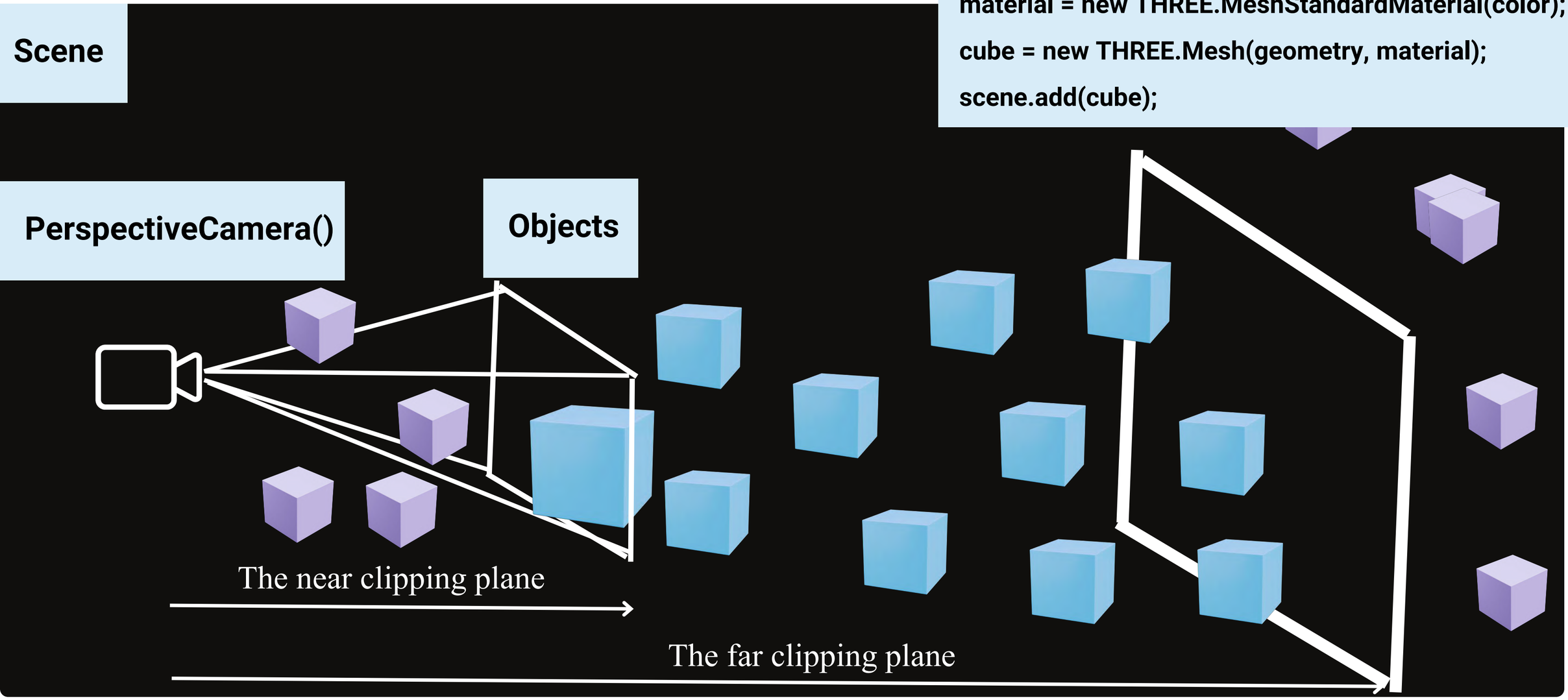
new THREE.PerspectiveCamera(FoV(field of view), aspect ratio, near, far);



https://zhuanlan.zhihu.com/p/27296011

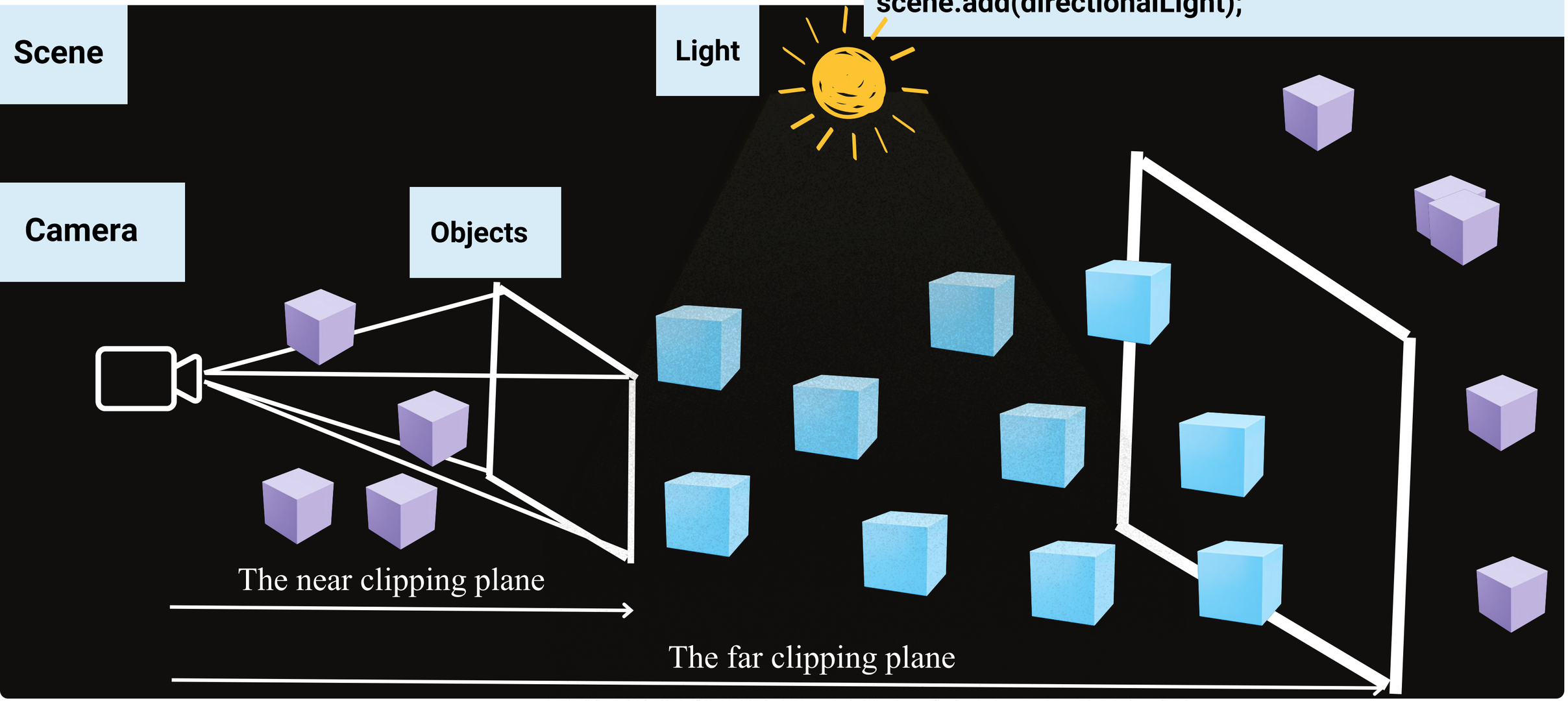
Three.js Objects

```
geometry = new THREE.BoxGeometry();
material = new THREE.MeshStandardMaterial(color);
cube = new THREE.Mesh(geometry, material);
scene.add(cube);
```

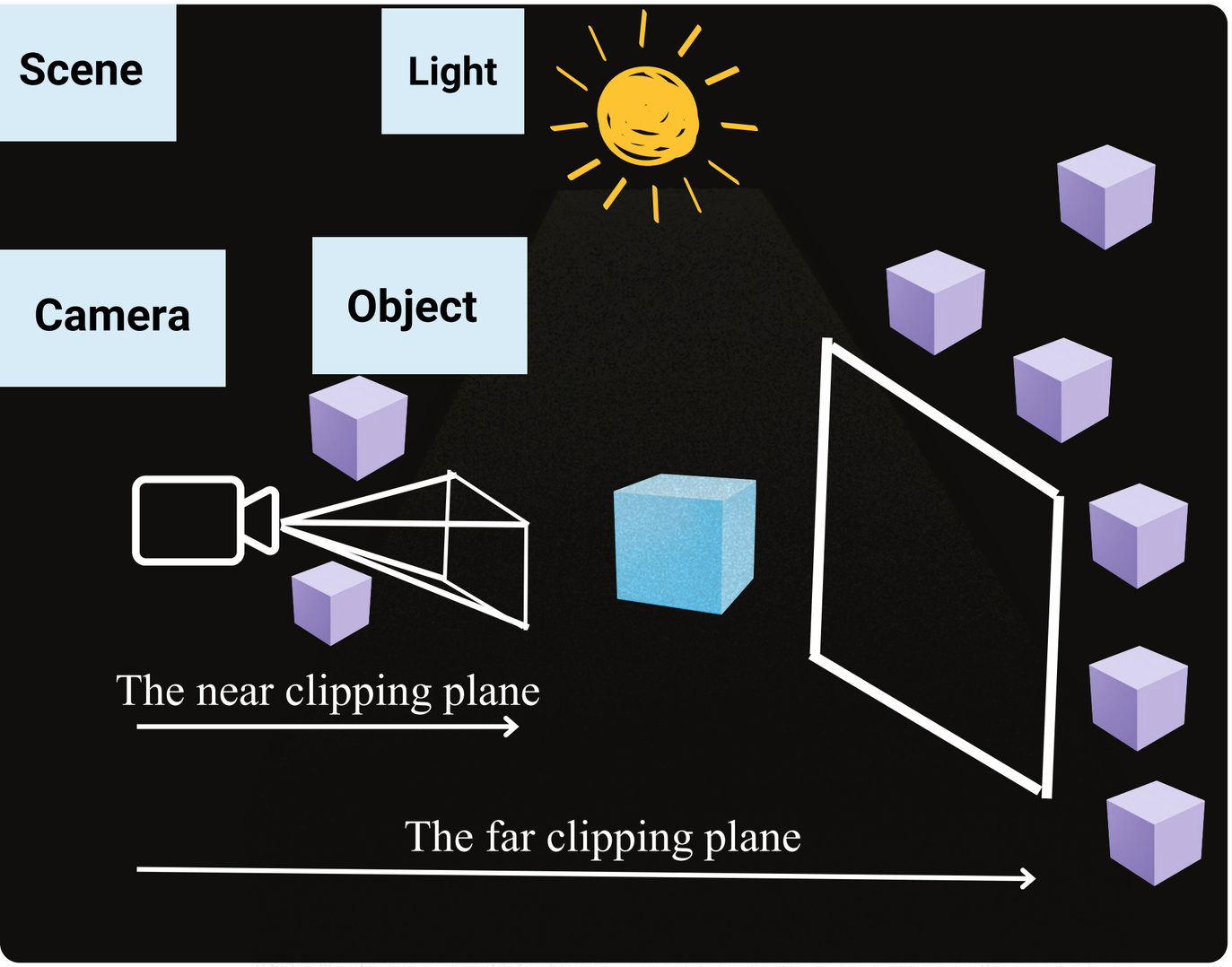


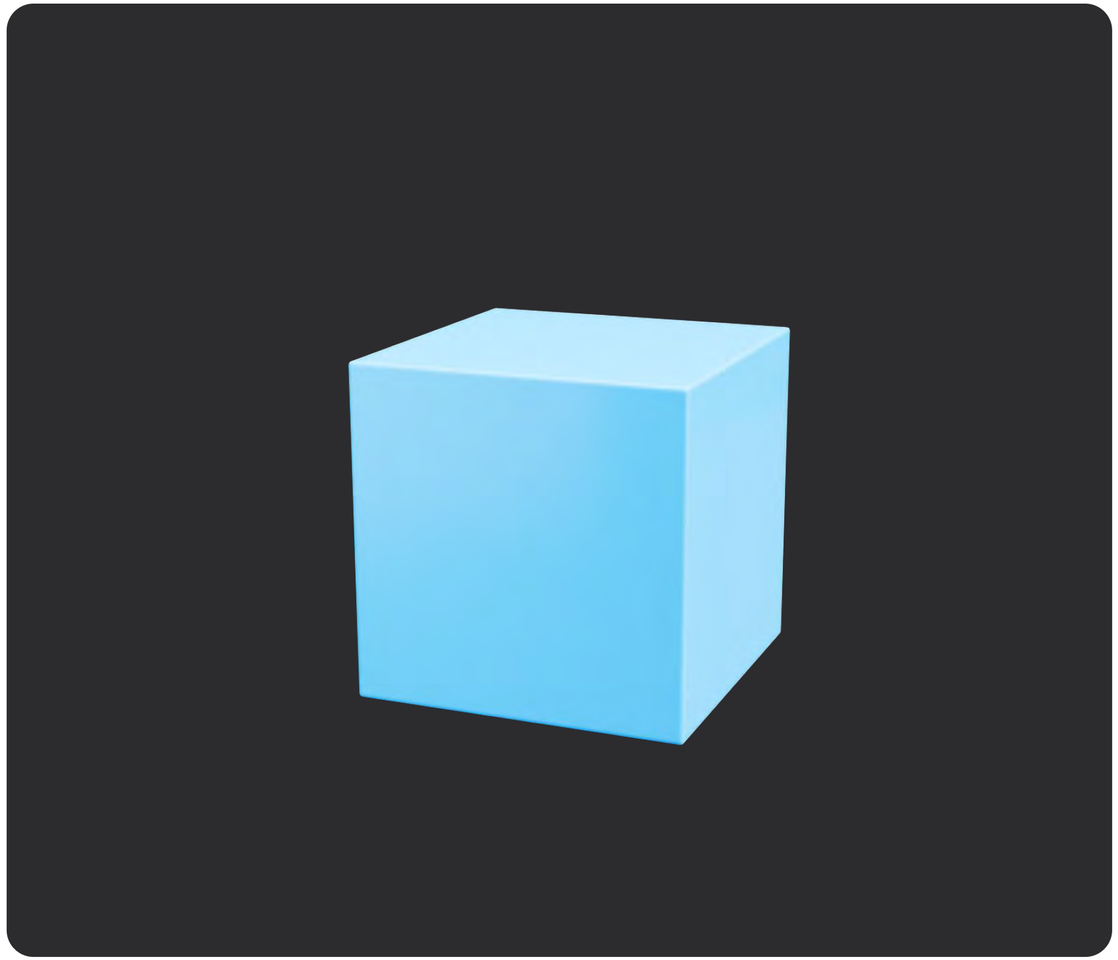
Three.js Light

```
light = new THREE.DirectionalLight(color, intensity);
light.position.set(1, 1, 1)
scene.add(directionalLight);
```



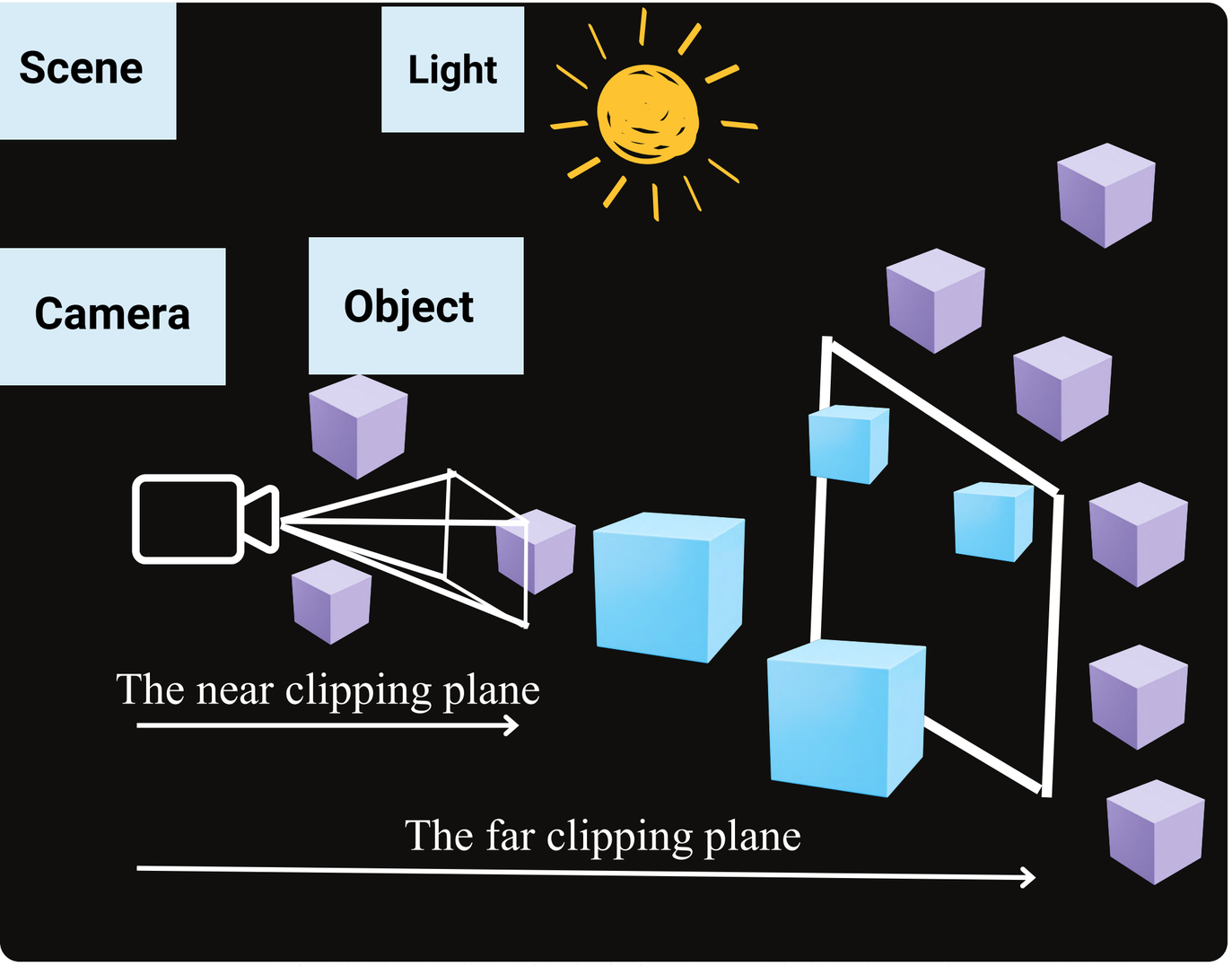
Three.js Renderer

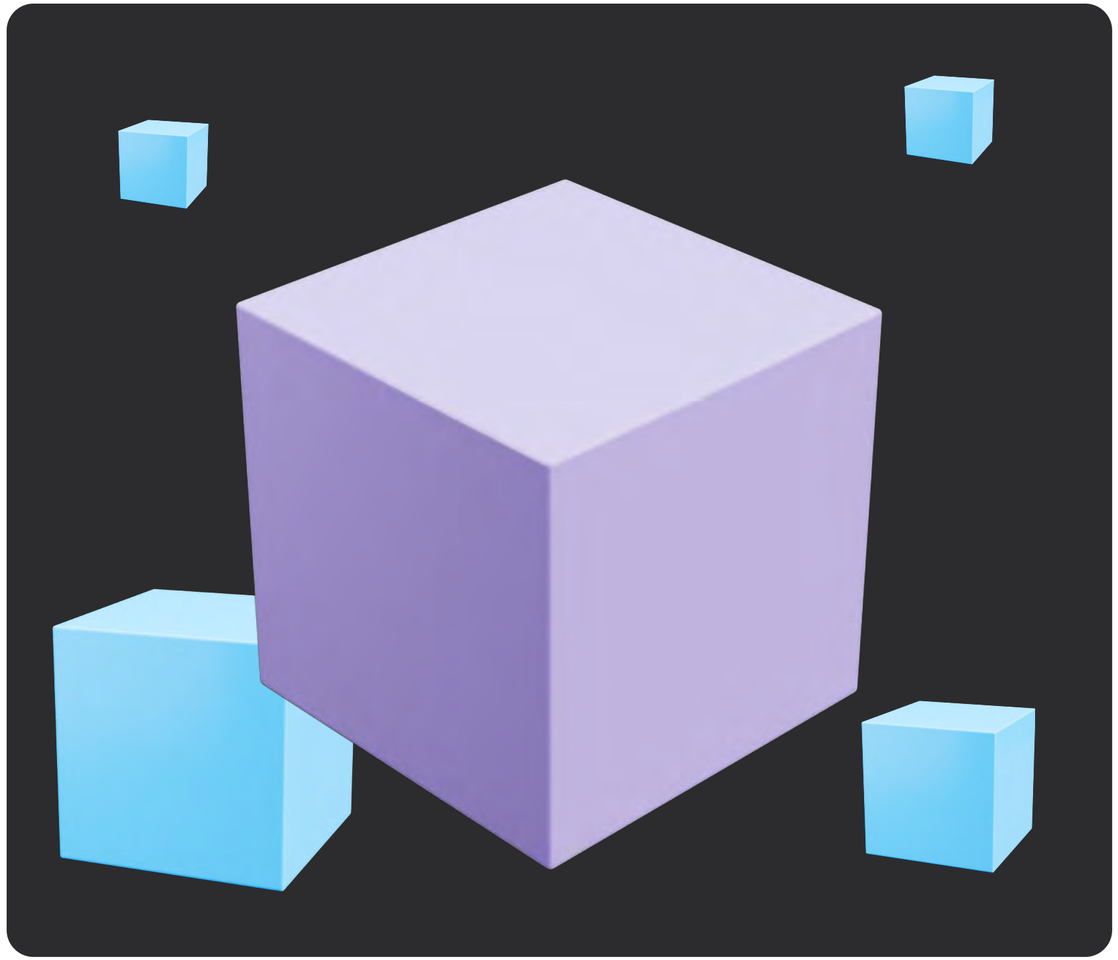




Rendered screen

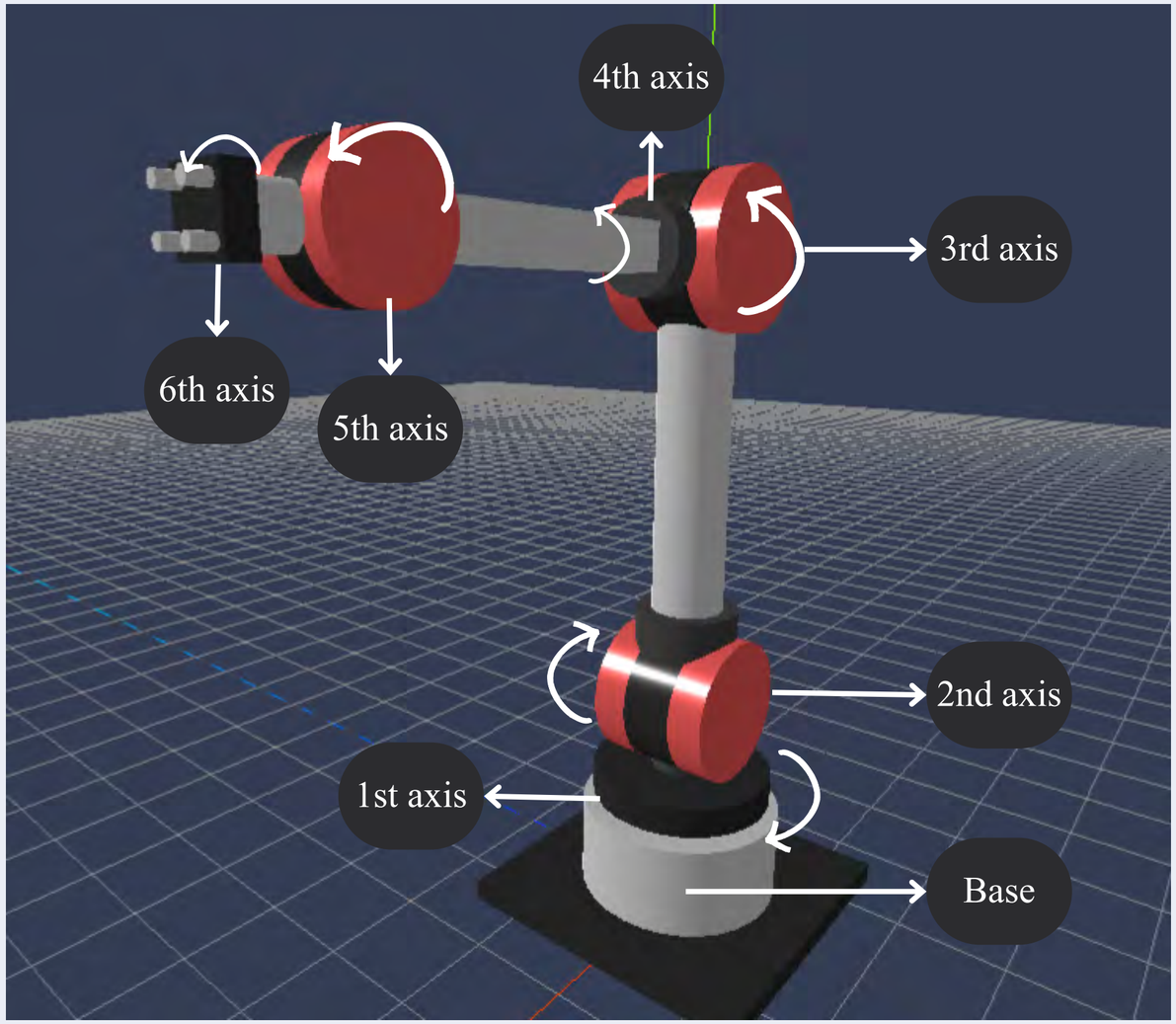
The display of PerspectiveCamera in Three.js

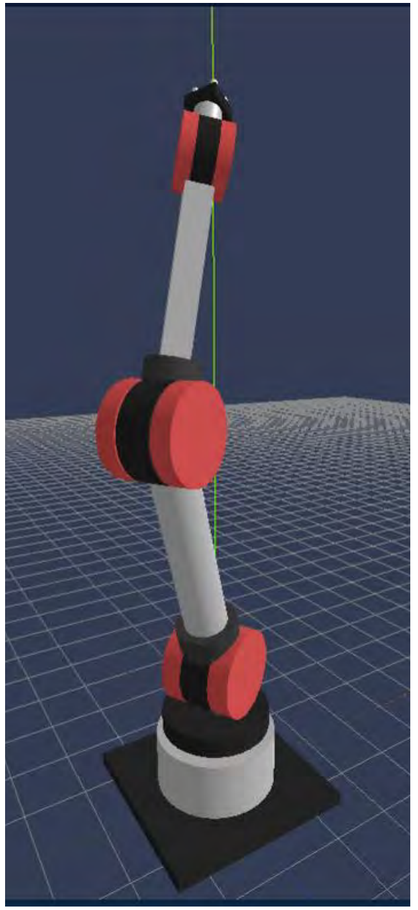


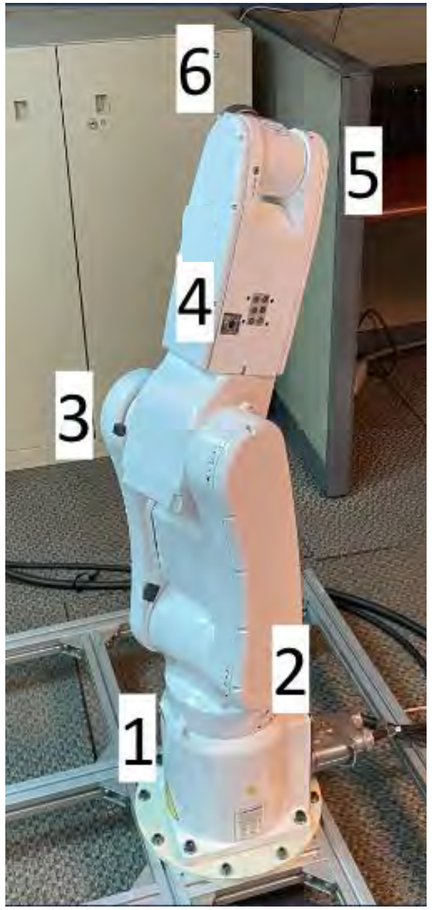


Rendered screen

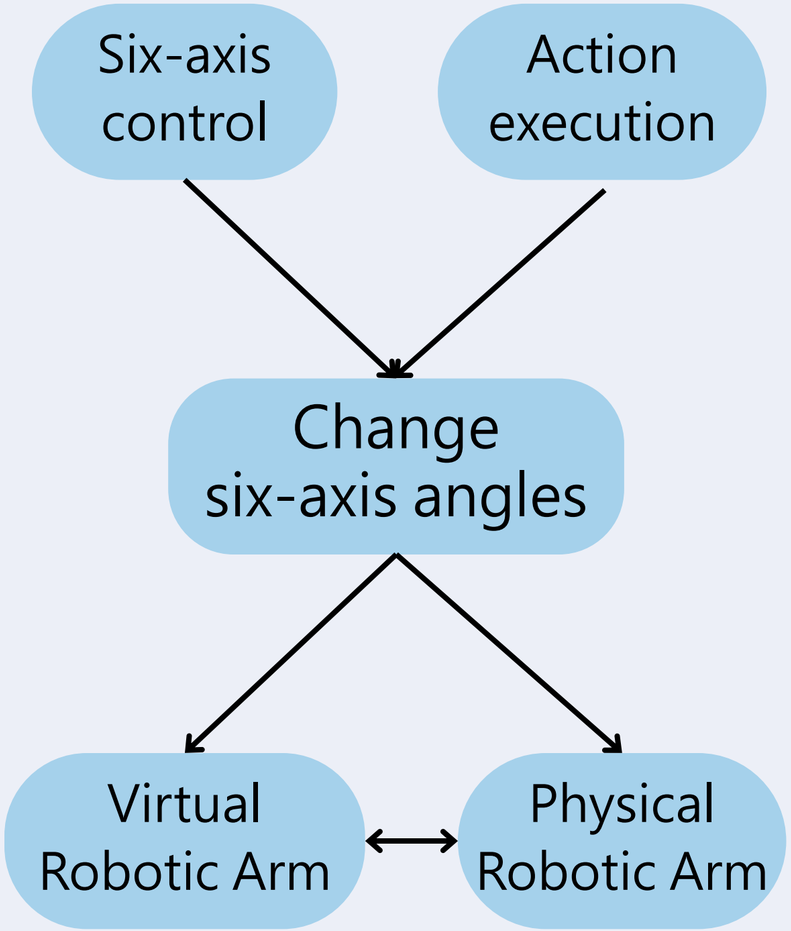
Integration of robotic arm with Three.js virtual environment

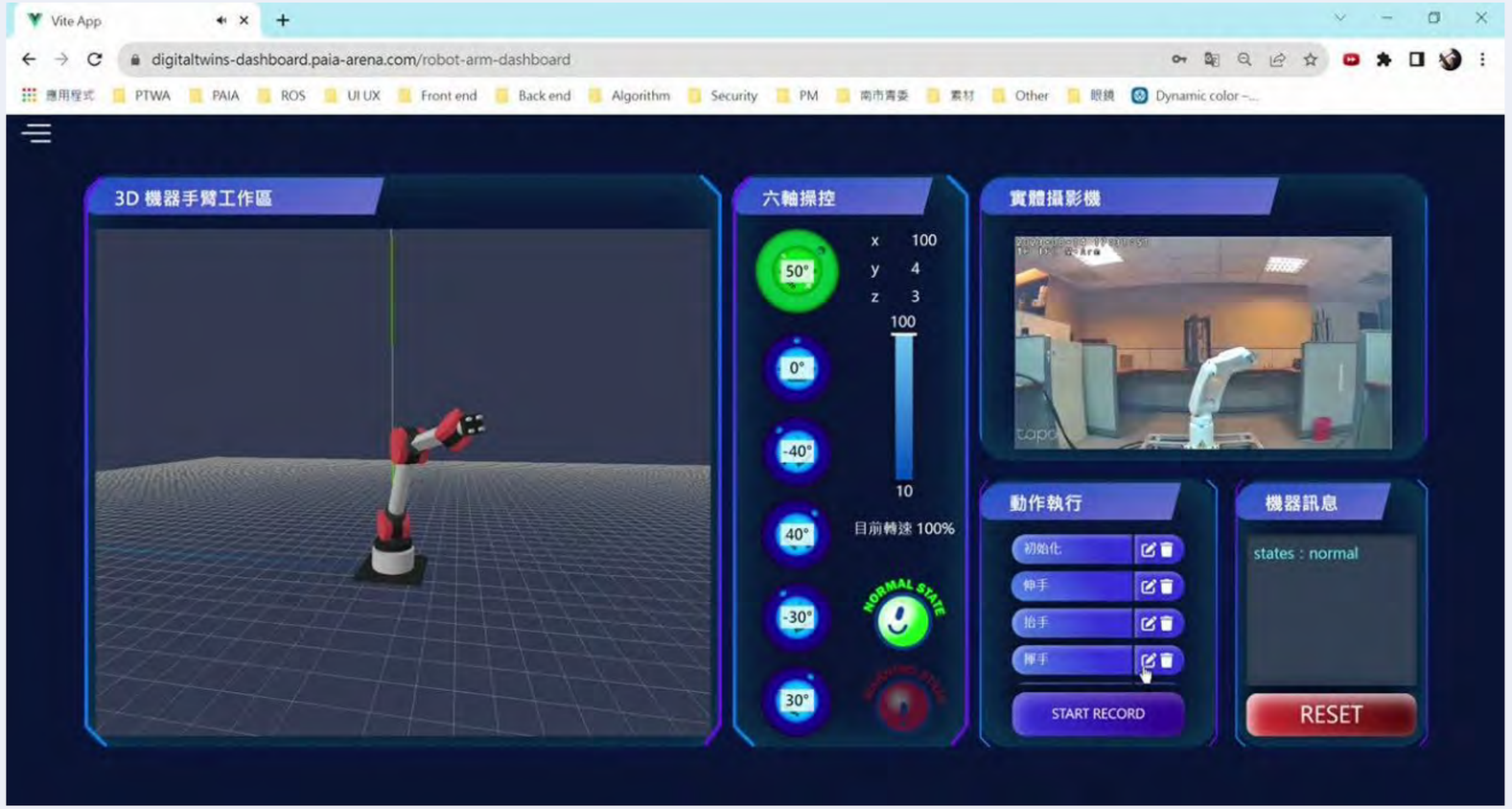






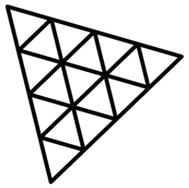
How to achieve synchronization of virtual and physical?



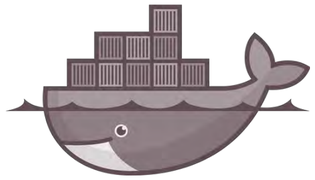


WebRTC

FrontEnd

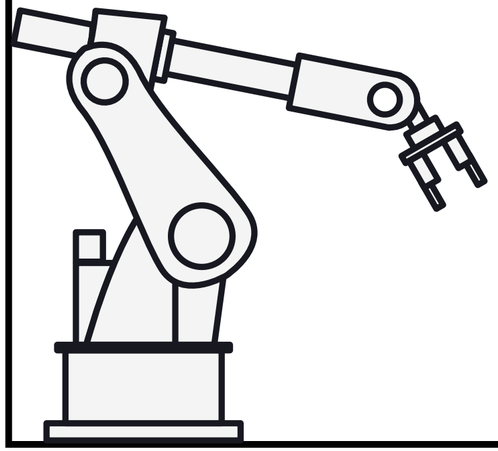
DevOps



Jenkins

BackEnd


Local

WebRTC bind IP cam – Hardware

TP-Link Tapo C320WS

- Support RTSP protocol

tp-link 家用產品 智慧家庭系列 商用產品 ISP用產品 優惠活動

How to view the IP camera on computer?

設定指南 更新10-13-2023

這篇文章適用於:

rtsp://username:password@<IP address>:554/stream1

Tapo攝影機的RTSP即時串流的URL(網址)為：

對於1080P（1920 * 1080）串流：rtsp：//使用者名稱：密碼@IP位址：554 / stream1

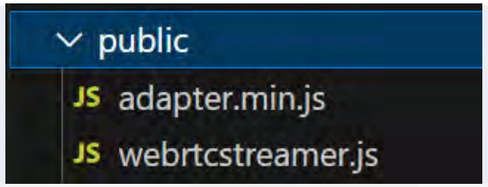
對於360P（640 * 360）串流：rtsp：//使用者名稱：密碼@IP位址：554 / stream2

WebRTC bind IP cam - Web

webrtc-streamer

- Supports RMTP/RTSP protocols
- Starts up with an HTTP server
- Compatible with Windows and Linux
- Provides Docker image

index.html




```
<script src="/adapter.min.js"></script>
<script src="/webrtcstreamer.js"></script>
```

README.md

WebRTC-Streamer

circleci passing | cirrusci passing | webrtc-streamer latest/stable v0.2.8+git451.2d0afce

C/C++ ci linux passing | C/C++ ci windows passing | C/C++ ci macos passing

release v0.8.3 | downloads 42k | docker pulls 674k

WebRTC bind IP cam - Web

```
data() {
  return {
    webRtcServer: null,
    webcamIp: import.meta.env.VITE_CAMERA_URL,
    webrtcIp: import.meta.env.VITE_WEBRTC_URL ||
      `${location.protocol}//${window.location.hostname}:8000`,
  }
},
mounted() {
  this.webRtcServer = new WebRtcStreamer('video', this.webrtcIp)
  this.webRtcServer.connect(this.webcamIp, null, "rtptransport=tcp&timeout=60")
},
beforeDestroy() {
  this.webRtcServer.disconnect()
  this.webRtcServer = null
},
```

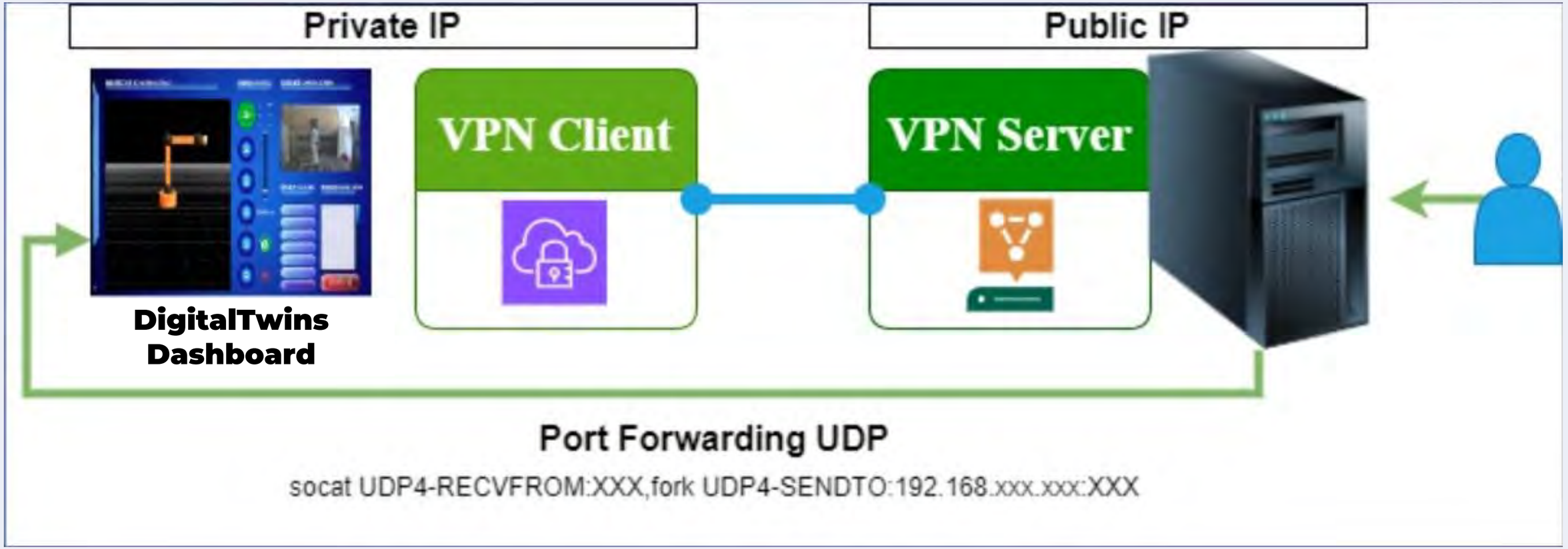
WebRTC bind IP cam - issues

The difference between webrtc-streamer in development and production.

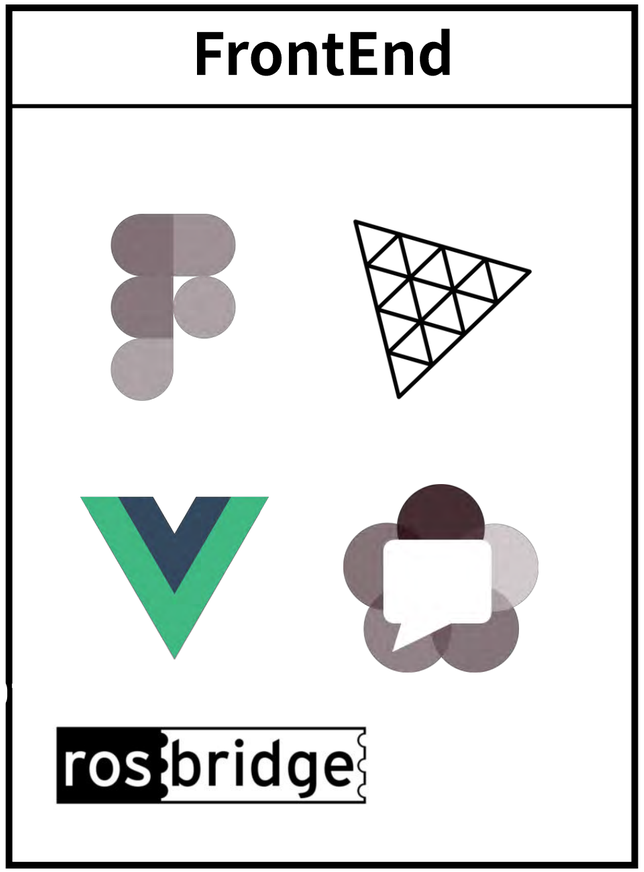
- During development：
  - The API server of webrtc-streamer runs locally.
- production：
  - The dashboard of DigitalTwins system and the API server of webrtc-streamer must be deployed simultaneously.
    - ➢ Solution: docker-compose
  - The screen on the website appear black during development
    - ➢ Solution: Change UDP to TCP for connections.
- IP Cam without an external IP
  - ➢ Solution: port forwarding

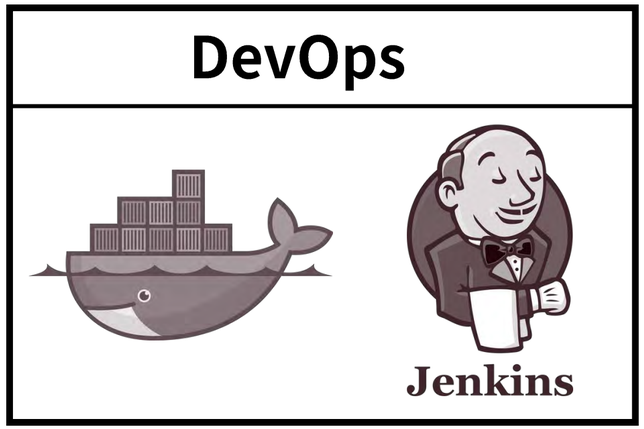
WebRTC bind IP cam - port forwarding

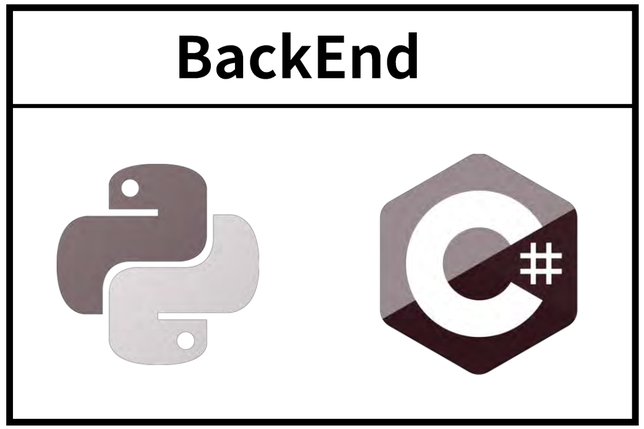
socat UDP4-RECVFROM:XXX,fork UDP4-SENDTO:192.168.xxx.xxx:XXX

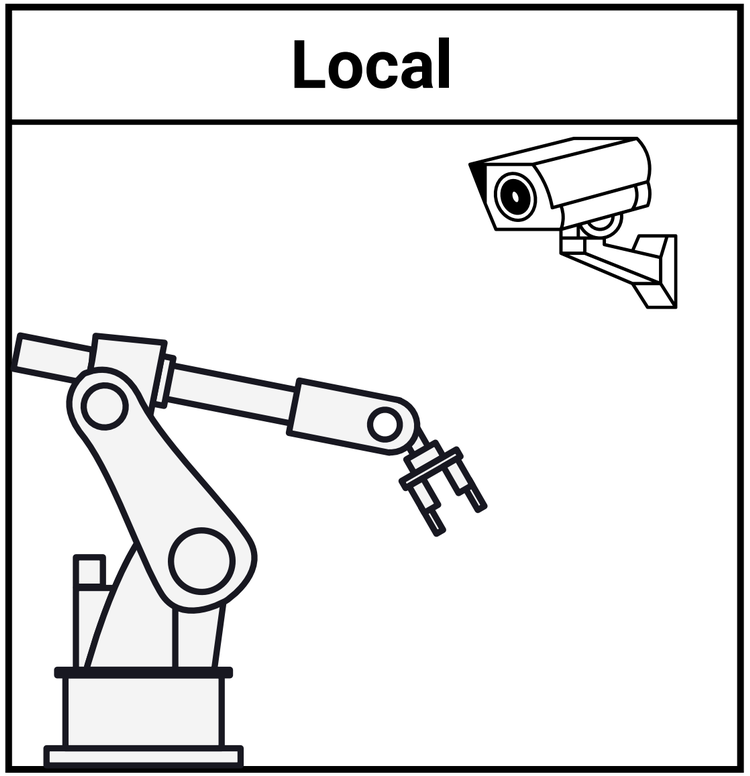


ROS bridge

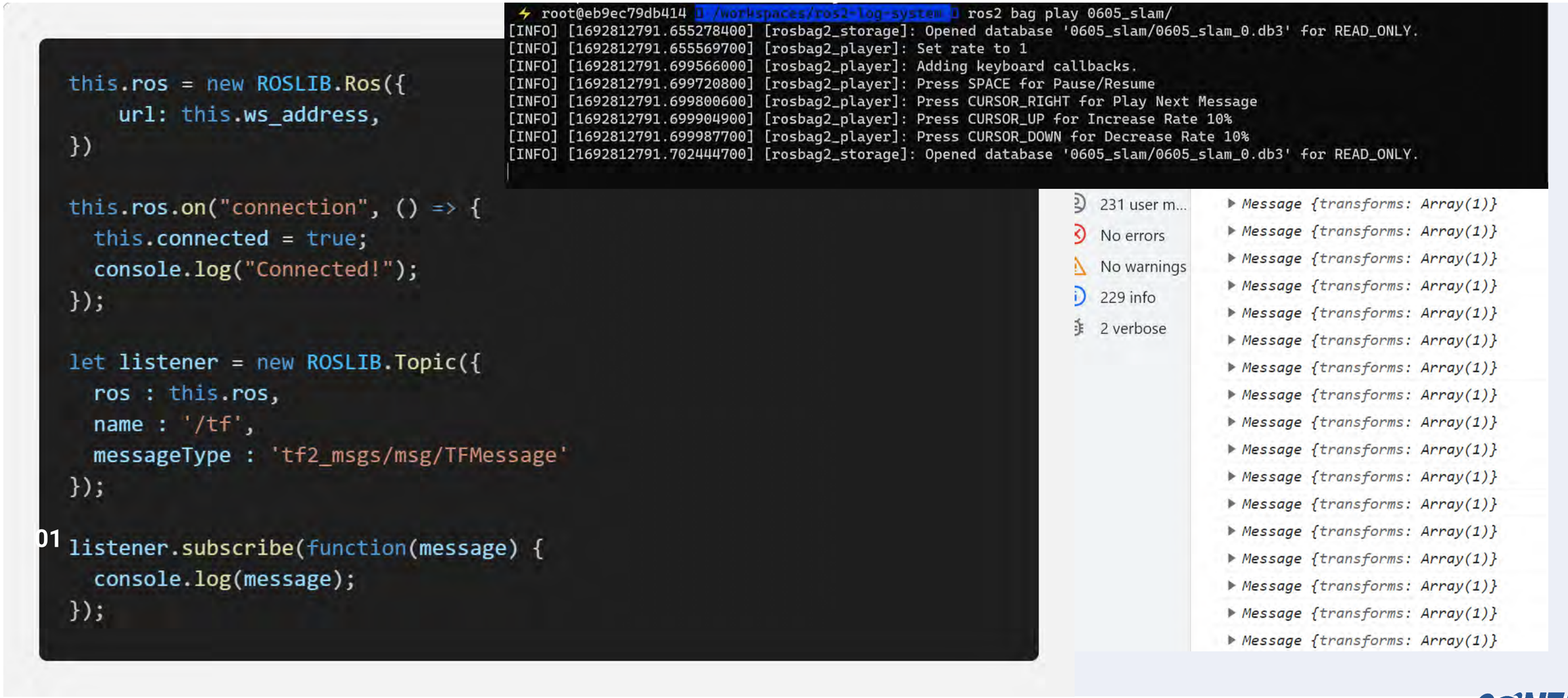








Integration with ROS cluster via Rosbridge



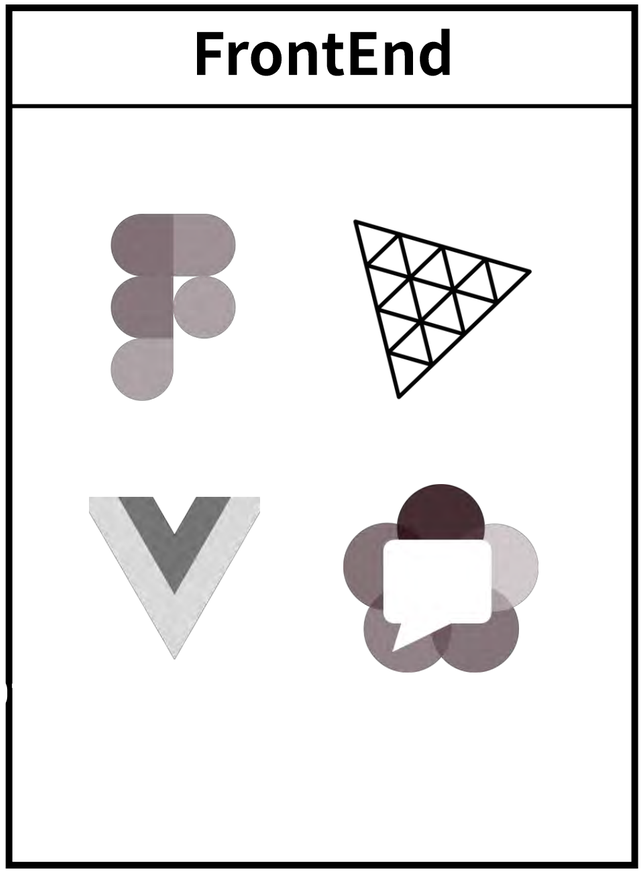
```
this.ros = new ROSLIB.Ros({
    url: this.ws_address,
})

this.ros.on("connection", () => {
  this.connected = true;
  console.log("Connected!");
});

let listener = new ROSLIB.Topic({
  ros : this.ros,
  name : '/tf',
  messageType : 'tf2_msgs/msg/TFMessage'
});

listener.subscribe(function(message) {
  console.log(message);
});
```

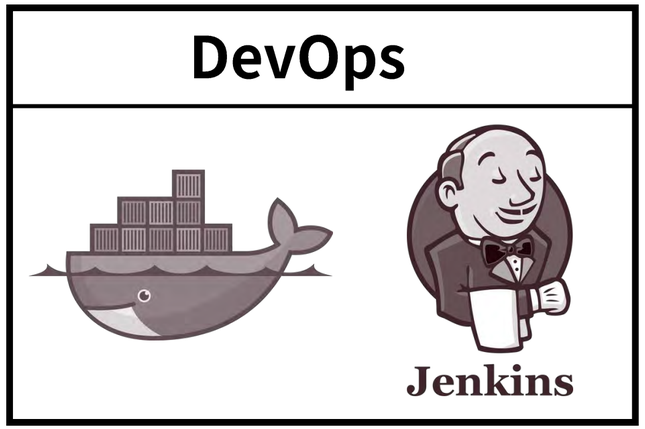
```
root@eb9ec79db414 /workspaces/ros2-log-system ros2 bag play 0605_slam/
[INFO] [1692812791.655278400] [rosbag2_storage]: Opened database '0605_slam/0605_slam_0.db3' for READ_ONLY.
[INFO] [1692812791.655569700] [rosbag2_player]: Set rate to 1
[INFO] [1692812791.699566000] [rosbag2_player]: Adding keyboard callbacks.
[INFO] [1692812791.699720800] [rosbag2_player]: Press SPACE for Pause/Resume
[INFO] [1692812791.699800600] [rosbag2_player]: Press CURSOR_RIGHT for Play Next Message
[INFO] [1692812791.699904900] [rosbag2_player]: Press CURSOR_UP for Increase Rate 10%
[INFO] [1692812791.699987700] [rosbag2_player]: Press CURSOR_DOWN for Decrease Rate 10%
[INFO] [1692812791.702444700] [rosbag2_storage]: Opened database '0605_slam/0605_slam_0.db3' for READ_ONLY.
```

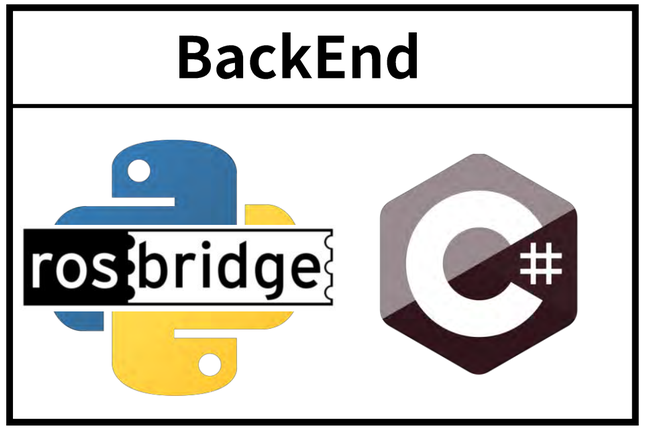
231 user m...
No errors
No warnings
229 info
2 verbose

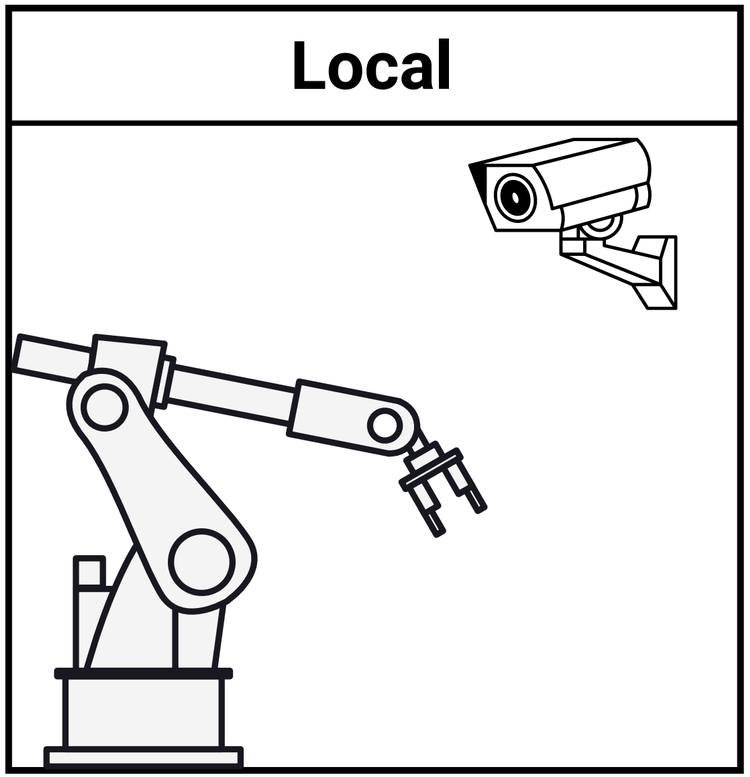
Message {transforms: Array(1)}

Backend - Python







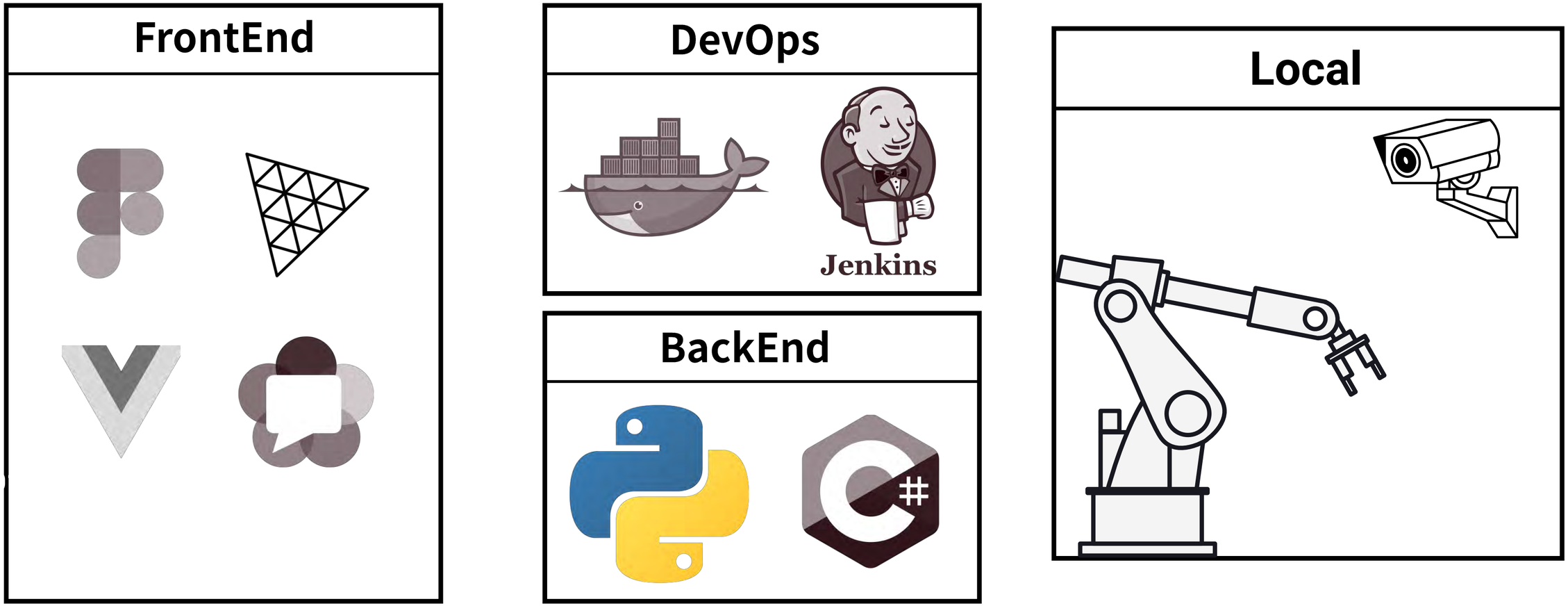


Subscribe ROS message in Python

```
        self.subscription = self.create_subscription(
            String,
            'topic',
            self.listener_callback,
            10)
```

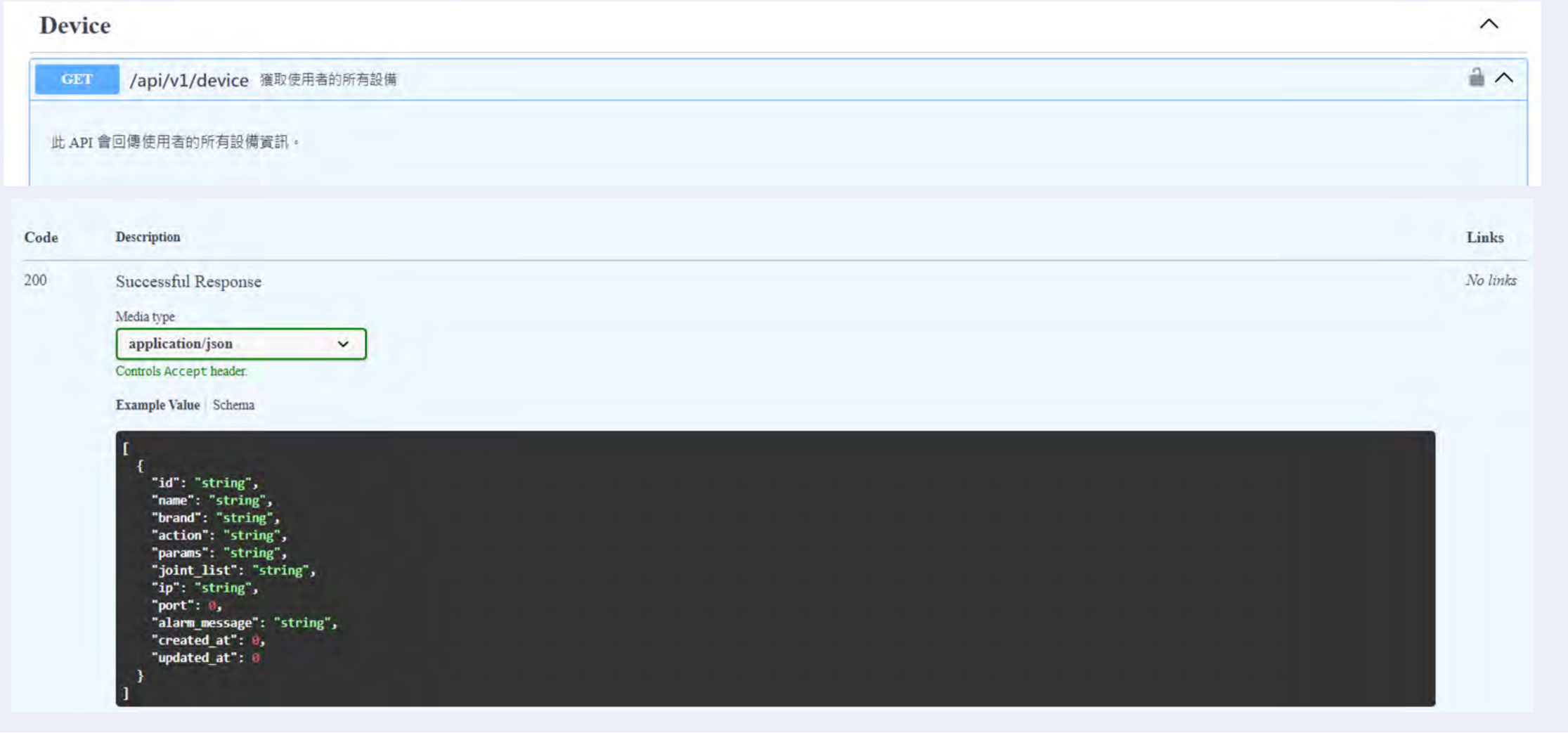
```
    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)
        data = json.loads(msg.data)
        alarm_code = data.get("alarm_code", None)
        if alarm_code != None:
            self.get_logger().info('Alarm Code: "%s"' % alarm_code)
            response = device_get(self.DEVICE_ID, self.token)
            device_patch(self.DEVICE_ID, {
                "name": response["name"],
                "brand": response["brand"],
                "ip": response["ip"],
                "port": response["port"],
                "alarm_message": alarm_code
                }, self.token)

        if "jointAngles" in data:
            jRot = data["jointAngles"]
            jPos = data["jointPos"]
            rpm = data["rpms"]
            torque = data["torqueValues"]
            self.get_logger().info('Joint Rot: "%s", Pos: "%s", RPM: "%s", Torque: "%s"' % jRot % jPos % rpm % torque)
            print(jRot, jPos, rpm, torque)
```

Backend - Python



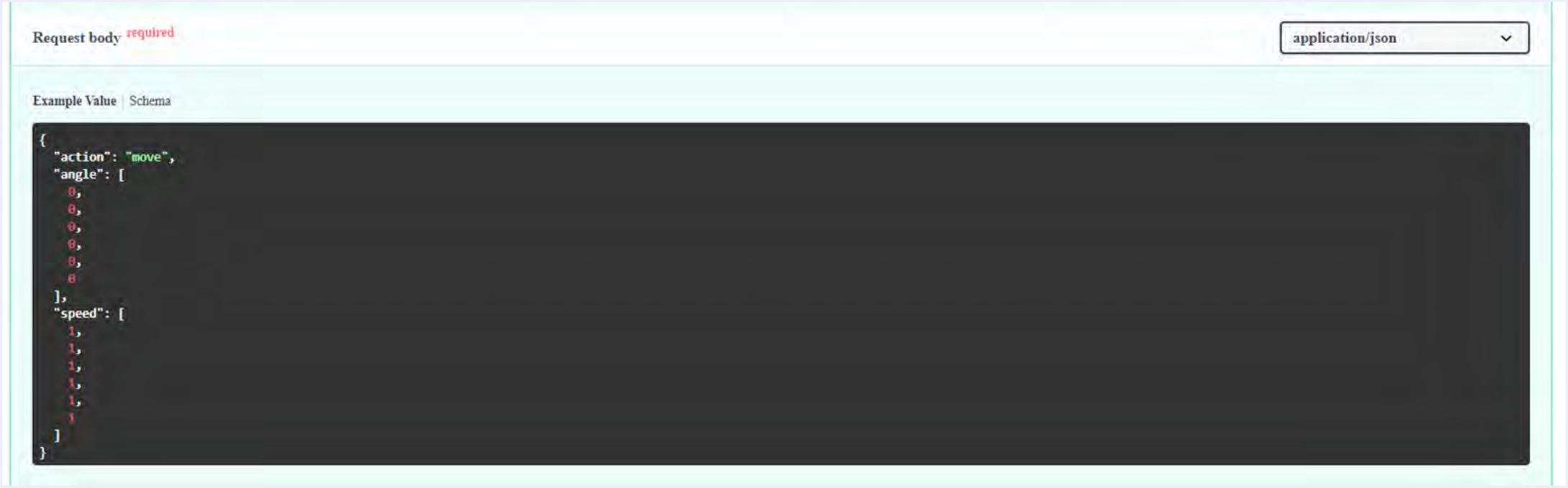
ROS Cloud API

ROS Cloud API



ROS Cloud API

- /api/v1/device
- /api/v1/device/{id}
- /api/v1/device/{id}/action



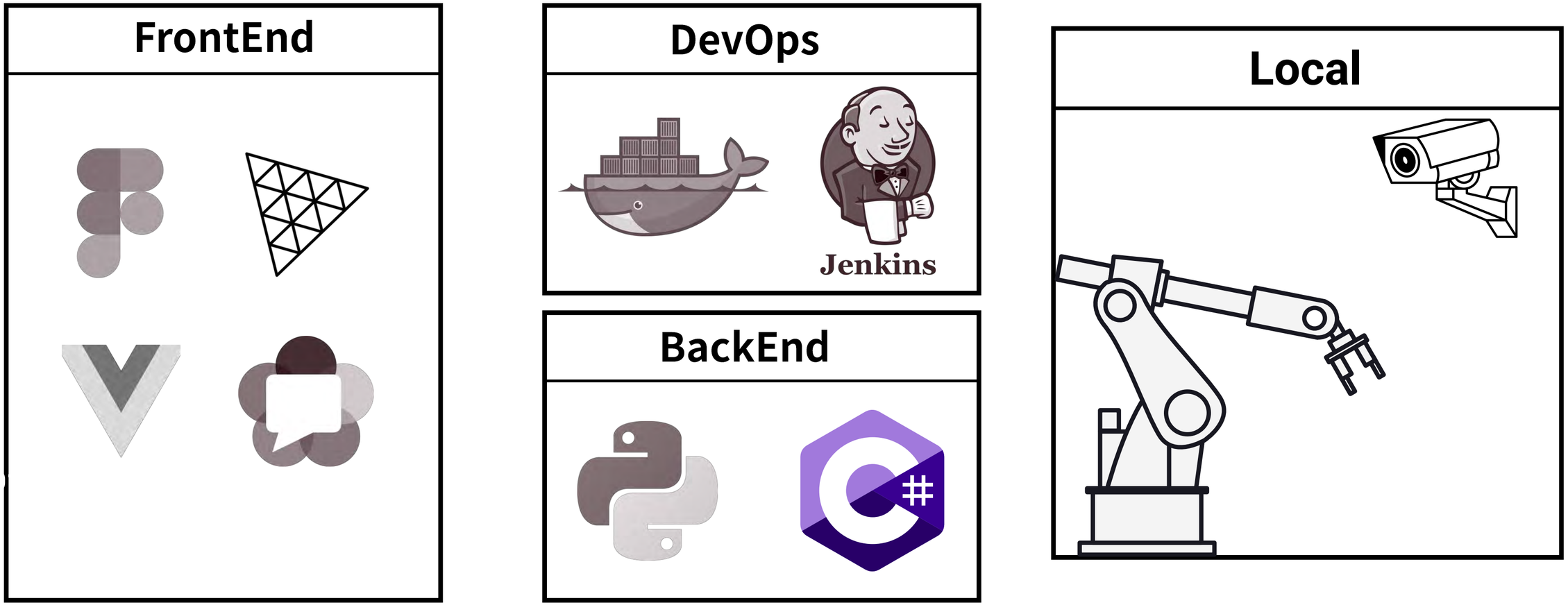
Python

- **Call ROS Cloud API**

```
if compare2angleList(response_angle, current_angle):
      try:
        await websocket.send(str(message))
        print("ws_sent")
      except websockets.ConnectionClosedOK:
        print("websockets ConnectionClosedOK error")
```

```
message = await asyncio.wait_for(websocket.recv(), timeout=0.1)
        data = json.loads(message)
        device_action_patch(DEVICE_ID, {
              "action":"move",
              "angle": data["jointAngles"]
              }, token)
```

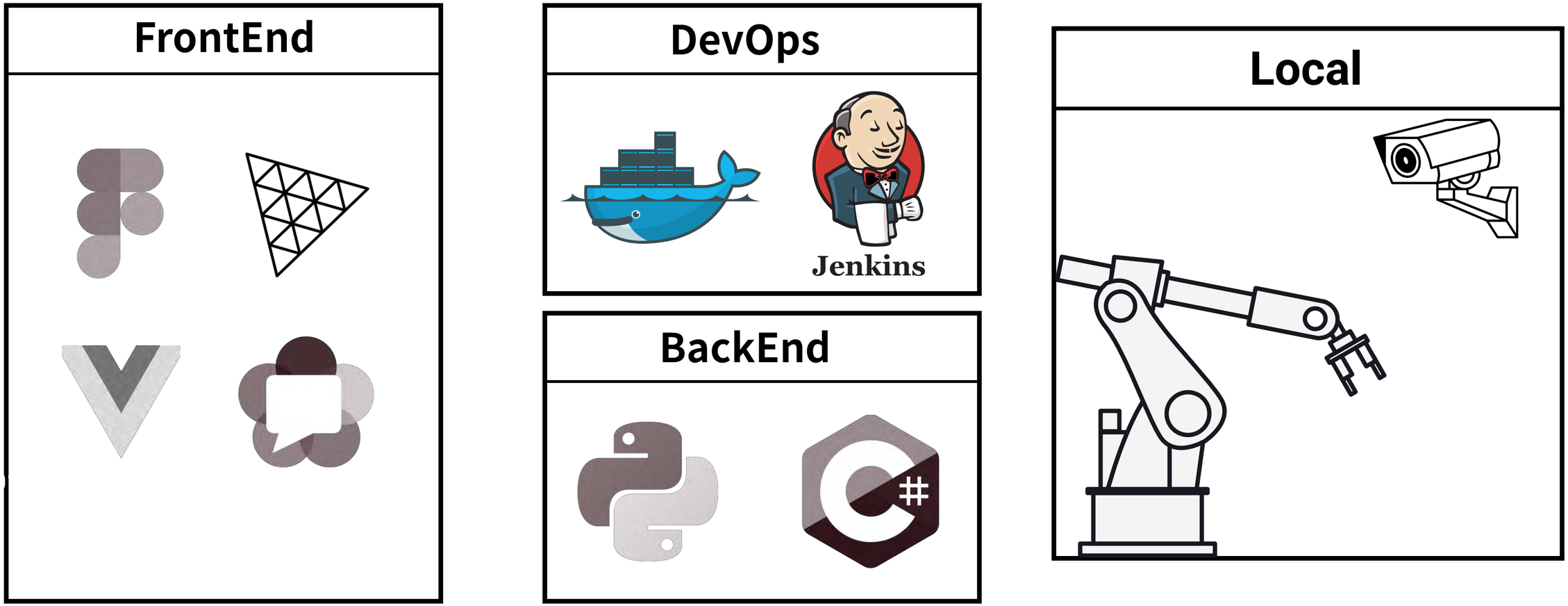
Backend – C#



C# SDK to ROS node

```
Movement_handle.RunPosAxis(a1to6.Take(6).ToArray());
Movement_handle.Current_Angles(jointAngles);
Movement_handle.Current_Pos(jointPos);
Movement_handle.Current_rpm(rpms);
Movement_handle.Motor_torque(torqueValues);
```

DevOps



Cloud and on-premises deployment

Jenkins & Dockerize – Build Image

```
        stage('Build Image') {
            steps {
                        sh "rm -r -f ${repo_name}"
                        sh "git clone -b ${BRANCH_NAME}
https://${GITLAB_API_TOKEN}@${GITLAB_URL}:${GITLAB_PORT}/${repo_name}.git"
                        sh "cd ${repo_name} && git pull origin ${BRANCH_NAME}"
                        sh "cd ${repo_name} && npm install"
                        script {
                            writeFile file: "${repo_name}/.env", text: "VITE_API_BASE_URL=${ROS_API}" +
"\n" + "VITE_CAMERA_URL=rtsp://${VITE_CAMERA_URL}"
                        }
                        sh "cd ${repo_name} && npm run build"
                        sh "cd ${repo_name} && docker build -t ${image_name}:${version} -f ./Dockerfile ."
                }
            }
        }
```

Cloud and on-premises deployment

Jenkins & Dockerize - Deploy

```
        stage('Deploy to EC2 via SSH') {
            steps {
                withAWS(credentials:'credentials') {
                    wrap([$class: 'MaskPasswordsBuildWrapper',
                     varPasswordPairs: [[password: AWS_ACCESS_KEY_ID], [password: AWS_SECRET_ACCESS_KEY]]]) {
                            sshagent(credentials: ['ssh-credentials-id']) {
                               sh """
                                   [ -d ~/.ssh ] || mkdir ~/.ssh && chmod 0700 ~/.ssh
                                   ssh-keyscan -t rsa,dsa ${host} >> ~/.ssh/known_hosts
                                   ssh ubuntu@${host} 'aws ecr get-login-password --region ${region} | docker
login --username AWS --password-stdin ${registry} && docker pull ${registry}/${image_name}:${version} && rm
-r -f ${repo_name} && git clone -b ${BRANCH_NAME}
https://${GITLAB_API_TOKEN}@${GITLAB_API_URL}/ros/${repo_name}.git && cd ${repo_name} && git pull origin
${BRANCH_NAME} && cp /env/.env .env && docker-compose --env-file .env up -d'
                               """
                           }
                    }
                }
            }
        }
```

Conclusion

- Basic concepts of ROS and Three.js
- Integration of DigitalTwins system with Virtual and Physical environments
- Cloud-based operation interface integrates with ROS cloud APIs and on-premises webcams
- Deployment mechanism using Docker + Jenkins