Assignment#2

25.02.10

0. OOP

https://velog.io/@hkoo9329/OOPObject-Oriented-Programming-%EA%B0%9D%EC%B2%B4-%EC%A7%80%ED%96%A5-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%EC%9D%B4%EB%9E%80

1.캡슐화 : 특정한 목적을 위한 필요한 변수나 메소드를 하나로 묶어 private 한 정보 접근 제한 (getter, setter) 등의 함수를 이용하여 접근가능케

2.추상화 : 객체들의 “공통적”인 부분을 묶어 공통사용

3.다형성 :

4.상속성, 재사용성 : 기존 클래스에 기능을 가져와 재사용할 수 있으면서도 동시에 새롭게 만든 클래스에 새로운 기능을 추가

1. For ffmpeg

변수:

IP 주소(private), ffmpeg, 재접속시도횟수, 재접속시도간격

기능:

재접속시도(함수), 프레임영상 불러오기(반복), 시각화

2. For gstreamer

변수:

IP 주소(private), codec(h264), latency(=200ms)

기능:

Gstreamer element(source, depay, parse, ...)간 연결,

element 집합인 pipeline 정의,

pad 정의(Pad : GStreamer 에서 **pad (패드, Pad)** 는 요소(element) 간의 데이터 연결점),

새로운 프레임 이미지 하나씩 읽어오는 on_new_sample,

이미지 한프레임씩 시각화,

에러메세지 표출,

위의 모든 기능 run 에 통합

[Additional Tasks]
- PYGI 에서 frame 을 실시간으로 가져오는 메소드 확인
- PYGI 코드에서 Frame 을 가져와 Display 하는 방식으로 코드 수정

```python
def on_new_sample(self, sink):
    """appsink로부터 새 샘플(프레임)이 들어오면 호출됩니다."""
    sample = sink.emit("pull-sample")
    if sample:
        buf = sample.get_buffer()
        caps = sample.get_caps()
        structure = caps.get_structure(0)
        width = structure.get_value("width")
        height = structure.get_value("height")

        # 버퍼 데이터를 읽기 전용으로 매핑
        success, map_info = buf.map(Gst.MapFlags.READ)
        if not success:
            print("버퍼 매핑 실패")
            return Gst.FlowReturn.ERROR

        # NumPy 배열로 변환 (BGR 포맷으로 캡스 설정했다고 가정)
        frame = np.frombuffer(map_info.data, dtype=np.uint8)
```

Gstreamer 에서 rstpsrc 기반 파이프라인을 생성하고 appsink 를 사용하여 새 프레임을 버퍼에 저장 및 numpy 로 불러옴.

3. Compare

메모리, 타임로그, cProfiler

GStreamer 의 **Bus** 를 활용하여 비동기 이벤트 기반 처리로 전환합니다.

예를 들어, 파이프라인의 상태 변화나 에러 메시지를 Bus 로 받아 처리하면 time.sleep()으로 기다리지 않아도 됩니다.

이를 통해 파이프라인이 실패하거나 중단될 때 즉시 콜백 함수로 처리할 수 있으며, 별도의 폴링(loop) 없이 재접속 로직을 구현할 수 있습니다.

OpenCV 자체는 GStreamer 의 Bus 나 GLib 의 MainLoop 처럼 내장된 이벤트 루프를 제공하지 않기 때문에, 비동기 처리를 위해서는 별도의 기법(예: 스레드, asyncio, 큐 등)을 활용해야 합니다.