RCVBadge1-Grabber

멘토:김태주,한대찬
멘티:정찬호

목차

- Grabber란
  - Camera system
  - Camera synchronize
- 나만의 Grabber 만들기
  - Spinnaker SDK
  - My MFC Grabber
- 결과분석

Grabber란
-Camera system

- Handling Exposure->셔터와 조리개

셔터는 시간으로 노출을 조절 조리개는 들오는 빛의 양으로 조절

- 셔터:필름의 바로 앞면에 위치하여 평상시에는 완전히 펼쳐져 있으면서 필름에 닿을 빛을 차단,카메라의 찍는 버튼을 꾹 누르면 이 셔터가 순간적으로 열리면서 필름이 빛에 노출되는 것
- 셔터를 통한 노출 시간을 셔터 스피드 라고함
- 조리개

F넘버에 따른 조리개의 크기변화. 좌에서 우로 갈 수록 노출은 줄어든다.

Grabber란
-Camera synchronize

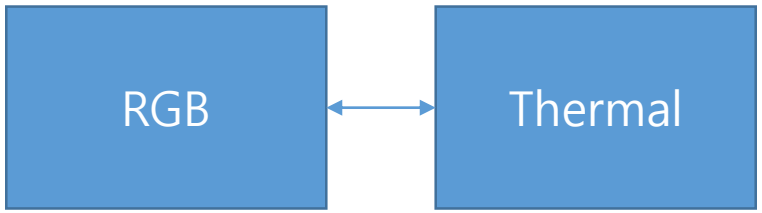
- 왜?

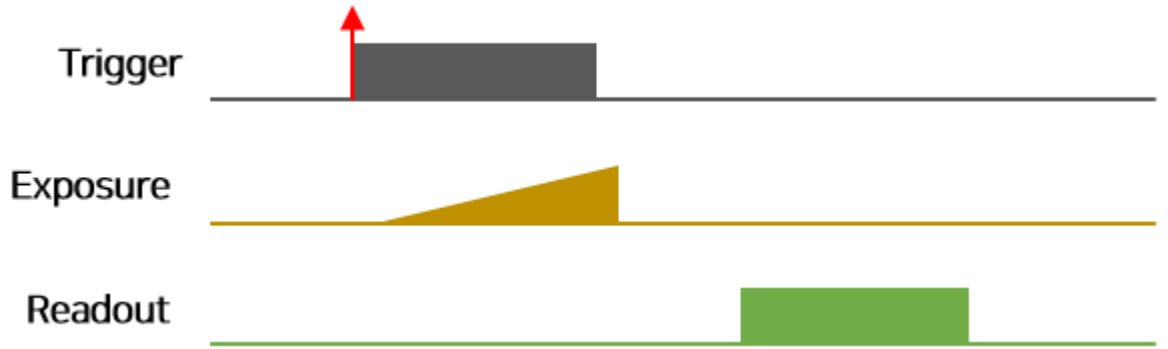
  현재 하나의 RGB 카메라 이미지로 detection model을 학습시켰을 때 보다 RGB,Thermal 두 개의 이미지로 학습시켰을 때 의미있는 성능향상을 보인다.

  따라서 원래 하나(RGB)의 였던 정보를 동일 '시각'에서의 추가 정보(Thermal)을 사용하기 위해선 두 개의 카메라가 동일 '시각'에 찍은 이미지가 필요하다.

- 어떻게?

  Hardware Trigger!(with electronic signal)



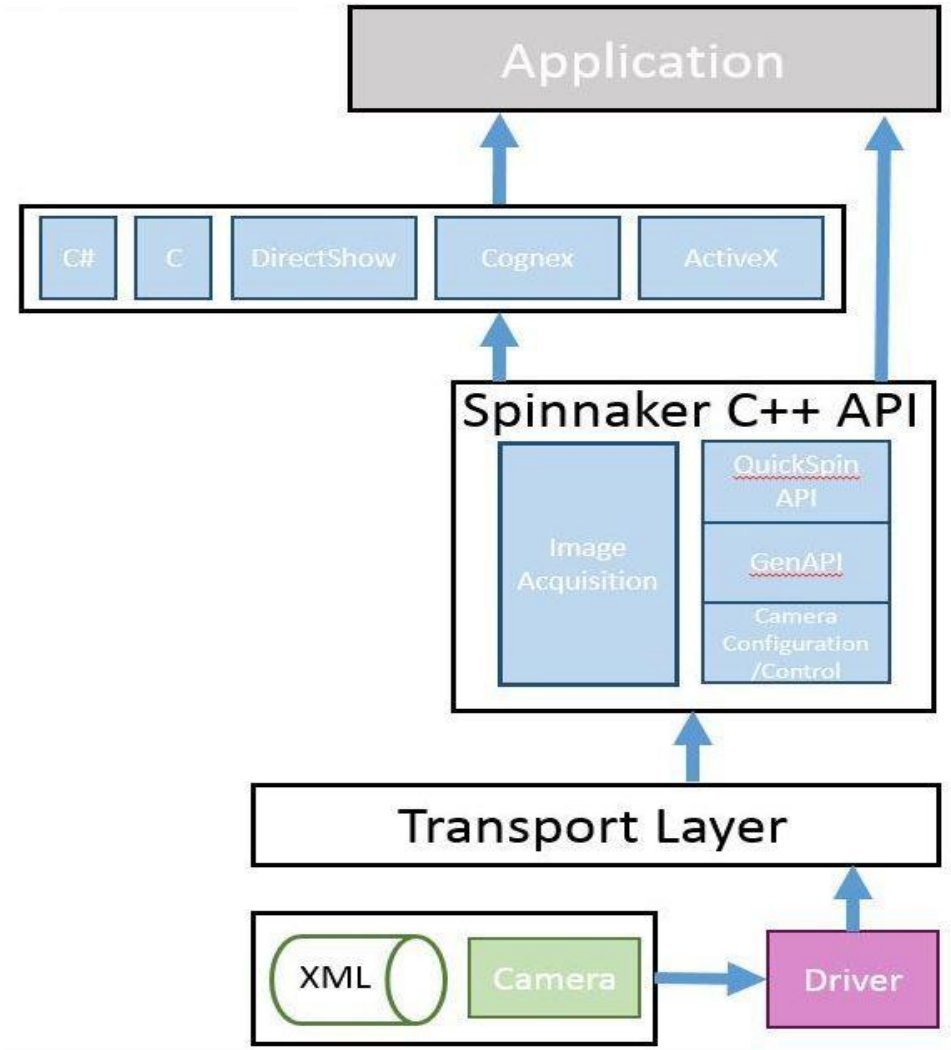


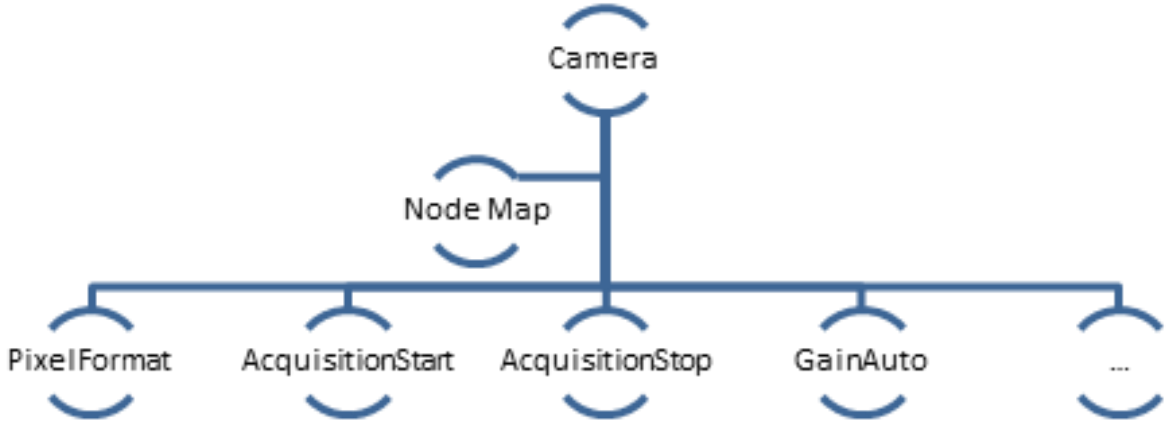
나만의 Grabber 만들기
-Spinnaker SDK

- 현재 매뉴얼에서 사용하고 있는 각각의 카메라처리를 위한 라이브러리로 FlyCapture SDK, eBUS SDK를 사용하고있는데 Spinnaker 라이브러리로 통합한다면 메뉴얼하여 전달하는 과정이 좀더 간략해지고 또한 python언어 지원도 하기 때문에 좀 더 쉽게 프로그래밍 할 수 도있다.

따라서 Spinnaker SDK를 사용하여 RGB,Thermal 카메라 동기화 하여 frame image를 grab할 수 있는 프로그램을 Spinnaker SDK 메뉴얼과 example들을 보며 MFC를 통해 c++로 만들어 보았다.

나만의 Grabber 만들기
-Spinnaker SDK



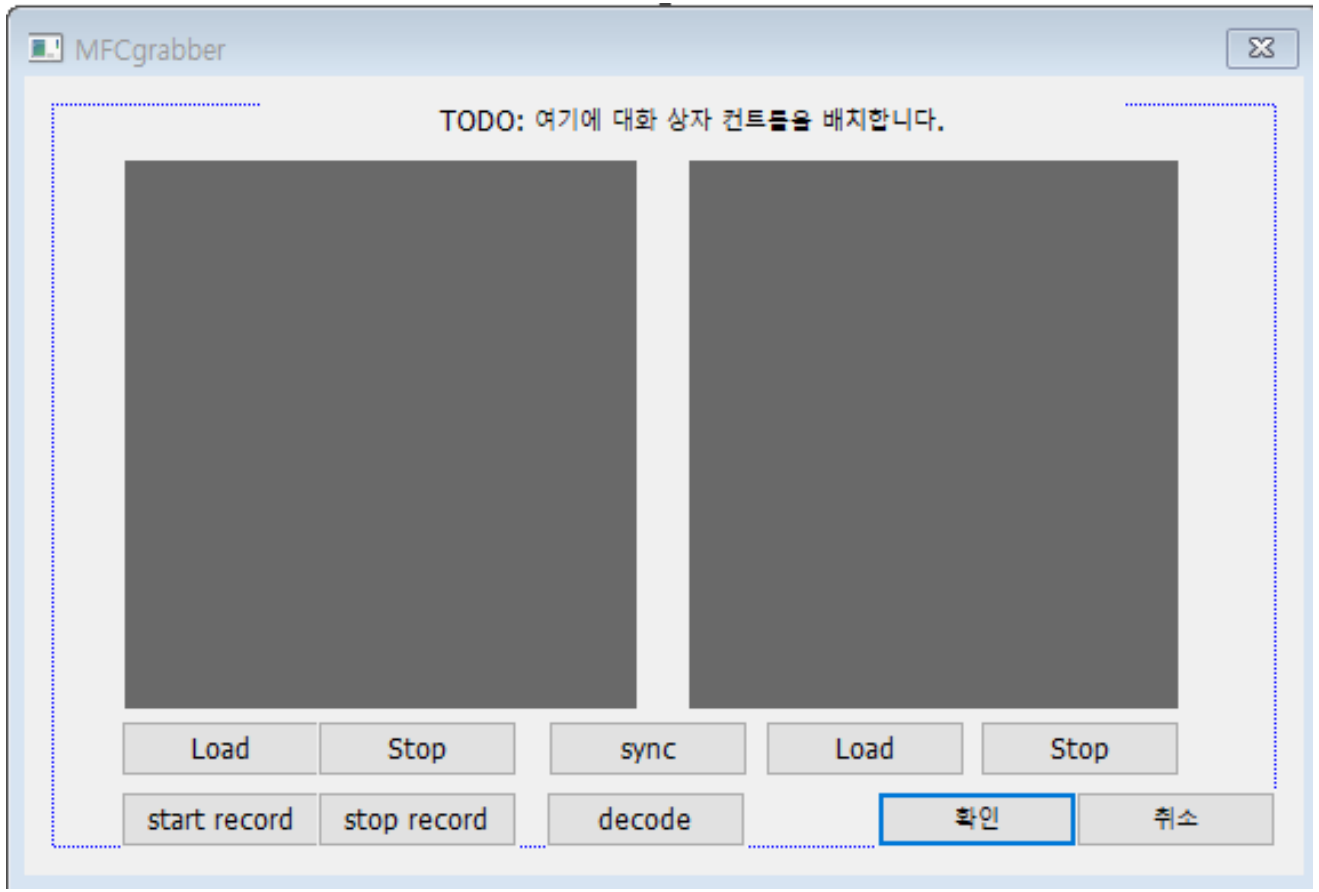



```
#include "Spinnaker.h"
#include "SpinGenApi#SpinnakerGenApi.h"
```

C++ GenAPI

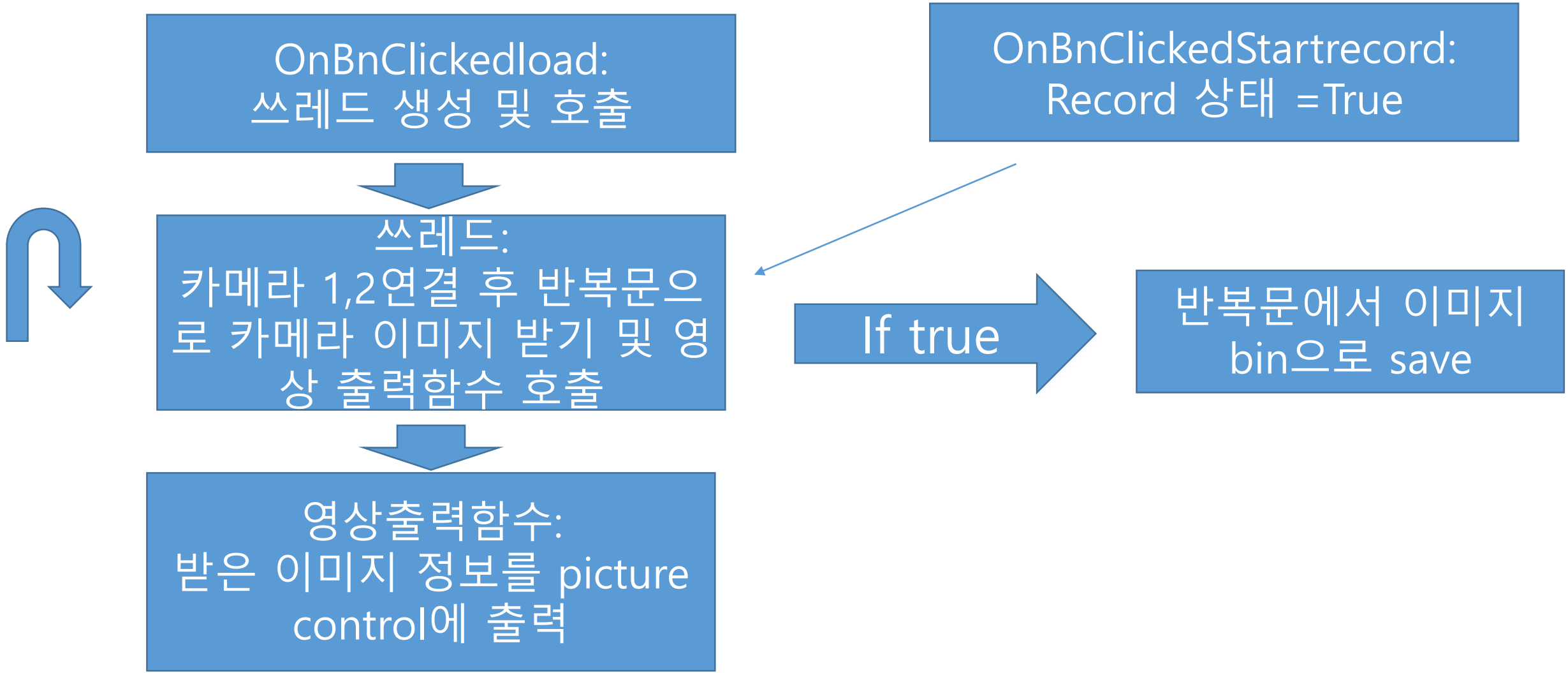
```
GenApi::INodeMap & nodeMap = cam.GetNodeMap();

CIntegerPtr width = nodeMap.GetNode("Width");

width->SetValue(new_width_val);
```

나만의 Grabber 만들기
-My MFC Grabber

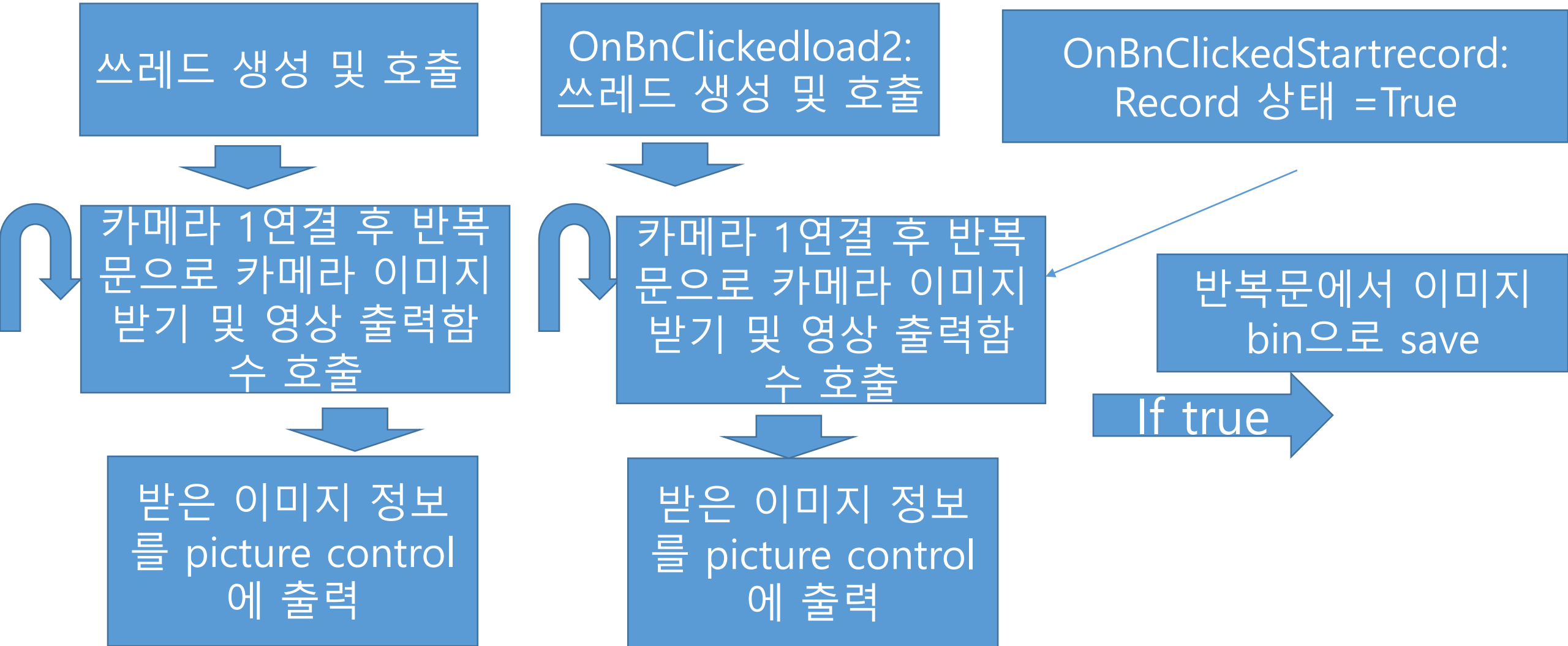


- Sync:카메라동기화
- Load:카메라연결,출력
- Stop:카메라해제
- Start record:save to bin
- Stop record:기록중지
- Decode:bin to png

나만의 Grabber 만들기
-My MFC Grabber-Process-one-thread

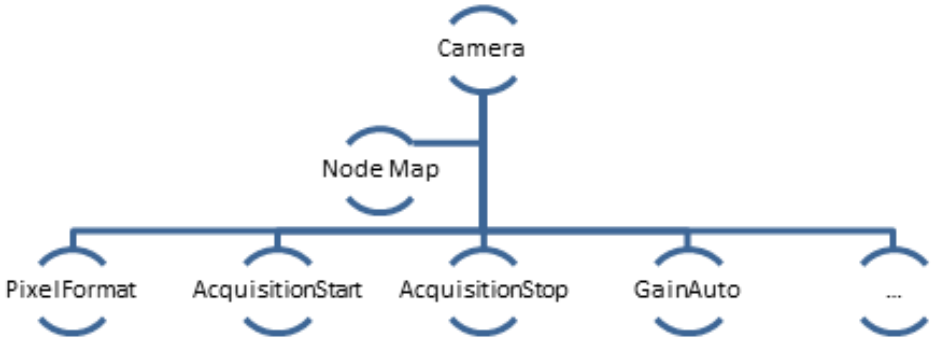


나만의 Grabber 만들기
-My MFC Grabber-Process-two-thread



나만의 Grabber 만들기
-My MFC Grabber-Sync:동기화

```
INodeMap& nodeMapTLDevice = pCam->GetTLDeviceNodeMap();
pCam->Init();
INodeMap& nodeMap = pCam->GetNodeMap();

CEnumerationPtr triggerMode = nodeMap.GetNode("TriggerMode");

triggerMode->SetIntValue(triggerMode->GetEntryByName("Off")->GetValue());

CEnumerationPtr exposureAuto = nodeMap.GetNode("ExposureAuto");
exposureAuto->SetIntValue(exposureAuto->GetEntryByName("Off")->GetValue());

CEnumerationPtr exposureMode = nodeMap.GetNode("ExposureMode");
exposureMode->SetIntValue(exposureMode->GetEntryByName("Timed")->GetValue());
```



나만의 Grabber 만들기
-My MFC Grabber-Sync:동기화

```
CFloatPtr exposureTime = nodeMap.GetNode("ExposureTime");
exposureTime->SetValue(1000);

triggerMode->SetIntValue(triggerMode->GetEntryByName("On")->GetValue());

CEnumerationPtr triggerSource = nodeMap.GetNode("TriggerSource");
triggerSource->SetIntValue(triggerSource->GetEntryByName("Line3")->GetValue());

CEnumerationPtr triggerActivation = nodeMap.GetNode("TriggerActivation");
triggerActivation->SetIntValue(triggerActivation->GetEntryByName("RisingEdge")->GetValue());
```

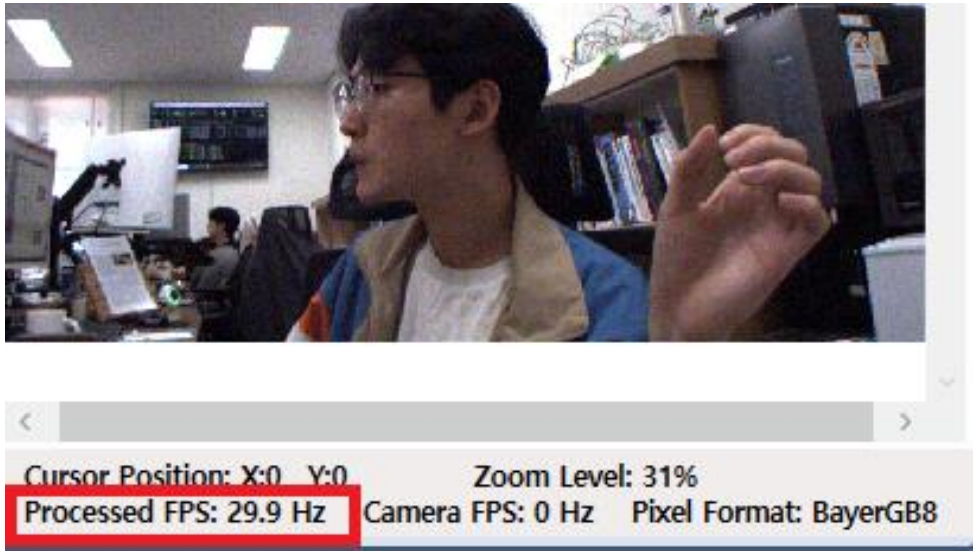
나만의 Grabber 만들기
-My MFC Grabber-Sync:동기화

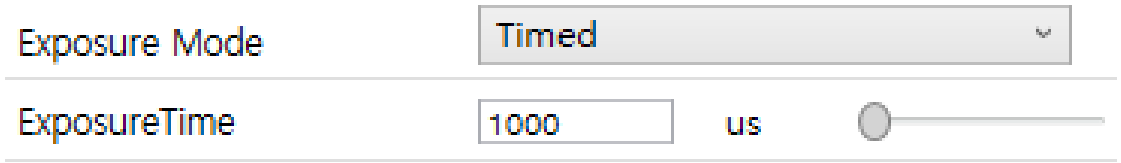
```
nodeMapTLDevice = pCam->GetTLDeviceNodeMap();
pCam->Init();
INodeMap&nodeMap2 = pCam->GetNodeMap();

CEnumerationPtr syncMode = nodeMap2.GetNode("SyncMode");
syncMode->SetIntValue(syncMode->GetEntryByName("SelfSyncMaster")->GetValue());
```

RGBcam:
Trigger mode on
Trigger Source->line3
Trigger Activation->RisingEdge

ThermalCam:
Self Sync Master

나만의 Grabber 만들기
-My MFC Grabber-Sync:동기화

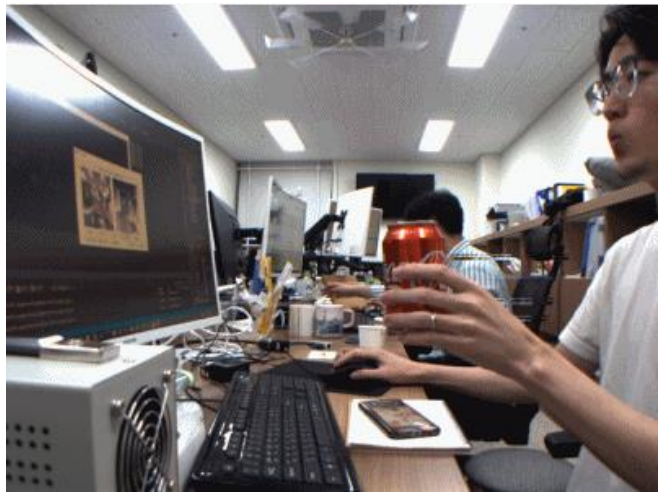




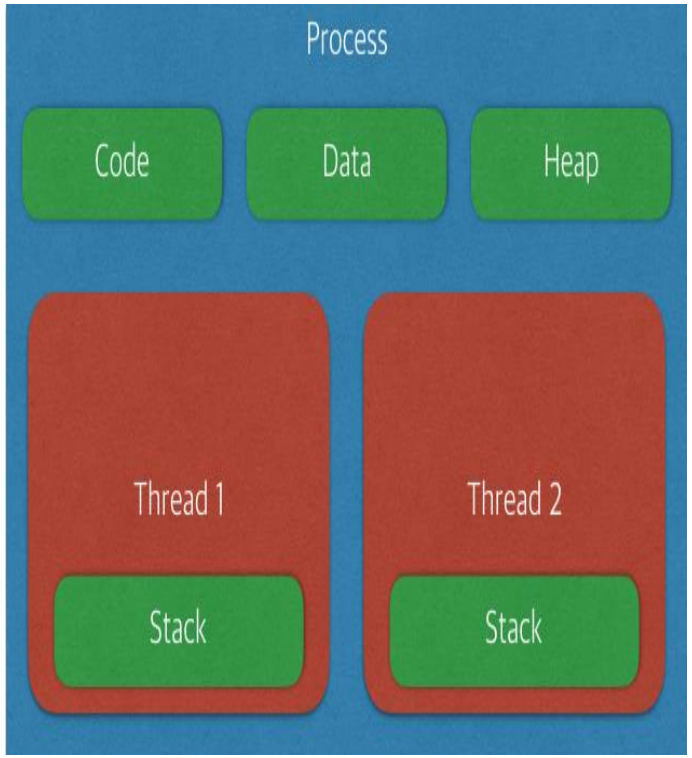
- Trigger signal의 주파수는 29.9Hz이다.따라서 이와 동일한 Processed FPS를 갖도록 ExposureTime을 설정하였다.

결과분석-Process-one-thread

결과분석-Process-two-thread



결과분석



- 하나의 쓰레드에서 두개의 캠을 키고 이미지를 띄우고 저장하면 우리가 원하는 순서대로 동기화 적용한대로 저장됨을 알 수 있다
- 하지만 하나의 쓰레드에서 4대의 캠을 키고 동일한 작업을 한다면 당연히 충돌이 날 수 있다.따라서 4대의 캠을 동기화 하기위해 쓰레드를 늘려보는 작업을 해보았다
- 쓰레드가 두개일때 각각의 쓰레드에 cout을 이용해서 순서를 확인해본결과 둘의 순서는 정해져있지 않았다 즉 우리가 원하는 순서대로 적용될 수 없다는 리스크가 있다.따라서 이경우 예외 처리를 통해 둘의 순서가 우리가 원할때와 같을때만 저장하는 방식을 택하는 것이 적절하다.