facebook

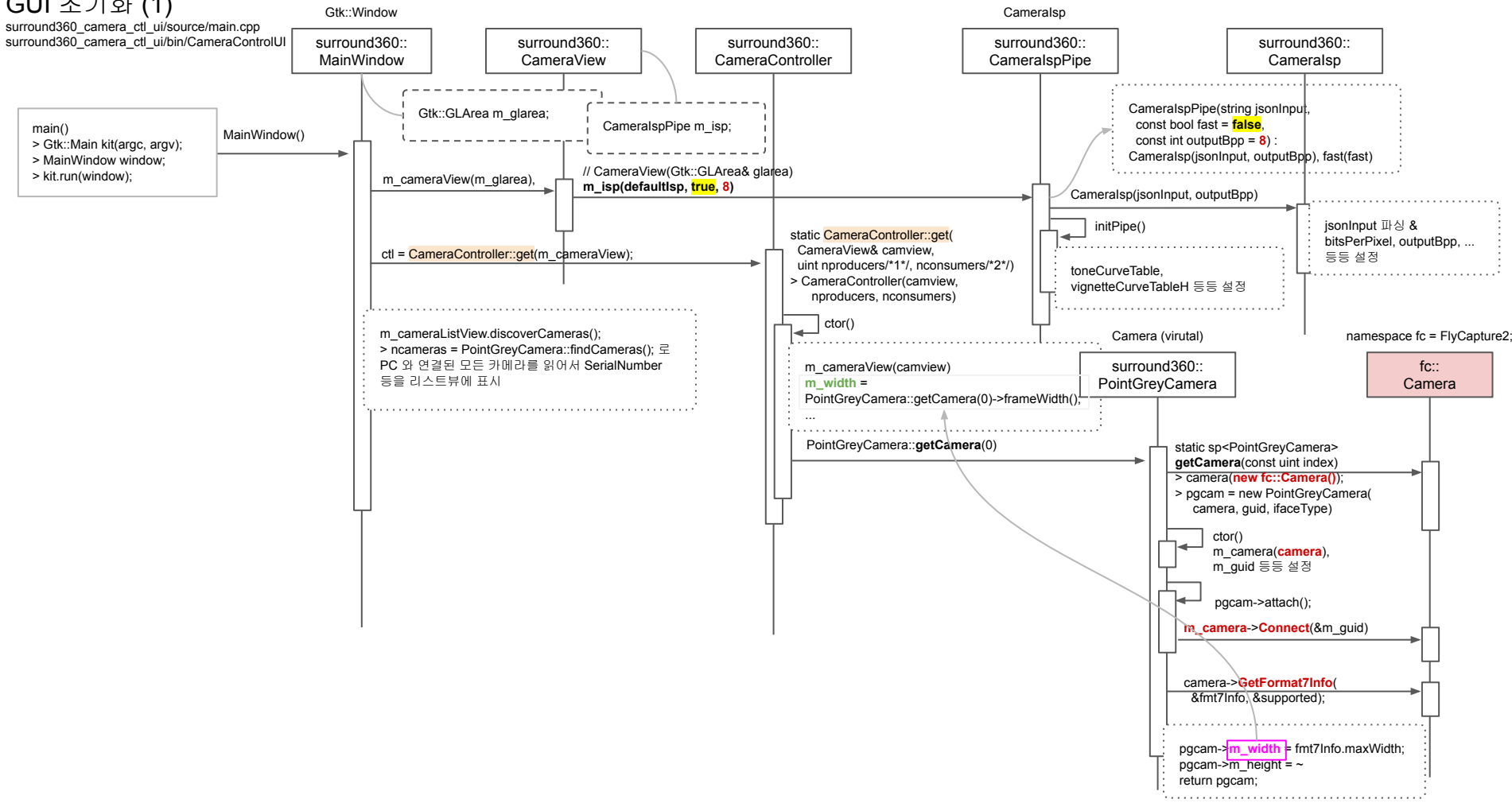
Surround 360 Src

24.05.08

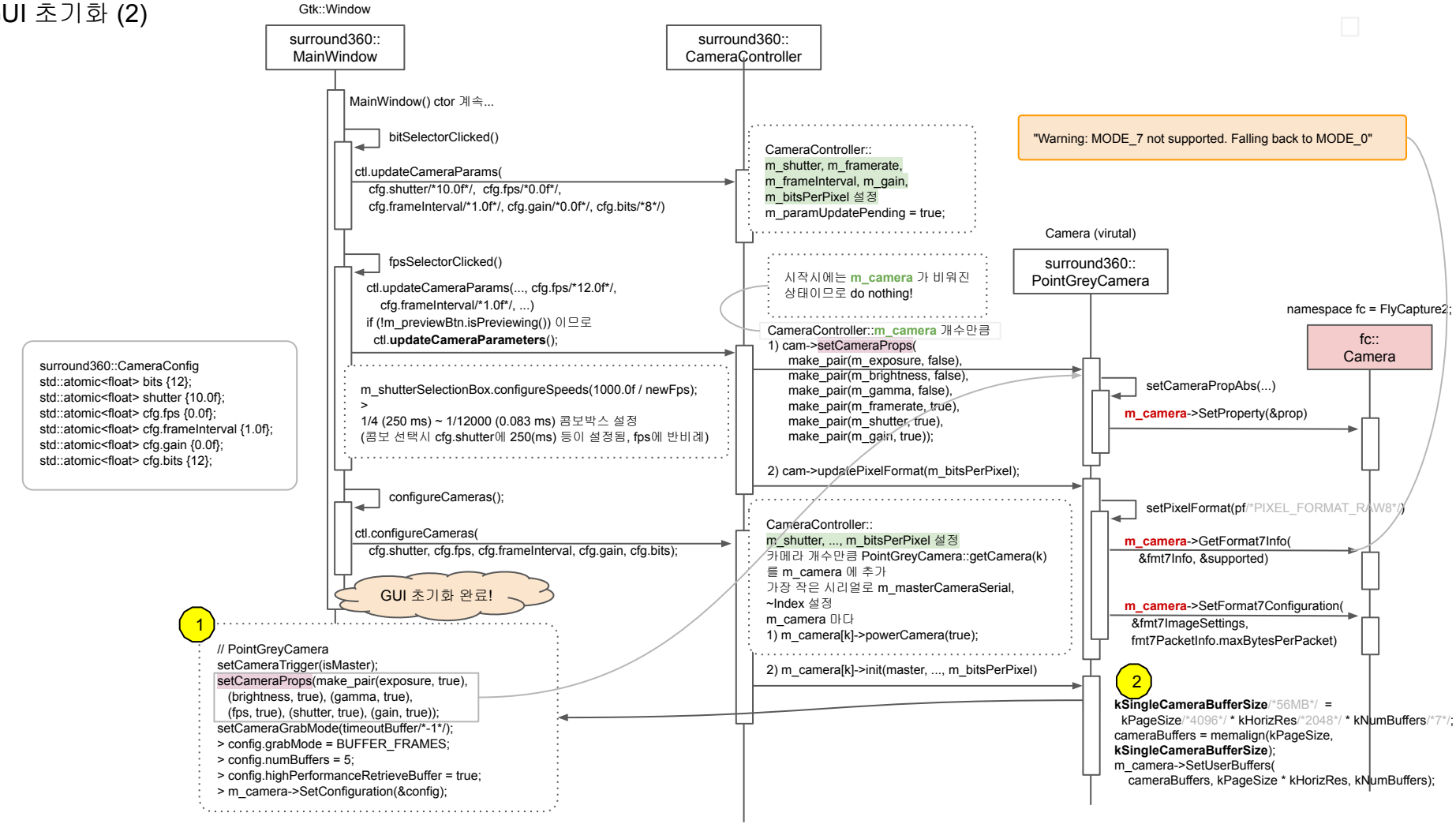
(https://docs.google.com/presentation/d/1Ji6pteoc8ca9dmvOFfTJtB87eUW2krZ9Wp1Jm6CnAsM/edit?usp=sharing)

GUI 초기화 (1)

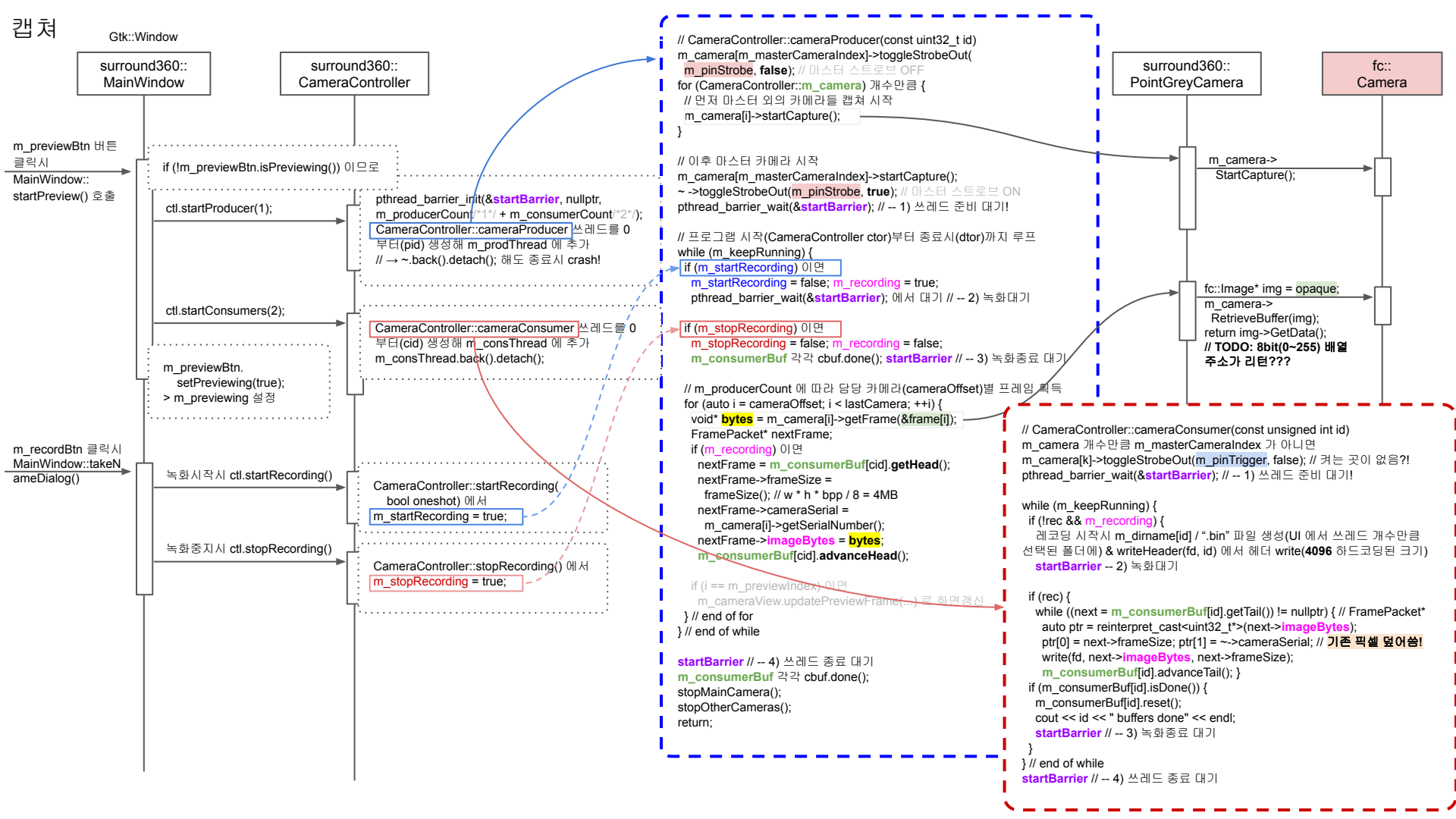
surround360_camera_ctl_ui/source/main.cpp
surround360_camera_ctl_ui/bin/CameraControlUI



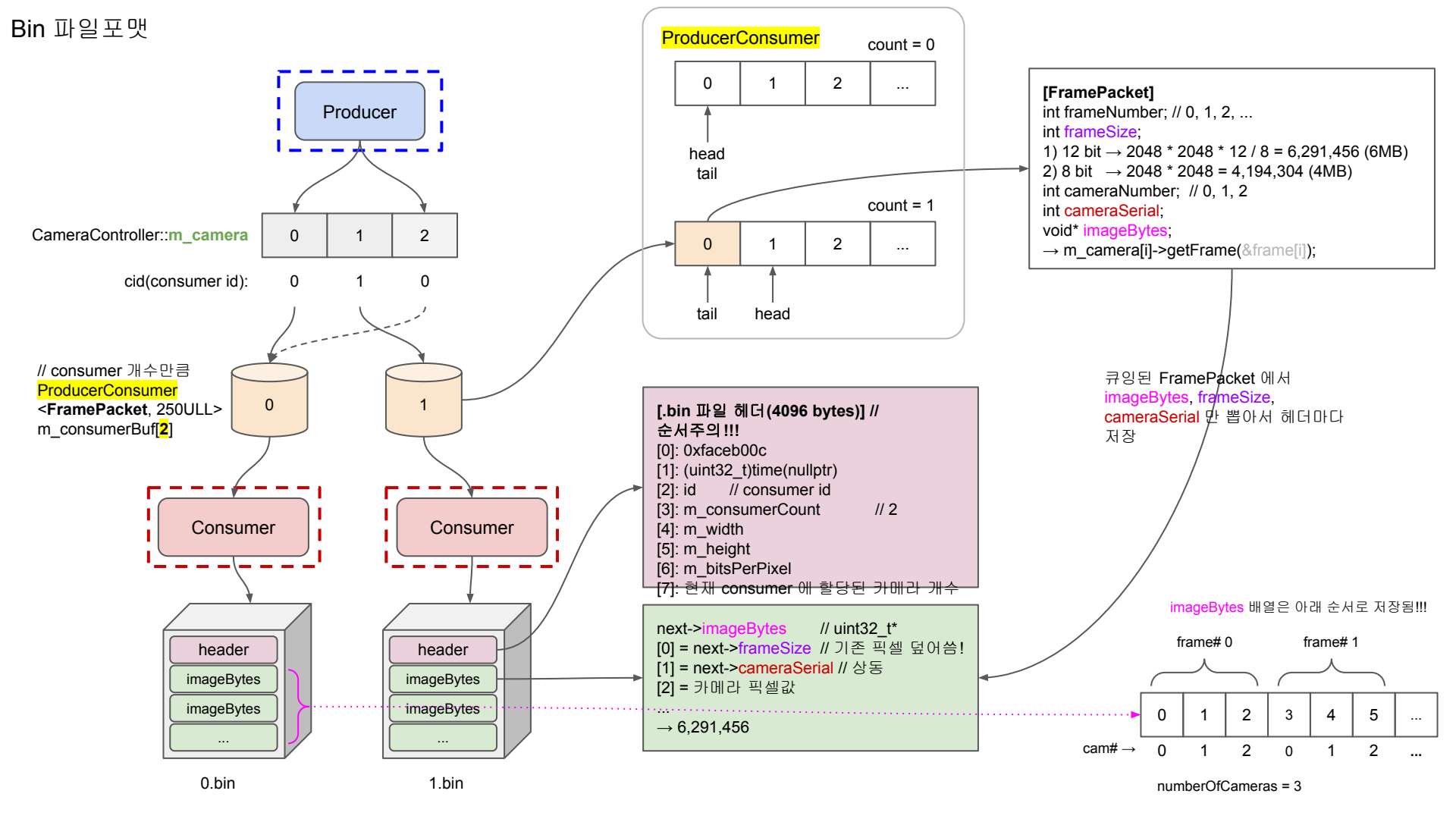
GUI 초기화 (2)


Gtk::Window
surround360::
MainWindow
surround360::
CameraController
MainWindow() ctor 계속...
bitSelectorClicked()
ctl.updateCameraParams(
cfg.shutter/*10.0f*/, cfg.fps/*0.0f*/,
cfg.frameInterval/*1.0f*/, cfg.gain/*0.0f*/, cfg.bits/*8*/)
CameraController::
m_shutter, m_framerate,
m_frameInterval, m_gain,
m_bitsPerPixel 설정
m_paramUpdatePending = true;
"Warning: MODE_7 not supported. Falling back to MODE_0"
Camera (virutal)
surround360::
PointGreyCamera
fpsSelectorClicked()
ctl.updateCameraParams(..., cfg.fps/*12.0f*/,
cfg.frameInterval/*1.0f*/, ...)
if (!m_previewBtn.isPreviewing()) 이므로
ctl.updateCameraParameters();
시작시에는 m_camera 가 비워진
상태이므로 do nothing!
CameraController::m_camera 개수만큼
1) cam->setCameraProps(
make_pair(m_exposure, false),
make_pair(m_brightness, false),
make_pair(m_gamma, false),
make_pair(m_framerate, true),
make_pair(m_shutter, true),
make_pair(m_gain, true));
2) cam->updatePixelFormat(m_bitsPerPixel);
namespace fc = FlyCapture2;
fc::
Camera
setCameraPropAbs(...)
m_camera->SetProperty(&prop)
setPixelFormat(pf/*PIXEL_FORMAT_RAW8*/)
m_camera->GetFormat7Info(
&fmt7Info, &supported)
m_camera->SetFormat7Configuration(
&fmt7ImageSettings,
fmt7PacketInfo.maxBytesPerPacket)
surround360::CameraConfig
std::atomic<float> bits {12};
std::atomic<float> shutter {10.0f};
std::atomic<float> cfg.fps {0.0f};
std::atomic<float> cfg.frameInterval {1.0f};
std::atomic<float> cfg.gain {0.0f};
std::atomic<float> cfg.bits {12};
m_shutterSelectionBox.configureSpeeds(1000.0f / newFps);
>
1/4 (250 ms) ~ 1/12000 (0.083 ms) 콤보박스 설정
(콤보 선택시 cfg.shutter에 250(ms) 등이 설정됨, fps에 반비례)
configureCameras();
ctl.configureCameras(
cfg.shutter, cfg.fps, cfg.frameInterval, cfg.gain, cfg.bits);
CameraController::
m_shutter, ..., m_bitsPerPixel 설정
카메라 개수만큼 PointGreyCamera::getCamera(k)
를 m_camera 에 추가
가장 작은 시리얼로 m_masterCameraSerial,
~Index 설정
m_camera 마다
1) m_camera[k]->powerCamera(true);
GUI 초기화 완료!
1
// PointGreyCamera
setCameraTrigger(isMaster);
setCameraProps(make_pair(exposure, true),
(brightness, true), (gamma, true),
(fps, true), (shutter, true), (gain, true));
setCameraGrabMode(timeoutBuffer/*-1*/);
> config.grabMode = BUFFER_FRAMES;
> config.numBuffers = 5;
> config.highPerformanceRetrieveBuffer = true;
> m_camera->SetConfiguration(&config);
2) m_camera[k]->init(master, ..., m_bitsPerPixel)
2
kSingleCameraBufferSize/*56MB*/ =
kPageSize/*4096*/ * kHorizRes/*2048*/ * kNumBuffers/*7*/;
cameraBuffers = memalign(kPageSize,
kSingleCameraBufferSize);
m_camera->SetUserBuffers(
cameraBuffers, kPageSize * kHorizRes, kNumBuffers);

캡쳐


Gtk::Window
surround360::
MainWindow
surround360::
CameraController
m_previewBtn 버튼
클릭시
MainWindow::
startPreview() 호출
if (!m_previewBtn.isPreviewing()) 이므로
ctl.startProducer(1);
pthread_barrier_init(&startBarrier, nullptr,
m_producerCount/*1*/ + m_consumerCount/*2*/);
CameraController::cameraProducer 쓰레드를 0
부터(pid) 생성해 m_prodThread 에 추가
// → ~.back().detach(); 해도 종료시 crash!
ctl.startConsumers(2);
CameraController::cameraConsumer 쓰레드를 0
부터(cid) 생성해 m_consThread 에 추가
m_consThread.back().detach();
m_previewBtn.
setPreviewing(true);
> m_previewing 설정
m_recordBtn 클릭시
MainWindow::takeN
ameDialog()
녹화시작시 ctl.startRecording()
CameraController::startRecording(
bool oneshot) 에서
m_startRecording = true;
녹화중지시 ctl.stopRecording()
CameraController::stopRecording() 에서
m_stopRecording = true;
// CameraController::cameraProducer(const uint32_t id)
m_camera[m_masterCameraIndex]->toggleStrobeOut(
m_pinStrobe, false); // 마스터 스트로브 OFF
for (CameraController::m_camera) 개수만큼 {
// 먼저 마스터 외의 카메라들 캡쳐 시작
m_camera[i]->startCapture();
}
// 이후 마스터 카메라 시작
m_camera[m_masterCameraIndex]->startCapture();
~ ->toggleStrobeOut(m_pinStrobe, true); // 마스터 스트로브 ON
pthread_barrier_wait(&startBarrier); // -- 1) 쓰레드 준비 대기!
// 프로그램 시작(CameraController ctor)부터 종료시(dtor)까지 루프
while (m_keepRunning) {
if (m_startRecording) 이면
m_startRecording = false; m_recording = true;
pthread_barrier_wait(&startBarrier); 에서 대기 // -- 2) 녹화대기
if (m_stopRecording) 이면
m_stopRecording = false; m_recording = false;
m_consumerBuf 각각 cbuf.done(); startBarrier // -- 3) 녹화종료 대기
// m_producerCount 에 따라 담당 카메라(cameraOffset)별 프레임 획득
for (auto i = cameraOffset; i < lastCamera; ++i) {
void* bytes = m_camera[i]->getFrame(&frame[i]);
FramePacket* nextFrame;
if (m_recording) 이면
nextFrame = m_consumerBuf[cid].getHead();
nextFrame->frameSize =
frameSize(); // w * h * bpp / 8 = 4MB
nextFrame->cameraSerial =
m_camera[i]->getSerialNumber();
nextFrame->imageBytes = bytes;
m_consumerBuf[cid].advanceHead();
if (i == m_previewIndex) 이면
m_cameraView.updatePreviewFrame(...) 로 화면갱신
} // end of for
} // end of while
startBarrier // -- 4) 쓰레드 종료 대기
m_consumerBuf 각각 cbuf.done();
stopMainCamera();
stopOtherCameras();
return;
surround360::
PointGreyCamera
fc::
Camera
m_camera->
StartCapture();
fc::Image* img = opaque;
m_camera->
RetrieveBuffer(img);
return img->GetData();
// TODO: 8bit(0~255) 배열
주소가 리턴???
// CameraController::cameraConsumer(const unsigned int id)
m_camera 개수만큼 m_masterCameraIndex 가 아니면
m_camera[k]->toggleStrobeOut(m_pinTrigger, false); // 켜는 곳이 없음?!
pthread_barrier_wait(&startBarrier); // -- 1) 쓰레드 준비 대기!
while (m_keepRunning) {
if (!rec && m_recording) {
레코딩 시작시 m_dirname[id] / ".bin" 파일 생성(UI 에서 쓰레드 개수만큼
선택된 폴더에) & writeHeader(fd, id) 에서 헤더 write(4096 하드코딩된 크기)
startBarrier -- 2) 녹화대기
if (rec) {
while ((next = m_consumerBuf[id].getTail()) != nullptr) { // FramePacket*
auto ptr = reinterpret_cast<uint32_t*>(next->imageBytes);
ptr[0] = next->frameSize; ptr[1] = ~->cameraSerial; // 기존 픽셀 덮어씀!
write(fd, next->imageBytes, next->frameSize);
m_consumerBuf[id].advanceTail(); }
if (m_consumerBuf[id].isDone()) {
m_consumerBuf[id].reset();
cout << id << " buffers done" << endl;
startBarrier // -- 3) 녹화종료 대기
}
} // end of while
startBarrier // -- 4) 쓰레드 종료 대기

Bin 파일포맷


Producer
CameraController::m_camera
0
1
2
cid(consumer id):
0
1
0
// consumer 개수만큼
ProducerConsumer
<FramePacket, 250ULL>
m_consumerBuf[2]
0
1
Consumer
Consumer
header
imageBytes
imageBytes
...
header
imageBytes
imageBytes
...
0.bin
1.bin
ProducerConsumer
count = 0
0
1
2
...
head
tail
count = 1
0
1
2
...
tail
head
[FramePacket]
int frameNumber; // 0, 1, 2, ...
int frameSize;
1) 12 bit → 2048 * 2048 * 12 / 8 = 6,291,456 (6MB)
2) 8 bit → 2048 * 2048 = 4,194,304 (4MB)
int cameraNumber; // 0, 1, 2
int cameraSerial;
void* imageBytes;
→ m_camera[i]->getFrame(&frame[i]);
[.bin 파일 헤더(4096 bytes)] //
순서주의!!!
[0]: 0xfaceb00c
[1]: (uint32_t)time(nullptr)
[2]: id // consumer id
[3]: m_consumerCount // 2
[4]: m_width
[5]: m_height
[6]: m_bitsPerPixel
[7]: 현재 consumer 에 할당된 카메라 개수
next->imageBytes // uint32_t*
[0] = next->frameSize // 기존 픽셀 덮어씀!
[1] = next->cameraSerial // 상동
[2] = 카메라 픽셀값
...
→ 6,291,456
큐잉된 FramePacket 에서
imageBytes, frameSize,
cameraSerial 만 뽑아서 헤더마다
저장
imageBytes 배열은 아래 순서로 저장됨!!!
frame# 0
frame# 1
0
1
2
3
4
5
...
cam# →
0
1
2
0
1
2
...
numberOfCameras = 3

Unpacker

surround360_render/bin/Unpacker
surround360_render/source/camera_isp/
Unpacker.cpp

```
// "--isp_dir", "~/Surround360/cubox/color_calibration/isp",
"--isp_dir", "NOT_EXISTING_DIR", // if --isp_dir 가 존재하지 않는 경로이면 tiff 까지만 생성됨
"--bin_list", "~/Desktop/0.bin,/home/cubox/Desktop/1.bin",
"--output_raw_dir", "~/Desktop/Surround360_output/unpacker@output_raw_dir",
"--output_dir", "~/Desktop/Surround360_output/unpacker@output_dir",

// bin_list is saved to output_raw_dir/23486816/000000.tiff and
unpacker@output_dir/cam0/000000.png (renamed from 23486816 to cam0 dir)
needs existing unpacker@output_dir, unpacker@output_raw_dir directory
```

main()

surround360::
BinaryFootageFile

footageFiles 에 bin_list 를 , 로 나눠서 추가

footageFiles.emplace_back(binFile);

fileDescriptor(-1), baseAddress(nullptr),
mappingSize(0),
filename, directory // from binFile 인자

```
// 각 bin 파일마다 loop
for (fileIdx < footageFiles.size()) {
  auto& footageFile = footageFiles[fileIdx];
  footageFile.open();
  cint numCameras =
    footageFile.getNumberOfCameras();
  cint endFrameMax =
    footageFile.getNumberOfFrames() - 1;

  // 각 bin 파일 내의 카메라마다
  for (camIndex < numCameras) {
    // 카메라별 모든 프레임을 비동기로
    for (frameIndex <= endFrame)
      auto frame = footageFile.getFrame(
        frameIndex, cameraIndex);
      upscaled = RawConverter::convert8Frame(
        frame, width, height); // uint16_t* 리턴
      upscaled 를
      output_raw_dir/23486816/000000.tiff 에 저장
      isp_dir 폴더 존재시, CameraIspPipe isp(json,
kFast, kOutputBpp) 로 png 도 생성
```

footageFile.open()

fileDescriptor = openFile(); // just open
mappingSize = getFileSize(); // just get size
baseAddress = mapFile(); → mmap(mappingSize) 로 메모리 맵핑!
readMetadataHeader()
→ baseAddress 로부터 metadata 멤버에MetadataHeader 크기만큼 copy!

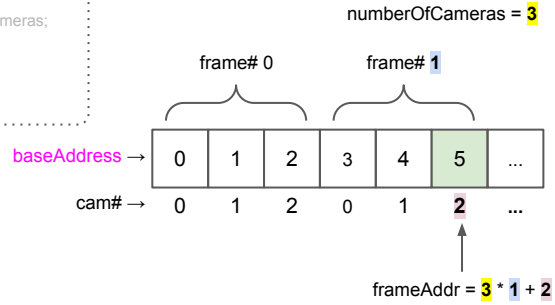
footageFile.getNumberOfFrames()

```
// 카메라 1대의 프레임 개수를 리턴(카메라마다 동일)
size_t kNumCameras = getNumberOfCameras(); // metadata.numberOfCameras;
size_t kSingleFrameSize = getFrameSize();
→ metadata.width * metadata.height * metadata.bitsPerPixel / CHAR_BIT
size_t kPageSize = 4096;
return (mappingSize - kPageSize) / kSingleFrameSize / kNumCameras;
```

footageFile.getFrame(
frameIndex, cameraIndex)

.tiff 저장

```
calculateFrameAddress(frameNumber, cameraNumber) 리턴
→ cuint8_t* framesBase(baseAddress + kMetadataSize/*4096*/);
frameAddr = framesBase + (metadata.numberOfCameras *
frameNumber + cameraNumber) * kSingleFrameSize; 를 리턴
```



Unpacker

surround360_render/bin/Unpacker
surround360_render/source/camera_isp/
Unpacker.cpp

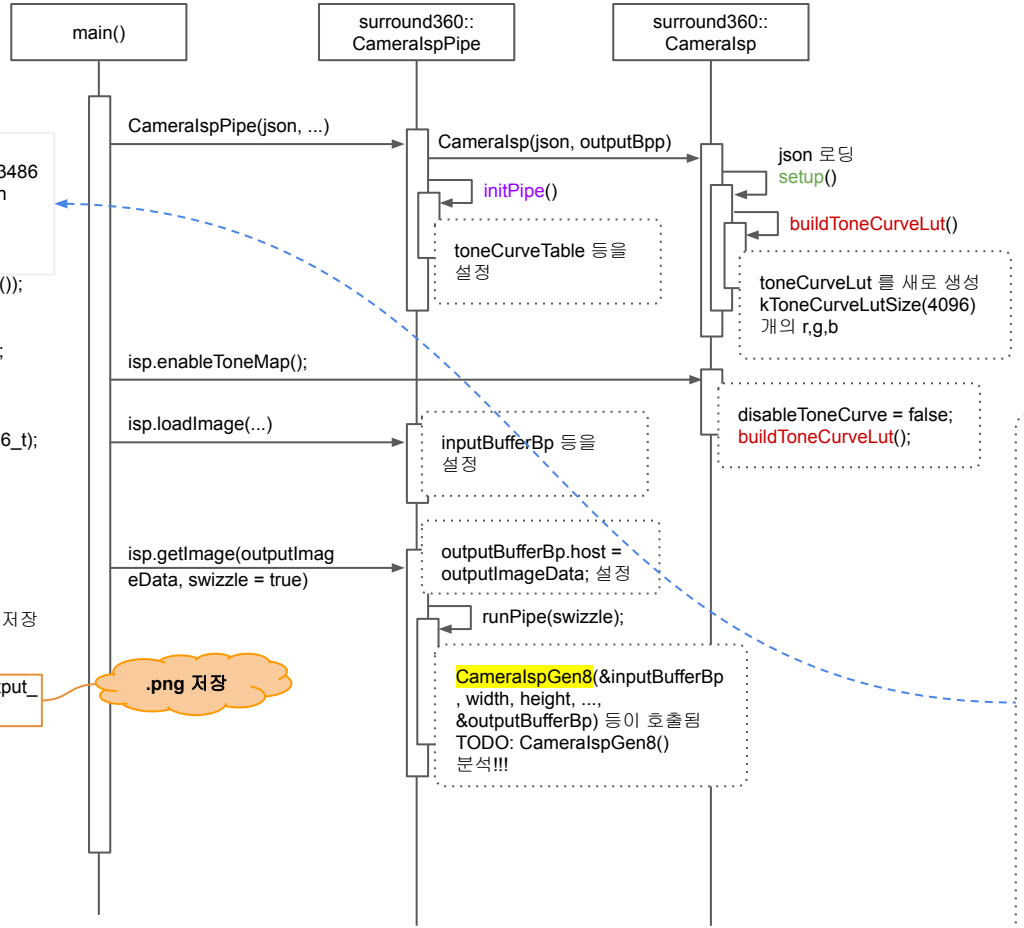
-- 계속 --

isp_dir 폴더 존재시,
"~/Surround360/cubox/color_calibration/isp/<23486816>.json" 을(color/optical vignetting calibration 결과물) json 객체에 읽어
CameraIspPipe isp(json, kFast/*false*/, kOutputBpp/*16*/);
isp.setBitsPerPixel(footageFile.getBitsPerPixel());
→ CameraIsp::bitsPerPixel 설정
isp.enableToneMap();
isp.loadImage(upscaled->data(), width, height);
isp.setup();
isp.initPipe();

int imageSize = width * height * 3 * sizeof(uint16_t);
auto coloredImage =
make_unique<vector<uint8_t>>(imageSize);
isp.getImage(coloredImage->data());

Mat outputImage(height, width, CV_16UC3, coloredImage->data());

"~output_dir/<23486816>/000000.png" 등으로 저장 후, 폴더명을 23486816 에서 cam0 식으로 변경
→
~/Desktop/Surround360_output/unpacker@output_dir/cam0/000000.png 에 저장




```
"CameraIsp" : {       "serial" : 0,
    "name" : "PointGrey Grasshopper",
    "bitsPerPixel" : 16,
    "compandingLut" :  [[0.0,  0.0, 0.0],
                        [0.6,  0.6, 0.0],
                        [0.7,  0.7, 0.0],
                        [1.0,  1.0, 0.0]],
    "blackLevel" : [0.0, 0.0, 0.0],
    "vignetteRollOffH" : [[1.1, 1.1, 1.1],
                          [1.0, 1.0, 1.0],
                          [1.0, 1.0, 1.0],
                          [1.0, 1.0, 1.0],
                          [1.1, 1.1, 1.1]],
    "vignetteRollOffV" : [[1.1, 1.1, 1.1],
                          [1.0, 1.0, 1.0],
                          [1.0, 1.0, 1.0],
                          [1.0, 1.0, 1.0],
                          [1.1, 1.1, 1.1]],
    "bayerPattern" : "GBRG"
}   res\config\isp\passthrough.json
```

```
"CameraIsp" : {
    "serial" : 0,       "name" : ~,
    "bitsPerPixel" : 16,
    "compandingLut" :  [[0.0,  0.0, 0.0], ...
    "blackLevel" : [1285.0, 1285.0, 1285.0],
    "vignetteRollOffH" : [[1.1, 1.1, 1.1], ...
    "vignetteRollOffV" : [[1.1, 1.1, 1.1], ...
    "whiteBalanceGain" : [1.1, 1.0, 1.65],
    "stuckPixelThreshold" : 5,
    "stuckPixelDarknessThreshold" : 0.11,
    "stuckPixelRadius" : 0,
    "denoise" : 0.6,
    "denoiseRadius" : 2,
    "ccm" : [[1.02169,  -0.05711,  0.03543],
             [0.16789,   1.13419, -0.30208],
             [-0.15726, -0.07864,  1.2359]],
    "sharpening" : [0.5, 0.5, 0.5],
    "saturation" : 1.2,
    "contrast" : 1.0,
    "lowKeyBoost" : [-0.2, -0.2, -0.2],
    "highKeyBoost" : [0.2, 0.2, 0.2],
    "gamma" : [0.4545, 0.4545, 0.4545],
    "bayerPattern" : "GBRG"
}       res\config\isp\cmosis_sunex.json
                (Side camera)
```

Color Calibration

surround360_render/scripts/color_calibrate_all.py

```
// INPUT:
//   <data_dir>/charts/23486816.tiff
// OUTPUT:
//   <data_dir>/isp/23486816.json
"--data_dir", "~/Desktop/color_calibration",
// Default "--output_data_dir", "~/Desktop/color_calibration/output",
"--illuminant", "D50",
"--num_squares_w", "6",
"--num_squares_h", "4",
"--min_area_chart_perc", "0.5",
"--min_area_chart_perc", "0.5",
"--max_area_chart_perc", "40.0",
"--black_level_hole",
// "--save_debug_images",
```

raw_charts = list_tiff(data_dir + "/charts") 로 모든 tiff 파일 경로를 저장 후
camera_names[i] 에는 serial 만 저장
out_dirs[i] 에는 <data_dir>/output/<serial> 를 저장

for () 각 tiff 마다 **bin/TestColorCalibration** 를 호출
(surround360_render\source\test\TestColorCalibration.cpp)
파라미터는 --image_path = <data_dir>/charts/23486816.tiff
--isp_passthrough_path = /res/config/isp/passthrough.json 등등

TestColorCalibration 에서는
1) colorPatches = detectColorChart(raw8, ...) 로 ColorPatch 벡터를 획득 후, 개수가
numPatchesExpected(6 * 4 = 24) 와 같은지 확인

"Number of patches found (21) different than expected (24)"

2) colorResponse = computeRGBResponse(...)
saveXIntercepts(colorResponse, FLAGS_output_data_dir); // **intercept_x.txt** 저장
3) obtainIspParams(...) 로 whiteBalance 등을 획득
4) writeIspConfigFile() 함수로 whiteBalance 등을 isp_passthrough_path 파일 내용을
기반으로 <data_dir>/output/<serial>/isp_out.json 을 생성

"Finding worst-case X-intercepts..."

일단 black_level_adjust 는 false 로(default) skip!

for () 모든 카메라 마다, **intercept_x.txt** 를 읽어
모든 이미지들의 최소, 최대 intercept_x_max, intercept_x_min 를 구한 후,

생성된 <data_dir>/output/<serial>/isp_out.json 을 <data_dir>/isp/<serial>.json 로 복사!
json 마다 --isp_dst, --update_clamps --clamp_min 등의 인자로 한번 더
bin/TestColorCalibration 를 호출

MacBeth ColorChecker

1_D50_raw.png

16_isp_out.png

카메라마다 1장씩만 촬영!

```
~/Desktop/color_calibration/ -> data_dir
├ charts/              // image_path
│  ├ 23486816.tiff
│  ├ ...
│  └ 23486833.tiff
├ output/              // out_dir
│  ├ 23486816/
│  │  ├ 1_D50_raw.png
│  │  ├ ...
│  │  ├ 16_isp_out.png
│  │  └ isp_out.json
│  ├ 23486817/
│  ├ ...
│  ├ 23486833/
│  └ runtimes.txt
└ isp/
   ├ 23486816.json
   ├ ...
   └ 23486833.json
```

INPUT
카메라별 1장만!

Vignetting Calibration 을 위해
이 color calibrated json 파일들을
~/Desktop/vignetting_calibration/isp/
하위에 복사해줘야 함!

Optical Vignetting Calibration

surround360_render\scripts\vignetting_calibrate.py

```
// data_dir should have serial_number folders!
// "--data_dir", "~/work/Surround360/cubox/vignetting_calibration",
"--data_dir", "~/Desktop/vignetting_calibration",
"--save_debug_images",
```

for serial_number in list_dirs_numbers(data_dir): // 숫자로만된 폴더들(serial) 각각을
isp_json = “<data_dir>/isp/<serial>.json”
<data_dir> = ~/Desktop/vignetting_calibration

print "[" + serial_number + "] Generating data for vignetting correction..."

"[23486816] Generating data for vignetting correction..."

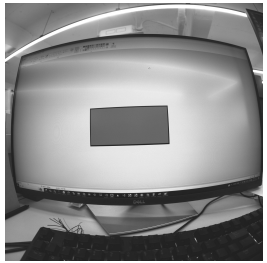
1) bin/TestVignettingDataAcquisition 를 호출
-> INPUT_DIR=<data_dir>/23486816
OUTPUT_DIR=<data_dir>/23486816/acquisition, ISP_JSON=<**isp_json**>
OUTPUT_DIR/**data.json** 에 image_id, location, rgbmedian 을 저장

"[23486816] Computing vignetting..."

2) bin/TestVignettingCalibration
-> OUTPUT_DIR=<data_dir>/23486816/calibration
DATA_PATH=<data_dir>/23486816/acquisition/**data.json**
TEST_ISP_PATH=**isp_json**, 를 로딩 후, ceres::Solver 로 curve fit 을 수행?!
TEST_IMAGE_PATH= e.g, 000000.tiff
<data_dir>/23486816/calibration/isp_out.json 이 생성됨

3) <data_dir>/isp_new/isp_out.json 를 <data_dir>/isp_new/<serial>.json 로 rename

카메라마다 20장 이상 촬영이 추천됨!!!



charts/000000.tiff

~/Desktop/vignetting_calibration/ -> data_dir

```
├ 23486816/          // input_dir
│ ├ acquisition/...
│ ├ calibration/...
│ │ ├ ...
│ │ └ isp_out.json                INPUT
│ └ charts/000000.tiff  // 20 장이상 필요
├ 23486817/
├ ...                              renamed!
├ 23486833/
├ isp/
│ ├ 23486816.json
│ ├ ...
│ └ 23486833.json
└ isp_new/
  ├ 23486816.json
  ├ ...                            OUTPUT
  └ 23486833.json
```

~/Desktop/color_calibration/isp/ 하위 json 파일들을 복사해와야 함

Copy to
~/Desktop/test/render/config/isp/23486816.json
and run run_all.py

Geometric Calibration

surround360_render\scripts\geometric_calibration.py

feature 가(선명한 경계선, 코너 등) 많은
scene 촬영이 추천됨!!!
(e.g, 사무실 인테리어)

~/Desktop/geometric_calibration/ 하위에서 이미지 파일들을 찾아서 8 bit 이미지로 변환 & 저장
(image 를 강제로 uint8 로 변경 후, cv2.imwrite(filename, image) 를 사용)
--> e.g, ~/Desktop/geometric_calibration/cam[0-16]/000000.png

COLMAP_DIR = "/usr/local/bin"
COLMAP_EXTRACT_TEMPLATE = {COLMAP_DIR}/feature_extractor
--General.image_path "{IMAGE_PATH}" // <data_dir> -> 저장된 8 bit 이미지 파일 디렉토리
--General.database_path "{COLMAP_DB_PATH}" // ~/Desktop/geometric_calibration/colmap.db

COLMAP_MATCH_TEMPLATE = {COLMAP_DIR}/exhaustive_matcher
--General.database_path "{COLMAP_DB_PATH}"

{SURROUND360_RENDER_DIR}/bin/GeometricCalibration
--json "{RIG_JSON}" // $PWD/res/config/camera_rig.json
--output_json "{OUTPUT_JSON}" // ~/Desktop/geometric_calibration/camera_rig.json
--matches "{MATCHES_JSON}" // ~/Desktop/geometric_calibration/**matches.json** (from colmap.db)
--pass_count {PASS_COUNT} // 10
--log_dir "{LOG_DIR}" // ~/Desktop/geometric_calibration/logs
--logbuflevel -1
--stderrthreshold 0
{FLAGS_EXTRA} // --save_debug_images
-->
~/Desktop/geometric_calibration/debug 도 생성됨
for () FLAGS_experiments(10) 만큼, **cameras** 을 흔들고(camera_rig.json 초기값을 perturb), **matches.json** 도 읽어서
refine(**cameras**, keypointMap, overlaps, pass, debugDir); 함수로 **cameras** 객체를(std::vector<Camera>) 최적화?! --> ceres::Solver 사용!
Camera::saveRig(FLAGS_output_json, **cameras**); 로 camera_rig.json 를 업데이트!

```
--data_dir  ~/Desktop/geometric_calibration \
--colmap_dir /usr/local/bin \
--rig_json $PWD/res/config/camera_rig.json \  // --> 기본 rig 파일
--output_json ~/Desktop/geometric_calibration/camera_rig.json \
--save_debug_images
```

```
~/Desktop/geometric_calibration/ -> data_dir
├─ colmap.db
├─ matches.json
└─ camera_rig.json
```

이후, Geometric Calibration 까진 마친 최종
camera_rig.json 를
~/Desktop/test/render/config/camera_rig.json 에 복사 후
run_all.py 를 실행!!!

Render

surround360_render\scripts\batch_process_video.py

run_all.py 에서는 총 2 단계를 수행
1) bin/Unpacker (이전 페이지 참고)
2) scripts/batch_process_video.py

```
// batch_process_video.py
for () 각 프레임 이미지에 대해서
  // “4k” 일 경우,
  render_params["SHARPENNING"]              = 0.25
  render_params["EQR_WIDTH"]                = 4200
  render_params["EQR_HEIGHT"]               = 1024
  render_params["FINAL_EQR_WIDTH"]          = 4096
  render_params["FINAL_EQR_HEIGHT"]         = 2048

// bin/TestRenderStereoPanorama 를 호출
--rig_json_file “{RIG_JSON_FILE}” // dest_dir/config/camera_rig.json
--imgs_dir "{SRC_DIR}/rgb"        // dest_dir/rgb
--frame_number {FRAME_ID}         // 000000
--output_data_dir "{SRC_DIR}"     // dest_dir/
--output_equirect_path "{OUT_EQR_DIR}/eqr_{FRAME_ID}.png"
// --> dest_dir/eqr_frames/eqr_000000.png
--output_cubemap_path "{OUT_CUBE_DIR}/cube_{FRAME_ID}.png"
// --> dest_dir/cube_frames/cube_000000.png
--eqr_width {EQR_WIDTH}           // 6300
--eqr_height {EQR_HEIGHT}         // 3072
--final_eqr_width {FINAL_EQR_WIDTH/HEIGHT}          // 6144 / 6144

main() 에서 renderStereoPanorama() 호출
```

```
surround360_render/res/  (default)
├ config/
│  ├ isp/
│  │  ├ cmosis_fujinon.json
│  │  ├ cmosis_sunex.json
│  │  └ passthrough.json
│  └ camera_rig.json
└ pole_masks/
   ├ cam15.png
   └ cam16.png
```

```
// run_all.py
--data_dir "~/Desktop"  // .bin 파일이 포함된 디렉토리
--dest_dir "~/Desktop/test/render"  // 하위에 config/camera_rig.json,
config/isp/ 등 필요

batch_process_video.py 를 호출함
--flow_alg {FLOW_ALG}             // "pixflow_low"
--root_dir "{ROOT_DIR}"           // dest_dir
--surround360_render_dir "{SURROUND360_RENDER_DIR}"
// → ~/cubox/Surround360/surround360_render
--quality {QUALITY}               // “4k”
--start_frame {START_FRAME}       // 0
--end_frame {END_FRAME}           // dest_dir/rgb/cam0/ 파일 개수 - 1
--cubemap_width {CUBEMAP_WIDTH}         // 0 (TODO: 512 ?)
--cubemap_height {CUBEMAP_HEIGHT}       // 0 (TODO: 512 ?)
--cubemap_format {CUBEMAP_FORMAT}       // 'video'
--rig_json_file "{RIG_JSON_FILE}"
// → dest_dir/config/camera_rig.json
{FLAGS_RENDER_EXTRA}          // --enable_top --enable_bottom
--enable_pole_removal --save_debug_images --verbose
```

```
~/Desktop/test/
├ 0.bin (10.76GB)
├ 1.bin (9.56GB)
└ render (823.6MB) // dest_dir(ROOT_DIR)
   ├ config          // config_dir
   │  ├ isp          // isp_dir
   │  │  ├ 15636976.json
   │  │  ├ 16237674.json
   │  │  ├ ...
   │  │  └ 16283829.json
   │  └ camera_rig.json
   ├ eqr_frames
   │  ├ eqr_000000.png
   │  └ eqr_000001.png
   ├ pole_masks/
   │  ├ cam15.png
   │  └ cam16.png
   └ rgb
      ├ cam0
      │  ├ 000000.png
      │  └ 000001.png
      ├ cam1
      ├ cam2
      ├ ...
      └ cam16
   ├ raw // if --save_raw
      ├ 15636976
      │  ├ 000000.tiff
      │  └ 000001.tiff
      ├ ...
      └ 16283829
   ├ cube_frames/
   │  ├ eqr_000000.png
   │  └ eqr_000001.png
   ├ flow/
   │  └ 000000/, 000001/, ...
   └ debug/
      ├ 000000
      │  ├ flow_images/
      │  └ projections/
```

if not exists, use default config

Render (Surround360\surround360_render\source\test\TestRenderStereoPanorama.cpp)

main()

RigDescription

Camera

```
main()
> renderStereoPanorama();
```

Camera::Rig
rig, rigSideOnly

```
RigDescription
rig(FLAGS_rig_json_file)
--> "camera_rig.json"

if (FLAGS_enable_bottom) 이면
prepareBottomImagesThread 생성!

if (FLAGS_enable_top) 이면
prepareTopImagesThread 생성

if (FLAGS_save_debug_images)
~/Desktop/test/render/debug/000000/pro
tions/ 생성

vector<Mat> projectionImages;
// "Projecting camera images to spherical"
projectSphericalCamImages(rig,
FLAGS_imgs_dir, FLAGS_frame_number
projectionImages);
...
```

```
ctor() 에서 rig =
Camera::loadRig(filename)
// RigDescription::rig 에 추가
```

```
group 명에 "side" 존재시
rigSideOnly.emplace_bac
k(camera); 에도 추가 #14
(rig 에는 모든 카메라 #17)
```

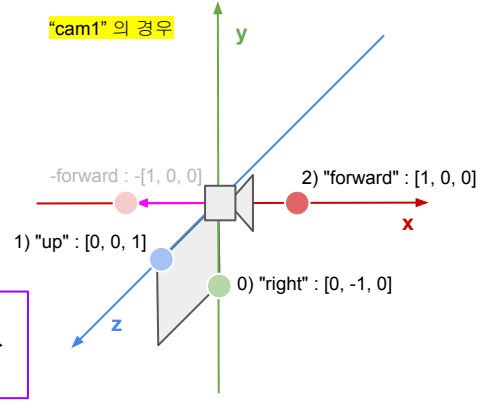
```
// surround360_render\source\render\Camera.h
using Real = double;
using Vector2 = Eigen::Matrix<Real, 2, 1>;
using Vector3 = Eigen::Matrix<Real, 3, 1>;
using Matrix3 = Eigen::Matrix<Real, 3, 3>;
using Ray = Eigen::ParametrizedLine<Real, 3>;
using Rig = std::vector<Camera>;
```

```
Camera::Rig Camera::loadRig(const std::string& filename)
> "camera_rig.json" 의 "cameras" 를
std::vector<Camera> cameras;에 로딩
return cameras; // 모든 카메라(#17) 리턴
```

```
Camera(const dynamic& json)
> position = deserializeVector<3>(json["origin"]);
setRotation(
   deserializeVector<3>(json["forward"]),
   ~["up"]), ~["right"]));
group, resolution(2048, 2048), principal(1024,
1024) 멤버 등등을 설정
```

rig(world ?) → camera
→ (distort) → sensor →
pixel (left, top 기준)

"cam1" 의 경우




```
void Camera::setRotation(const Vector3& forward, up, right)
>
CHECK_LT(right.cross(up).dot(forward), 0) << "rotation must be right-handed"; // 음수
rotation.row(2) = -forward; // +z is back (Ray(빛) 방향???)
rotation.row(1) = up; // +y is up
rotation.row(0) = right; // +x is right
Eigen::AngleAxis<Camera::Real> aa(rotation);
rotation = aa.toRotationMatrix(); // Matrix3 rotation; 멤버변수
```

0: [0, -1, 0]
1: [0, 0, 1]
2: -[1, 0, 0]
→ "cam1" 의 경우

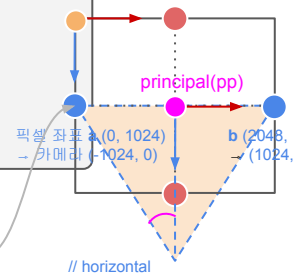
```
// compute rig coordinates, returns a ray, inverse of pixel()
Ray Camera::rig(const Vector2& pixel) const {
  // transform from pixel to distorted sensor coordinates
  Vector2 sensor = (pixel - principal).cwiseQuotient(focal); // 뮦
  // transform from distorted sensor coordinates to unit camera vector
  Vector3 unit = sensorToCamera(sensor); // 렌즈 undistort !!!
  // transform from camera space to rig space
  return Ray(position, rotation.transpose() * unit);  } // R, t
```

```
void projectSideToSpherical(
    Mat& dst, // projectionImages[camIdx] 에 그려짐
    const Mat& src, // camImages[camIdx] 에 해당
    const Camera& camera,
    const float leftAngle,    const float rightAngle,
    const float topAngle,     const float bottomAngle)
상하 각각 side_alpha_feather_size(100) 만큼 alpha feather 를 한 후,
bicubicRemapToSpherical(dst, tmp/*=src*/, ...) 를 호출해서 각 점을
FOV 각도 범위내에서 구면 좌표 Mat warp 에 매핑 후,
remap(tmp, dst, warp, Mat(), CV_INTER_CUBIC,
BORDER_CONSTANT); 로 warpping!!! (cv2.remap)
```

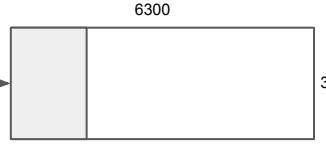
```
// side 카메라 이미지를 spherical coordinates 에 투영
void projectSphericalCamImages(
    const RigDescription& rig, const string& imagesDir, // ~/Desktop/test/render/rgb
    const string& frameNumber, vector<Mat>& projectionImages)
vector<Mat> camImages = rig.loadSideCameraImages(imagesDir, frameNumber);
> side 카메라마다 ~/Desktop/test/render/rgb/cam1/000000.png 등을 읽어옴
projectionImages.resize(camImages.size()); // 14개
const float hRadians = 2 * approximateFov(rig.rigSideOnly, false); vRadians ...
projectionImages[camIdx] 마다 create(...) // h: 180, w: 360도 partial 이미지가 설정됨!
threads.emplace_back(projectSideToSpherical,
    ref(projectionImages[camIdx]), // <-- 여기에 그려짐!!!
    cref(camImages[camIdx]),       cref(camera),
    direction + hRadians / 2,      direction - hRadians / 2,
    vRadians / 2,                  -vRadians / 2);

> side 카메라 이미지(projectionImages[camIdx])마다
"cubox_out/debug/000123/projections/crop_<camIdx>.png" 에 저장
```

```
// measured in radians from forward → 최대 FOV 절반 각도를 리턴!!!
float approximateFov(const Camera& camera, const bool vertical) {
  Camera::Vector2 a, b = camera.principal;
  if (vertical)   a.y() = 0;    b.y() = camera.resolution.y();
  else            a.x() = 0;    b.x() = camera.resolution.x();
  return acos(max(          // a, b FOV 가 다를수도 있나? 오차???
    camera.rig(a).direction().dot(camera.forward()),
    camera.rig(b).direction().dot(camera.forward())));  }
```




```
// horizontal
a(0, 1024), b(2048, 1024)
// vertical
a(1024, 0), b(1024, 2048)
```



hRadians 가 90도이면, 6300 / 4 = 1575,
vRadians 가 90도이면, 3072 / 2 = 1536
실제
render\debug\000000\projections\crop_c
am1.png 이미지의 경우, 1666 x 1460

Render (Surround360\surround360_render\source\test\TestRenderStereoPanorama.cpp)

```
// renderStereoPanorama()                                  -- 계속 --
  projectSphericalCamImages(...) 호출 이후에
  Mat sphericalImageL, sphericalImageR;
  const double fovHorizontal = 2 * approximateFov(
  rig.rigSideOnly, false) * (180 / M_PI); // 카메라 중 최대값
generateRingOfNovelViewsAndRenderStereoSpherical(
    rig.getRingRadius(),
    fovHorizontal,          projectionImages/*side camera 이미지*/,
    sphericalImageL/*out*/,  sphericalImageR,
    opticalFlowRuntime,                 novelViewRuntime);

padToheight(sphericalImageL, FLAGS_eqr_height); // 상하패딩
padToheight(sphericalImageR, FLAGS_eqr_height);

topFlowThreadL/R 쓰레드 생성(poleToSideFlowThread)
> topSphericalWarpedL 에 결과 저장
bottomFlowThreadL/R 쓰레드 생성(poleToSideFlowThread)

sphericalImageL = flattenLayersDeghostPreferBase(sphericalImageL,
topSphericalWarpedL); // 이미지 합치기???
imwrite~("cubox_out/debug/000123/eqr_sideL.png", sphericalImageL);
_eqr_sideL_sharpened.png 도 저장

if (FLAGS_cubemap_width > 0) 이면,
Mat stereoCubemap 이미지를 생성해
~/Desktop/test/render/cube_frames/cube_000000.png 에 저장
최종으로 Mat stereoEquirect 이미지를 생성해
~/Desktop/test/render/eqr_frames/eqr_000000.png 에 저장 // dest_dir
```

```
float overlapAngleDegrees = // 등등 정의부분!
   (camFovHorizontalDegrees * float(numCams) -
360.0) / float(numCams/*14*/); // 겹치는 앵글?

const int camImageWidth = projectionImages[0].cols;
int overlapImageWidth =
  float(camImageWidth) * (overlapAngleDegrees /
camFovHorizontalDegrees); // 겹치는 폭?
```

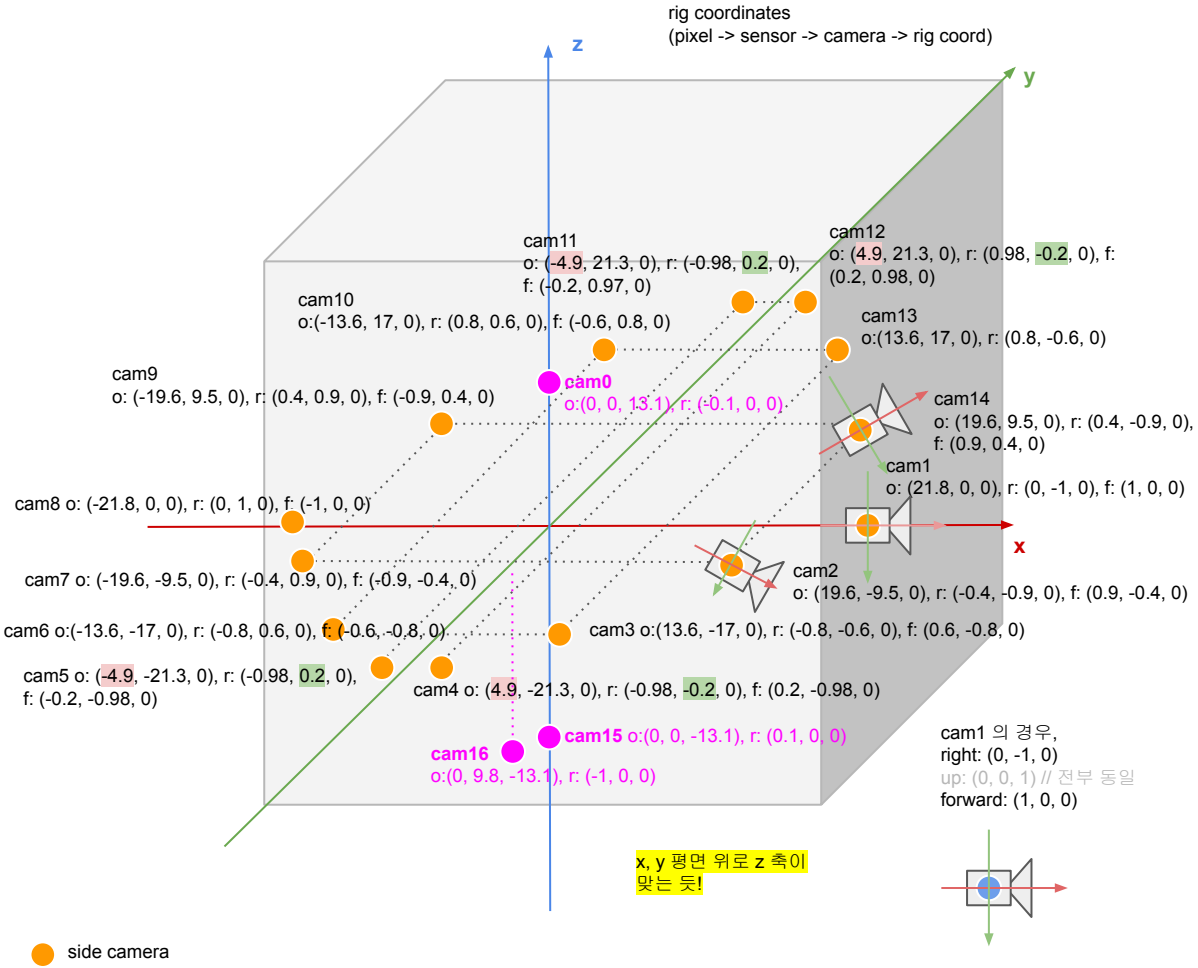
최종 결과 이미지
eqr_000000.png
cube_000000.png

```
// generates a left/right eye equirect panorama using slices of novel views
void generateRingOfNovelViewsAndRenderStereoSpherical(
    const float cameraRingRadius,              const float camFovHorizontalDegrees,
    vector<Mat>& projectionImages,
    Mat& panoImageL,                    Mat& panoImageR,
    double& opticalFlowRuntime,     double& novelViewRuntime)
> const int numCams = projectionImages.size(); // 14개
const int overlapImageWidth =
    float(camImageWidth) * (overlapAngleDegrees / camFovHorizontalDegrees);
for (int leftIdx = 0; leftIdx < projectionImages.size(); ++leftIdx) {
  novelViewGenerators[leftIdx] =
      new NovelViewGeneratorAsymmetricFlow(FLAGS_side_flow_alg);
  threads.push_back(std::thread(prepareNovelViewGeneratorThread,
      overlapImageWidth,      leftIdx,
      &projectionImages[leftIdx],     &projectionImages[rightIdx],
      novelViewGenerators[leftIdx]
    )); }
for (int leftIdx = 0; leftIdx < projectionImages.size(); ++leftIdx) {
    panoThreads.push_back(std::thread(renderStereoPanoramaChunksThread,
      leftIdx,     numCams,     camImageWidth,     camImageHeight,
      numNovelViews,     fovHorizontalRadians,
      vergeAtInfinitySlabDisplacement,     novelViewGenerators[leftIdx],
      &panoChunksL[leftIdx],     &panoChunksR[leftIdx]
    )); }
panoImageL/R = stackHorizontal(panoChunksL);
panoImageL/R = offsetHorizontalWrap(panoImageL, zeroParallaxNovelViewShiftPixels);
```

```
void prepareNovelViewGeneratorThread(
    const int overlapImageWidth,
    const int leftIdx, // only used to determine debug image filename
    Mat* imageL,          // side camera 이미지
    Mat* imageR,
    NovelViewGenerator* novelViewGen) {
  novelViewGen->prepare( // 여기서 flowLtoR 멤버에 optical flow 저장?!
    overlapImageL/R, // imageL/R 에서 겹치는 overlapImageWidth 영역
    prevFrameFlowLtoR/RtoL, prevOverlapImageL/R); // 선언만 된 변수
```

```
void renderStereoPanoramaChunksThread(
    leftIdx, ..., Mat* chunkL/R) {
> *chunkL = lazyNovelChunksLR.first;
```

Render (/surround360_render/res/config/camera_rig.json)



surround360_render\res\config\camera_rig.json

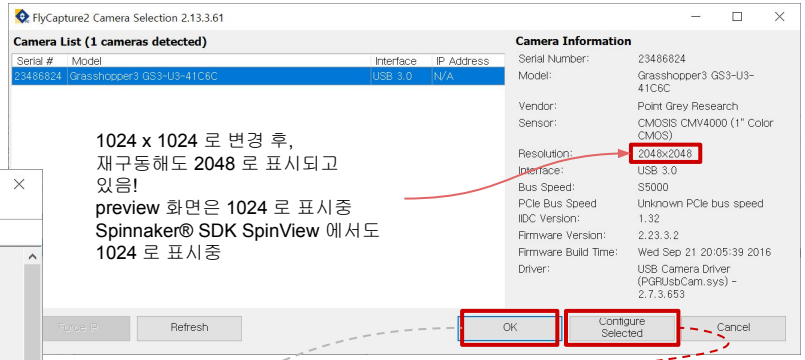
```
{
  "cameras" : [
    {
      "group" : "side camera",
      "id" : "cam1",
      "origin" : [ // rig 중심 기준 cm 좌표
        21.799999237060547,
        2.4202861089795465E-15,
        -7.26085832693864E-15
      ], // --> Camera::position
      "principal" : [                --> 전부 동일!
        1024,       1024
      ],
      "right" : [                    --> -Y axis
        3.3306690738754696E-16,
        -1,
        1.1102230246251565E-16
      ],
      "up" : [                       --> Z axis
        1.1102230246251565E-16,
        3.3306690738754696E-16,
        1          --> 전부 동일(side cameras)
      ],
      "forward" : [                  --> X axis
        1,
        1.1102230246251565E-16,
        -3.3306690738754696E-16
      ],
      "focal" : [ 초점거리? side cameras 모두 동일
        1269.580673376528,
        -1269.580673376528
      ],
      "resolution" : [
        2048,       2048
      ],
      "type" : "RECTILINEAR",
      "distortion" : [
        0,       0
      ],
      "version" : 1
    },
    {
      "group" : "side camera",
      "id" : "cam2",
```

...

```
    {
      "group" : "side camera",
      "id" : "cam14",
      "origin" : [
        19.641120632888047,
        9.45866518173573,
        -1.2101430544897733E-15
      ],
...
    {
      "group" : "",      "id" : "cam0",
      "origin" : [
        0,       0,       13.1
      ],
      "principal" : [       1024,       1024 ],
      "right" : [       -1,       0,       0      ],
      "up" : [       0,       1,       0      ],
      "forward" : [       0,       0,       1      ],
      "focal" : [
        483.76220324,       -483.76220324
      ],
      "resolution" : [       2048,       2048 ],
      "type" : "FTHETA",
      "distortion" : [
        0,       0
      ],
      "fov" : 1.61443, --> all fisheye same
      "version" : 1
    },
    {
      "group" : "",
      "id" : "cam15",
      "origin" : [
        -0,       -0,       -13.1
      ],
...
    {
      "group" : "",
      "id" : "cam16",
      "origin" : [
        0,       9.8,       -13.1
      ],
...
```

해상도 변경 (TODO: 실패 원인 검증)

윈도우 시작 / Point Grey FlyCap2 실행 후,
Configure Selected 버튼을 클릭 후,
Custom Video Modes 에서 Mode 를 1로 변경 후, 하단의 Apply 버튼을 클릭(해상도가 1024 x 1024 로 변경됨)



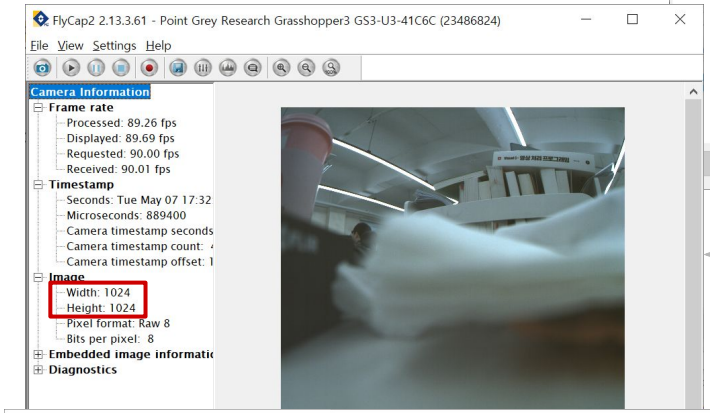
1024 x 1024 로 변경 후,
재구동해도 2048 로 표시되고 있음!
preview 화면은 1024 로 표시중
Spinnaker® SDK SpinView 에서도 1024 로 표시중

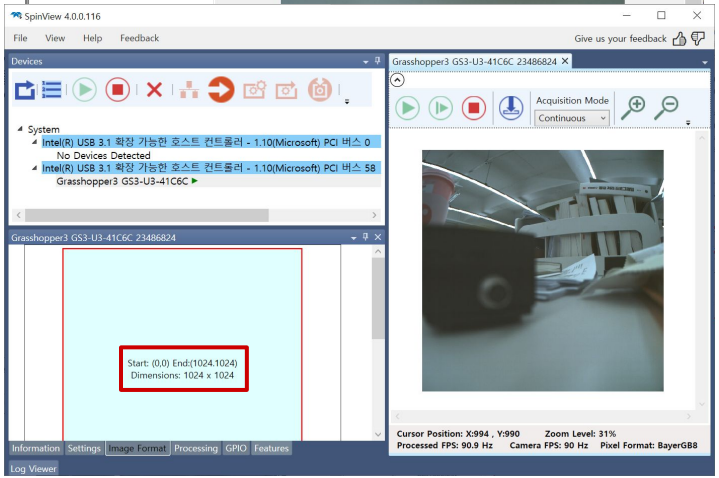
하지만, 이렇게 해상도를 변경할 경우,

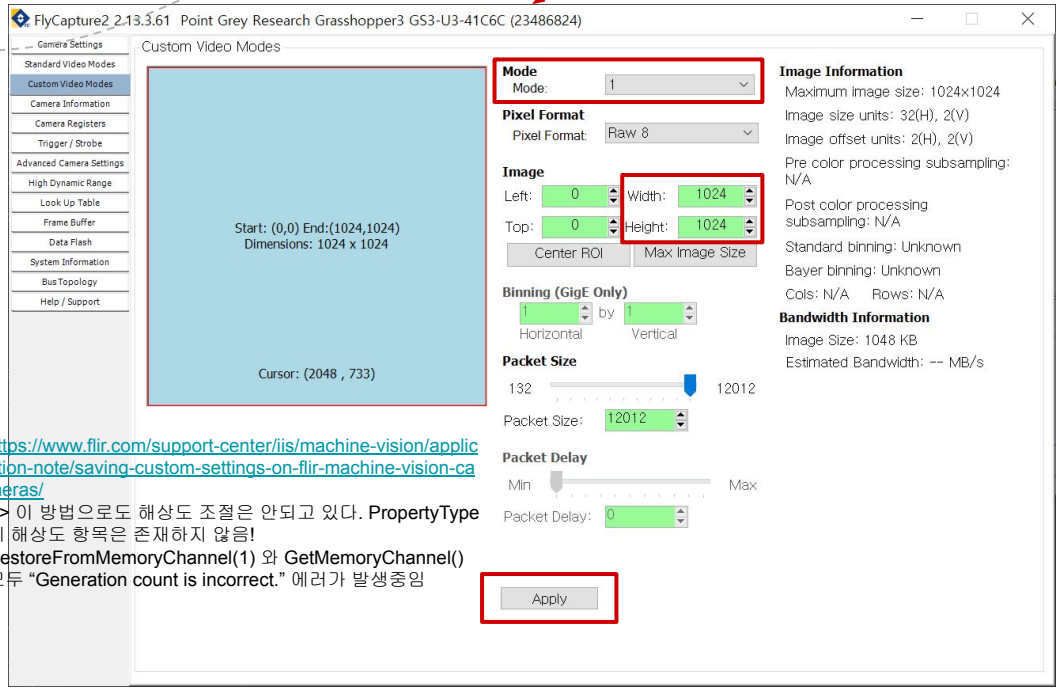
1) CameraControlUI 를 실행시 PointGreyCamera::getCamera(index) 에서 pgcam->m_width = fmt7Info.maxWidth; 가 2048 로 리턴되고 있음

2) 소스 전반적으로 2048 로 하드코딩된 부분들 수정이 필요해보임

3) calibration 도 1024 해상도에서 다시 해줘야 함(e.g, camera_rig.json 의 resolution, principal 등등)







https://www.flir.com/support-center/iis/machine-vision/application-note/saving-custom-settings-on-flir-machine-vision-cameras/
--> 이 방법으로도 해상도 조절은 안되고 있다. PropertyType 에 해상도 항목은 존재하지 않음!
RestoreFromMemoryChannel(1) 와 GetMemoryChannel() 모두 “Generation count is incorrect.” 에러가 발생중임

EOD