영상처리를 활용한
가상 악기 및 리듬 게임

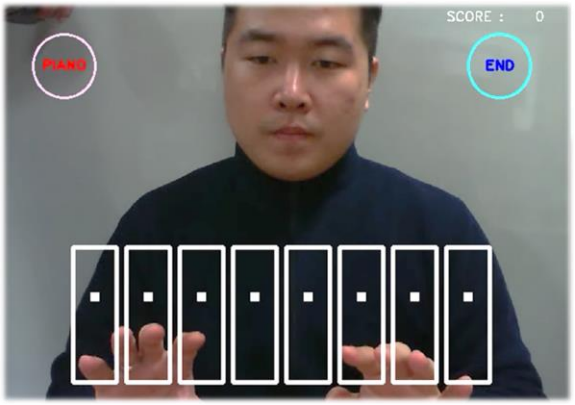
홍길동

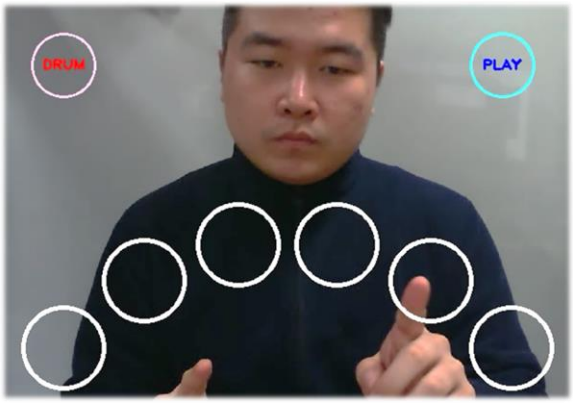
시연 영상

https://youtu.be/3qcJlSSl2Q0

- 프로젝트 명 : CH2 Band
- 개 발 기 간 : 2016.05.09 ~ 2016.05.20
- 개 발 목 적 : 웹 캠을 통해 사용자의 손 모션을 인식하여 손쉽게 다룰 수 있는 가상 악기(피아노, 드럼) 구현 및 사용자에게 재미를 줄 수 있는 리듬게임 구현.

프로젝트 개요





- OpenCV를 활용한 손 검출 및 모션인식. 가상악기 구현
- Fmod를 활용한 사운드 처리
- 영상과 사운드를 병합한 리듬게임 구현

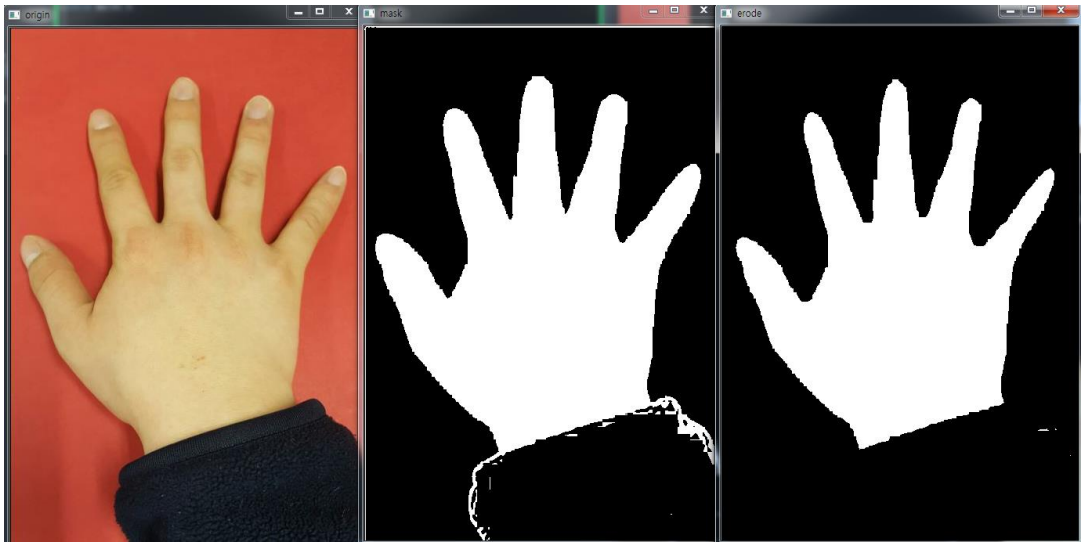
개발 환경

OS	Window 7
Language	C++
Library	OpenCV 3.0 / Fmod 4.4
Tool	Visual Studio 2013
Device	MS Life 3000 WebCAM

OpenCV

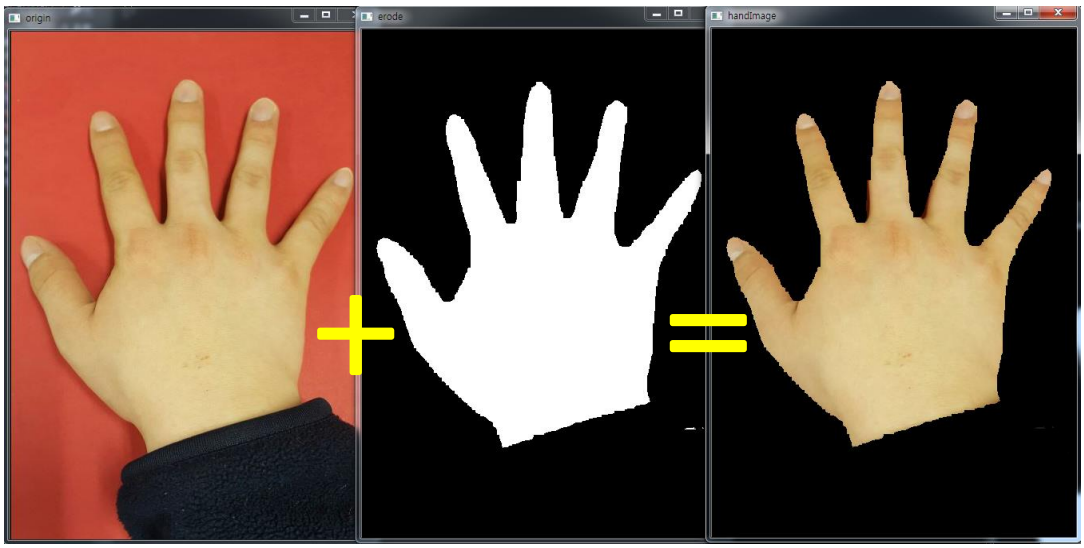
Visual Studio

손 검출



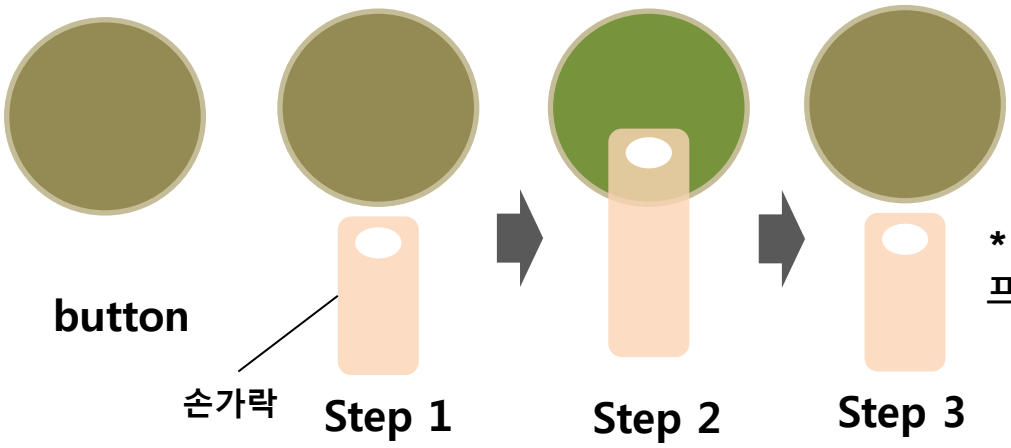
- 색상 값 YCrCb 범위를 통한 피부색 마스크 검출.
- erode() 를 통한 마스크 침식 연산

손 검출



- 원본 영상과 마스크(이진화)된 영상의 연산을 통한 손 영역 검출

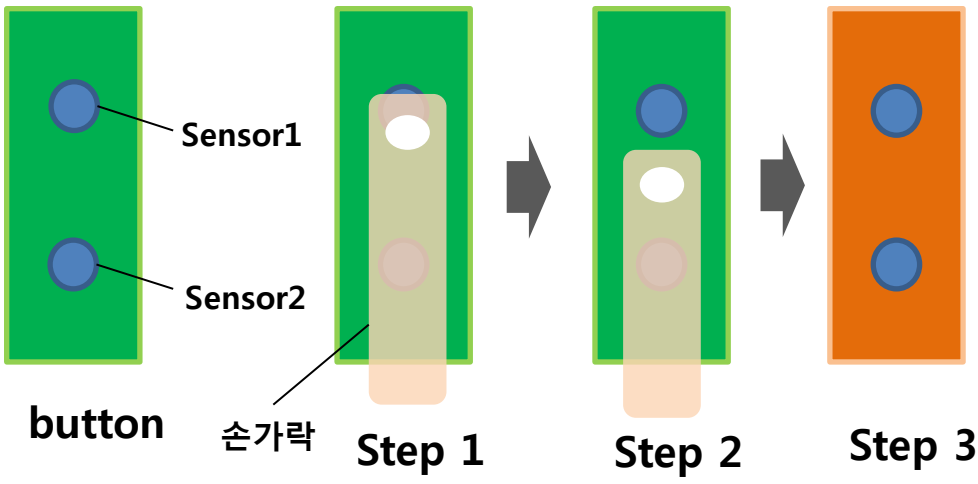
모션 인식 - 버튼 & 드럼



* 각 Step가 바뀔땐, 프레임 단위로 시분할.

- 손가락이 버튼을 가린 후 일정 프레임(시간)이 지나면 Button_Ready = True
- Button_Ready = True 일 때, 손가락이 버튼을 벗어나면 Button_On = True

모션 인식 - Piano



* 각 Step가 바뀔땐, 프레임 단위로 시분할.

- 손가락이 센서 2개를 가리면 Button_Ready = True
- Button_Ready = True 일때 1번 센서가 눌리지 않으면 Button On = True

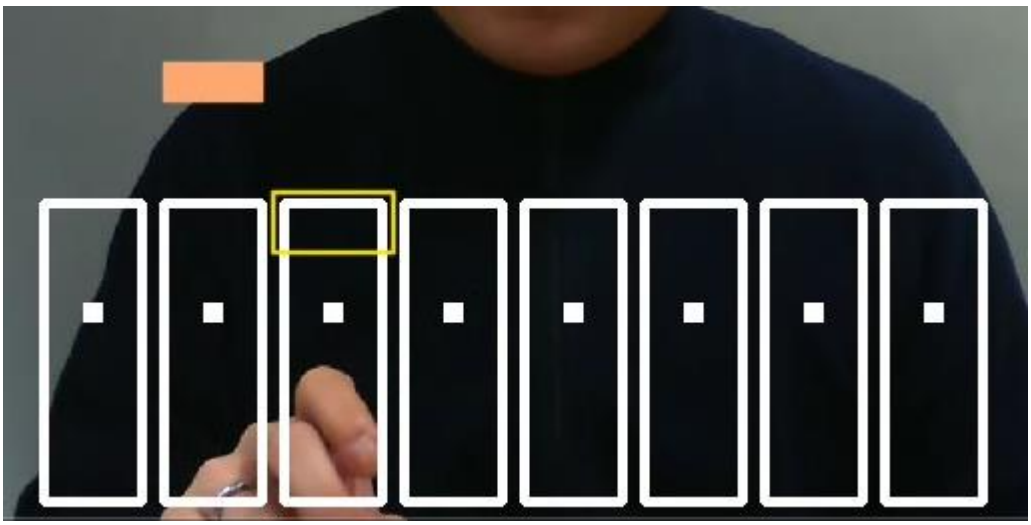
Fmod - 음향 처리

```
#include "inc/fmod.hpp"
#pragma comment (lib, "fmodex_vc.lib")
#define MAX_PIANO       8
using namespace FMOD;
class PIANO
{
private:
    System* g_pSystem;
    Sound* g_pSound[MAX_PIANO];
    Channel* g_pChannel;
public:
    PIANO(){
        System_Create(&g_pSystem);
        g_pSystem->init(8, FMOD_INIT_NORMAL, 0);
        g_pSystem->createSound("piano₩₩HighDo.wav", FMOD_HARDWARE, 0, &g_pSound[7]);
        g_pSystem->createSound("piano₩₩Si.wav", FMOD_HARDWARE, 0, &g_pSound[6]);
        g_pSystem->createSound("piano₩₩La.wav", FMOD_HARDWARE, 0, &g_pSound[5]);
        g_pSystem->createSound("piano₩₩Sol.wav", FMOD_HARDWARE, 0, &g_pSound[4]);
        g_pSystem->createSound("piano₩₩Pa.wav", FMOD_HARDWARE, 0, &g_pSound[3]);
        g_pSystem->createSound("piano₩₩Mi.wav", FMOD_HARDWARE, 0, &g_pSound[2]);
        g_pSystem->createSound("piano₩₩Re.wav", FMOD_HARDWARE, 0, &g_pSound[1]);
        g_pSystem->createSound("piano₩₩Do.wav", FMOD_HARDWARE, 0, &g_pSound[0]);
```

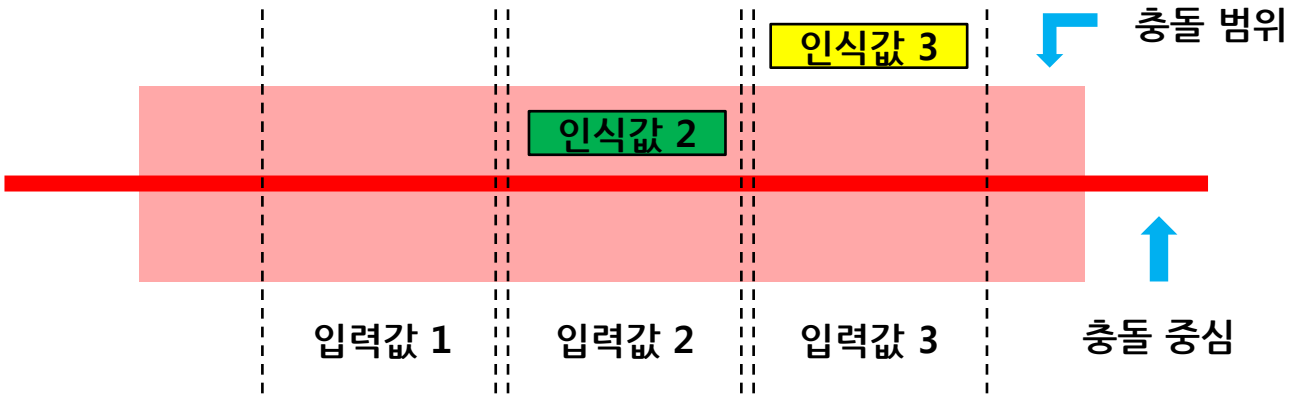
```
void Play(int n)
{
    g_pSystem->playSound(FMOD_CHANNEL_FREE, g_pSound[n], 0, &g_pChannel);
    g_pSystem->update();
}
```

- 클래스 생성자에 createSound를 추가하여 각 음계 생성 및 배열에 저장
- Play() 함수의 playSound를 통해 배열에 저장된 음원 재생.
- Update를 통해 각 음계의 동시재생.

리듬 게임 - 충돌처리

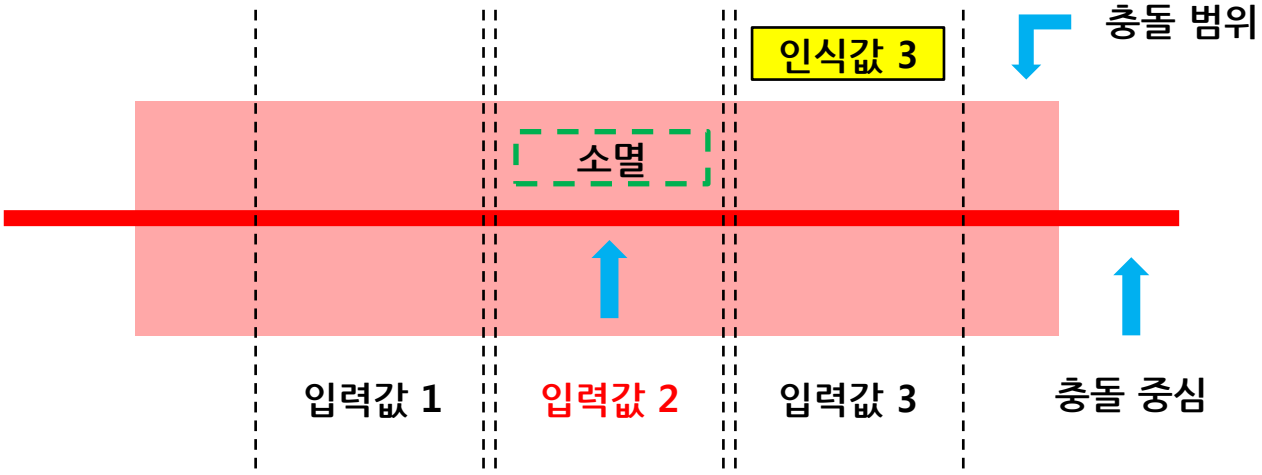


리듬 게임 - 충돌처리



- 비트 박스와 동작 음계에 각 음계에 맞게 인식하는 변수를 부여함.
- 비트 박스가 충돌 처리할 범위 내로 위치할때 입력을 체크하는 클래스를 실행.

리듬 게임 - 충돌처리



- 해당 범위 내에서 박스의 변수와 입력된 변수가 동일할 시 박스를 소멸시키는 함수를 실행하여 박스를 소멸시킨다.
- 박스가 소멸되거나 범위를 벗어날 시 입력 체크 상태를 종료하고 입력되었던 값을 0으로 초기화 한다.