Project – Final Report

[10조] 영화관 매표 시스템

과 목	임베디드시스템S/W설계
담당교수	김태석 교수님
학 과	컴퓨터공학과
학 번	2013722058, 2012722073
성 명	김현지, 정수찬
날 짜	2017. 06. 12 (월)

광운대학교 KwangWoon University

1. 개요

- 이번 프로젝트에서 구현한 주제는 영화관 매표 시스템으로써 하나의 티켓 판매소가 있는 상황에서, 일반 손님들과 VIP 손님들을 받는다. 일반 손님의 경우, 들어온 순서대로 티켓을 구입하기 위해 줄을 서지만 VIP 손님들의 경우는 들어오는 즉시, 창구가 비자마자 바로 티켓을 구입할 수 있다. 또한 매표소는 한 번에 하나의 티켓만 발급해줘서 vip가 티켓을 받을 때는 일반 손님은 잠시 대기하도록 동작하였다.

2. 구현 내용

A. 프로젝트 설명

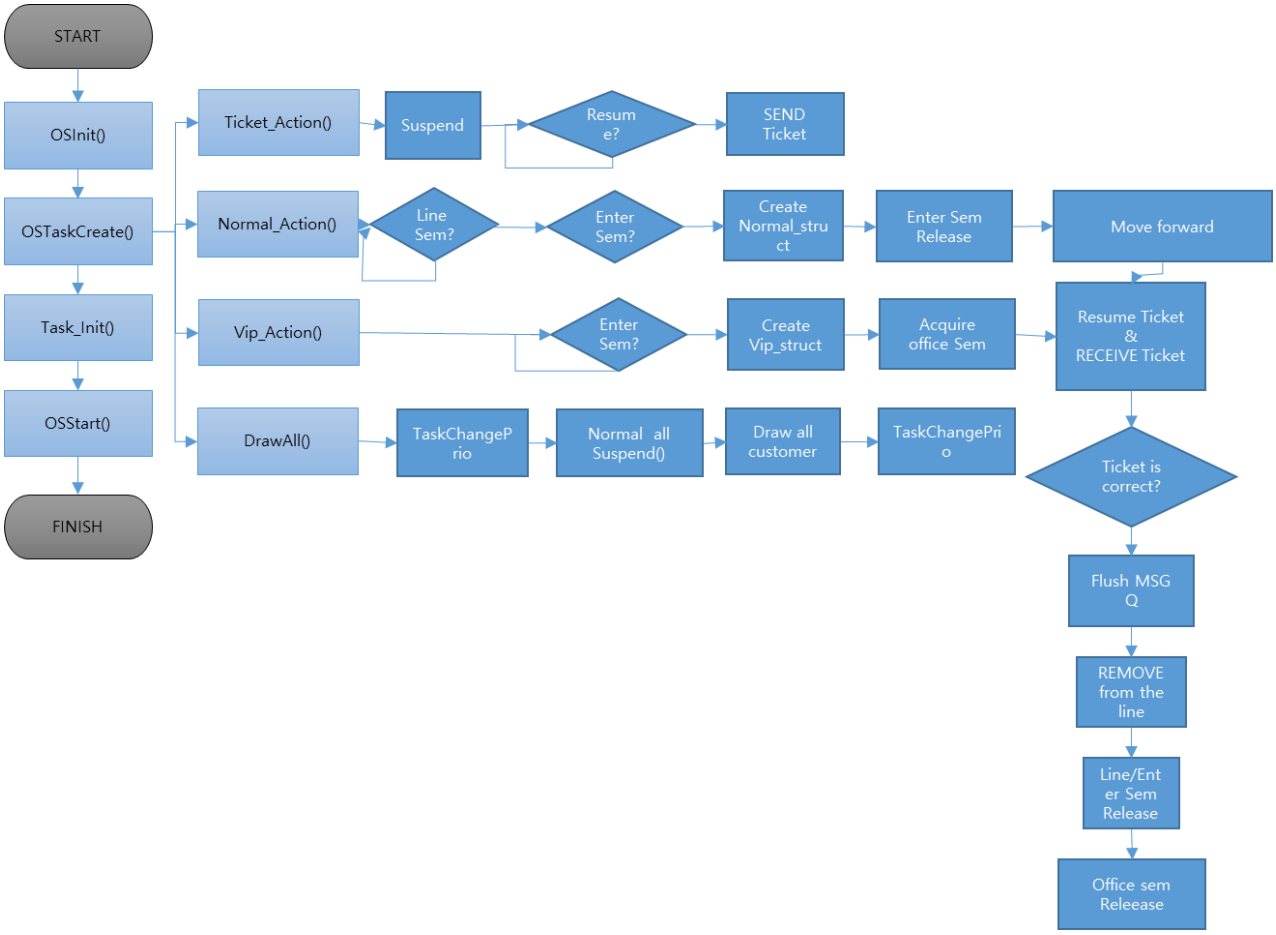
- 영화관 매표 시스템은 영화 티켓 구매를 위해, 입장 순서에 따른 대기 줄을 세워 대기 순서대로 우선순위를 주어 손님들의 더욱 빠른 대기를 위해 만들었다. 영화관에서는 조금 더 편한 시스템을 위해 VIP 손님들에게는 보다 좋은 혜택을 제공하는데, 그 혜택 중 한 가지가 바로 대기 줄 없이 바로 창구에서 티켓을 구매 가능하다는 점이다. 따라서, 한 창구에서 손님이 사용 중이라면, 세마포어를 사용 불가한 상태로 만들고, 손님이 티켓을 발급 받는 순간 창구가 사용 가능한 상태가 되므로 세마포어를 해제시킨다. 이 때, 줄을 서 있는 대기 손님들 보다 VIP 손님이 먼저 창구를 사용할 수 있도록 세마포어를 가져가게 된다.
- 각각의 손님들은 현재 줄을 서 있는 위치와, 서로 다른 손님들을 구별하기 위해 랜덤으로 주어지는 색, 현재 고객이 티켓을 받고 있는 중인지 등의 고객 상태 등을 Customer라는 구조체 내의 변수들로 가지고 있게 된다.

B. 사용 기법 설명

- 구현한 프로그램에서의 창구는 하나이기 때문에 일반 손님의 대기 줄 역시 한 줄로 세우며, 한 줄로 서는 세마포어를 갖거나 해제할 수 있다. 또한 일반 손님은 30명이 줄에 들어갈 수 있도록 경쟁하며 한 줄에는 최대 10명을 세울 수 있으며 10명이 넘어가게 되면 대기 줄에 못 서도록 세마포어를 만들었다. 그러나, VIP 손님의 경우는 들어오자마자 티켓을 구입할 수 있으므로 줄을 서지 않아서 VIP손님들만의 세마포어는 1개의 자원만 할당해줬다. 또한 매표소에서 티켓은 한 번에 한 장씩 발급해주기 때문에 매표소 세마포어에도 1개의 자원만 할당해줬다.
- 매표소에서 티켓을 발급해 주는 것을 메시지 큐로 구현하였으며 티켓은 한번에 하나의 티켓만 부여해줄 수 있도록 메시지를 하나 보내는 즉시 suspend시켰다. 또한 vip가 티켓을 메시지 큐로 받는 동안 일반 손님들이 다른 동작을 하지 못하도록 모두 suspend 시키고 vip가 줄을 섰는지 파악한 후에 resume 시킬지를 판단하였다.
- 일반 손님들이 이동하는 도중에 그리기 태스크가 끼어들게 되면 하나의 자리에 두 명

이상이 차지할 수 있어서 그리기 함수의 시작 우선순위를 61로 부여한 후에 그리고 나서 손님 태스크들을 깨우는 중에도 방해를 받지 못하게 하기 위해 동작하는 도중에는 우선순위를 0으로 바꾸어주었다. 이를 반복하여 우선순위에 의해 움직이는 동작과 그리는 동작이 서로 방해받지 못하도록 설정하였다.

C. FLOW CHART



D. Pseudo Code

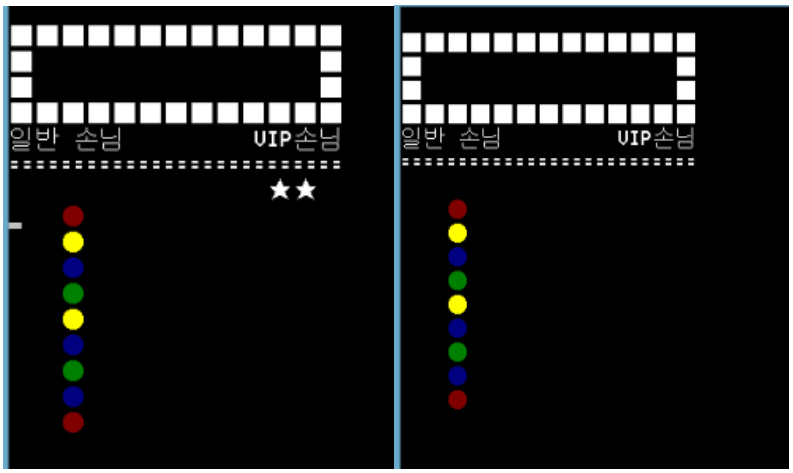
```
int main(void){
        INT8U i; //변수 i선언

        OSInit(); // uC/OS-II 초기화
        OSTaskCreate(DrawAll) 으로 그리기 태스크 생성
        OSTaskCreate(Ticket_Action) 으로 매표소 태스크 생성
        For 0 to 10
                OSTaskCreate(Vip_Action) 으로 vip 태스크 생성
                VIP_INFO[i].state를 REMOVE로 초기화
```

```
        For 0 to 30
                OSTaskCreate(Normal_Action) 으로 일반 손님 태스크 생성
                VIP_INFO[i].state를 REMOVE로 초기화

        세마포어 및 메시지 큐 초기화

        태스크 시작

        return 0;
}


/**
* 일반 고객 행동 함수
*/
void Normal_Action(void *pdata)
{
        normal_num = 해당 태스크에 해당하는 Normal_Info의 번호를 부여

        while (true){
                한 줄 세마포어 획득
                줄 접근 세마포어 획득


        Normal_Info[normal_num] 의 위치를 현재 줄의 가장 끝으로 설정
        Normal_Info[normal_num] 상태를 이동 가능(줄에 있음)으로 설정
        Normal_Info[normal_num]의 색를 4가지 색 중 랜덤으로 설정

        1초 딜레이

        줄 접근 세마포어 반환

                While(맨 앞줄이 아니라면)
                {
                        if (이동 가능하면)
                                한 칸씩 앞으로 이동

                        if (맨 앞줄이면) break;

                        화면을 그리기 위해 해당 태스크 suspend
```

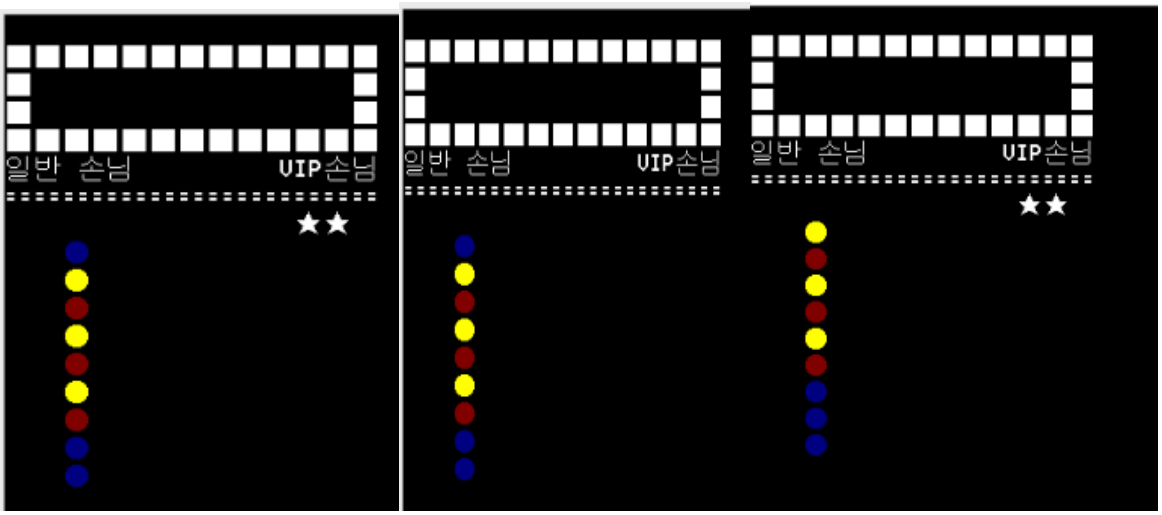
```
                }
        창구세마포어 얻기
        매표소 깨움

        msg =메시지 큐에 ticket이 들어오는 것을 기다림

        if (메시지가 있다면)
                메시지 큐 비우기

        Normal_Info[normal_num]의 state를 REMOVE로 설정
        한줄 세마포어 반환
        창구세마포어 반환
}}

/**
* vip 고객 행동 함수
*/
void Vip_Action(void *pdata)
{
        vip_num를 pdata로 설정
        while (1){

                vip는 100초 중 랜덤으로 뜰 수 있도록 딜레이

                vip 한줄 세마포어 획득
                Vip_Info[vip_num]의 위치를 맨 앞줄로 설정
                Vip_Info[vip_num]의 상태를 MOVE로 설정
                Vip_Info[vip_num]의 색을 두가지 중 랜덤으로 설정

                1초 딜레이

                해당 태스트 suspend
                창구세마포어 획득
                매표소 태스크 resume

                msg = 메시지 큐로 ticket 받기

                if (메시지가 있다면)
                        메시지 큐 비우기

                Vip_Info[vip_num]의 상태를 REMOVE로 설정
```

```
                창구 세마포어 반환
                vip 한 줄 세마포어 반환
        }}

/**
* 매표소 행동 함수
*/
void Ticket_Action(void *pdata){
        while (1){
                suspend
                티켓을 메세지로 보냄
        }
}

// 일반 고객 태스크 모두 멈춤
void Normal_All_Suspend()
{
        일반 고객 태스크 모두 suspend
}

//화면 그리는 함수
void DrawAll(void *pdata)
{
        while (1)
        {
                그리기 태스크의 우선순위를 0으로 바꾸기
                Normal_All_Suspend 함수로 모든 일반 고객 태스크 멈춤

                모든 화면을 공백으로 그리기

                티켓 창구 그리기
                줄 서 있는 일반고객 그리기
                줄 서 있는 VIP고객 그리기

                Vip가 줄을 섰는지 확인
                줄을 서지 않았으면 모든 일반 고객 태스크 resume
                모든 vip 태스크 resume
                1초 딜레이
                해당 태스크의 우선순위를 61로 낮춤
        }
}
```

3. 동작 예시



- 위 그림과 같이 vip가 오는 경우 왼쪽의 빨간색으로 표기된 일반 손님이 기다리고, 하얀별로 표기된 vip 손님이 먼저 처리되어 오른쪽 그림과같이 vip 손님이 티켓을 받아서 줄에서 삭제된 것을 볼 수 있다.



- 역시 맨 왼쪽처럼 vip가 들어오면 맨 앞의 파란색 손님은 잠시 대기하고, vip 손님 처리가 수행된다. Vip 손님 처리가 끝나면 두번째 그림처럼 vip 손님은 사라지고, 세번째 그림에서 보면 파란색 손님이 처리된 것을 볼 수 있다. 그 후 vip 손님이 들어왔으니 노란색 손님은 대기하고 vip가 먼저 처리될 것이다.

4. 고찰

- 생각보다 semaphore를 넣어서 처리하는 게 쉽지 않았고, 태스크가 따로 따로 하나씩 수행되는 것이 아니라 동시에 수행되는 것이라 코드 짜기에 굉장히 어려움이 컸던 것 같다. 각 손님마다의 태스크를 부여해줘서 동작시키고 그리는 태스크도 따로 만들었기 때문에 서로 동작하는 부분에 있어서 침범하는 것을 제어하는 것에 가장 고민을 많이 하였다. 또한 어느 부분에 조건을 걸어줘야 할지 생각하는 것이 제일 어려웠고, 우선순위와 줄 서 있는 순서가 다르다 보니, 그림으로 표현할 때 생각대로 잘 표현되지 않아서

여러 번의 시도를 거쳐야 했다.

또, semaphore와 message queue를 모두 써야 해서 어떻게 이용을 해야 할지도 굉장히 어려운 고민이었고, 원래 여러 창구를 두어 여러 창구가 빌 때마다 세마포어를 해제하여 운영하는 식으로 계획을 했었으나, 하나의 창구만으로도 세마포어나 현재 테스크, 우선순위 등 복잡하여 창구는 하나로 축소하게 되었다. 그래서 하나의 창구에서 일반 손님과 vip 손님을 동시에 받는 것으로 구성하였으며 vip가 동작하는 도중에는 일반 손님이 동작하지 못하도록 하여야 하는데 해당 부분은 vip 부분에서 제어가 힘들어서 draw 태스크에서 제어하도록 설정하였다. 그래서 결국 draw 태스크에 전체적인 동작을 어느 정도 제어하는 역할도 같이 부여해주게 되었다.