간단함에 미치다

대학원생 공돌이 하이머딩거

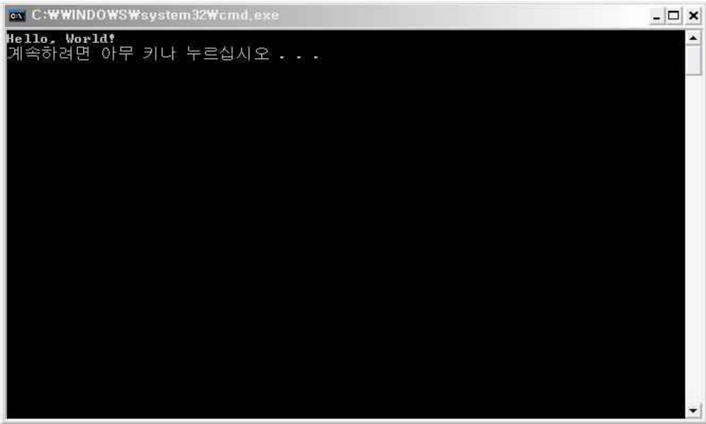
Daum 로그인

ㄱ” 그림을 그려보자

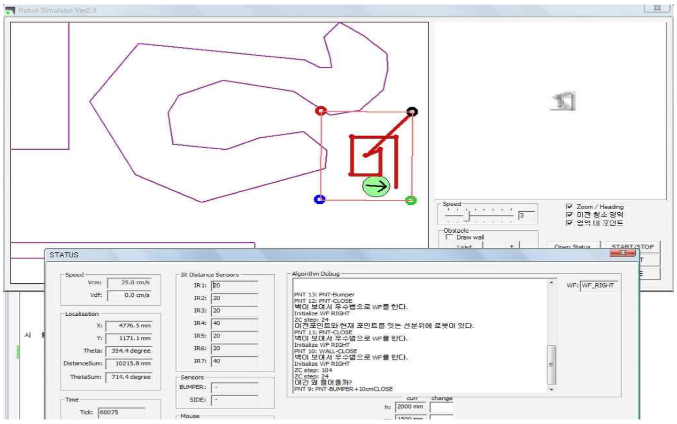
robot_society.zip

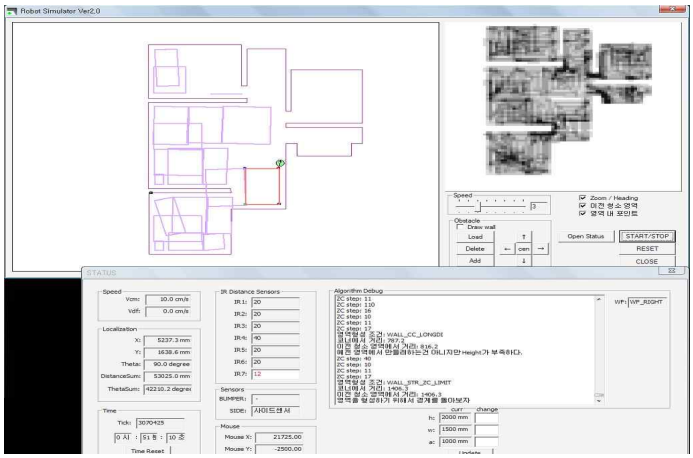
학교를 다니던 3년동안 Visual studio는 죽어라 했었지만 매번 검은 바탕에 흰 글씨가 띄워지는 것 외에는 하지 못했었다. encapsulation이다, poly morphism이다, template 등의 알면 좋지만 꼭 몰라도 괜찮은 것을 배우느라 시간을 쓰다가, 회사에 와서 mfc를 다루게 되었다.

화면에서 몬가 움직이는 것을 보았을 때의 그 즐거움이란 ! (아 코덕후같군..)
흡사 처음 Hello World! 를 띄웠을 때와 비슷한 느낌이었달까.



↓↓ 회사에서 만든 로봇시뮬레이터 ↓↓
화살표 달린 초록색 동그라미가 *청소 로봇*이다. (클릭해서)





몇 번의 비슷한 프로그램을 짜는 와중에서 그림을 그리고, 시뮬레이션 할 수 있는 프로그램을 짜는 것은 익숙해 졌으나 할 때마다 느낀 것은 이 것을 정리해 놓으면 좀 더 좋을 텐데 하는 생각이었다.

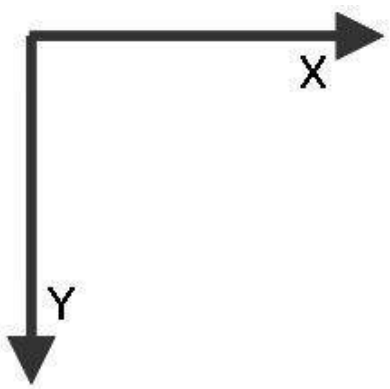
다이얼로그에 그림을 그리기 위해서는 크게 두 가지를 알아야 하고, 좀 더 잘 그리기 위해서는 몇가지 더 알아야 한다.
먼저 그림을 그리기 위해서 알아야 할 두 가지는 다음과 같다.

하나는 그림을 그릴 줄 알아야 하고, 다른 하나는 그림을 제대로 그릴 줄 알아야 한다. -_-;;

이런 짤방 하나 넣어보고 싶었다. 모 어차피 나 혼자 볼텐데 흠

그림을 그릴 줄 알아야 한다는 것은 CPaintDC dc(this);와 CPen, CBrush등을 이용해서 다이얼로그에 그림을 그릴 줄 알아야 한다는 것이다. dc의 개념에 대해서는 나도 잘 이해하지 못해서 *카페엔페이스트*를 쓰고있는 입장이라, 그냥 이렇게 하면 되더라 정도만 쓰겠다.

그림을 제대로 그릴 줄 알아야 한다는 것은 좌표 변환을 뜻한다. 우리가 수학시간에 배웠던 Cartesian Coordinate에서는 x는 우측을 y는 위를 향하고 있다. 왠진 모르겠지만 mfc에서는 x는 그대로지만 y는 아래를 향하게 되어있다.



그래서 X는 그대로 Y는 -Y로 각도는 -각도로 바꿔줘야 한다.

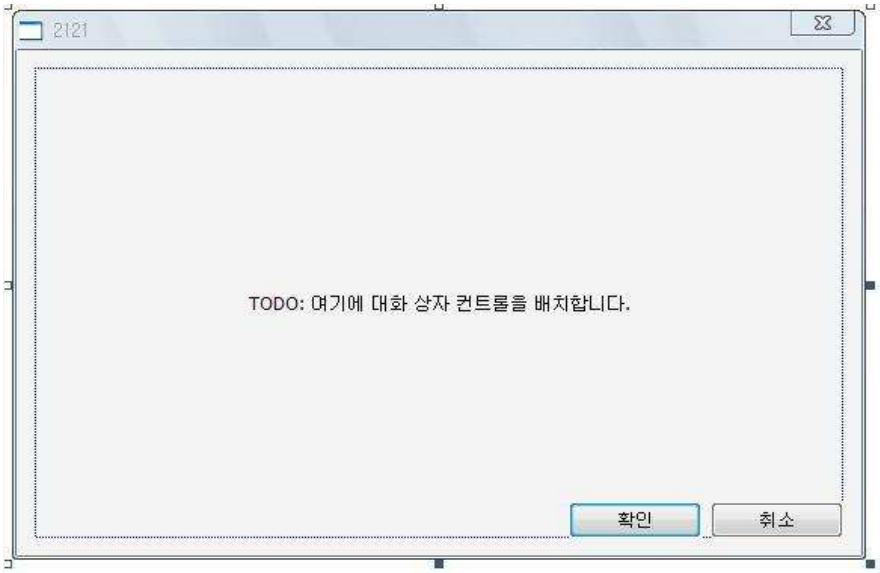
좀 더 잘 그리기 위해서는 그림을 보기 편하게 마우스 휠을 이용한 줌과 방향키를 이용한 그림 움직이기, 그리고 연속적인 사진이 끊어지지 않게 하기위한 더블 버퍼링이 필요하다.

물론 더 많겠지만, 그냥 간단한 시뮬레이터 정도 만들기 위해서는 이정도면 충분한..게 아니라 아는게 이정도 밖에 없다.

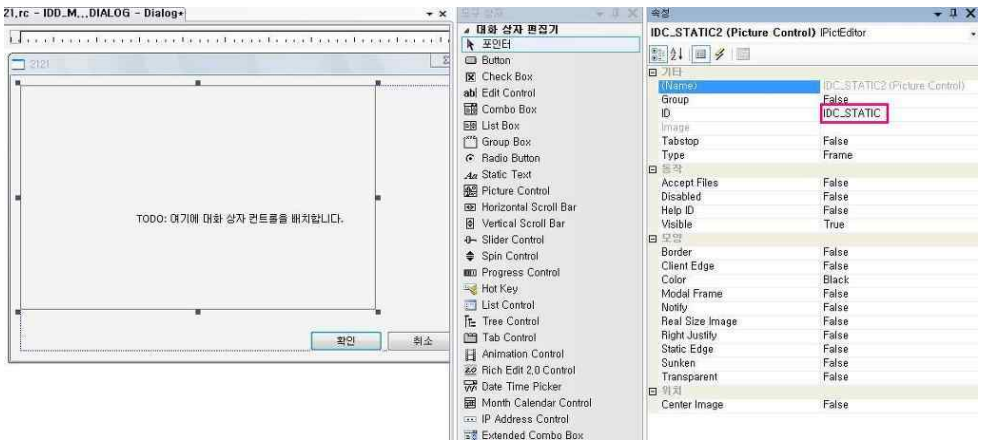
자 그럼 시작해보자 !

먼저 본인이 소장하고 있는 버젼의 visual studio를 연다. 그리고 프로젝트를 만든다. 나와 같은 커리큘럼을 들었다면 손이 자연스레 win32에 콘솔로 가겠지만 이번엔 mfc로 가보자.

MFC -> MFC 응용 프로그램 -> 이름 적고 -> 응용 프로그램 종류에서 대화 상자 기반 을 선택한다. 해본게 그것 뿐이다. -_ -;;
대충 다음 다음 마침 누르고 나면 프로젝트 하나가 완성 된다. Ctrl+F5를 눌러서 실행을 해보면 덩그라니 화면 하나가 뜨고 확인 취소 버튼 하나씩 있을 것이다.



그러고 나서 도구 상자에 Picture Control을 눌러서 내가 그림을 그리고 싶은 영역 만큼 네모를 그린다. 실은 이 부분은 그리 필요하지는 않는데, 코딩 할 때 조금 편하라고 하는 것이다. 그 이유는 하다보면 안다. 여튼 그렇게 하고나면 다음과 같은 화면을 얻을 수 있다.



Picture Control 화면은 그림의 왼쪽 위에 맞춘다. 일단 그렇게 해야 좀 편한다.

그림에서 빨간 네모에 있는 IDC_STAIC이 바로 Picture Control의 이름이다. 이것을 내가 보기 편하게 바꿔주면 좋다. 난 IDC_STATIC_STAGE1 이렇게 바꾼다.

TODO: 여기에 대화 상자 컨트롤을 배치합니다.

라고 되 있는건 거슬리니 지워버린다.

아 그림 올리는건 너무 귀찮다.
누가 이거 보려나 .

그림을 그리기 전에 몇가지 추가할 변수가 있다.
stage관련 변수인데, 이는 개인 style이니 내가 한 것을 따라하면 안좋다.

```
// STAGE1
int stage1_offset_x, stage1_offset_y;    //STAGE1에서 초기 화면의 offset
int stage1_top,stage1_btm,stage1_right,stage1_left;
int stage1_w, stage1_h;
int stage1_view_offset_x, stage1_view_offset_y;  //STAGE1에서 보여지는 offset
int stage1_view_center_x, stage1_view_center_Y;
double stage1_view_zoom;
```

자 여기까지 추가하고 나서는

BOOL CrobotsocietybetaDlg::OnInitDialog()

에서

```
RECT stage1_rect;
CStatic *pictureBox = (CStatic *)GetDlgItem(IDC_STATIC_STAGE1);
pictureBox->GetClientRect(&stage1_rect);
stage1_offset_x = stage1_offset_y = 12; //MFC Dialog 경계선
stage1_w  = stage1_rect.right;
stage1_h  = stage1_rect.bottom;
stage1_top  = stage1_rect.top + stage1_offset_y;
stage1_btm  = stage1_rect.bottom + stage1_offset_y;
stage1_right = stage1_rect.right + stage1_offset_x;;
stage1_left  = stage1_rect.left + stage1_offset_x;
```

이렇게 초기화를 시킨다.
아까 지정한 Picture Control의 크기를 읽어와서 그림을 그릴 크기를 잡는 부분이다.
점점 지루해지고 있기 때문에 대충쓰겠다.

```
stage1_view_offset_x = stage1_view_offset_y = 0;   //STAGE1에서 보여지는 offset
stage1_view_center_x = stage1_view_center_Y = 0;
stage1_view_zoom = 1.0;
```

이것들도 초기화 시킨다. 그리고

```
void CrobotsocietybetaDlg::DrawCircle(CDC* pDC, CPoint point, int radius)
{
    pDC->Ellipse( _X((int)(point.x) - radius)
        , _Y((int)(point.y) - radius)
        , _X((int)(point.x) + radius)
        , _Y((int)(point.y) + radius) );
}

void CrobotsocietybetaDlg::DrawCircle(CDC* pDC, CBrush* brush, CPoint point, int radius)
{
    pDC->SelectObject(brush);
    DrawCircle(pDC, point, radius);
}
```

위의 함수도 추가한다. 위의 함수는 동그라미를 그리는 함수이다. 내용은 별거 없다.

근데 이렇게까지 추가하고 컴파일 하면 에러가 뜰 것이다. _X, _Y함수가 없기 때문이다.
_X, _Y함수는 거창하게 말하자면 아까 위에서 말한 좌표계를 변환하고, zoom과 이동을 가능하게 하는 함수랄까..

```
int CrobotsocietybetaDlg::_X(double x)
{
    return (int)(x*stage1_view_zoom + stage1_view_offset_x + stage1_w/2);
}

int CrobotsocietybetaDlg::_Y(double y)
{
    return (int)(stage1_h/2 - y*stage1_view_zoom + stage1_view_offset_y);
}

int CrobotsocietybetaDlg::_X(int x)
{
    double d = (double)x;
    return _X(d);
}

int CrobotsocietybetaDlg::_Y(int y)
{
    double d = (double)y;
    return _Y(d);
}

CPoint CrobotsocietybetaDlg::_P(CPoint p)
{
    CPoint r;
    r.x = _X(r.x);
    r.y = _Y(r.y);
    return r;
}
```

여튼 위의 함수를 추가하면 컴파일은 될 것이다 하지만 그림이 그려지지 않을 것이다.
초기화 하고 함수만 추가했을 뿐 실제 그림을 그리는 부분은 없지 않는가.

그래서

void CrobotsocietybetaDlg::OnPaint()

에서 다음의 소스를 추가한다.

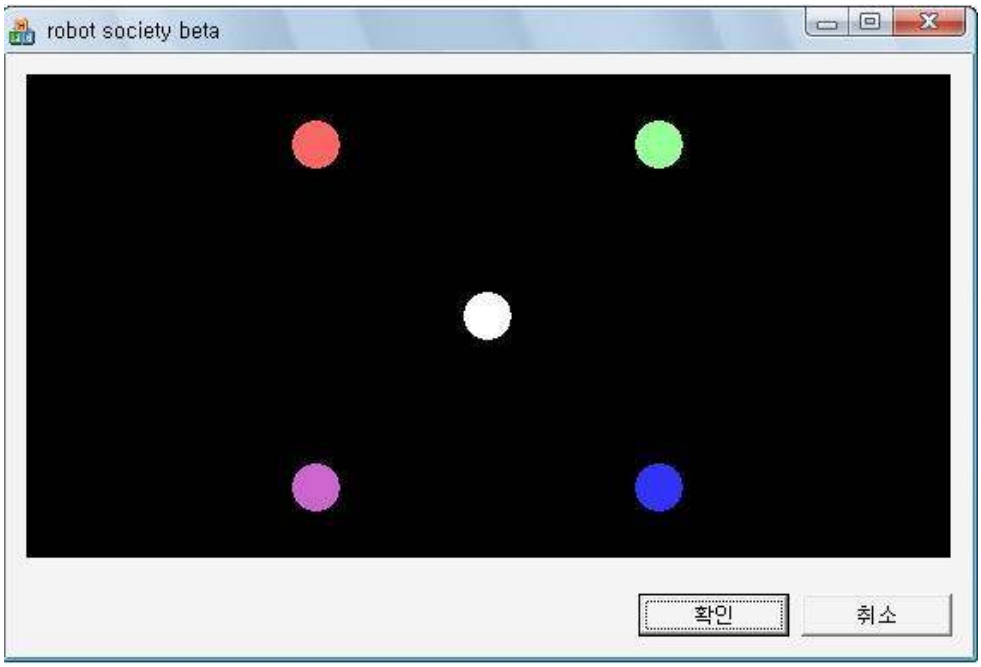
```
void CrobotsocietybetaDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // 그리기를 위한 디바이스 컨텍스트
        SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
        // 클라이언트 사각형에서 아이콘을 가운데에 맞춥니다.
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // 아이콘을 그립니다.
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CPaintDC dc(this);      // 그리기를 위한 디바이스 컨텍스트
        CRect rect;             // 클라이언트 영역의 크기
        CDC memDC;              // CDC for STAGE
        CBitmap imgBit;
        GetClientRect(&rect);   // 클라이언트 영역의 크기를 얻음
        memDC.CreateCompatibleDC(&dc);
        imgBit.CreateCompatibleBitmap(&dc, rect.Width(), rect.Height());
        memDC.SelectObject(&imgBit);

        CBrush brush_black(COLOR_D_BLACK);
        CBrush brush_lgreen(COLOR_L_GREEN);
        CBrush brush_lred(COLOR_L_RED);
        CBrush brush_lblue(COLOR_L_BLUE);
        CBrush brush_lpurple(COLOR_L_PURPLE);
        CBrush brush_white(COLOR_WHITE);

        memDC.SelectObject(brush_black);
        memDC.Rectangle(stage1_left, stage1_top, stage1_right, stage1_btm);

        CPoint point0(0,0), point1(100, 100),point2(-100, 100),point3(100, -100),point4(-100, -100);
        DrawCircle(&memDC, &brush_white, point0, 15);
        DrawCircle(&memDC, &brush_lgreen, point1, 15);
        DrawCircle(&memDC, &brush_lred, point2, 15);
        DrawCircle(&memDC, &brush_lblue, point3, 15);
        DrawCircle(&memDC, &brush_lpurple, point4, 15);
        dc.BitBlt(stage1_left, stage1_top, stage1_w, stage1_h, &memDC, 0, 0, SRCCOPY);
        //메모리 반환
        memDC.DeleteDC();
        imgBit.DeleteObject();
        CDialog::OnPaint();
    }
}
```

(위의 소스에는 더블 버퍼링도 숨어 있다. 잘 찾아보길.)

물론 이렇게 추가하고 나서 컴파일 하면 또 에러가 뜰텐데 이는 색에 해당하는 define이 안되어 있기 때문이다. 다음의 글을 어딘가에 추가하면 된다.

```
#define COLOR_L_PURPLE  RGB(0xCC,0x66,0xCC)
#define COLOR_L_GREEN   RGB(0x99,0xFF,0x99)
#define COLOR_L_RED     RGB(0xFF,0x66,0x66)
#define COLOR_L_BLUE    RGB(0x33,0x33,0xFF)
#define COLOR_D_BLACK   RGB(0x00,0x00,0x00)
#define COLOR_WHITE     RGB(0xFF,0xFF,0xFF)
```

자 여기까지 하고 나서 컴파일을 하고 실행을 하면 다음과 같은 화면이 뜰 것이다.



일단 그림을 띄우는데 성공했다 !

(참고로 말하자면 흰색 원이 원점 나머지 네 점은 (100, 100) (-100, 100) (100, -100) (-100, -100) 이다.)

이제 아래의 소스를 추가한다. 아래 소스들은 그냥 추가하면 안되고 다이얼로그 속성에 가서 추가를 해야한다. naver에 함수 이름치면 잘 설명해준다.

```
void CrobotsocietybetaDlg::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.
    m_pMousePt.x = point.x -5;
    m_pMousePt.y = point.y -10;
    CDialog::OnMouseMove(nFlags, point);
}
BOOL CrobotsocietybetaDlg::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
    // TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

    if(m_pMousePt.x>=stage1_left && m_pMousePt.x<=stage1_right && m_pMousePt.y>=stage1_top && m_pMousePt.y<=stage1_btm)
    {
        if(zDelta>0)
        {
            stage1_view_zoom += 0.03;
        }
        else
        {
            stage1_view_zoom -= 0.03;
            if(stage1_view_zoom<0.1)    stage1_view_zoom = 0.1;
        }
    }
    Invalidate(FALSE);
    return CDialog::OnMouseWheel(nFlags, zDelta, pt);
}
BOOL CrobotsocietybetaDlg::PreTranslateMessage(MSG* pMsg)
{
    // TODO: 여기에 특수화된 코드를 추가 및/또는 기본 클래스를 호출합니다.
    if(m_pMousePt.x>=stage1_left && m_pMousePt.x<=stage1_right && m_pMousePt.y>=stage1_top && m_pMousePt.y<=stage1_btm)
    {
        if(pMsg->message==WM_KEYDOWN)
        {
            switch(pMsg->wParam)
            {
            case VK_UP:
                stage1_view_offset_y -= 5;
                return TRUE;
            case VK_DOWN:
                stage1_view_offset_y += 5;
                return TRUE;
            case VK_RIGHT:
                stage1_view_offset_x += 5;
                return TRUE;
            case VK_LEFT:
                stage1_view_offset_x -= 5;
                return TRUE;
            }
        }
    }
    Invalidate(FALSE);
    return CDialog::PreTranslateMessage(pMsg);
}
```

이제 마우스 휠을 돌리면 줌도 되고 키보드 방향키를 눌르면 움직이기도 한다 !

참 쉽다. MFC.

'Enginius > C / C++' 카테고리의 다른 글

MFC – 에디트박스를 써보자	2009/11/23
MFC – 창을 하나 더 띄워볼까? 모달리스	2009/11/23
MFC – Slider Control을 써보자	2009/11/06
MFC – 포커스를 옮겨보자	2009/11/05
MFC – 그림을 그려보자.	2009/11/04
C++에서 클래스 상호 참조 문제 해결	2009/11/02

Filed under : Enginius/C / C++
Tag : DC,mfc,그림 그리기,다이얼로그,더블 버퍼링,시뮬레이션