1. 그리기

시작. 벡터 가 아닌 리스트로 구성

그림을 그릴 때 shape클래스에 만들어 각각rect, circle, curve, star의 도형을 넣어줄 때 shape 리스트에 추가해 주는 방식으로 진행하고자 했다. 하지만 이때 vector로 만드는 것에 익숙해져있었지만 후에 지우고 순서를 바꾸는 등의 작업을 하기 위해서는 vector가 아닌 list를 사용하는 방식을 취해주었다. 나는 MFC자체에서 지원해주는 CList클래스를 이용하여 만들기 시작했다.

가장 먼저 생각한 것은 각 도형들이 동시에 받을 수 있는 멤버 변수가 무엇인지, 멤버 함수가 무엇인 지었다. 이에 나온 답은 처음 찍은 점과 색 그리고 그리는 멤버함수와 점을 찍는 함수들이었다. 후에 추가되는 선택등의 함수또한 추가해주었다.

사각형

사각형은 처음 도형시작점, 도형 끝점을 받아 그려주었다.

원

원은 처음 도형 시작점에서 마우스 포인터 방향까지의 반지름(x축의 차이의 제곱, y축의 차이의 제곱의 합의 루트)로 계산하여 각각 더해주었다.

커브

커브에서 처음 고비가 있었는데 처음 커브를 만들 때 클래스로 만들어 주지 않고 단순하게 moveto, lineto로 하여 커브가 연결되서 출력되는 오류가 있었다. 이런 오류를 해결하기 위해 처음에는 리스트안에 리스트를 넣어 만들어보고자 하였지만 실패했고, 두번째는 클래스를 제작하여 만드는 것이었다. 이후 새로 만들어 shape클래스를 제작, 이후 커브를 추가하여 포인터를 준 이후 클래스 안에 점들의 벡터를 만들어 주어 그림 그릴 때 반복문으로 출력해주었다.

원래는 커브를 여기서 끝낼 생각이었지만, 그림을 선택한 후 움직여줄 때 커브는 shape의 멤버변수 m_pt에 구애받지 않는 녀석이다보니 움직일 수 없는 문제가 생겼다. 이에 m_pt를 기준으로 각 점들의 차이를 그려주는 방식으로 변경하게 되었다.

별

별을 그리기 위해서는 먼저 원을 그리는 방식으로 반지름을 구해주었다. 이후 각도를 재기 위하여 360도의 호도법인 2pi에서 각 점인 10으로 나누어 나눈 값들 중 2의 배수의 녀석들은 반지름 그대로 가져가도록 하였고, 2의 배수가 아닌 녀석들은 반지름에서 1/3의 값을 가져가도록 하였다. 이후 polygon함수를 이용하여 별 내부를 채워주려고 하였다.

그런데 단순하게 각 점을 연결하여 주면 될 줄 알았지만 이는 실패하였고 각 점들의 배열을 만들어 폴리곤 함수를 써야한 다는 사실을 뒤늦게 깨닳았다 그래서 코드들을 처음부터 다시 쓰기 시작했는데, 간단하게 배열로 주가하여 그려주었다.

하지만 이것도 안된 것이, CPoint클래스를 이용하여 초기화 해 주는데에 문제가 있었고 이를 고치기 위해 x축y축을 따로따로 초기화해주었다.

그림 순서 유지

이것을 하기 위하여 위에서 shape 부모클래스로 준 것인데, 그린 순서대로 AddTail함수를 이용하여 뒤에 연결해 주면서 출력할때는 HeadPosition부터 Tail순서대로 출력되므로 자연스럽게 그린 순서대로 나중에 그린 그림이 가장 앞에 나오는 순서를 유지하게 될 수 있었다.

더블버퍼링

그림이 깜빡거리게 하지 않기 위해서는 앞에서 그려주는게 아닌 뒤에서 그린후 붙여주는 더블버퍼링을 이용해야 하는데, 더블버퍼링을 하기 위해서 CDC클래스의 변수 memDC를 만들어 주어 dc와 연결해 준후 전체 창의 크기를 구하여 먼저 memDC에 저장해준 후 이 memDC를 dc에 그대로 붙여주었다.(bitblt을 이용)

2.선택하기

시작. shape클래스에 각 함수 추가

선택을 하기 위해서 각 도형에는 필요한 것이 있었는데 마우스 포인터가 해당 도형의 내부에 있는가 판단하는 것이었다. 이를 위해서 shape클래스에 도형 내부에 있는지 확인하는 가상함수를 제작하였다. 아래의 작업들은 대부분 어떻게 안에 있는

가 파악하는 내용이 담겨있다.

이 도형을 선택하는 함수는 bool값을 반환하는데 만약 도형 안에 있다면 true를 반환하여 해당 도형을 선택해주도록 할 것이다.

그리고 여기서 더 중요한 점은 도형을 여러 개 선택할 가능성에 대비하여 선택된 도형들만 모아두는 리스트를 만들어주는 것이다. 해당 리스트는 여러 개를 선택한 후 bring front를 할때도 유용하게 쓰였다.

Shape를 이용하여 각 도형들을 상속해준 것은 선택시 하나의 리스트에 저장할 수 있는 장점이 있다.

사각형 선택

사각형 내부를 파악하는 일이 가장 쉬웠는데 그 방법은 사각형의 왼쪽위점의 x축, y축을 본 후 마우스 포인터의 위치와 비교하여 bool값으로 리턴해주면 되기 때문이었다.

원 선택

원을 선택하는 것은 사각형과는 많이 달랐다. 예제 문제를 보면 정확하게 원을 누르지 않으면 선택이 작동하지 않는 것을 볼 수 있었는데, 이를 위해서는 x축 비교y축비교가 아닌 다른 방식을 써야 하는 것이었다. 이를 위해서 나는 고민하다 m_pt를 중점으로 하면서 마우스 포인터 까지의 반지름을 계산해준후 원래 반지름과 비교해보는 방식을 썼다.

커브 선택

커브를 할 때 생각한 것은 커브의 점과 점사이를 연결하는 선을 sin과 cos을 이용하여 구해주는 방식을 하고 싶었다. 그러나 구현을 실패한 후 커브를 연결하는 점주변에 범위를 주어 그 범위에 해당되면 선택되도록 해주었다.

별 선택

별의 경우는 다각형 내부에 있는가를 확인하는 알고리즘을 이용하면 되는데, 해당 점의 x축 방향을 연결하여 확인한다. 이후 해당 선을 지나가는 선의 개수를 확인하여 2개이상의 짝수개일 경우 내부에 있다는 뜻이며, 홀수개일 경우 외부에 있다는 뜻이다. 이 알고리즘을 이용하여 계산해주었다.

선택 표시

각 선택 표시를 그려주기 위해서 먼저 shape의 클래스의 멤버변수로 사각형을 그리기 위한 최대점, 최소점을 만들어주었다. 이후 최대점 최소점에서 +a를 한 값을 넣어주어 잘 보이도록 해준 후 선택이라는 것을 알수 있도록 pen클래스를 이용하여 PS_DOT, 1, RGB(255,0,0)을 해주었다. 해당 최소점을 계산하는 방식은 각 도형마다 다른데

사각형 선택표시

사각형의 경우는 원래 도형의 모양과 같으므로 해당 도형의 점과 너비 높이값을 이용하여 삼항연산자를 이용하여 비교하여 최대최소값을 멤버변수에 넣어주었다

원 선택표시

원 의 경우는 점에서 반지름을 더한 값이 최댓점이고 반지름을 뺀 값이 최솟점이었다.

커브 선택표시

커브의 경우 각 점들이 그려질 때마다 점들의 위치를 파악 그후 점들의 위치가 최댓점보다 커지면 갱신 최솟점보다 작아지면 갱신하는 식으로 최댓값 최솟값을 계산했다.

별 선택표시

별은 원의 반지름으로 하고자 했지만 사실상 튀어나오는 부분이 생기기 때문에 점의 배열을 이용하여 최대 최솟값을 갱신해주는 방향으로 진행하였다.

여러 개 선택

쉬프트 왼쪽클릭을 하려면 먼저 쉬프트를 눌러야만 작동할 수 있는 조건문을 추가해준 후 이것이 작동 될 경우 선택할 때 만들어둔 select 리스트에 저장해준다. 하지만 이런것들로는 구현하기 힘들기 때문에 shape내부에 멤버변수 bool타입의 select변수를 만들어 선택되면 true로 변경해주는 식으로 바꿔주었다 그러나 밖을 선택하면 이것이 false로 변경되어야 하는데 이것을 구할 때는 nowSelect 변수를 chileview에 만들어주어 반복문 내부에서 선택이 된적있으면 ture가 되고 없으면 false가 되도록 하는 방식을 이용했다.

여기서 또 생각했던 것은 여러 개를 선택한 후 다시 해당 도형을 선택할 경우 도형이 한 리스트에 여러 번 쌓이는 오류가 있었는데 이것은 마찬가지로 shape함수에 멤버변수 bool타입 select를 추가해 준 것으로 고칠 수 있었다.

드래그로 여러 개 선택

드래그로 여러 개 선택하는 것은 아까 전에 구해뒀던 최댓값 최솟값을 이용하여 구현했는데, 만약 드래그 한 사각형이 while반복문 내부에 돌아간 도형의 최대 최솟값보다 클 경우 사각형안에 있다고 판단, m_select 리스트에 추가해주었다.

문제에서 나와있던 선택된 도형의 개수 또한 m_select의 리스트의 크기를 보고 판단해줄 수 있었다.

3. **이동 및 삭제**

이동

도형을 이동할때 생각한 것은 마우스가 도형위에 있는가? 마우스의 시작점은 어디인가? 마우스가 시작점으로부터 얼마나 움직였는가 이다. 이것을 구하기 위해 childview에 변수 move1,2를 만들어주었는데 move1은 시작한 위치 move2는 움직인 위치이다. 그러나 move를 하는동안 끊임없이 초기화 해주어야 하는데 처음에 이것을 = 으로 계산해주었다.(도형위치=move1-move2)이런식으로 계산하였더니 당연하게도 틀렸다. 이에 더해주는 방식으로 했더니 (도형위치 += move1-move2) 도형이 발산했다. 이것을 어떻게 해결할까 했는데

Move1의 위치를 끊임없이 갱신해주는 방식을 이용하였다. 이후 더해주는 방식에 갱신해주었더니 잘 움직였다.

하지만 문제는 선택 사각형이었는데

선택사각형 움직임

선택사각형은 따로 움직임을 구현하지 않았기 때문에 움직이지 않고 그자리에 있었다. 심지어는 사각형은 움직인 것 처럼 보였는데 최대 최솟값은 그자리 그대로있어서 움직이려면 다시 그 부분을 드래깅해야했다. 이에 어떻게 해야할까 생각했는데 다시 초기화해주는 함수를 만들어주는 것이다.

해당 함수는 움직임 범위만큼 m_pt에 더해준 후 다시 최대최소 멤버변수를 초기화 해주는 방식으로 진행되었다.

곡선 선택 움직임

그러나 이런 방식에서 곡선의 움직임은 많이 불편했다. 계속하여 곡선의 점의 모임을 움직이려다 보니 반복문을 너무 많이 쓰게 되었다. 그러나 이러한 방식은 m_pt에 연결된 곡선으로 변경하면서 m_pt의 값만 변경해주면 나머지가 다 따라오기 때문에 해결할 수 있었다.

삭제

이를 위해서 벡터가 아닌 리스트로 설정해 둔 것인데 리스트를 이용하여 멤버변수 select가 true인 녀석들을 모두 찾아 해당 녀석들은 removeAt(position)함수를 이용하여 지워주었다. 그냥 지워줄경우 해당 리스트가 연결하던 것이 사라지기 때문에 리스트는 당황하게 된다. 그렇기 때문에 POSITON변수를 하나 더 만들어준 후 원래 POSITION변수는 GetNext를 이용해 다음으로 간다 이후 새로 만든 포지션 변수를 없애주어 연결도 원활하도록 만들어주면 된다.

Align

Bring Back

이것을 만들기 위해 리스트로 했는데 선택한 도형을 꺼내오는 일이다. 이것을 하기 위해서 위의 삭제 방식을 이용하였는데 현재 것을 삭제해주기 전 MyShape 변수에 POSITION의 값을 저장해 두었는데 삭제한 후 그 값을 AddHead로 다시 넣어주는 것이다.

bring Front

똑같이 만들었다 그러나 나는 오류가 나는 것에서 당황했다 오류는 두개이상을 선택한후 front를 하면 오류가 발생하며 튕기는 것이다 그 오류는 내가 리스트가 끊겼을때와 유사하게 생겼다. 나는 이것이 끊겼다고 생각하고, null포인터에서 끝나는 것이 아니라 마지막길이를 계산하고 끝내도록했다 그러나 이것은 뭔가 이상했다. 왜 이런일이 생기는지 모른 상태로 끝나는 것 같았다.

이에 차분히 생각해보니 여러 개를 하면 head에서 tail 로 순서대로 가다가 tail로 밀어준 것이 다시 이것또한 select되어있기 때문에 다시 뒤로 가고 이것이 반복하게 된다.(사실상 끊어짐)

이것을 방지하기 위해 bring Front를 만들어줄때는 반복문을 tail에서 앞으로 오도록 gettailposition함수와 getprev함수를 이용하여 코딩해주었다.

4.UI

메뉴바 툴바

UI는 단순히 menu와 툴바 설정 탭에서 조정해주면 되어 조정해주었다. 그러나 이것이 현재 선택되어있다는 것을 알려야하므로 Updateui를 넣어주었다. 이후 툴바에 메뉴바를 연결해주었따.

컨텍스트메뉴

컨텍스트 메뉴 추가하기 위해 OnContextMenu를 이용하여 추가해주었고 순서를 바꾼만큼 배열의 3번을 넣어 주었다.