설명서

1. 작업 내용

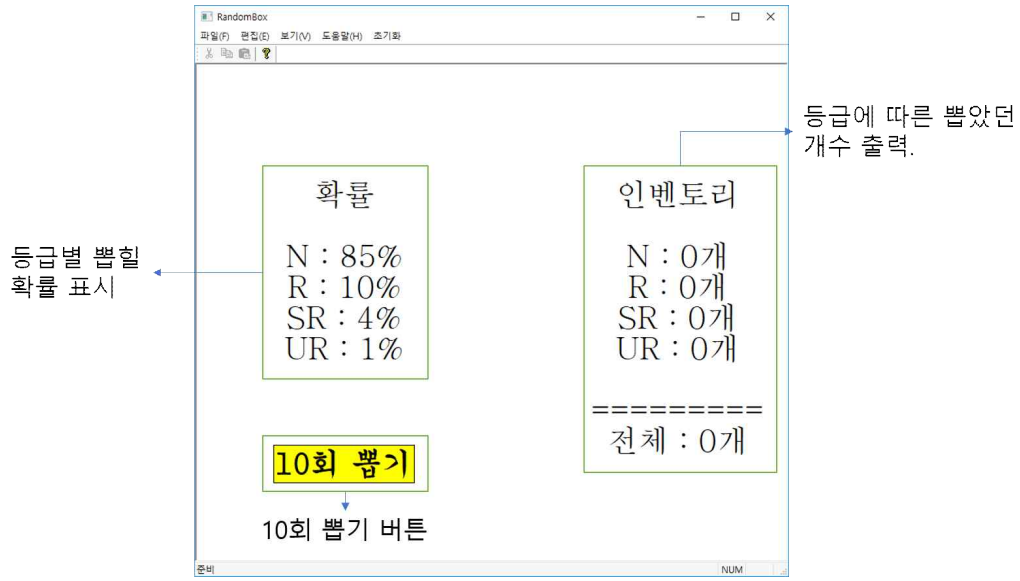


그림 1 메인 화면

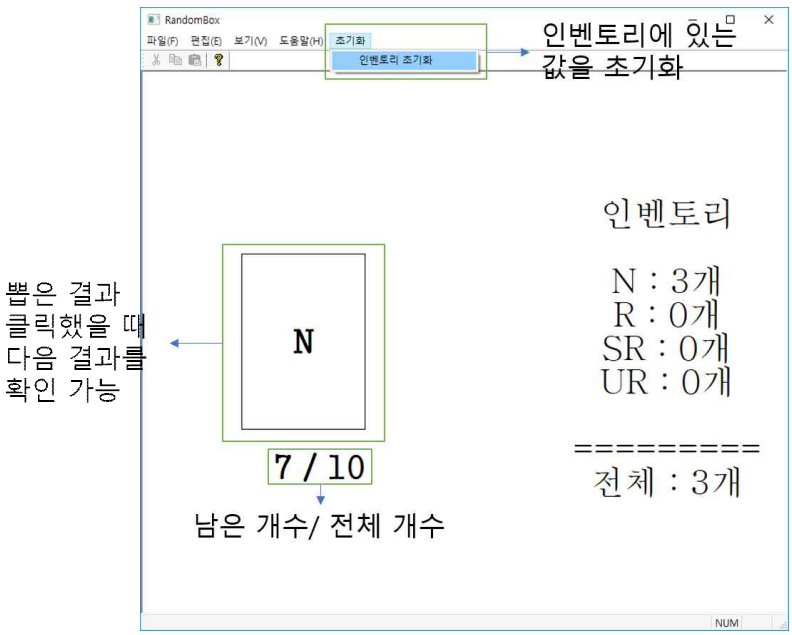
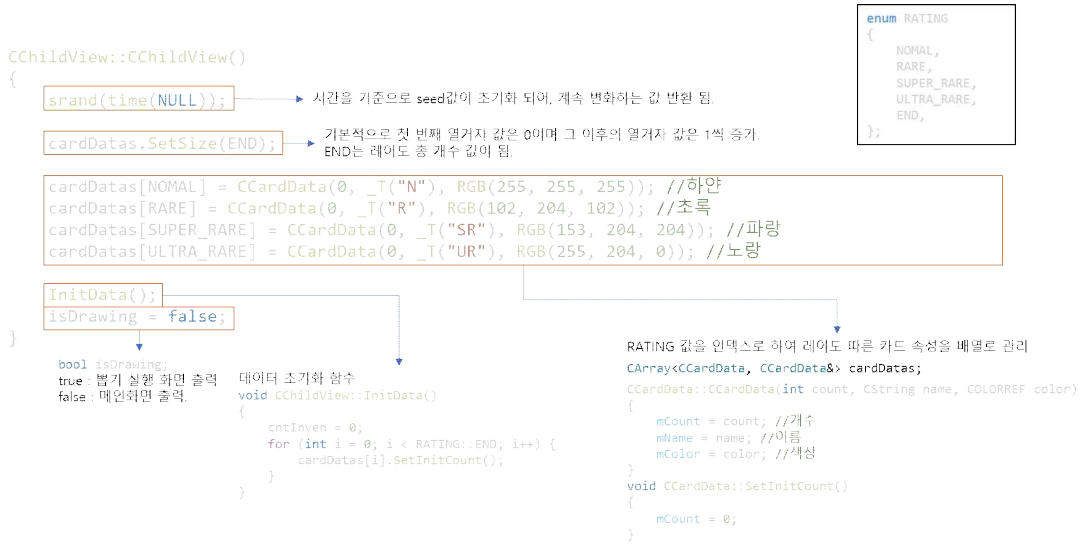


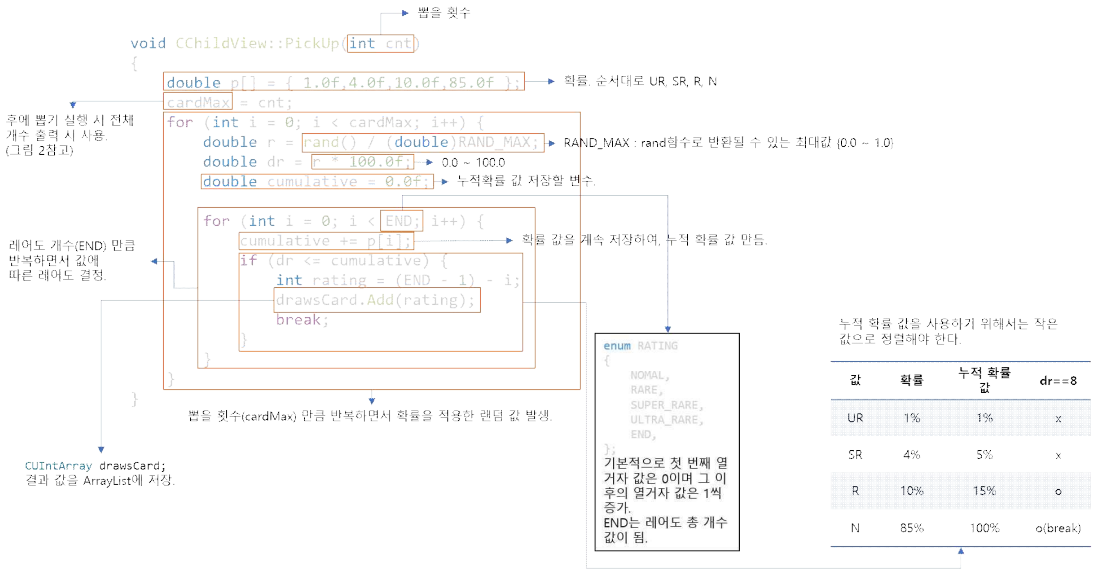
그림 2 뽑기 실행 시

2. 주요 코드

1. ChileView 생성자.



2. 확률을 적용한 랜덤 값. (뽑기 버튼 눌렸을 시 동작)



값	확률	누적 확률 값	dr==8
UR	1%	1%	x
SR	4%	5%	x
R	10%	15%	o
N	85%	100%	o(break)

3. 레어도에 따른 카드 결과 값 출력.

```
void CChildView::DrawCardInfo(CPaintDC * dc)
{
    cardRect.left = (windowRect.right / 4) - 74;
    cardRect.top = (windowRect.bottom / 2) - 105;
    cardRect.right = (windowRect.right / 4) + 74;
    cardRect.bottom = (windowRect.bottom / 2) + 105;

    //카드 그리기
    CBrush btnBrush(cardDatas[drawsCard.GetAt(0)].GetColor());
    dc->SelectObject(&btnBrush);
    dc->Rectangle(&cardRect);

    //폰트 작업.
    CFont font;
    //폰트 설정
    font.CreatePointFont(300, _T("궁서"));
    dc->SelectObject(&font);
    //폰트 색상 및 배경 설정.
    dc->SetBkColor(cardDatas[drawsCard.GetAt(0)].GetColor());
    //글자 출력
    dc->DrawText(cardDatas[drawsCard.GetAt(0)].GetName(), &cardRect, DT_CENTER | DT_VCENTER | DT_SINGLELINE);
    //남은 개수/전체 개수 출력
    dc->SetBkColor(RGB(255,255,255));
    CString count;
    count.Format(_T("%d / %d"), drawsCard.GetCount(), cardMax);

    dc->SetTextAlign(TA_RIGHT); //오른쪽 정렬
    dc->TextOutW(cardRect.right, cardRect.bottom+25, count);
}
```

카드 사이즈 설정

뽑기에서 첫번째로 저장한 값의 색상 값을 가져온다.

```
COLORREF CCardData::GetColor()
{
    return mColor;
}

CString CCardData::GetName()
{
    return mName;
}
```

뽑기에서 첫번째로 저장한 값의 색상 값을 가져온다.

뽑기에서 첫번째로 저장한 값의 이름을 가져온다.

전체 개수 : 확률에 적용된 랜덤 값 코드 참고

남은 개수 : 카드 클릭 시 코드 참고

4. 카드 클릭 시

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
    //생략
    else if (isDrawing && PtInRect(&cardRect, point)) {

        if (drawsCard.GetSize() > 0) {
            int rating = drawsCard.GetAt(0);
            cardDatas[rating].AddCount();
            drawsCard.RemoveAt(0);
            cntInven += 1;
        }

        if (drawsCard.GetSize() <= 0)
            isDrawing = false;
    }

    Invalidate();
}
```

bool isDrawing;
true : 뽑기 실행 화면 출력
false : 메인화면 출력.
PickUp 함수 호출 후 true로 설정.

BOOL RtInRect(const RECT * lprc, POINT pt);
영역 내에 점이 위치해 있다면 TRUE, 아니라면 FALSE 반환.

첫번째에 있는 결과값(레어도)을 저장.

인벤토리 전체 값 증가

첫번째 값 삭제. 위에 사용했기 때문에

해당 레어도 개수를 더한다.

```
void CCardData::AddCount()
{
    mCount += 1;
}
```

DrawCardInfo 함수에서 리스트에 값이 없는데 호출하는 것을 방지하기 위해.
Ex : drawsCard.GetAt(0)