객체지향 별 움직이기

BaseObject.h

```
#ifndef __BASE_OBJECT__
#define __BASE_OBJECT__

class CBaseObject
{
public:
	CBaseObject() : _X(0)
	{
	}
	virtual bool Update(void) = 0;
	virtual void Render(void) = 0;
protected:
	int	_X;
};
#endif
```

OneStar.h

```
#ifndef __ONE_STAR__
#define __ONE_STAR__

class COneStar : public CBaseObject
{
public:
	COneStar();
	virtual bool Update(void);
	virtual void Render(void);
};
#endif
```

OneStar.cpp

```
#include <iostream>
#include "BaseObject.h"
#include "OneStar.h"

COneStar::COneStar()
{
}
bool COneStar::Update(void)
{
	_X++;
	if ( _X > 79 )
	{
		return false;
	}
	return true;
}
void COneStar::Render(void)
{
	for ( int iCnt = 0; iCnt < _X; iCnt++ )
	{
		std::cout << " ";
	}
	std::cout << "*";
}
```

TwoStar.h

```
#ifndef __TWO_STAR__
#define __TWO_STAR__

class CTwoStar : public CBaseObject
{
public:
	CTwoStar();
	virtual bool Update(void);
	virtual void Render(void);

};

#endif
```

TwoStar.cpp

```
#include <iostream>
#include "BaseObject.h"
#include "TwoStar.h"

CTwoStar::CTwoStar()
{
}

bool CTwoStar::Update(void)
{
	_X += 2;
	if ( _X > 79 )
		return false;
	return true;
}

void CTwoStar::Render(void)
{
	for ( int iCnt = 0; iCnt < _X; iCnt++ )
	{
		std::cout << " ";
	}
	std::cout << "**";
}
```

ThreeStar.h

```
#ifndef __THREE_STAR__
#define __THREE_STAR__

class CThreeStar : public CBaseObject
{
public:

	CThreeStar();
	virtual bool Update(void);
	virtual void Render(void);

};
#endif
```

ThreeStar.cpp

```
#include <iostream>
#include "BaseObject.h"
#include "ThreeStar.h"

CThreeStar::CThreeStar()
{
}

bool CThreeStar::Update(void)
{
	_X += 3;
	if ( _X > 79 )
		return false;
	return true;
}
void CThreeStar::Render(void)
{
	for ( int iCnt = 0; iCnt < _X; iCnt++ )
	{
		std::cout << " ";
	}
	std::cout << "***";
}
```

Main.cpp

```
#include <windows.h>
#include <conio.h>
#include <iostream>
#include "BaseObject.h"
#include "OneStar.h"
#include "TwoStar.h"
#include "ThreeStar.h"

CBaseObject *g_ObjectArray[20] = {0,};


void Action()
{
	for ( int iCnt = 0; iCnt < 20; iCnt++ )
	{
		if ( g_ObjectArray[iCnt] != NULL )
		{
			if ( !g_ObjectArray[iCnt]->Update() )
			{
				delete g_ObjectArray[iCnt];
				g_ObjectArray[iCnt] = NULL;
			}
		}
	}
}

void Draw()
{
	for ( int iCnt = 0; iCnt < 20; iCnt++ )
	{
		if ( g_ObjectArray[iCnt] != NULL )
			g_ObjectArray[iCnt]->Render();

		std::cout << std::endl;
	}
}

void KeyProcess()
{
	CBaseObject *pObject = NULL;

	if ( _kbhit() )
	{
		switch ( _getch() )
		{
		case '1':
			pObject = new COneStar;
			break;
		case '2':
			pObject = new CTwoStar;
			break;
		case '3':
			pObject = new CThreeStar;
			break;
		}
```

```
                if ( pObject != NULL )
                {
                        for ( int iCnt = 0; iCnt < 20; iCnt++ )
                        {
                                if ( g_ObjectArray[iCnt] == NULL )
                                {
                                        g_ObjectArray[iCnt] = pObject;
                                        return;
                                }
                        }
                        delete pObject;
                }
        }
}


int main(void)
{
        while ( 1 )
        {
                KeyProcess();
                Action();

                system("cls");
                Draw();
                Sleep(100);
        }
        return 0;
}
```