C++ 프로그래밍 설계 과제 #2: 다형성에 기반한 도형 드로잉 툴 만들기

국민대학교 컴퓨터공학부
2013년 1학기

과제 설명:

본 과제에서는 설계과제 #1의 내용을 확장하여, 객체지향 설계(Object-oriented design)를 통해 다양한 도형을 포스트스크립트 (PostScript) 파일로 그릴 수 있는 도형 드로잉 툴을 만든다. 특히, 본 과제에서는 다형성(polymorphism)에 기반한 객체지향 설계를 경험해 봄으로써 다형성을 활용한 프로그램의 구조가 매우 효과적임을 체험하도록 한다.

과제 목표:

다형성에 기반한 도형 드로잉 툴을 만들기 위해 다음의 세부 목표를 모두 만족하는 프로그램을 디자인하도록 한다.

- Shape 클래스를 만들고, 각각의 도형 클래스(Circle, Rectangle, Triangle)는 Shape 클래스로부터 상속받도록 한다.
- Shape 클래스 및 각각의 도형 클래스에 draw(…)라는 함수를 정의하고, draw(…) 함수를 다형성을 통해 구현한다.
- Shape 클래스는 순수 추상 클래스(pure abstract class)로 구현한다.
- 각 도형 클래스의 인스턴스(instance)는 모두 동적할당(dynamic allocation)을 통해 생성하도록 한다.
- 표준 템플릿 라이브러리(Standard Template Library, STL)에서 제공하는 링크트 리스트인 std::list를 활용하여, 동적할당된 도형의 인스턴스를 Shape *를 통해 관리하도록 한다.
- 이미 만들어진 도형 중 하나를 선택(selection)할 수 있는 기능을 구현한다.
  - 도형의 선택 기능을 구현하기 위해, 각 도형은 만들어지는 순서대로 ID 값을 부여 받는다.
  - ID 값은 0 이상의 정수값이며, 도형들은 서로 다른 ID값을 가진다.
  - ID 값을 통해 만들어진 도형 중 하나를 선택할 수 있다.
- 선택한 도형을 삭제(deletion)할 수 있는 기능을 구현한다.

과제에서 작성할 프로그램은 [Table 1]과 같은 대화식 입력을 통해, 사용자가 그리기 원하는 여러 개 도형들의 정보를 입력 받고, 이를 기반으로 [Table 2]와 같이 포스트스크립트 언어 문법을 따르는 텍스트 파일을 std::ofstream을 이용하여 현재 디렉토리에 생성하도록 한다.

Table 1: 설계과제 프로그램의 대화식 입력 예 (사용자의 입력은 밑줄 친 볼드체로 표시, 설계과제 #1 대비 추가된 사항은 붉은색으로 강조)

> my_drawtool.exe
도형 드로잉 툴을 구동합니다.
모드를 선택하십시오.
(1) 도형 그리기, (2) 도형 선택, (x) 프로그램 종료: **1**
다음 중 그리고 싶은 도형 하나를 선택하십시오.
(a) 원, (b) 삼각형, (c) 사각형: **a**
원의 중심 좌표 (x, y) 값을 입력하세요: **3.0 5.0**
원의 반지름 R 값을 입력하세요: **2.0**
원의 색깔 (r, g, b) 값을 입력하세요: **0.2 0.6 0.9**
원(도형 ID: 0)이 생성되었습니다.
모드를 선택하십시오.
(1) 도형 그리기, (2) 도형 선택, (x) 프로그램 종료: **1**
다음 중 그리고 싶은 도형 하나를 선택하십시오.
(a) 원, (b) 삼각형, (c) 사각형: **c**
사각형의 좌상단 좌표 (x, y) 값을 입력하세요: **4.0 5.0**
사각형의 우하단 좌표 (x, y) 값을 입력하세요: **7.0 3.0**
사각형의 색깔 (r, g, b) 값을 입력하세요: **0.8 0.2 0.5**
사각형(도형 ID: 1)이 생성되었습니다.
모드를 선택하십시오.
(1) 도형 그리기, (2) 도형 선택, (x) 프로그램 종료: **1**
다음 중 그리고 싶은 도형 하나를 선택하십시오.
(a) 원, (b) 삼각형, (c) 사각형: **b**
삼각형의 세 꼭지점 좌표 (x, y) 값을 반시계 방향 순서로 입력하세요: **2.0 4.0 6.0 4.0 4.0 8.0**
삼각형의 색깔 (r, g, b) 값을 입력하세요: **0.2 0.7 0.3**
삼각형(도형 ID: 2)이 생성되었습니다.
모드를 선택하십시오.
(1) 도형 그리기, (2) 도형 선택, (x) 프로그램 종료: **2**
선택할 도형의 ID를 입력하세요: **1**
선택된 도형에 대해 수행할 작업을 입력하세요.
(d) 삭제: **d**
모드를 선택하십시오.
(1) 도형 그리기, (2) 도형 선택, (x) 프로그램 종료: **x**
지금까지 그린 도형을 저장할 파일 이름을 입력하세요: **my_polygons_02**
현재 디렉토리에 my_polygons.ps 파일이 생성되었습니다.
도형 드로잉 툴을 종료합니다.
>

Table 2: 사용자 입력에 기반한 도형정보가 담긴 포스트스크립트 파일 my_polygons_02.ps (사용자의 입력이 반영된 부분은 밑줄 친 볼드체로 표시함)

```
%!
50 50 scale                          % just scale the coordinate. DO NOT TOUCH THIS LINE.

% draw a circle (ID: 0)
newpath
        3.0 5.0 2.0 0 360 arc        % x y r start_angle end_angle
closepath
0.2 0.6 0.9 setrgbcolor              % r g b
fill                                 % draw a shape by filling with the current color

% draw a triangle (ID: 2)
newpath
        2.0 4.0 moveto
        6.0 4.0 lineto
        4.0 8.0 lineto
closepath
0.2 0.7 0.3 setrgbcolor              % r g b
fill                                 % draw a shape by filling with the current color

showpage
```

[Table 2]와 같이 포스트스크립트 언어 문법을 따르는 텍스트 파일은 GSview 프로그램을 이용하면 [Figure 1]과 같이 그 결과를 가시적으로 확인할 수 있다.

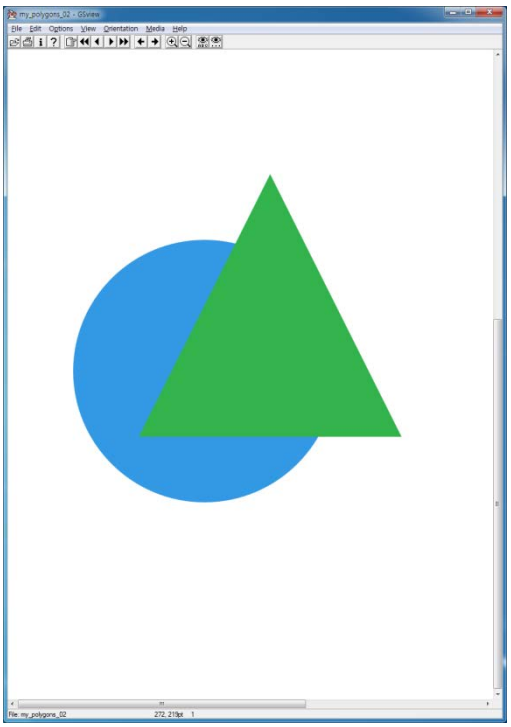

Figure 1: GSview를 활용하여, 설계 과제 프로그램이 생성한 my_polygon_02.ps 파일을 가시화 한 예

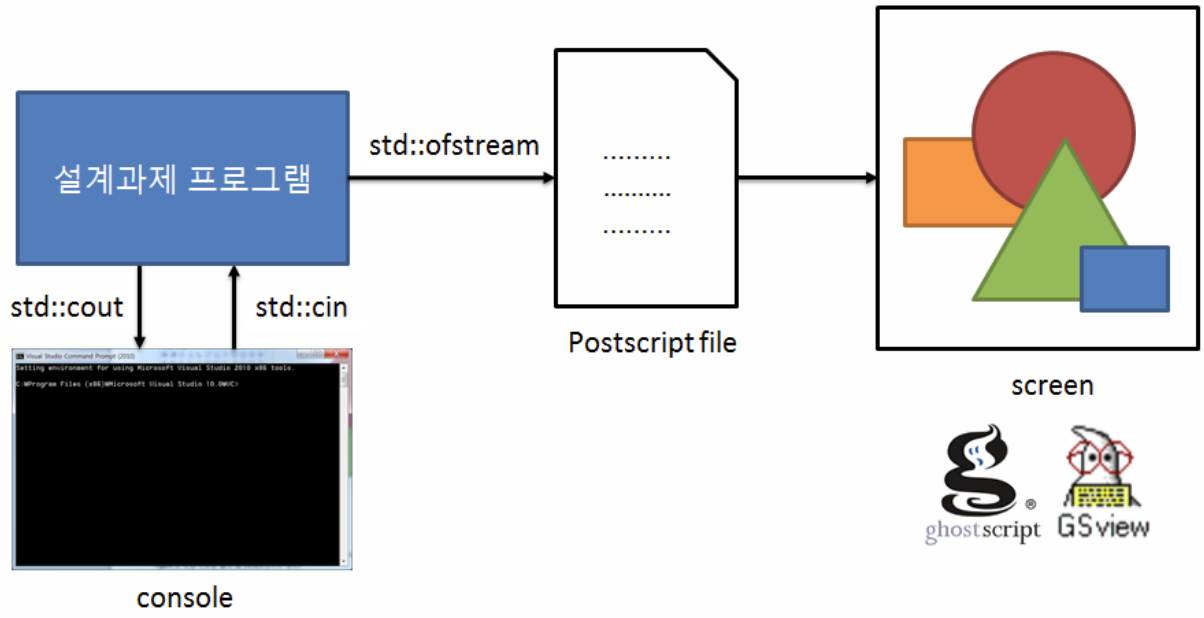

Figure 2: 설계 과제 프로그램의 입출력 형태

과제의 목표를 달성하도록 하는 프로그램의 개략적인 코드 전개는 [Table 3]과 같다. (**주의사항**: [Table 3]은 프로그램의 개략적인 뼈대만을 알려줄 뿐이며, 학생 개개인은 [Table 3]을 참조하여 과제의 요구사항 및 목표를 만족하도록 자신만의 코드를 구현해야 함.)

Table 3: 설계과제 #2의 개략적인 코드 전개

```
#include <iostream>
#include <fstream>
#include <list>

class Shape
{
private:
  int   id_;
public:
  Shape() : id_(-1)  {}
  virtual void draw(std::ofstream& ofs) const
  {}
};

class Circle : public Shape
{
private:
  /// data for specifying a circle
public:
  virtual void draw(std::ofstream& ofs) const
  { ofs << "Circle::draw()" << std::endl; }
};

class Rectangle : public Shape
{
private:
  /// data for specifying a rectangle
public:
  virtual void draw(std::ofstream& ofs) const
  { ofs << "Rectangle::draw()" << std::endl; }
};

class Triangle : public Shape
{
private:
  /// data for specifying a triangle
public:
  virtual void draw(std::ofstream& ofs) const
  { ofs << "Triangle::draw()" << std::endl; }
};

void main()
{
  /// 도형 관리를 위한 링크트 리스트 ls
  std::list<Shape *>           ls;
  std::list<Shape *>::iterator  it;

  /// 링크트 리스트 ls를 활용하여 도형 생성, 선택, 삭제하는 기능 구현
  ls.push_back(new Triangle());
  ls.push_back(new Rectangle());
  ls.push_back(new Circle());
  ls.push_back(new Circle());

  /// 파일에 도형들 그리기
  std::ofstream   outStream("out.txt");
  for (it = ls.begin(); it != ls.end(); ++it)
  {
    const Shape*  s = (*it);
    s->draw(outStream);
  }
  outStream.close();
}
```

참고할 정보:

- 설계과제 #1의 '**참고할 정보**'를 참고하도록 한다.
- STL이 제공하는 링크트 리스트 std::list<…>의 사용법을 더 자세히 알고 싶은 경우, 다음을 참고하도록 한다.
  - http://www.cplusplus.com/reference/list/list/
  - 설계과제 #2에 첨부된 stl_list_example 예제를 참고하도록 한다.

과제 수행 시 주의사항:

본 과제를 성공적으로 수행하기 위해 아래의 요소들을 고려해야 한다 .

- [Table 1]은 [Figure 2]에서 std::cout, std::cin 부분에 해당한다. 채점의 일관성을 위해, [Table 1] 에 나타난 사용자 입력 부분의 형식을 그대로 따른다.
  - 학생의 편의에 의해 사용자 입력 부분을 임의로 수정하는 것은 허용하지 않음
- [Table 2]는 [Figure 2]에서 std::ofstream을 통해 만들어진 PostScript 파일에 해당한다. [Table 2]에 이미 본 과제를 성공적으로 수행하기에 충분한 포스트스크립트 언어 문법이 표현되어 있으므로, [Table 1]에서 나타난 사용자의 입력이 [Table 2]에 어떻게 반영되었는지 비교분석 해 보도록 한다.
- GSview 프로그램을 통해, 생성된 PostScript 파일을 가시적으로 확인해 보도록 한다.
- **마감일은 2013년 5월 24일(금) 자정(23:59)이며, 가상대학을 통해 제출하도록 한다.**