객체지향 프로그래밍 소스 구현 설명

202104361
ICT공학부
천승환

1. 문제 정의 : 그래픽 편집기의 기능인 "삽입", "삭제", "모두보기", "종료"를 출력 부분에 나타내고 추가적으로 각 생성된 도형 객체들을 vector에 삽입하여 관리하는 과정이다.

2. 문제 해결 방법 :
벡터를 사용하여 Shape객체의 관리와 작업을 훨씬 수월하게 할 수 있다.
GraphicEditor 헤더파일에서 vector<Shape*> v; 를 선언함으로써 선택한 도형을 포인터로 저장하고 동적 메모리(new)로 생성된 객체들을 관리하여 그래픽 편집기 기능인 삽입 및 삭제, 모두보기 등을 효율적으로 수행할 수 있게 된다.
GraphicEditor.cpp에서 도형 삽입 작업을 할 때 push_back 메소드를 사용함으로써 동적으로 생성된 도형 객체 포인터를 벡터의 끝에 추가하는 형식으로 코드를 짰다. 이어서 삭제 작업을 할 때 erase 메소드를 사용함으로써 특정 위치를 제거하고 나머지 도형들이 앞쪽으로 이동하여 빈 공간을 채워주는 형식으로 코드를 짰다. 모두보기 작업 같은 경우(show), v.size()를 사용하여 벡터에 저장된 도형의 개수까지 반복문을 사용하여 각 도형을 호출하였다.

3. 아이디어 평가 :
- 벡터를 사용하게 되면 동적으로 생성된 객체들의 크기를 미리 지정할 수 있기 때문에 관리에 용이하다.
- 그래픽 편집기 기능 작업을 수행할 때(삽입, 삭제) 동적 메모리를 직접적으로 관리할 필요가 없다.
- 한 눈에 봐도 알아볼 수 있는 메소드(push_back, erase, size 등)를 사용하게 됨으로써 가독성을 높일 수 있고, 이러한 메소드를 통해 데이터를 효율적으로 관리할 수 있다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명 :
벡터를 Shape* 타입으로 선언을 하게되면, 각 도형 객체들을 하나의 컨테이너처럼 관리를 할 수 있게 된다. 그리고 vector는 동적 배열이기 때문에 자동으로 크기를 확장시킬 수 있어 Line, Circle, Rect 외에도 다른 도형들이 있어도 무관하다. 3.아이디어 평가에서도 설명했지만 벡터 라이브러리는 적절한 그래픽 편집기의 기능(push_back(), erase())을 가지고 있기 때문에 코드가 간결해진다.