CSED232 Object-Oriented Programming (Spring 2018)

Assignment #3

Polymorphism (with QT)

Due date: 5/18 (Fri) 23:59:00

담당 조교: 차호준 (hersammc@postech.ac.kr)

이번 과제에서는 QT환경에서 GUI를 활용해 "얼굴 만들기"를 구현한다. 이번 과제를 통해 Polymorphism의 개념을 활용한 class 디자인을 하고 이를 실제 프로그래밍에 적용해본다.

0. QT 환경 설정

QT 환경 설정은 미리 배포한 QT Install Guide 2018을 확인하세요.

QT Creator 버전: 4.6.0

QT 버전: 5.10.1

1. 개요

이번 과제에서 구현할 프로그램은 얼굴 각 요소의 이미지를 활용하여 얼굴을 자유롭게 구성하는 프로그램이다.

사용자에게 얼굴 요소의 이미지를 합친 화면이 주어지고, 이 화면을 클릭하여 얼굴의 요소(FaceFeature) 중 얼굴 형태(Face), 눈(Eyes), 코(Nose), 입(Mouth)을 각각 선택할 수 있다. 이미지 화면 옆의 선택 툴을 활용하여 Face를 제외한 선택한 요소의 크기를 가로, 세로로 늘려 변형할 수 있다.

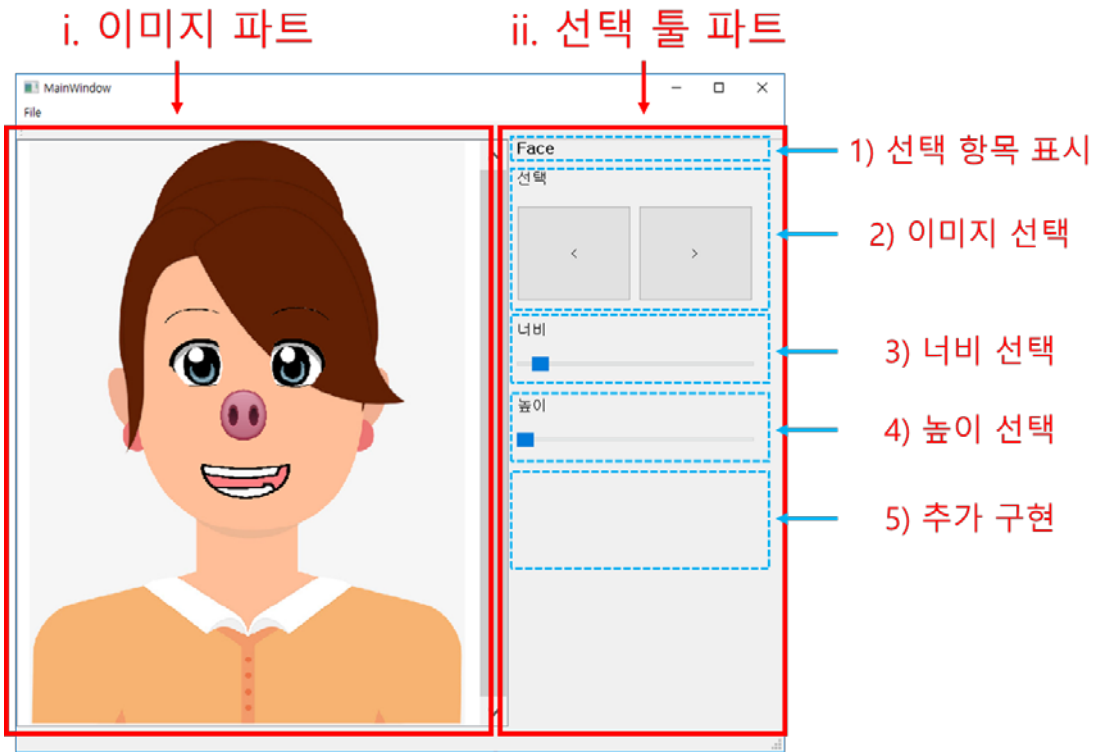

Figure 1. 얼굴 고르기 프로그램 화면 구성

2. 프로그램 디자인

A. 프로그램의 구성 (GUI)

프로그램은 크게 두 파트로 구성된다.

i. 이미지 파트 (QGraphicsView)

이미지 파트에서는 얼굴의 이미지가 드러난다. 이 때 표현된 이미지는 얼굴 형태 위에 눈, 코, 입이 모두 구현된 형태이다.

이 이미지 위의 요소를 클릭하여 얼굴 요소를 선택할 수 있도록 한다. 선택한 요소는 ii. 선택 툴 파트에서 변경 및 변형이 가능하도록 한다.

ii. 선택 툴 파트 (QLabel, QPushButton, QSlider, QTextBrower, QRadioButton 등)

선택 툴 파트에서는 선택한 요소의 이미지를 변경하거나 크기를 조절할 수 있다. 이를 사용자가 클릭하여 변경하면 즉시 이미지 파트에 반영이 된다.

선택한 항목의 경우 반드시 각 얼굴 요소의 클래스의 객체가 아닌 (Face, Eyes, Nose, Mouth 등) 상위 클래스(FaceFeatures)를 사용해야 한다. (자세한 내용은 3. 클래스 디자인과 4. 제약사항에 소개)

1) 선택한 항목의 표시

마우스로 클릭하여 선택한 얼굴 요소가 어떤 항목인지를 표시한다.

- 아무것도 선택하지 않았을 경우 'Select a face component.' 또는 '얼굴 요소를 선택하세요'와 같은 문구로 설정한다.
- 얼굴을 전체를 선택하였을 경우 'Face' 또는 '얼굴', 눈을 선택하였을 경우 'Eyes' 또는 '눈'과 같은 문구로 설정한다.

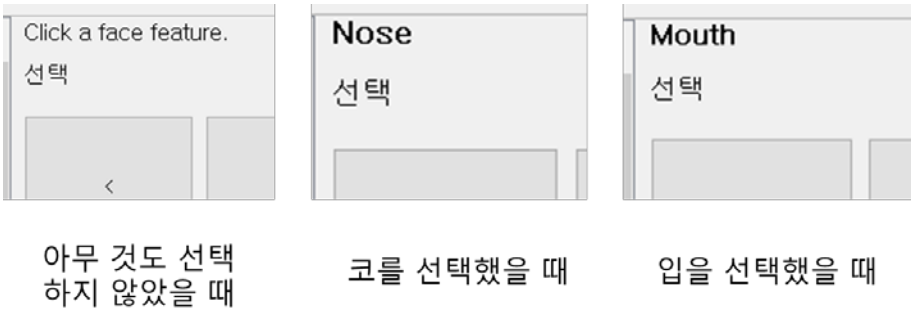

Figure 2. 선택 항목의 표시 예시

2) 이미지 선택

이미지 선택에서는 좌우 버튼 등 다양한 방법으로 여러 이미지 중에 원하는 이미지를 선택하도록 한다.

Figure 3. 이미지 선택 예시

3) 너비 선택

너비 선택에서는 이미지의 너비(좌우)를 조절한다. 슬라이더, 수치 입력 등을 자유롭게 선택하여 넣을 수 있다.

- 얼굴과 같이 크기가 변하기 어려운 요소의 경우에는 이를 변경해도 아무런 변화가 일어나지 않는다.
- 눈, 코, 입의 경우 변화가 있을만한 요소의 경우에는 이를 변경하여 적절한 크기를 만든다.

Figure 4. 너비 선택 예시

4) 높이 선택

높이 선택은 너비 선택과 마찬가지지만 이미지의 높이(상하)를 조절한다. 이 또한 툴을 자유롭게 선택하여 만들 수 있다.

Figure 5. 높이 선택 예시

5) 추가 구현

추가 구현은 1개를 선택하여 구현한다. 다음의 항목 중 하나를 구현할 수 있다.

- 현재 이미지를 저장

- 선택한 요소의 위치(x, y)를 이동
- 눈 이미지를 둘로 잘라 미간의 크기를 변경
- 얼굴 요소의 회전
- 색상 변경
- 새로운 얼굴 요소의 적용 (예: Snow 등의 포토 앱에서의 필터 같은 효과)
- 기타 자신이 생각하기에 얼굴을 구성하기에 필요한 기능 추가

B. 프로그램의 흐름도

프로그램의 전반적인 흐름은 다음과 같다. (추가적인 정보일 뿐, 이대로 따라가지 않아도 감점요인이 되지 않는다.)

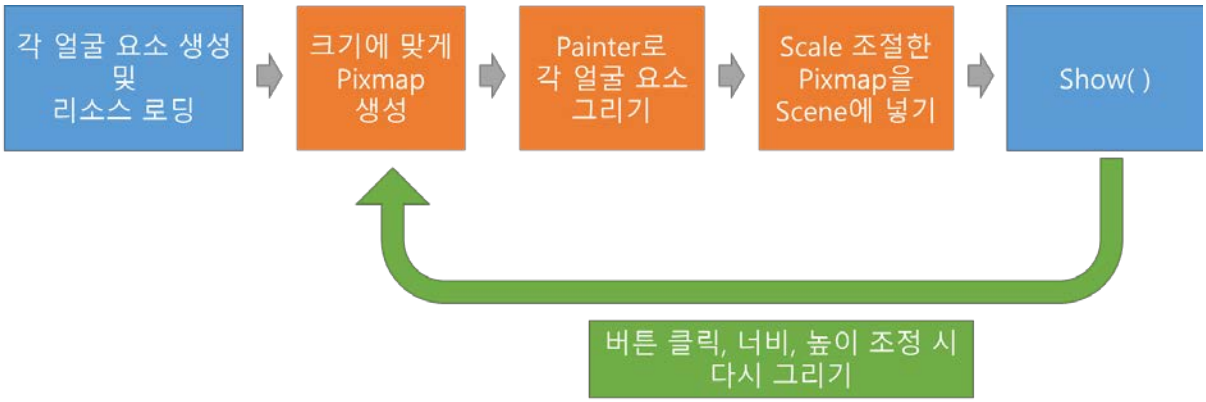

Figure 6. 프로그램 흐름도

여러 그림을 덧입혀서 그릴 때, 위의 Figure 6.과 같은 방법을 사용할 수 있는데 더 좋은 방법이 있다면 사용해도 좋습니다. (보고서 작성시 간략히 설명할 것)

이를 자세히 설명한 것은 아래에 있습니다.

i. 각 얼굴 요소 생성 및 리소스 로딩

얼굴 요소를 디자인에 맞게 생성하고 리소스를 불러오는 초기 과정을 거친다.

다양한 방법이 있지만 QImage 객체를 사용하는 것이 보편적이다.

ii. 크기에 맞게 Pixmap 생성

1) 얼굴 크기에 맞게 QPixmap을 생성한다.

2) 우리의 경우 face를 제외한 얼굴 요소 (eyes, nose, mouth)는 얼굴 위에 그려야 하는

특성상 투명하기 때문에 이를 QPixmap을 투명화 한다.

- Ex) pixmap->fill(Qt::transparent);

iii. Painter로 각 얼굴 요소 그리기

1) 각 얼굴 요소를 QPainter 객체를 사용하여 그려낸다.

2) Ex) painter->drawPixmap(x, y, w, h, QPixmap::fromImage(image));

iv. Scale 조절한 Pixmap을 Scene에 넣기

1) 그려낸 Pixmap을 우리가 넣을 QGraphicsView의 높이 또는 너비에 맞게 크기 배율을 조절한다. (* 추후 마우스 클릭할 때 중요함)

2) 새로운 QGraphicsPixmapItem을 크기를 조절한 Pixmap을 사용하여 만들고, 이를 새로운 QGraphicsScene에 넣는다.

3) 이를 QGraphicsView 객체에 scene으로 등록한다.

참고: https://stackoverflow.com/questions/4375433/how-to-add-an-image-on-the-top-of-another-image

3. 클래스 디자인

A. 각 얼굴 요소의 클래스는 FaceFeatures 라는 클래스의 하위 클래스로 들어간다.

B. 하지만 FaceFeatures은 어떤 객체도 이 클래스를 직접적으로 사용하여 만들면 안되기 때문에 반드시 pure virtual method를 1개 이상 포함하여 사용한다.

C. 좋은 클래스 디자인은 반복적인 함수 사용을 줄이고 구현 시간을 단축시킬 수 있습니다. 어떤 변수와 어떤 함수가 어떻게 사용되어야 할지 잘 고민하면 좋습니다.

D. 아래에 그려진 클래스 구조에 더해서 더 세분화된 구조 또는 새로운 클래스를 만들어도 좋습니다.

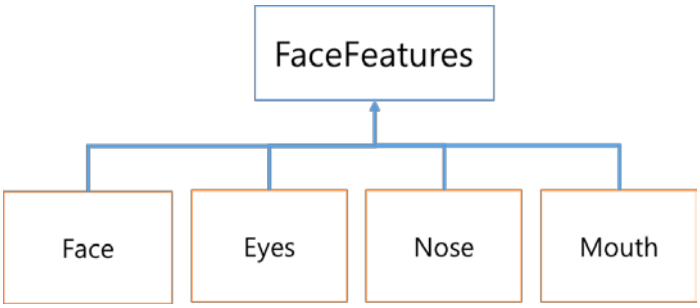

Figure 7. 클래스 디자인 예시

4. 힌트

A. 마우스 클릭 힌트

i. mousePressEvent 발생 시, 현재 마우스 위치를 확인할 수 있는 방법이 여러 가지가 있지만, 하나의 QWidget 안에서 상대적인 마우스 위치를 알아내기 위해서는 다음의 2가지 좌표의 차이로 알아낼 수 있습니다.

1) 현재 전체 화면에서 QWidget의 좌상단이 위치한 곳의 좌표

Ex) QPoint topLeft = [[QWidget]]->mapToGlobal([[QWidget]]->rect().topLeft());

2) 현재 전체 화면에서 마우스가 위치한 곳의 좌표

B. 그림 위치 힌트

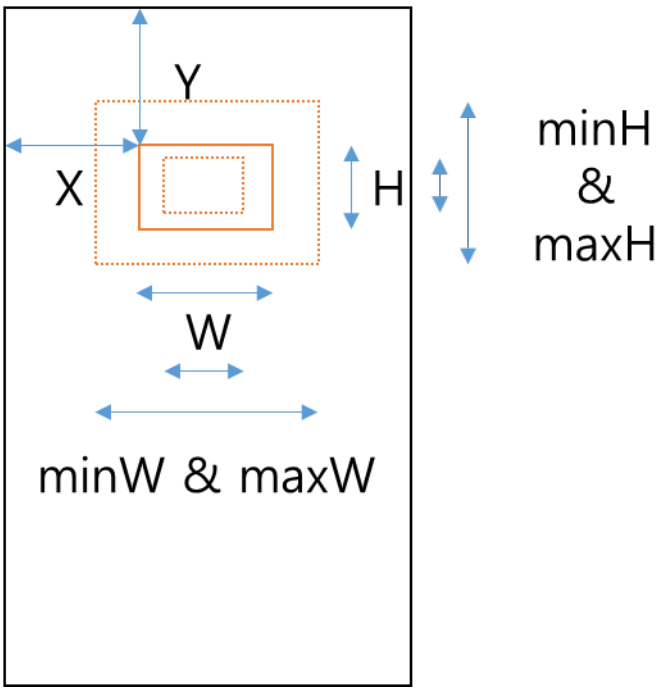

Figure 8. 얼굴 요소 위치 예시 힌트

얼굴 요소를 그려낼 때에는 반드시 중앙에 위치해야 하는데, 이 때 고려해야하는 변수를 위의 Figure 8.과 같이 생각해볼 수 있습니다. 이를 고려하여 클래스 디자인을 하면 문제 해결에 도움이 될 것입니다.

5. 제약사항 및 주의사항

A. 각 얼굴 요소를 사용할 때에는 FaceFeatures 클래스로 upcasting하여 사용한다. (Face, Eyes, Nose, Mouth 클래스의 객체로서 직접 사용 금지)

   이와 같이 작성하지 않을 경우 polymorphism 개념을 적용하지 않은 것으로 보고 최대 총점의 20%까지만 줄 예정입니다.

   i. 각 얼굴 요소를 draw할 때

   ii. 한 얼굴 요소를 선택한 후 다음 사진으로 넘길 때 (해당 항목: 2. A. ii. 2) 이미지 선택)

   iii. 한 얼굴 요소를 선택한 후 크기를 조절할 때 (해당 항목: 2. A. ii. 3), 4) 너비 및 높이 선택)

B. FaceFeatures의 클래스에서의 1개 이상의 method를 pure virtual method로 만들어서 반드시 하위 클래스(derived class)에서 해당 method를 구현하도록 한다.

C. 얼굴 요소는 현재 크기 외에도 크기를 키우거나 줄였을 때에도 반드시 중앙에 위치해야 한다. (힌트 4. B. 참고)

D. 얼굴 요소 그림은 더 추가해도 무방하나, 제출시 추가로 제출해야 한다.

E. 컴파일이 되지 않는 경우 0점 처리된다.

F. 어싸인 중 exception을 잡지 않아서 실행 시 segmentation fault 등의 에러 메시지가 뜨면서 종료되는 경우는 감점 요인이 된다.

   i. 단, 여러 번 draw 하다보면 QImage 또는 QPainter 쪽에서 문제가 발생하여 더 이상 이미지가 렌더링하지 못하고 메시지가 뜨면서 흰 배경이 나올 수 있는데 해당 문제는 감점 요인이 아닙니다.

G. STL은 사용 할 수 없지만, QT Library에서의 유사한 기능의 사용은 가능하다. (예: QList ...)

H. 보고서에 코드 전체를 그대로 복사하지 않는다.

I. 채점 기준은 Assn3ReadMe 파일을 참고한다.