CSED232 Object-Oriented Programming (Fall 2017)

Assignment#3

Polymorphism & Operator Overloading (with QT)

Due date: 12/03(Sun) 23:59:00

담당 조교: 조성원 (kardy04@postech.ac.kr)

이번 과제에서는 "세포 증식 게임"을 구현한다. 이번 과제를 통해 다형성과 연산자 오버로딩의 개념에 대해 공부하고 이를 실제로 프로그래밍에 적용해본다. 또한 이번 과제에서는 Graphic User Interface (GUI) Programming을 위해 QT를 사용한다.

1. QT 환경 설정

- QT 다운로드 링크: https://www.qt.io/download-qt-for-application-development
(ID 생성 후 30일 무료 버전 다운로드 가능, Open source package로도 다운 가능, Fig. 1 참고)

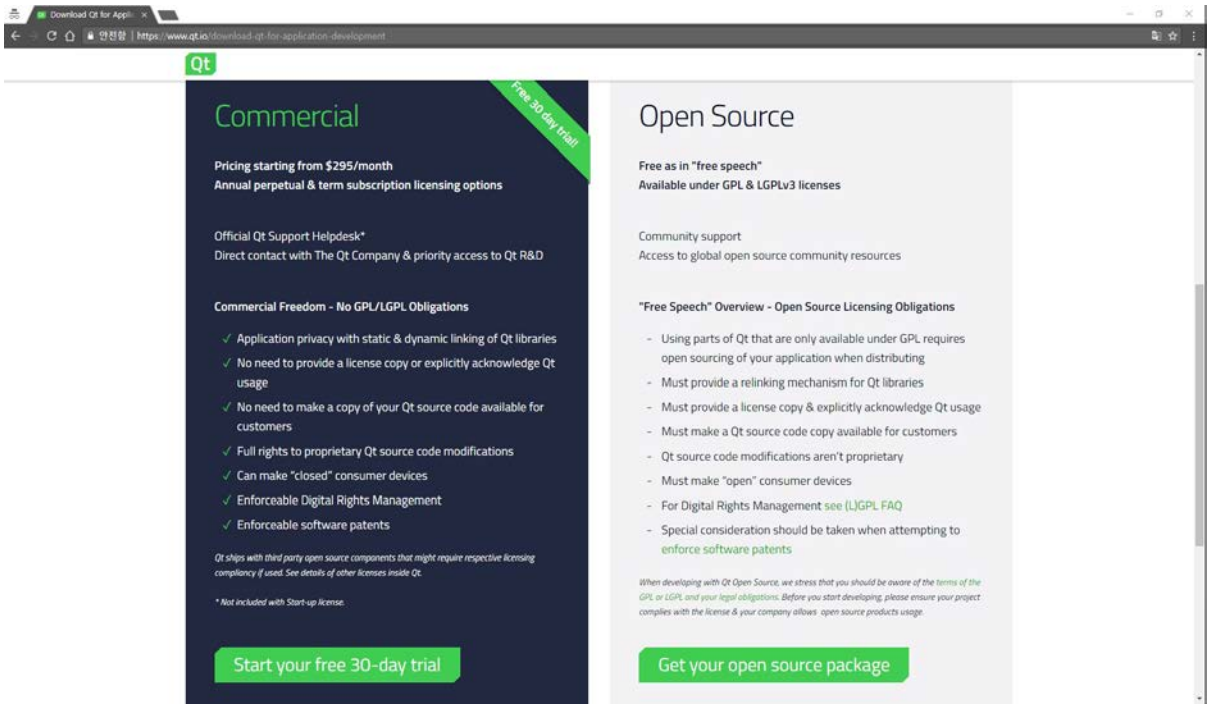

Figure 1. 다운로드 링크로 들어갔을 때의 화면. 무료 30일 commercial 다운 혹은 Open source package 다운로드 가능.

- QT Creator 4.4.1 version, QT 5.9.2 version 사용을 권장하며 자세한 QT 설치 및 환경 설정에 관해서는 추후 추가 문서를 통해 공지할 예정이다. Online installer를 통해 QT 5.9 버전과 QT creator를 설치한다.

- 가능한 한 위에서 명시한 환경에서 과제를 구현할 것을 권장한다. 하지만 부득이한 이유로 인해 (Windows 외 다른 OS 환경, QT Creator가 아닌 다른 프레임워크로 QT를 써온 경험 등) 위에 명시한 것과 다른 환경에서 작업하게 된 경우, 보고서에 본인이 작업한 환경과 환경 설정 과정에 대해 상세히 서술한다. (OS, 사용한 Framework 및 버전, QT version, 환경 변수 설정 등) 이를 명시하지 않은 경우, 불이익이 있을 수 있다.

2. Abstract of Game

A. 이번 과제에서 구현할 게임은 세포를 조작하여 다른 생물체들을 잡아먹으며 세포의 크기를 키워나가는 게임이다.

B. 플레이어 외 다른 생물체에는 적 세포 (Enemy), 바이러스 (Virus), 먹이 (Feed) 가 있다. 각 생물체는 그 종류에 따라 모양, 색, 움직이는 경향성 등이 다르다. 이에 대한 자세한 설명은 3번 Game Design 파트에서 다룬다.

C. 플레이어 세포와 적 세포는 자신보다 크기가 작은 다른 생물체를 잡아먹을 수 있다. 다른 생물체를 잡아먹기 위해선 잡아먹으려는 생물체와 충돌해야 한다. 하지만 자신보다 크기가 큰 생물체와 충돌한 경우, 세포가 사망하게 된다.

D. 다른 생물체를 잡아 먹을 경우, 일반적으로 잡아먹은 생물체의 크기에 비례해 세포의 크기가 증가한다. 하지만 바이러스를 잡아 먹은 경우 바이러스의 크기에 비례해 세포의 크기가 감소한다. 바이러스는 특정한 패턴을 가지고 움직이며, 시간이 지남에 따라 크기가 감소하다 소멸하게 된다.

3. Game Design

A. 플레이어 세포 (Cell)

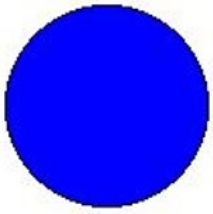

Figure 2. 게임 내 플레이어 세포 예시

- 모양: 원, 색: 파란색, 크기: 원의 지름을 의미 (Fig. 2 참고)

- 따로 조작을 하지 않아도 일정한 방향과 속력으로 이동한다. 모든 생물체 중에 가장 빠른 속력으로 움직인다.

- 방향키를 눌러 세포의 이동 방향을 바꿀 수 있다. (상, 하, 좌, 우)

- 스페이스 바를 눌러 Booster 기능을 켜고 끌 수 있다. Booster 사용 시 이동 속력이 1.5배가 되지만, Booster를 사용하는 동안 0.5초 간격으로 세포의 크기가 현재 크기의 0.95배로 줄어든다.

- 자신보다 크기가 작은 생물체와 충돌 시 그 생물체를 잡아먹는다. 그 후, 잡아먹은 생물체의 크기의 0.8배만큼 자신의 크기가 증가한다. (Fig. 3 참고) (단, 바이러스를 먹은 경우 0.8배만큼 크기가 감소한다.)

- 자신보다 크기가 큰 생물체와 충돌 시 플레이어 세포가 게임에서 사라지고 프로그램이 종료된다.

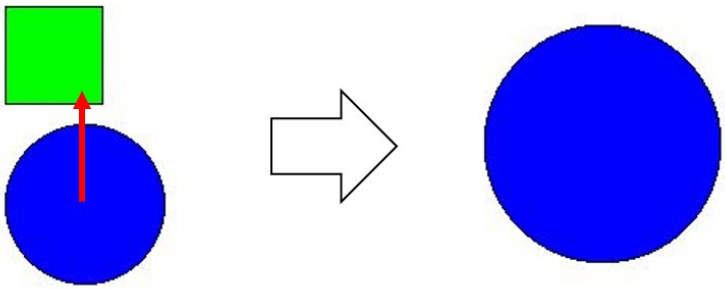

Figure 3. 실제 게임 내에서 플레이어 세포가 먹이를 잡아 먹고 커지는 과정 예시. 왼쪽 그림에서 플레이어 세포가 위로 이동하여 먹이를 잡아 먹었고, 그 후 크기가 커진 것을 확인할 수 있다.

B. 적 세포 (Enemy)

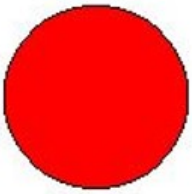

Figure 4. 게임 내 적 세포 예시

- 모양: 원, 색: 빨간색, 크기: 원의 지름을 의미 (Fig. 4 참고)

- 일정한 속력으로 이동한다.

- 3초 간격으로 이동 방향이 상, 하, 좌, 우 중 랜덤하게 변한다.

- 플레이어 세포와 마찬가지로 다른 생물체와 충돌 시 서로의 크기에 따라 잡아 먹거나 잡아 먹힌다. 플레이어 세포와는 다르게 적 세포가 다른 생물체에 잡아 먹혀도 프로그램이 종료되지 않는다.

C. 바이러스 (Virus)

Figure 5. 게임 내 바이러스 예시

- 모양: 정사각형, 색: 검은색, 크기: 변의 길이를 의미 (Fig. 5 참고)

- 1초 간격으로 오른쪽, 위, 왼쪽, 아래 방향의 순으로 일정 간격 이동한다. (Fig. 6 참고)

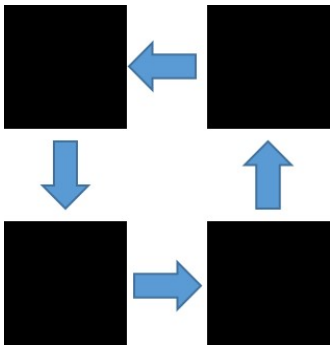

Figure 6. 바이러스의 움직임

- 4초 간격으로 현재 크기의 75%로 크기가 감소한다. (Fig. 7 참고) 4번째로 크기가 감소할 때 바이러스가 소멸한다. (즉, 생성 후 16초 후에 소멸한다.)

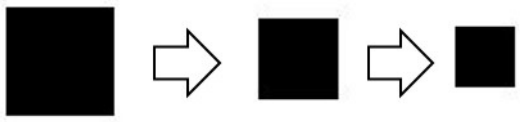

Figure 7. 시간에 따른 바이러스의 크기 감소 예시

- 바이러스는 다른 생물체를 잡아먹을 수 없지만, 바이러스를 잡아 먹은 생물은 바이러스 크기의 0.8배만큼 크기가 감소한다. (Fig. 8 참고)

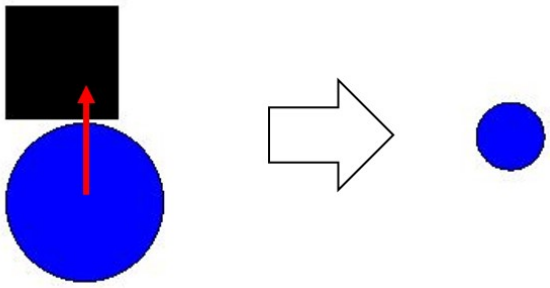

Figure 8. 플레이어 세포가 바이러스를 잡아먹어 크기가 감소하는 과정 예시. 플레이어 세포가 위로 이동하여 바이러스를 잡아먹고 크기가 작아진 것을 확인할 수 있다.

- 바이러스를 잡아 먹어 생물체의 크기가 0 이하가 된 경우, 그 생물체는 죽는다. 만약 그 생물체가 플레이어 세포였을 경우, 프로그램을 종료한다.

D. 먹이 (Feed)

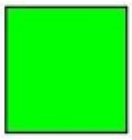

Figure 7 게임 내 먹이 예시 (사이즈가 다른 4개)

- 모양: 정사각형, 색: 초록색, 크기: 변의 길이를 의미 (Fig. 8 참고)

- 다른 생물체들에 비해 가장 크기가 작은 생물체이다.

- 다른 생물을 잡아먹을 수 없고, 움직이지 않는다.

- 먹이는 바이러스와 충돌하여도 아무 일이 발생하지 않는다.

E. Others

- 시작 화면을 만들고, 시작 화면에 시작 버튼과 종료 버튼을 넣어 구성한다. 시작 버튼을 누르면 게임 화면으로 전환되고, 종료 버튼을 누르면 프로그램이 종료된다. (Fig. 9 참고)

- 게임 화면의 크기, 생물체의 초기 크기와 속도의 예시는 다음 페이지의 표와 같다. 이 표의 수치를 정확히 따라할 필요는 없지만 아래 명시된 사항이 지켜져야 한다.

    A. 화면 크기는 한 화면 안에 15개 이상의 생물체가 그려질 수 있도록 설정한다.

    B. 플레이어 세포는 처음 크기를 고정하며, 다른 생물체는 초기 크기의 범위를 정해 두고 생물체를 생성할 때마다 생성자에서 랜덤으로 크기를 설정한다.

    C. 먹이의 크기는 다른 생물체의 크기보다 작아야 한다.

    D. 플레이어 세포의 속도가 적 세포의 속도보다 같거나 빨라야 한다.

- 처음 시작 시 플레이어 세포 외의 30개의 생물체를 화면 내에 랜덤으로 생성하며, 각 생물체는 70% 확률로 먹이, 15% 확률로 적 세포, 15% 확률로 바이러스가 된다.

- 게임 시작 후 15초 간격으로 5개의 생물체를 랜덤으로 추가 생성한다. 단, 생성되는 생물체가 **플레이어와** 겹치지 않도록 한다. 생물체의 종류는 위와 같은 확률을 사용해 결정한다.

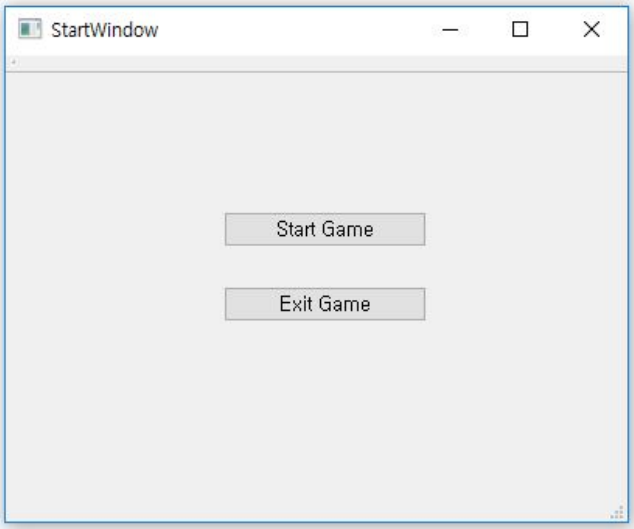

Figure 8. 실행 첫 화면 예제. 시작 버튼과 종료 버튼을 포함해 자유로운 디자인으로 구현.

- 잡아 먹힌 생물체는 게임 화면 내에서 사라져야 한다. Fig. 3의 경우, 플레이어 세포가 먹이를 잡아 먹은 이후 먹이가 사라진 것을 확인할 수 있으며 Fig. 8의 경우, 플레이어가 바이러스를 잡아 먹은 이후 바이러스가 사라진 것을 확인할 수 있다.

- 게임 실행 예제 사진은 Fig. 10과 같다. 이때 사용된 각 생물체의 크기, 속도 등의 조건은 그 아래에 있는 표에서 확인할 수 있다.

종류	크기	속도
Cell	20	50ms 간격으로 2 이동
Enemy	10~40	50ms 간격으로 1 이동
Feed	7~12	X
Virus	15~20	1초 간격으로 20 이동
화면	800*800	

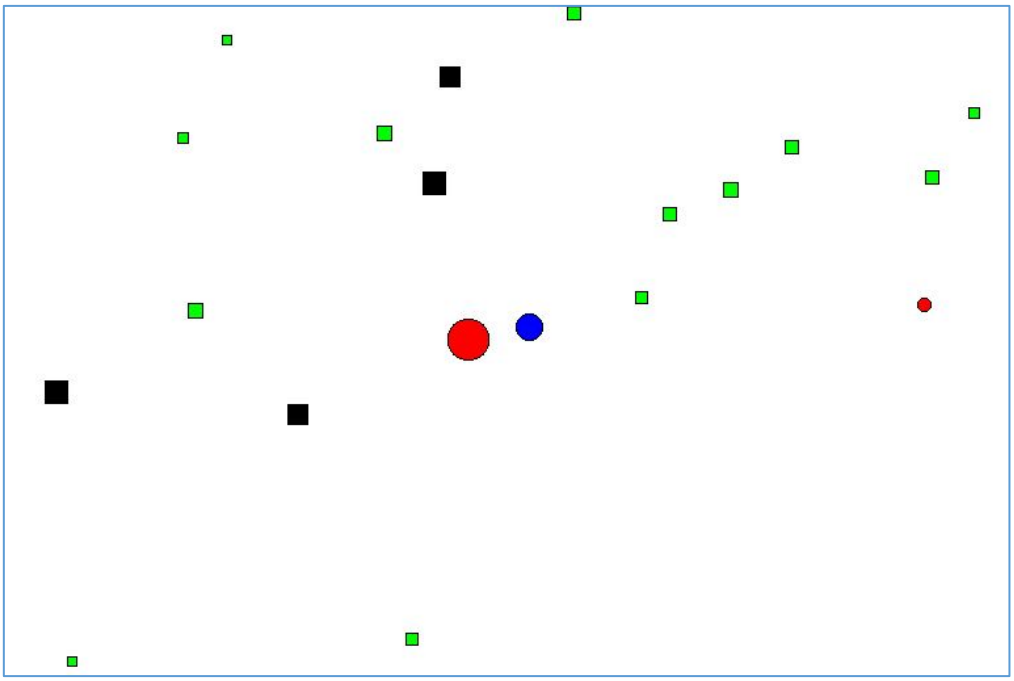

Figure 9. 게임 실행 화면 예시 (일부분 캡쳐)

(파란색: 플레이어 세포, 빨간색: 적 세포, 초록색: 먹이, 검은색: 바이러스)

4. Class Design

- 기본 클래스 디자인은 Fig. 11과 같다. QT Library를 활용하기 위해 QT 클래스를 각 클래스에 추가로 상속하여 사용하는 것을 권장한다. (e.g. QObject, QGraphicsItem, etc.)

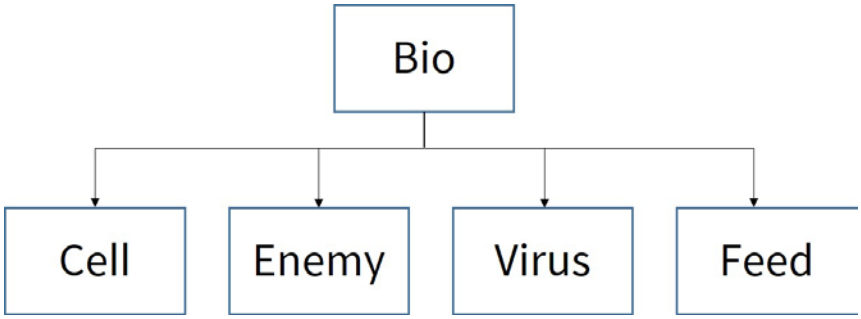

Figure 10 Class Design

- 각 클래스에 필수로 들어가야 할 변수와 메소드는 아래 표에서 확인할 수 있다.

- 아래 표에서 Bio의 Move 함수와 Draw 함수는 virtual 함수이다. 따라서 Bio 클래스는 Abstract class이며, Bio 클래스를 상속하는 Cell, Enemy, Virus, Feed 클래스에는 Move와 Draw 함수에 대해 정의해야 한다.

- 아래 표의 각 메소드를 구현하는데 필요한 인자가 있다면 사용해도 무관하다. 이 경우, 추가한 인자에 대해서 보고서에 서술한다.

- Cell 클래스의 keyPressEvent 함수는 QT에서 사용자의 입력을 관리하는 함수이다. 사용자가 방향키를 누를 때 움직임의 방향을 바꾸고, 스페이스바를 눌렀을 때 Booster 기능을 켜고 끌 수 있도록 구현한다.

- 아래 표에 언급된 필수 구현 조건 이외에 추가적으로 구현에 필요한 클래스 멤버와 메소드를 추가해도 무관하며 이에 대해 보고서에 서술한다.

- 모든 생물체 객체는 추상화 클래스인 Bio 클래스의 포인터로 선언하여 사용한다 (e.g. Bio* bios[30]). 필요한 경우 dynamic cast 등을 사용할 수 있다.

Bio		
Private	float size	생물체의 크기를 나타내는 변수
Public	virtual void Draw()	QT Library를 사용하여 생물체를 화면에 그리는 함수
	virtual void Move()	각 생물체의 움직임을 조절하는 함수

Enemy		
Private	Enum Direction	상, 하, 좌, 우 중 현재 객체가 움직이는 방향을 나타내는 변수
Public	void Draw()	Enemy를 게임 화면 내에 그리는 함수. Private 변수인 size에 비례한 크기로 그려야 하며, 빨간색 원의 형태이다.
	void Move()	Direction 변수에 따라 Enemy를 움직이는 함수 (Slot & QTimer 활용 가능)
	void changeDir()	3초 간격으로 Direction을 랜덤하게 바꿔주는 함수 (Slot & QTimer 활용 가능)

Cell		
Private	Enum Direction	상, 하, 좌, 우 중 현재 객체가 움직이는 방향을 나타내는 변수
	bool boost	Booster On/Off 여부를 나타내는 변수
Public	void Draw()	Cell을 게임 화면 내에 그리는 함수. Private 변수인 size에 비례한 크기로 그려야 하며, 파란색 원의 형태이다.
	void Move()	Direction, boost 변수에 따라 Cell을 움직이는 함수. (Slot & QTimer 활용 가능)
	void keyPressEvent()	유저의 키 입력을 관리. 방향키 (상, 하, 좌, 우) 입력에 대해 Direction을 바꾸고, 스페이스바 입력에 대해 boost 값을 바꾼다.

Virus		
Public	void Draw()	Virus를 게임 화면 내에 그리는 함수. Private 변수 size에 비례한 크기로 그리며, 검은색 정사각형의 형태이다.
	void Move()	Virus를 움직이는 함수. Virus는 1초 간격으로 오른쪽, 위, 왼쪽, 아래 순서대로 반복하여 움직인다. (Slot & QTimer 활용 가능)
	void Shrink()	4초 간격으로 size를 감소시키는 함수. (Slot & QTimer 활용 가능)

Feed		
Public	void Draw()	Feed를 게임 화면 내에 그리는 함수. Private 변수 size에 비례해 그려야 하며, 초록색 정사각형 형태이다.
	void Move()	아무 동작도 하지 않고 return한다.

5. Operator Overloading

- 생물체의 크기 비교와 생물체를 잡아먹는 연산을 연산자 오버로딩을 통해 구현한다. 다음 페이지의 표를 참고한다.

- 생물체가 충돌했을 때, 생물체 간의 크기 비교는 > 연산자를 오버로딩하여 비교한다.

- 생물체를 잡아먹어 크기가 커지는 과정을 + 연산자의 오버로딩을 통해 구현한다.

- 아래 표에 명시되지 않았으나 게임을 구현하는 데 연산자 오버로딩이 필요한 경우, 자유롭게 구현하여 사용한다.

- 보고서에 본인이 구현한 연산자 오버로딩에 대해 상세히 서술한다.

Cell / Enemy		
>	Enemy	자기 자신(Cell/Enemy)와 대상 객체(Enemy/Virus/Feed)의 size 변수를 비교하여 크면 true, 작거나 같으면 false를 return하도록 구현.
	Virus	
	Feed	
+	Enemy	자기 자신(Cell/Enemy)의 size에 대상 객체(Enemy/Feed)의 size를 더하도록 구현
	Feed	
	Virus	자기 자신(Cell/Enemy)의 size에 Virus의 size를 빼도록 구현. 이 연산으로 size가 0 이하가 된 경우, 소멸하도록 구현.

6. 주의 사항

A. 기본적으로 STL은 사용할 수 없으나, Qt Library의 충돌 체크 함수를 사용하기 위해 **QList의 사용은 가능**하다.

B. 다형성 개념이 적용되지 않은 경우 0점 처리된다.

C. QT를 사용하지 않고 다른 방식으로 콘솔 창 내에서 visualize하여 게임 진행이 가능하도록 구현한 경우, 일부 점수를 인정한다.

D. 컴파일 되지 않는 경우 0점 처리된다.

E. 보고서에 코드 전체를 그대로 복사하지 않는다.

F. 채점 기준은 Assn3ReadMe 파일을 참고한다.