객체

자료구조

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

코드 규칙

1. 작명

함수 이름, 클래스 이름, 구조체 이름은 Pascal Casing으로 작성한다. Ex) Player, MoveLeft(), IsHit()

변수 이름은 GNU Naming Convention으로 작성한다. Ex) attack\_speed, hdc\_buffer

열거형이나 상수 이름은 모든 문자를 대문자로 사용하는 GNU Naming Convention으로 작성한다. Ex) DEFAULT\_MAX\_HP, LEFT

1. 객체와 자료구조의 구분

객체는 멤버 변수를 공개하지 않는다. 공개 멤버 함수는 최대한 추상적으로 만들어 멤버 변수가 있는 그대로 클래스 외부에 노출되는 것을 피한다. 예를 들어, 사망 처리를 할 때에 Character 객체에 GetHp()를 만들어 리턴값이 0인 경우를 외부에서 체크하기보다 Character 객체에 IsHpZero()를 만드는 편이 좋다. IsHpZero()는 새로운 bool을 반환하지 기존 멤버 변수인 hp를 반환하지 않는다.

자료구조는 멤버 변수를 공개한다. public 멤버만 있는 클래스나 struct로 구현한다.

1. 함수 작성

여러 개의 함수를 만드는 것을 두려워하지 않는다. 단, 무작정 많이 만들지는 말고 다음의 네 가지 원칙을 따른다.

첫째, 한 함수의 모든 라인은 추상화 수준이 비슷해야 한다. 한 곳은 추상적이고, 한 곳은 구체적이면 안 된다.

if(damaged\_character->IsHpZero()) - 올바른 예

CharacterDieWithAnimation(damaged\_character);

if(damaged\_character->IsHpZero()) { - 잘못된 예

delete damaged\_character;

Animation(damaged\_character.GetRect(), DIE);

if (dynamic\_cast<Monster>(Character))

--monster\_num;

else

GameOver();

}

둘째, 한 함수는 최소한의 동작만 담아야 한다. (1~20라인 정도면 적당하다. 하드코딩적 요소가 필요한 경우는 예외이다.)

player.Move(LEFT); 보단 player.MoveLeft();가 낫다.

셋째, 탐색 함수와 실행 함수를 분리한다.

if(CanCharacterMove(player)) - 올바른 예

player.MoveRight();

if(!player.MoveRight()) - 잘못된 예

player.MoveLeft(); // 오른쪽으로 움직이는 것이 불가능했으니 이동 전으로 돌아간다.

넷째, 코드를 캡슐화한다. 한 라인에 여러 동작을 하지는 않는지 검토하고, 가독성을 위해 변수를 추가 선언하는 것을 꺼리지 않는다.

int next\_num = current\_num + 1; - 올바른 예

if (next\_num > 20)

DoSomeThing();

if (current\_num + 1 > 20) - 잘못된 예

DoSomeThing();

코드를 읽으면서 짐작했던 기능을 각 루틴이 그대로 수행한다면 깨끗한 코드라 불러도 되겠다.

코드가 그 문제를 풀기 위한 언어처럼 보인다면 아름다운 코드라 불러도 되겠다. – 워드 커닝햄

모든 객체와 변수는 이름대로의 정보를 담아야 하고, 모든 함수는 이름대로의 동작을 해야 한다.

코드는 문학 서적의 한 페이지로 옮겨도 이상하지 않을 만큼 서술적이어야 한다.

Git 사용법

<https://gitforwindows.org/> 에서 git을 설치한다.

앞으로 윈도우 검색창에 GitBash를 검색, 실행하여 git 명령어를 처리한다.

git config –-global user.name “유저 이름”

git config –-global user.email “가입할 때 사용한 이메일 주소”

ls와 cd를 활용하여 내가 작업할 폴더로 이동 (cd에서 띄어쓰기 때문에 인자 오류 뜰 땐 파일명을 “”로 감싸기)

git clone <https://github.com/MyeongWoonJang/-.git> 하면 작업할 폴더 안에 레포지터리 폴더가 생김 (이름이 WinTeam)

cd WinTeam 하면 레포지터리 폴더로 이동

git init

이후 git을 사용할 때마다 레포지터리 폴더로 이동하여

git pull 현재 브랜치의 가장 최근 내용 다운로드

이후 내용 편집

git add “파일 이름”, git add “파일 이름”, git add”파일 이름” …

git commit -m “올릴 때 남길 메시지”

git push origin “현재 브랜치”

브랜치 명령어

git branch 현재 브랜치 확인

git checkout “브랜치 이름” 브랜치 이동

git merge “브랜치 이름” 현재 브랜치에 인자로 받은 브랜치를 합침

파일 삭제

git rm “파일 이름”

git commit -m “삭제할 때 남길 메시지”

git push origin “현재 브랜치”

참고

<https://nevertrustbrutus.tistory.com/m/153>

<https://goddaehee.tistory.com/274?category=381481>

<https://goddaehee.tistory.com/275?category=381481>

<https://gmlwjd9405.github.io/2018/05/17/git-delete-incorrect-files.html>

main 브랜치는 완전한 파일들만 push하기

장명운은 devJang 브랜치에서 작업

이시우는 devLee 브랜치에서 작업

main 브랜치에 다른 브랜치를 merge할 때에는 서로 동의했을 때!