슈팅 로봇의 탄도궤적 연산량과 메모리 사용량 개선을 위한 다수 개체 생성 설계 기법에 대한 연구

A Study on Design Approach of Multiple Entities(Objects) Generation for improving Ballistic Trajectory Computation Volume and Memory Usage of Game Robots

목차

결론 및 향후 과제 (주장/시사점)

참고문헌 / 연구 관련 용어집

3000 words

I. 서론(연구목적/연구필요성, 현황파악) - 탄도 객체 발사 시 느려지는 문제 정의 : 탄도 궤적 연산, 메모리 사용량

1. 연구배경 : 탄도슈팅게임 소개

요즘 청소년들에게 게임의 위상은 부정적으로 생각되었던 예전과 달리 학업 스트레스를 풀거나 여가 시간을 보내기 좋은 수단 중 하나로 부상하고 있다. 미래에 사람들에게 도움이 되는 로봇과 소프트웨어을 만들고 싶은 연구자 본인도 공부에 지쳐 복잡해진 마음을 치유하기 위해 게임 자체를 즐기기도 하지만, 개인적으로 직접 프로그래밍을 통해 원하는 게임이나 검증 소프트웨어를 만드는 과정 또한 즐기기도 한다.

얼마 전 슈팅 게임을 즐기던 중, 게임 속 로봇이 발사한 다수의 탄환이 특정 개수를 넘어서면 전체 게임 동작이 심각하게 느려지거나 심지어 몇몇 기능들은 정상적으로 동작하지 않기도 하는 상황이 발생하고는 하였다. 이에 해당 문제점을 인식하고 소프트웨어 관점에서 관련된 분석을 시도해 보고자 한다.

2. 연구목적(의의/공헌점) : 품질 문제 정의 - 효율성/성능 - 탄도궤적 연산, 메모리 사용량

그림1은 소프트웨어 품질에 영향을 미치는 다양한 특성을 나타낸다.

소프트웨어 응용 분야에서 소프트웨어 동작, 즉 소프트웨어 가용성에 영향을 미치는 요인에 대한 여러 연구가 있었다[1, 2, 3]. 일반적으로 소프트웨어의 가용성 품질에 가장 영향을 크게 미치는 주요 요인은 연산 또는 메모리이다. 또한, 현재 컴퓨터 하드웨어 사양이 발전하면서 고성능의 기기에서 동작하는 게임 플랫폼 또한 복잡해지는 경향을 보이고 있다. 게임 플랫폼 환경의 고급화와 컴퓨터 하드웨어 사양의 고급화는 서로 영향을 주어 더욱 고사양을 원하는 소프트웨어들이 계속 나오고 있다. 하지만 소프트웨어 자체는 스마트폰과 같은 모바일 기기나 컴퓨터 내 자원들에 대한 사용 제약이 존재하기 때문에 연산이나 메모리의 영향이 클 수 밖에 없다. 즉, 대기시간이나 반응속도에 대한 품질을 보장할 수 있는 효율적인 연산 기법과 메모리 관리 방안이 필요하다.

앞서 슈팅 게임에서 문제에 대한 가설을 설정해 보면 다음과 같은 고려 사항을 가진다.

첫째, 탄 개체들이 생성되면서 메모리를 차지한다. 즉, 다수의 탄 개체의 생성에 대한 메모리 사용량이 로봇 동작에 영향을 줄 수 있다.

둘째, 각 탄 개체들이 궤적에 대한 연산시간을 필요로 한다. 즉, 탄 개체의 궤적에 대한 연산시간이 로봇 동작에 영향을 줄 수 있다.

결과적으로 다수 탄 개체들에 할당된 메모리와 각 탄도 궤적 연산시간을 확인해보면 문제 분석과 검증이 가능하리라 판단된다. 즉, 게임 내에 다양한 종류의 수많은

개체들이 존재하므로 그 개체들을 표현하기 위한 효율성, 특히 연산시간이나 메모리에 대한 효율성이 주 연구 대상이다. 따라서, 본 연구에서는 게임 플랫폼에 독립적인 소프트웨어 관점에서 탄도궤적 연산시간, 탄환 객체 생성 메모리 사용량을 확인하고 가능하다면 개선 방안을 찾아 보다 나은 소프트웨어를 설계/구현해보고자 한다.

그림2는 이 연구의 대상 환경을 나타낸다. 소프트웨어 설계 관점의 접근을 위해 가장 대중적인 python언어를 프로그래밍 언어로 선택하였고, 개발환경은 본 연구자의 MS Windows 기반 PC 또는 노트북과 Visual Studio Code 프로그램이다. 설계할 슈팅 로봇은 다수의 탄을 생성할 수 있고, 생성된 탄들은 여러 조건들에 따라 탄도궤적 연산 과정을 거쳐 발사된다. 본 연구의 검증을 위해 GUI는 구현하지 않을 것이고 탄환과 발사장치를 가진 슈팅로봇, 다른 속성을 가질 수 있는 탄환, 탄환 발사를 위한 발사장치(예. 총), 연산함수와 메모리 측정함수 등을 설계/구현하고자 한다.

3. 연구구성 : 관련연구-분석/개선전략-설계/구현/테스트/실험- 평가-결론/향후

본 연구의 구성은 다음과 같다. 먼저 2장에서는 관련 연구를 기술하고, 3장에서는 객체 지향 프로그래밍 설계 관점에서 일반적인 개체 생성과 연산 방법을 분석해보고, 연산이나 메모리에 대한 효율성을 고려한 다수의 탄도 개체 생성 전략을 도출하고, 4장에서는 실제 개선을 검증할 탄 개체, 궤적 연산 함수와 메모리 사용 함수를 설계/구현하고, 5장에서는 일반적인 설계와 개선 설계의 연산시간과 메모리 사용량의 변화를 측정하여 비교/분석한 후 개선 효율성을 검증/평가해보고자 한다. 6장에서는 결론 및 향후 개선과제를 제안한다. 또한, 연구 관련 용어집 정리와 관련 자료 리서치는 연구 중에 지속적으로 수행하여 추후 본 연구를 통한 추가 연구의 기반을 마련하고자 한다.

II. 관련연구(이론적배경/선행연구 : 연구가설과 관련된 정보) - 탄도 궤적 논문, 객체 메모리 사용 논문

이 장에서는 먼저 게임 소프트웨어 품질평가 연구를 통해 소프트웨어 효율성의 원인을 재확인하고[], 탄도 궤적 연산과 메모리 관련 선행 연구를 분석하여 분석과 개선전략을 위한 기반 지식을 쌓고[], 실제 설계/구현을 위해 OOD와 디자인 패턴에 대한 자료들을 활용[]하여 본 연구에 적용한다.

1. 게임 소프트웨어 품질평가

게임 소프트웨어의 품질 중 사용자 요구사항이 가장 많은 부분은 규정된 조건에서 사용되는 자원의 양에 따라 요구된 성능을 제공하는 소프트웨어의 능력인 효율성[1]이다.

본 연구에서는 효율성에 영향을 주는 주요인인 궤적 연산과 메모리 사용 기법들에 대해 구체적인 분석을 통해 실제 소프트웨어로 구현하여 검증해보고자 한다.

[1] 게임 소프트웨어의 품질 평가 모델

: 정혜정한국인터넷정보학회인터넷정보학회논문지*8(6)pp.115~1252007.12*컴퓨터학

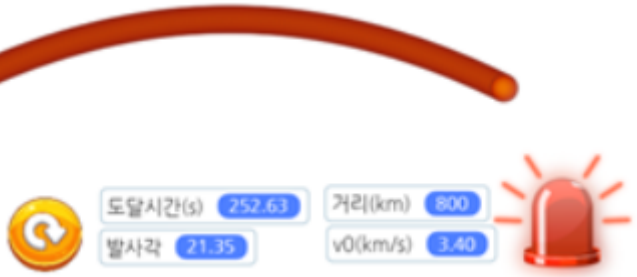
2. 탄도궤적연산

로켓의 궤적은 미적분에 의해서 계산할 수 있으며[2], 세계적으로 유명한 게임인 '앵그리버드'에서 적용된 미적분과 간단한 탄도학에 대한 지식[3]은 본 연구와 병행했던 '미적분의 원리가 적용되어 속도, 방향 등을 제어할 수 있는 탄두를 구현할 수 있을까?' 라는 연구[4]에서 활용할 수 있었다.

본 연구에서 셀프참고하는 연구자가 병행한 연구[4]에서는 포물선(탄도궤적) 운동을 수직운동과 수평운동으로 구분하고, 각 운동들에 대해 미분, 적분, 삼각함수 등 수학적 개념을 활용하여 탄도궤적에 대한 x좌표, y좌표의 위치함수를 유도하였다. 또한, 위치함수를 유도하기 위하여 물체의 예상 도달거리, 예상 도달시각, 발사각을 계산하였고, 발사되는 탄도궤적 그래프를 엔트리 프로그래밍으로 표현하였다.

우선 탄두의 운동을 수평운동과 수직운동으로 나누고 공기저항을 무시한 조건에서 수평운동은 등속운동이라고 가정, 수직운동은 등가속운동이며 중력의 영향을 받는다고 가정하였다. 탄두의 속도를 시간에 대하여 적분하면 탄두의 위치를 나타내는 x,y좌표값이 나오는데 탄두를 발사한 당시의 초기속도를 고정된 값으로 사용하여 시간의 흐름에 따라 x,y값을 좌표평면에 표시하면 탄두의 궤적이 된다.

탄두의 예상 도달거리를 입력받은 경우 발사각과 예상 도달시각을 계산하고 탄두의 x,y좌표로 탄두의 궤적을 표현하였다. 예상 도달거리만 입력하는 경우 초기발사속도를 10 마하, 즉 3.4km/s로 고정 하였다. 탄두의 목표물은 정해져 있으므로 예상 도달거리는 항상 입력해야 한다는 가정을 하였고 탄두의 발사 초기속도와 도달거리를 사용자가 정하여 입력하면 발사각과 예상 도달시각을 계산하는 과정을 거쳐서 탄두의 궤적을 좌표평면에 표시하였다. 탄두가 목표물에 도달해야 하는 예상 도달시각을 예상 도달거리와 함께 입력하는 경우 초기속도와 발사각을 계산하였고, 탄두의 예상 도달거리와 발사각을 입력하는 경우 초기속도와 예상 도달시각을 계산하여 탄두의 궤적을 좌표평면에 표시하였다.



[2] 미적분의 쓸모 *(the use of calculus)*

: 한화택*(Hwataik Han)* 출판사-더퀘스트 *ISBN-9791165219550*

[3] 이렇게 흘러가는 세상 *(This is how the world goes)*

: 송현수*(Hyunsoo Song)* 출판사*-MID ISBN-9791190116213*

[4] 수학을 이용한 실생활 연구 보고서

: 정휘준 *https://hweejoon-chung.github.io*

3. 메모리 사용

자원에 대한 사용 제약이 존재하는 환경에서 게임 소프트웨어의 질을 높일 수 있는 효과적인 메모리 관리 및 활용 기법이 필요하다. 이를 통해 메모리와 밀접한 관계에 있는 게임 반응 속도 역시 효과적으로 높일 수 있다.[5]

[5] 메모리가 제한적인 자바가상기계에서의 지역 재사용

: 김태인김성건한환수한국정보과학회정보과학회논문지 *:* 소프트웨어 및 응용*34(6)pp.562~5712007.06*컴퓨터학

4. OOD와 디자인 패턴

대규모의 게임을 만들 때 객체지향개념을 충분히 활용할 수 있는 설계기법이 유용하다.[7]

본 연구에서는 객체 지향 디자인 패턴 중 탄도 개체의 생성절차를 추상화하기 위한 생성패턴에 대해서만 분석하여 활용하고자 한다. 다양한 탄도 개체들 중 특정 개체가 생성 또는 변경되어도 프로그램 구조에 영향을 주지않도록 일반적으로 다음과 같은 5가지 생성패턴이 존재한다.[6]

(1) 추상 팩토리 패턴 : 동일한 분류의 다른 팩토리들을 묶어줌
(2) 빌더 패턴 : 생성과 표현을 분리하여 복잡한 객체를 생성
(3) 팩토리 메서드 패턴 : 생성할 객체의 클래스를 특정하지 않고 객체 생성 가능
(4) 프로토타입 패턴 : 기존 객체를 복제함으로써 객체 생성
(5) 싱글턴 패턴 : 하나의 클래스에 하나의 객체 인스턴스만 존재

실제 상용 게임에서 다양한 디자인 패턴들을[7] 활용하여 객체지향 설계를 지향한다.

[6] Design Pattern (Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides, Addison-Wesley)

*[7] Observer*패턴을 적용한*MMORPG*파티 시스템 아이템 배분 방법

:
김태석김신환김종수한국멀티미디어학회멀티미디어학회논문지*10(8)pp.1060~10672007.08*전자/정보통신공학

III. 분석/개선전략(연구내용/가설설정/연구방법, 전략 설정) - 탄도 객체 발사 관련 연산과 메모리 사용 확인방안 및 개선 전략 설정

기본 설계

1. 탄도궤적 연산 기법 도출

슈팅 게임에서 슈팅로봇이 탄도를 발사하면 발사체의 경로를 계산할 필요가 있는데 이 때 복잡한 연산을 수행한다. 구체적으로 거리를 입력받아 각도, 도달시간, 탄도궤적 등을 계산하는데 미분과 적분 개념을 이용한 연산이 필요하고[3], 이는 전체 프로그램의 시간이나 속도에 대한 효율성에 영향을 미친다.

병행연구[4] 내용을 기반으로 본 연구에서는 탄두의 궤적을 연산하려면 발사각과 초기속도, 도달시각, 도달거리가 필요한데 4가지 조건 중 도달거리는 슈팅로봇이 알고있다고 가정한다. 따라서, 도달거리를 제외한 3가지 조건 중 2 이상의 조건이 주어지면 3가지 탄도 궤적 연산 계산식들을 (삼각함수를 이용하여) 도출 가능하다.

본 연구에서는 3가지 탄도 궤적 연산 계산식들의 연산량(연산시간)을 비교하여 가장 최적의 식을 선택하고자 한다.

그림은 3가지 탄도궤적 연산 계산식을 나타낸다. 첫번째 식은 ~~~, 두번째 식은 ~~~, 세번째 식은 ~~~이다.

2. 개체 생성 방안 분석 및 적용패턴 선정

소프트웨어 개발 시, 가장 기본적인 개체 생성 방식은 탄 개체 갯수를 입력받아 해당 개수의 탄도 개체를 만드는 것이다. 다수의 탄을 만들게 되면 메모리 사용량도 그에 비례하여 증가하게 된다.

기본적인 개체 생성방식의 문제 개선 전략을 마련하기 위해, 객체 지향 설계 관점에서 탄도 개체 생성의 특징을 살펴보면 다음과 같다.

1. 동일한 수많은 탄도 개체가 독립적으로 생성
2. 탄도는 발사체에 따라 다른 특성을 가질 수 있음
3. 탄도 개체의 생성 제약은 해당 소프트웨어가 동작하는 시스템에 따라 달라짐

과 같다.

객체지향 설계방법 중 위의 특징들을 개선할 수 있도록 본 연구에서 도출한 전략은 다음과 같다.

1. 디자인 패턴을 활용하여 독립적으로 생성되도록 하자.
2. 탄도 개체는 생성 시 여러 발사체 형태가 가능하도록 팩토리 메서드 패턴이 필요하다.
3. 특정 상태값들을 가지지 않는 단순한 탄도 궤적 표현을 위한 탄도 개체 생성이 가능하도록 Flyweight 패턴을 적용한다.

개선 설계

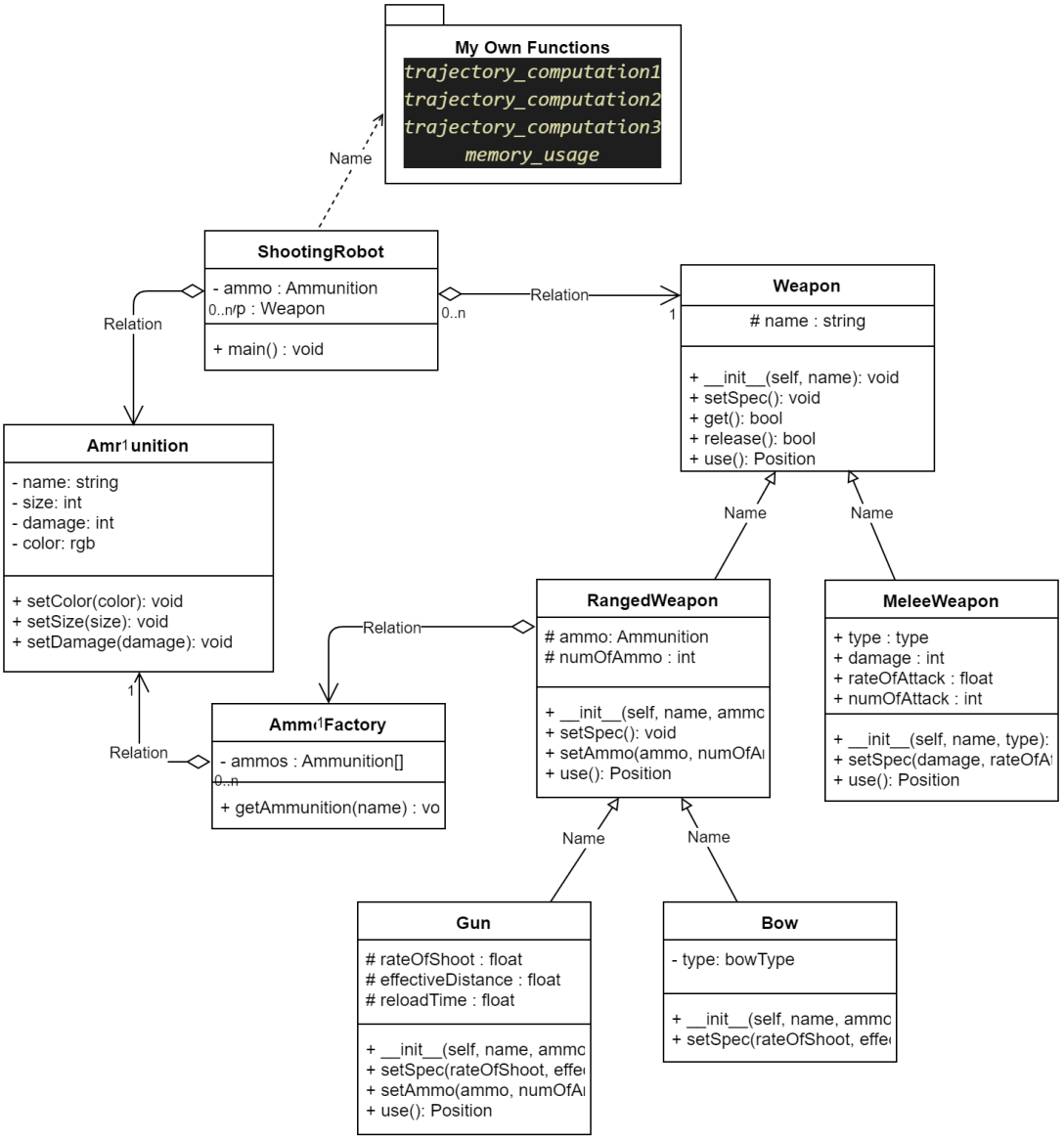
IV. 연구결과(분석결과/가설검정, 탄도개체 설계) - 모의 테스트(실험)

다수의 탄도 개체 생성 설계 및 구현 (테스트 용도)

시스템 구현/테스트 환경

본 연구의 시스템 구현 환경은

탄도 개체 생성 수에 따른 탄도 궤적 연산량과 메모리 사용량 확인 (설계/구현)



V. 결과 고찰 - 평가

평가 방법

발사체 객체를 생성하여 메모리의 사용량을 비교해본다.객체가 필요한 만큼 생성될 때마다 메모리가 할당되는 방법과 패턴을 사용하여 객체를 생성할 때 메모리가 할당되는 방법 중 메모리 사용량이 적은 쪽이 더 효율적인 방법이라 하겠다.

```python
def getAmmunition(name):
    ammo = AmmunitionFactory._ammo_mapping.get(name, None)

    if ammo is None: #해당 타입의 탄도가 없다면
        ammo = Ammunition(name) #새로운 객체 생성
        AmmunitionFactory._ammo_mapping[name] = ammo
        print(f"===새로운 객체 생성 [{name} / {len(AmmunitionFactory._ammo_mapping)}] 발사체 ===")

    return ammo
```

```python
def getAmmunitionNaive(name):
    ammo = Ammunition(name) #새로운 객체 생성
    AmmunitionFactory._ammo_list.append(ammo)
    print(f"===새로운 객체 생성 [{name} / {len(AmmunitionFactory._ammo_list)}] 발사체 ===")

    return AmmunitionFactory._ammo_list
```

궤도를 계산하는 세 가지 방법을 기술하는 함수의 연산량을 비교해본다. 앞서 언급한 궤도를 계산하는 세 가지 방법은 다음과 같다.

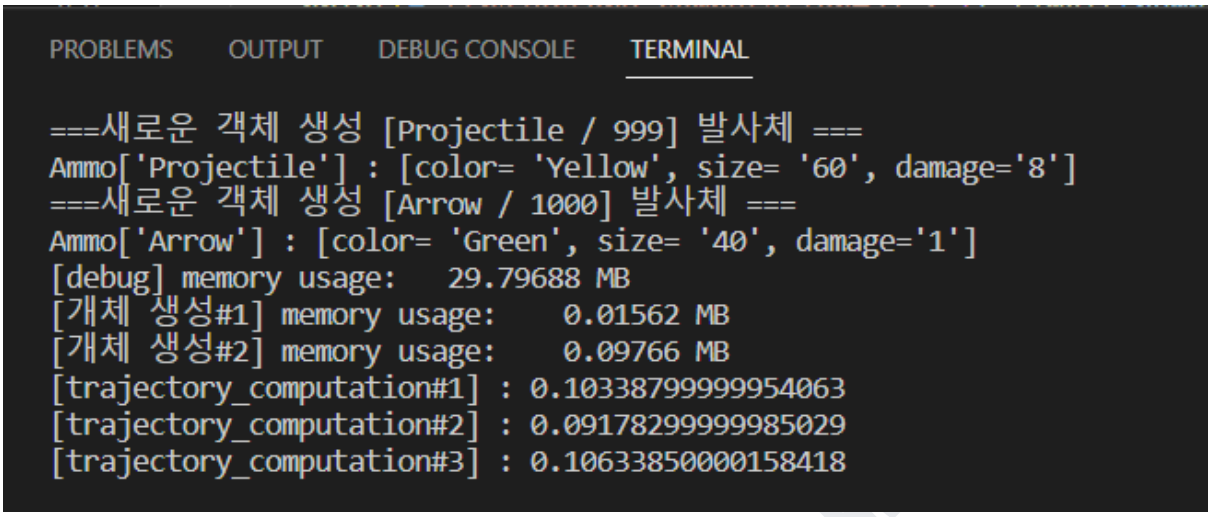
[tragectory_computation#1]	탄두의 예상 도달거리와 초기속도를 입력받아 탄두의 x,y좌표를 계산하는 함수
[tragectory_computation#2]	탄두의 예상 도달거리와 예상 도달시각을 입력받아 탄두의 x,y좌표를 계산하는 함수
[tragectory_computation#3]	탄두의 예상 도달거리와 발사각을 입력받아 탄두의 x,y좌표를 계산하는 함수


```python
def trajectory_computation1(r, v0) :                # 궤도 연산 함수1 : r(km), v0(km/s)
    theta = np.arcsin(r*0.0098/v0)/2
    t = v0*np.sin(theta)*2/0.0098
    x = v0*np.cos(theta)*t
    y = v0*np.sin(theta)*t - 0.0049*t**2
    return x, y

def trajectory_computation2(r, t) :                 # 궤도 연산 함수2 : r(km), t(s)
    v0 = math.sqrt((r/t)**2 + (0.0049*t)**2)
    theta = np.arcsin(r*0.0098/v0)/2
    x = v0*np.cos(theta)*t
    y = v0*np.sin(theta)*t - 0.0049*t**2
    return x, y

def trajectory_computation3(r, theta) :             # 궤도 연산 함수3: r(km), theta(radian)
    v0 = math.sqrt(r*0.0098/np.sin(2*theta))
    t = v0*np.sin(theta)*2/0.0098
    x = v0*np.cos(theta)*t
    y = v0*np.sin(theta)*t - 0.0049*t**2
    return x, y
```

성능평가 - 전략에 대한 연산과 메모리 사용량 비교



[개체 생성#1] memory usage는 패턴을 사용하여 다수의 개체들을 생성할 때의 메모리 사용량이며 [개체 생성#2] memory usage는 개체를 생성할 때마다 메모리를 할당해야 할 때의 메모리 사용량이다.

VI. 결론 및 향후 과제 (주장/시사점)

참고문헌 / 연구 관련 용어집