|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 2 주차 | **기간** | 2025.09.08  ~ 2025.09.14 | **지도교수** | (서명) |
| 이번주 한일 요약 | 1. 멀티코어 프로그래밍 학습  2. 셰이더 프로그래밍 학습  3. 서버 외부 강의 수강  4. 데이터베이스 학습  5. 매주 팀원 정기 회의 | | | | |

<상세 수행내용>

1. 서버 외부 강의를 통해 Client – Server 연동 실습

2. 셰이더 프로그래밍 학습

Vectex, Fragment Shader 작업 수행

-> 삼각형을 그린 후 정점의 데이터를 Shader에게 처리할 수 있게 프로그래밍

-> Color을 입혀 삼각형 각 정점에 RGBA값을 넣은 후 Fragment Shader에서 처리

Vectex의 데이터가 바뀌지 않더라도 형태를 바꾸거나 특정 부분만 그리는 방식을 학습함

->

3. 멀티코어 프로그래밍 학습

4. 데이터베이스 학습

데이터베이스 시스템의 구성 요소, 메타 데이터(시스템 카탈로그)를 활용하여 형식을 확인 후 데이터 베이스에 접근하는 방식 학습

파일 시스템의 장점과 단점, 데이터의 형식이 바뀌어도 그에 대한 의존도를 낮추는 방법에 대해 학습함

-> 메타 데이터를 생성하고 이 곳에 형식을 저장 후, 찾아가 해당 위치 데이터베이스에 데이터를 저장하는 방식, 그리고 찾고자 하는 형식을 찾은 후 데이터에 접근하는 것임을 인지함, 유지보수에 용이함

5. 매주 정기 팀원 회의

매주 토요일 오후 9시, 정기적으로 팀원과 만나 회의 진행

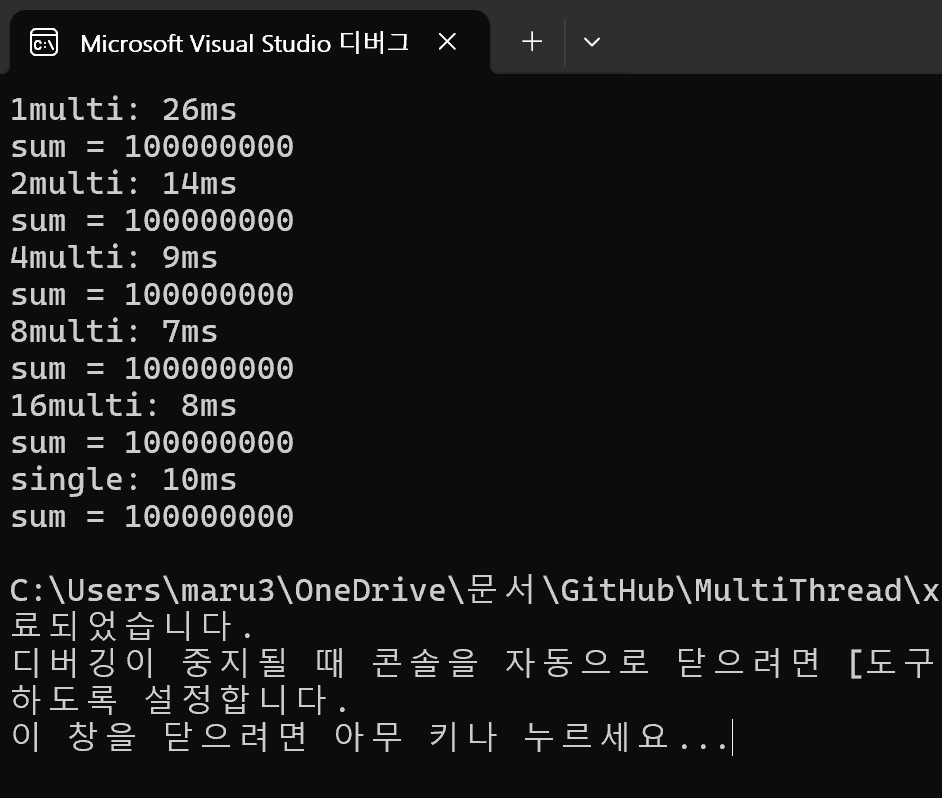
각자가 이번주에 공부한 내용 및 기획서 공유

Lock을 사용하지 않고 Data Race를 최소화하기 위해 alignas와 volatile을 사용해 구조체 형태로 선언하고 배열 형태로 선언하여 각 쓰레드가 자기 번호의 영역만을 접근하여 데이터를 쓰는 방식으로 방법을 바꿔보았다.

그러나 여전히 싱글 쓰레드에 비해 일부는 느린 경향을 보였다. 그리고 lock을 사용하지 않았기에 컴파일러가 volatile을 사용했음에도 불구하고 나머지 코드는 최적화하여 순서를 바꾸거나 아예 안 돌릴 가능성도 있었다.

이를 해결하기 위해 상호 배제를 유지하고 lock과 volatile을 사용한 상태로 Cache Thrashing이 이루어지지 않게 조치를 해야 한다.

이에 대한 조치로 Bakery algorithm이라는 알고리즘이 있었다. 이를 학습하여 적용하면 lock을 사용하면서 상호 배제 상태를 유지할 수 있을 것 같다.



|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | | |
| **해결방안** |  | | |
| **다음주차** | 3주차 | **다음기간** | 2025.09.15 ~ 2025.09.21 |
| **다음주 할일** | 1. 멀티코어 프로그래밍 학습  2. 셰이더 프로그래밍 학습  3. 에코 서버 테스트  4. 매주 팀원 정기 회의 | | |
| **지도 교수**  **Comment** |  | | |