|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **주차** | 19주차 | **기간** | | 2025.01.05 ~  2026.01.11 | **지도교수** | (서명) | |
| **이번주 한일 간단 요약** | 1. 열화상 카메라 제작  2. 머터리얼 변경 기능 추가  3. 배터리 및 전선 메쉬 적용  4. 열 전달 공식 정리 | | | | | | |
| **상세 내용** | **1. 열화상 카메라 제작**    Post Process Material을 활용한 열화상(Thermal Vision) 카메라 기능을 구현하였습니다. 오브젝트의 온도 값을 색상 그라데이션으로 시각화하여, 온도 분포를 직관적으로 확인할 수 있도록 하였습니다.    문제점 : Post Process 기반 열화상 카메라를 구현하는 과정에서, 효과가 특정 오브젝트에만 적용되지 않고 하늘 및 배경을 포함한 화면 전체에 적용되는 문제가 발생했습니다.  개발하기 위해 Custom Stencil(어떤 오브젝트에서 왔는지 구분하는 번호표(ID))을 이용했는데, Post Process 단계에서는 화면 전체를 대상으로 렌더링이 수행되기 때문에 Stencil 값이 설정된 오브젝트뿐만 아니라 하늘 및 배경 영역에도 동일한 열화상 연산이 적용되는 문제가 발생했습니다.    그래서 Custom Stencil 값이 설정된 오브젝트만을 열원으로 간주하여 화면에 표시하고, 그 외의 오브젝트는 열화상 처리에서 완전히 제외하는 방식으로 구현을 진행할려고 했습니다. 그러나 이 열이 존재하는 오브젝트만 단독으로 표시되어 현실적인 열 확산이나 환경 반응을 표현하기 어렵다고 판단되어,    다시 이 부분을 고쳐서 나가기 위해 여러 가지 방식을 고려했습니다.  첫째, Custom Stencil 값을 기준으로 열 객체와 열이 없는 객체를 분리하여, 열원이 되는 오브젝트에만 열화상 효과가 적용  둘째, 플레이어와의 거리를 기준으로 일정 범위 내에 위치한 오브젝트에만 열화상 효과가 적용되도록 하는 방식  셋째, 플레이어가 조준한 대상에 열이 있으면 열화상 카메라로 나오는 방식 등으로 해결방안을 모색하고 있습니다.  **2. 머터리얼 변경 기능 추가**      플레이어가 블록을 직접 선택하여 재질(폼/Mode)을 변경할 수 있도록 Line Trace 기반 머터리얼 변경 기능을 구현했습니다. 입력키(I)를 누르면 캐릭터 카메라 방향으로 Line Trace(레이저)를 발사하도록 구성했으며, 충돌한 오브젝트가 바꿀 수 있는 객체인 경우 해당 블록의 상태를 변경하도록 처리했습니다.  **3. 배터리 및 전선 메쉬 적용**    (전선은 작아서 임의로 확대했습니다)    모델러가 제작한 배터리 및 전선 메쉬를 프로젝트에 적용했습니다. 또한 전선 머터리얼도 전원 시스템과 연동하여 전선의 상태를 제어할 수 있도록 구성했습니다.  **4. 열 전달 공식 정리**  현재 열 객체가 얼음에 전달되는 열을 계산하기 위해 열 전달 방식에 대해 정리했습니다. 열 전달 방식에는 전도, 대류, 복사가 있으나, 본 시스템에서는 **복사열 방식**을 중심으로 구현했습니다.  전도 방식은 물체 간의 직접적인 접촉이 필요하며, 공기를 통한 전도 효과는 매우 제한적이기 때문에 본 시스템에는 적합하지 않았습니다.  대류 방식의 경우 공기 흐름, 방향, 환경 조건에 따라 열 전달량이 불규칙하게 변화하며, 거리 증가에 따른 열 감소를 일관된 수식 구조로 적용하기 어려웠습니다.  이에 따라 매질의 영향을 받지 않고 온도와 거리만으로 열 전달량을 표현할 수 있는 복사열 방식을 적용했습니다.  복사열의 기본 개념은 스테판–볼츠만 법칙을 참고했습니다.    P : 단위 시간 동안 주변으로 방출하는 총 복사 에너지, ε (방사율) : 열 객체의 재질 특성을 반영한 계수 σ : 물리적으로 정해진 스테판–볼츠만 상수이며, A : 열 객체의 표면적,  T : 열원의 절대온도입니다.  해당 식을 기준으로 온도가 증가할수록 방출되는 열량이 T⁴에 비례해 급격히 증가하도록 설계했으며, 이를 통해 고온 열 객체와 저온 객체 간의 열 영향 차이가 명확하게 드러나도록 구현했습니다.  복사열은 열원에서 방출된 에너지가 공간으로 확산되며 거리 증가에 따라 감소하는 특성을 가집니다. 이를 설명하기 위해 역제곱 법칙을 참고했습니다.  q(r) : 단위 면적당 복사 열량  P : 총 복사 에너지의 양  r : 거리  4πr² : 구의 표면적  이로 인해 열원에서 멀어질수록 동일한 에너지가 더 넓은 면적에 분산되어, 대상 객체가 받는 열량은 거리의 제곱에 반비례하여 급격히 감소하게 됩니다.  이러한 특성을 기반으로 열 객체와 객체 사이의 거리를 함께 고려한 열 감쇠 구조를 적용했으며, 열 객체에 가까운 얼음은 강한 열 영향을 받아 빠르게 반응하고, 거리가 증가할수록 열 전달량이 빠르게 감소하도록 구현했습니다.  텍스트, 스크린샷이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  (개발하면서 임의로 정해둔 온도)   |  |  | | --- | --- | | 항목 | 값 | | 열 객체 표면적 | 1.0 m² | | 표면 온도 | 600 °C (873 K) | | 방사율 | ε = 0.3 | | 스테판–볼츠만 상수 | σ = 5.67 × 10⁻⁸ |   실제 저희가 사용하고 있는 얼음의 부피가 1 m³ 이고  융해 잠열(고체를 액체로 상태만 바꾸는 데 쏟아붓는 에너지)가 334,000J/kg  총 필요한 에너지가 3.34×108J 로 나왔고  1m 거리를 유지한다고 가정해서 열 전달을 계산한 결과 1840J/s가 나왔습니다.  이를 얼음에 계속 전달하면 다 녹는데 46시간이라는 결과가 나왔습니다.  그래서 온도를 높이거나. 얼음크기를 줄이는 등 조금 더 속도를 높여야 할 것 같다고 결과가 도출되었습니다.    (해당 이미지는 수식 입력이 어려워서 GPT 도움을 받았습니다) | | | | | | |
| **다음주차** | 20주차 | | **다음기간** | | | | 2026.01.04 ~  2026.01.11 |
| **다음 주 할 일** | 1. c++ 개발  2. 열화상 카메라 개선 | | | | | | |
| **지도 교수의**  **피드백** |  | | | | | | |