|  |  |  |  |
| --- | --- | --- | --- |
| Magician War | | | |
| 15주 | 2021. 4. 04 ~ 2021. 4. 10 | 작성자 | 김덕규 |
| 주간 목표 | 김덕규  **1. 로비 씬 제작**  홍범도  **1. 하드웨어 인스턴싱 & 동적 인덱싱** | | |
| 이번주 한 일 | 김덕규  **1. 로비 씬 제작**  - 버튼 UI 제작    버튼이 눌리지 않았을 때, 마우스를 올렸을 때, 버튼이 눌렸을 때 3가자 상태로 구분하여 다른 텍스쳐를 띄우게 만듬.  - 서버 로직 제작    서버 구동을 위한 로직 제작, 필요한 패킷을 범도와 함께 정리했다,  추후에 패킷은 변동 가능성이 있다.  홍범도   1. **하드웨어 인스턴싱 & 동적 인덱싱** 2. **동적 인덱싱**   기존에는 각 Object마다 Constant Buffer를 들고 있었다.  (ex.Object Constant Buffer, Skinned Constant Buffer, Material Constant Buffer)  그래서 그래픽스 파이프라인에 Set Constant Buffer도 각 오브젝트마다  각각의 버퍼를 연결해줘야해서 한 오브젝트 마다 최소 1~3개의  Constant Buffer를 CPU에서 일일이 그래픽스 파이프라인에 묶어줘야했고  이로 인한 부담이 발생하고 있는 것 이였다.  그래서 이번에 **동적 인덱싱**을 통해 shader에서 StructuredBuffer로  Material를 처리할 수 있게 해주었다.  원래 Object들이 Constant Buffer를 들고 있었는데 Material Manager에서  모든 Material 종류별로 1개씩의 SRV 버퍼를 들고 처리해 주었다.  이제 CPU는 그래픽스 파이프라인에 SetSRV 한번으로 묶어줄 수 있게 되었다.   1. **하드웨어 인스턴싱**   인스턴싱은 한 장면에서 같은 물체를 여러 번 그리되 위치나 방향, Material등을 각자 다르게 해서 “한 번의 그리기”를 목표로 하는 것이다.  지금까지는 한번의 그리기 호출로 1개의 Object를 그렸다.  그래서 동일한 오브젝트여도 정점 자료와 인덱스 자료를 계속해서 바꿔서 그래픽스 파이프라인에 묶어주어야 했고 이는 곧 CPU가 낭비되는 것 이였다.  이제 하나의 mesh는 동일한 정점 자료와 인덱스 자료를 사용하므로  mesh별로 Buffer를두고 그리기를 할 때 지금 동일한 mesh를 사용하는 object들의 개수(즉, Instance )들을 파악하여 버퍼에 저장해서 CPU에서 그래픽스 파이프라인에 묶어줄 수 있게 **동적 인덱싱**을 적용하였고, 그래픽스 파이프라인에 정점 자료와 인덱스 자료를 Mesh별로 1번씩만 묶고 DrawIndexedInstance를  사용해서 1번의 Draw 호출로 같은 mesh를 사용하는 오브젝트들을 다 그리게 해주었다.  이로써 하드웨어 인스턴싱을 적용하였다. | | |
| 다음주 할 일 | 공동작업   1. **TCP 서버 개발**  * **로비에 필요한 패킷 교환 기능 구현** * **인 게임 플레이어 이동 패킷 제작**   김덕규   1. **로비 씬 제작**   홍범도   1. **스킬 개발** | | |
| 문제점 | 홍범도  인스턴싱을 구현하는데 기존의 Framework구조와 충돌되는 부분들이 많아서 많은 부분을 바꿔야 할 것 같았고 실제로도 바꿧었는데, 중간에 STL을 적절히 활용하여서 기존의 구조를 대부분 유지하면서 인스턴싱을 적용할 수 있게 되어서 다행이였다. | | |