|  |  |  |  |
| --- | --- | --- | --- |
| [Cranding(크랜딩)] | | | |
| **주차** | **2024.01.17 ~ 2024.01.27** | **작성자** | **정경서** |
| **주간 회의** | | | |
| **회의 날짜** | | 2024.01.25 | |
| 각자 방학 계획을 돌아보고 계획 수정하기  정경서  6주차: 애니메이션 관련 오류 수정 + 디퍼드 렌더링 공부  7주차: 애니메이션 관련 오류 수정 + 프레임워크 수정 + 디퍼드 렌더링 구현  8주차: 프레임워크 수정 + 디퍼드 렌더링 구현  9주차: 프레임워크 수정 + 디퍼드 렌더링 구현  10주차: 프레임워크 수정 + 디퍼드 렌더링 구현  홍예나  6주: 여행 + 애니메이션 블렌딩 구현  7주: 제발 애니메이션 블렌딩 구현 마무리 + 가우시안 블러 공부  8주 프레임워크 수정 + 가우시안 블러 구현 시작  9주: 프레임워크 수정 + 가우시안 블러 구현  10주: 프레임워크 수정 마무리 + 가우시안 블러 구현 | | | |
| **이번주 한 일** | | | |
| 1. 신동엽(서버)   월 ~ 금 : 일본 여행  서버 프레임워크를 다듬었다.  만들어 둔 SocketUtils클래스를 통해 리슨소켓과 클라이언트와 작용할 클라이언트 소켓을 만들어주고 Over\_exp라는 클래스로 overlapped 클래스를 만들었다 기존에는 overlapped 클래스를 상속받아 만들었지만 멤버변수에 WSAOVERLAPPED 변수를 두어 관리하기로 정했다.  Overlapped 클래스는 처음 생성될 때 accept type을 iocp 에 등록하고 이를 iocpcore에 Dispatch라는 함수에서 처리하도록 만들었다.  또한 기존에 만들었던 ThreadManager를 활용하여 클래스 멤버함수를 쓰레드 함수로 실행시키는 것에 성공하였다. 이전에 게임서버 텀프로젝트에선 모든 함수와 모든 클래스들이 main에 있던것이 맘에 들지않았지만 파일분할을 얼추 완성하고 네트워크에 필요한 부분과 쓰레드를 관리하는 부분, 전역으로 다루는 부분 등 모두를 분할시켜 정리하니 비교적 많이 깔끔해졌다.  Threadmanger를 사용하는 부분에서 바인딩 함수를 처리하는 대에 있어서 생각보다 시간을 많이 사용하였다. 결과는 람다를 통해 클래스 멤버함수를 실행시켜주도록 유도하였다.  파일을 분할하면서 발생한 문제들  겨우 클라이언트와 connect를 성공시켰지만 네트워크가 굉장히 불안하고 어쩔땐 자기멋대로 프로그램이 죽는다. 디버깅을 통해 에러를 확인해 보니 errcode: 10061가 발생하여 강제종료되는 에러를 뱉었다.  또한 서버에선 connect를 완료하였지만 서버에서 제대로 accept가 되어 컨테이너 안으로 클라이언트가 들어가지 않는 모습이다.  스크린샷, 텍스트, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진  자동 생성된 설명  유니티를 통해 맵 제작을 시작  ‘불’ 지형 리소스를 구하는데 시간이 더 걸릴 것 같다.  ‘얼음’ 지형 맵 제작 중   1. 정경서(클라이언트)   3D게임프로그래밍 개념 공부  구조 변경을 위해 3D 게임 프로그래밍1의 온라인 강의를 다시 수강하기로 계획하였다  강의는 총 33개가 올라와 있는데 필요한 강의만 수강할 예정이다.  1차시\_3D 그래픽스 기초 ( 수학적 기초 개념 - 01)  렌더링 엔진(렌더러)   * 가상적인 게임 세계의 3차원 표현을 스크린에 2차원으로 그려내는 소프트웨어. 그러한 과정 * 수학적 3차원 표현과 처리 방법이 필요하다 * 그림 그리기 알고리즘(Painter’s Algorithm)   게임 세계를 스크린에 그려낼때의 알고리즘  모델, 메쉬, 객체   * 메쉬: 물체의 **외관**을 표현하기 위한 다각형들의 집합. 구조화 된 점들의 집합 * 다각형: 연결된 선분들의 집합 * 선분: 연결된 점들의 집합 * 메쉬는 삼각형들의 집합으로 표현 가능하다. * 면: 메쉬의 각 다각형   직육면체: 6개의 면을 가짐. 한번에 최대 3개의 면만 볼 수 있다. 한 면은 4개의 점으로 구성된다.  2차원에서의 기하학  좌표계   * 화면 좌표계 * 직교 좌표계: 각 축들이 직각을 이루는 좌표계   3차원에서의 기하학   * 왼손 좌표계: 다렉 좌표계 * 오른손 좌표계   메쉬의 표현   * 매쉬: 다각형들의 집합 * 다각형: 순서화된 3차원 점들의 집합   위치와 방향  좌표계   * 원점, 축, 축의 방향, 단위   모델의 표현  모델 좌표계/객체 좌표계/로컬좌표계   * 모델을 표현하기 위한 좌표계 * 모델마다 자체적인 별도의 좌표계를 가지고 있다. * 객체 로컬 공간 * 일반적으로 객체의 중심을 원점으로 하는 좌표계   월드 좌표계   * 게임 세계를 하나의 통일된 좌표계로 표현 * 모든 객체들이 동일한 하나의 좌표계로 표현   모델과 객체   * 모델 = 메쉬(기하학적 모델) * 객체 = 모델의 인스턴스   매쉬와 텍스쳐 매핑   * 맵: 2D 이미지(ex) 높이맵, 텍스쳐 맵, 법선 맵, 범프 맵, 조명 맵) * 매핑: 2D 이미지를 사용하는것 * 텍스쳐: 3D 모델에 그릴 2D 이미지를 뜻한다   게임 월드의 단위   * 모든 객체들이 일관된 크기가 아닐때를 대비하여 만들기   은면 제거   * 관찰자(카메라)가 볼 수 없는 면을 그리지 않는 것   와인딩 순서   * 다각형의 정점을 나열하는 순서를 나타내는것 * DirectX에서는 시계방향이 기본이다 * 반시계 방향은 은면   렌더링  Scene(장면)   * 게임 월드 자체 * 게임 월드에서 현재 화면에 그려져야 하는 부분 * 게임 월드 = 객체(메쉬)들   렌더링   * 장면을 렌더링 한다 = 게임 월드의 모든 객체를 그린다 * 모든 다각형들을 그리는 것 * 모든 픽셀들을 그리는 것(색칠하는 것) * 다각형의 3D 모델 좌표를 2D 화면 좌표(픽셀)로 변환하는 과정 * T&L (Transformation & Lighting)   2차시\_3D 그래픽스 기초 ( 수학적 기초 개념 - 02)  **변환 파이프 라인(Transformation Pipeline)**   * 파이프 라인이란 작업이 순차적으로 수행 된다는 뜻이다. * 3D 모델 좌표를 2D 화면 좌표(픽셀)로 변환하는 코드(함수)      * 기본적으로 4개의 단계로 구성  1. 월드 변환 2. 카메라 변환 3. 투영 변환 4. 화면 변환  * 정점이 4개의 단계를 거치면서 화면 좌표를 얻는 과정이다   변환 파이프라인의 목적   * 3D로 표현된 매쉬를 2D 표현으로 변환하여 화면에 그리도록 하는 것 * 2D 화면에서 3D를 느낄 수 있게 원근감을 가지도록 하는 것 * painter’s algorithm      * 월드 변환, 카메라 변환, 투영 변환은 Vertex Shader(VS) 단계에 속한다 * 화면 변환은 Rasterizer(RS) 단계에 속한다 * 각 픽셀의 색을 정하는 것은 Pixel Shader 단계에 속한다   **오브젝트에 대해 렌더링 파이프라인에서 대표적으로 일어나는 변환은 SRT(Scale, Rotate, Translation)**  **변환:평행이동(Translation)**   * 모델 좌표계의 점을 월드좌표계의 위치로 이동 * 인스턴싱: 하나의 메쉬를 여러 객체에서 공유하는 방법 * GPU에서 실행이 되는 인스턴싱: 하드웨어 인스턴싱   변환:회전(Rotation)   * 원점을 기준으로 회전하는 것을 가정 * 3D 점을 원점에서 하나의 축을 기준으로 회전 = 2D 평면에서 원점을 기준으로 회전   → 회전축의 좌표는 변하지 않는다.  ex) X축을 중심으로 회전 할때 X좌표는 바뀌지 않는다   * 평행이동과 회전을 동시에 할때는 순서가 중요하다.      * 왼: 이동 → 회전(공전) 오: 회전 → 이동(자전) * 순서에 대한 특별한 언급이 없다면 회전 → 이동 순서로 진행된다   3차시\_3D 그래픽스 기초 ( 변환 파이프라인 - 01)  **카메라 변환 -** 가상 카메라   * 게임 월드를 보기 위해서는 가상의 카메라가 필요하다 * 일반적으로 가상 카메라는 플레이어 캐릭터에 부착되어 있다 * 카메라 정보는 위치(Position), 방향(Direction), 화각(Field Of View)이 있다   **카메라 변환** - 카메라 좌표계   * 게임 월드에 있는 모든 객체를 카메라 중심으로 한 상대적인 좌표계로 표현하는 것 * 카메라에 보이는 객체를 쉽게 알 수 있다   **카메라 변환**     1. 카메라를 월드 좌표계의 원점으로 평행이동 2. 카메라 좌표계의 축이 월드 좌표계의 축과 일치하도록 카메라를 회전한다. 이 회전을 모든 객체에 적용한다 3. 객체는 카메라를 중심으로 한 좌표계로 표현된다   **원근 투영 변환(Perspective Projection Transformation)**   * 원근감이 느껴지게 * 카메라에서 멀리 있는 객체는 작게 그리고 가까운 객체는 크게 그려진다 * 멀리있는 객체는 카메라 중심선 근처에 위치한다   **투영 좌표계**     * 카메라의 중심선을 기준으로 상하좌우를 하나의 좌표계로 표현     4차시\_3D 그래픽스 기초 ( 변환 파이프라인 - 02)  **투영 변환**   * 모든 객체의 점에 대해 원근 투영 나누기를 해서 카메라 좌표계에 Z=1인 평면에 그린것   **화면 좌표 변환**   * 투영 좌표 공간을 화면으로 매핑 * 뷰포트: 투영 좌표 공간이 실제로 매핑될 화면 영역   **벡터**   * 점 또는 방향과 크기를 나타낸다 * 한 점은 항상 벡터이다 * 점 벡터는 원점에서 방향과 크기를 가지고 정의될 수 있다. * 시작 점이 원점이 아니면 위치를 나타낼 수 없다 * 단위 벡터: 크기가 1인 벡터 * 벡터 정규화: 크기가 1이 아닌 벡터를 단위 벡터로 만드는 것  1. 프레임워크 수정   문제 1번: 코드가 계층구조를 무시하고 배열로 뼈를 불러온다  문제 2번: 지금 단계에서는 사용하지 않지만 이후에 사용할 함수/변수/클래스 등이 많다  이러한 문제들을 혼자 해결하는 것 보다는 회의를 통해 정하는 것이 맞다고 생각하여 기반 프로젝트에서 수정사항이 있는 부분을 찾아 두고 다음 회의에 어떤 방식으로 처리할지 생각을 해두었다.   1. 홍예나(클라이언트)   애니메이션 블렌딩 구현의 방향성을 잡음  CAnimation의 advanceTime 함수에서 블렌딩을 해야 하는 경우와 아닌 경우를 나누어서 실행되도록 구현  advanceTIme함수는 애니메이션 track의 keyframe 사이의 transform 정보를 보간하여 보여지도록 계산하고 반환해주는 역할을 하는 함수이다  애니메이션 블렌딩을 하는 목적은 애니메이션1의 실행 도중 애니메이션2로 변경되어야 하는 경우에 애니메이션1의 변경 지점과 애니메이션2의 시작 지점 사이의 연결되는 transform들을 만들어주어 보다 자연스러운 움직임을 구사할 수 있도록 하는 것이다  따라서 keyframe 사이의 transform 정보를 보간하는 역할을 하는 함수인 advanceTime 함수에 애니메이션 블렌딩을 해야하는 경우와 아닌 경우로 나누면 애니메이션 블렌딩을 구현할 수 있을 것이라고 생각한다    현재 player에 isMove라는 애니메이션의 변경 여부를 확인할 수 있는 멤버 변수를 생성하였다.    상, 하, 좌, 우 키 입력이 받아질 때마다 isMove가 ture가 되고  True가 된 경우 애니메이션1의 변경 지점과 애니메이션2의 시작 지점의 transform 정보를 받아와서 보간을 진행하도록 코드를 작성하였다  추가로 구현해야 할 사항들은   1. Interpolate의 t를 elapsedtime에 근거하여 증가하도록 수정 2. Animation set에서 뛰기, 점프 애니메이션을 어떻게 가져올지 지정 3. 두 애니메이션의 사이에 보간된 transform 정보가 얼마나 있어야 하는가?   등이 있다. | | | |
| **다음주 할 일** | | | |
| 1. 신동엽(서버)   - 기존 프레임워크와 비교하며 Workerthread로 다시 들어가는지 확인,  - Send Recv  - Echo 섭 테스트  - C++ 가상함수 다시보기   1. 정경서(클라이언트)  * 애니메이션 관련 오류 수정 * 디퍼드 렌더링 및 개념 공부  1. 홍예나(클라이언트)  * 애니메이션 블렌딩 구현 * 1/30 ~ 2/2 일본 여행 | | | |
| **비고** | | | |
|  | | | |