|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 5주차 | **기간** | 2020.04.13~ 2020.04.19 | **지도교수** | (서명) |
| 이번주 한일 요약 | 김영완:  충돌처리 구현, 게임 내 폰트출력을 위한 폰트매니저 구현  박건호 : 로비 씬, 로그인 씬 텍스처 이미지 구현, 버튼 이미지 구현  박정만: 고속으로 회전하는 오브젝트에 대한 물리처리 구현 | | | | |

<상세 수행내용>

김영완:

-충돌처리-

캐릭터의 이동에 관한 물리는 아직 해결을 못하여 미루고 있음(계산식 상 오차로 인해),

충돌 처리는 추가하였음 -> 충돌 체크가 확인될 경우 두 물체의 질량과 속도를

비교해서 질량이 더 적은 물체가 더 많이 움직이도록 velocity의 값을

변경함, 방향은 velocity진행의 반대방향으로 작용함.

프레임 or 속력의 차이로 물체가 겹치게 되는 경우,

한번 충돌처리를 적용해도 다음 프레임에 충돌처리를 적용해야 하는 문제가 있었다.

충돌처리 후에는 두 물체를 떨어뜨려 놓아서 중복되는 충돌체크를 없앨 필요가 있음.

해결방법으로 멤버로 이전 프레임의 위치 값을 갖도록 하고나서

충돌 발생 시 현 위치를 이전프레임의 위치로 변경한 뒤 충돌처리를 적용하도록 하였다.

-폰트 매니저 추가-

방목록에 대한 정보 및 로그인에서 쓰일 텍스트를 출력하기 위해서 폰트가 화면에

나올 필요가 있음. 그래서 폰트 이미지들을 담고 있는 스프라이트 식의 이미지 파일을

이용하여 화면에 출력하는 방식을 선택하였음

createManager와 같이 단 한번만 생성하고 프로그램이 끝날 때까지 존재하며 모든 씬의

폰트들을 출력하는 역할을 담당할 폰트매니저를 구현, 폰트매니저는 폰트를 출력하는 셰이더를

멤버로 갖고 있다. 폰트 셰이더는 문자열, 문자열의 크기, 색깔, 위치 값을 받고

해당 위치에 색상, 크기가 적용된 문자열의 문자를 하나씩 스프라이트 이미지에 맞춰서

화면에 출력한다. 이때 정점 버퍼는 텍스트가 바뀔 때마다 position, uv를 수시로 바꿔줘야

하므로 동적으로 다루게 하였다. 때문에 디폴트 힙 없이 오로지 업로드 힙으로만 사용.

매니저형태로 만든 이유는 각 씬에 셰이더로 존재하게 할 경우 매 씬 생성 시마다 생성을 해야하는데 동일한 폰트 이미지와 루트 시그너처를 사용하기 때문에 하나만 만들어서 그 객체를 공유하고 사용하는 것이 더 효율적이라고 생각하였음.

박건호 : 게임에 사용할 로그인 화면과 로비 화면을 텍스처 이미지로 구현을 하기로 정해서 로그인과 로비에 사용할 텍스처 이미지를 제작 하였다. 로그인 화면에서는 우리 게임이름의 로고를 크게 만들고 아이디와 비밀번호를 받아올 수 있는 창을 생각해서 만들었다. 로비 화면에서는 현재 게임의 접속중인 플레이어를 볼 수 있는 유저목록 창과 현재 대기중이거나 게임중인 방을 볼 수 있는 방 목록을 볼 수 있는 로비 화면으로 구성을 하여 제작하였다.

박정만:

클라이언트 물리처리 작업 수행: 회전 운동에 대하여 기존의 회전 운동을

적용하였지만 고속으로 회전하는 오브젝트에 한해서 방향 전환이 되지

않았으며 오브젝트의 방향만 변하고 계속 직선운동을 하고 있었기에

클라이언트와 함께 계속 회의하며 작업을 수행하였습니다. 또한 이번

작업을 성공적으로 마치면 드리프트에 대하여 새로 논의할 필요가 있다

판단하여 최대한 빠르게 완수를 해야될 것 같습니다.

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | | |
| **해결방안** |  | | |
| **다음주차** | 6주차 | **다음기간** | 2020.04.20 ~ 2020.04.26 |
| **다음주 할일** | 김영완: 씬들의 전환 구현 및 물리 구현.  박건호: 사운드 구현  박정만: 드리프트 로직 새로 구현, 서버 프레임워크 보완 및 데이터베이스 제작 시작 | | |
| **지도 교수**  **Comment** |  | | |