Singularity

김홍준 기능성게임프로그래밍

Singularity 는 적의 탄을 피하여 최대한 오래 살아남는 것이 목표인 게임이다. 이 레포트는 해당 게임에 대한 제작 과정 및 지금까지 한 개발 및 구현 중에 어려웠던 부분(알게 된 부분)을 설명한다. 기획하게 된 계기로는 Waves, Soundodger 라는 두 가지 게임 분석에서 시작했다. 두 게임의 공통점으로 첫 번째는 필드에서 적의 탄막을 피하며 최대한 살아남는 것이 목표인 점, 두 번째는 슬로우 모션 등 특수 능력으로 게임의 진입 장벽을 낮추었다는 점이다. 이 공통점들로 지금의 Singularity 게임을 설계하는 발판이 되었다. Sinugularity 의 기획은 한 필드에서 아이템을 주워 게임 클리어 조건을 달성하는 형식의 샘플 게임을 확장하자는 발상에서 시작했다. 해당 샘플 게임과 분석한 게임의 공통점은 한 필드에서 진행된다는 것이었으며 여기서부터 확장을 시작하였다. 그리고 해당 샘플 게임과의 차별점을 두기 위해 첫 번째로는 적의 탄막을 피하여 클리어 하는 게임을 만들려 하였고 두 번째는 탄막 게임은 진입 장벽이 높은 편에 속하기 때문에 특수 능력을 부여하여 쉽게 적응할 수 있도록 만들도록 기획하였다. 기초적인 게임을 구현할 때 크게 구현한 항목들은 적, 탄막, 클리어 조건 및 탈락 조건, 특수 능력인 슬로우 모션 및 대시까지 총 4 개이다. 첫 번째로 적은 샘플 게임에서 플레이어에게 있는 움직임을 담당하는 코드 및 탄막을 관리하는 탄막 매니저 객체를 가지게 하여 정상적으로 움직이거나 플레이어에게 탄막을 쏠 수 있도록 구현하였다. 두 번째로 탄막은 속도, 방향, 플레이어와의 상호작용을 구현하였다. 세 번째로 열쇠를 먹으며 열쇠를 먹은 개수를 세어 클리어 조건을 구현하고 일정 이상의 시간 및 탄막 맞은 횟수를 계산하여 달성률이 일정 퍼센티지 이하로 내려 갈 경우 탈락하는 조건을 구현하였다. 마지막 네 번째는 특수 능력들 중 대시를 플레이어가 움직이는 코드에서 일정 시간동안 조작을 하지 못하게 하고 속도를 높이도록 만들었으며 이 동안 탄막에 부딪혀도 부딪히지 않도록 구현하였으며 슬로우 모션은 유니티의 TimeScale 을 조정하여 구현하였다. 추가로 샘플 게임에서 아이템을 생성하여 상호작용하는 부분을 이용하여 각각의 특수 능력을 채워주는 아이템을 생성하여 그 수단으로만 특수 능력을 다시 채울 수 있도록 구현하였다. 위와 같은 과정을 거쳐 기초만 만든 게임을 만들었으나 적이 단순히 움직이며 탄막을 주는데 적과의 거리와 가까워도 메리트는 없고 리스크만 증가하는 현상이 생겼다. 그 해결법으로 적과의 거리가 가까우면 리스크에 대한 보상을 주도록 적과의 거리에 따라 점수 계산을 하도록 만들었지만 전체 적의 평균값으로 하니 체감이 안되어 적이 일정 거리 이상 플레이어에게 못 붙도록 움직이게 하였다. 여기서 확장으로 적이 바다에 빠졌을 때 점수를 얻는 것을 생각하였다. 한편 스테이지로 나누어 진행하니 전 스테이지에 유지하던 긴장이 풀려 다음 스테이지에서 체감상 어렵게 느껴졌다. 이에 따라 키를 먹어 스테이지를 클리어하는 것에서 끝까지 살아남는 endless 게임으로 바꾸었다. 그렇게 게임을 완성하고 유저 피드백으로 받은 유의미했던 피드백 사항으로는 플레이어가 적을 미는 것이 힘든 점, 목숨 회복 수단의 부재, 아이템의 가시성, 메뉴와 게임 Scene 의 전환 간에서의 옵션에 관한 버그들이었다. 플레이어가 적을 미는 것이 힘든 점에 대해서는 최적화를 위해 캡슐이 아닌 원형 콜라이더의 위치를 바꾸어 해결하였고 목숨 회복 수단의 부재로는 목숨 회복 수단이 있으면 게임 난이도가 너무 쉬워지지 않았나 싶었지만 구현하고 보니 난이도는 같지만 진입장벽을 낮추는데 도움이 되었다. 아이템의 가시성은 아이템 위에 아이콘을 띄움으로써 가시성이 좋도록 구현하였다. 메뉴와 게임 Scene 의 전환 간에서의 옵션에 관한 버그들은 Scene 을 삭제하기 전 DontDestroyOnLoad 함수를 통해 필요한 정보를 Scene 끼리 교환할 수 있도록 구현하여 해결하였다. Singularity 게임을 만들면서 어려웠던 점(알게 된 점)은 콜라이더를 통해 작동하는 OnTriggerExit, OnTriggerEnter 함수가 조건에 따라 작동하지 않는다는 점을 알았으며 해당 문제를 OnTriggerStay 함수에서 거리에 따라 Exit, Enter 기능을 작동하도록 만들어 해결하였다. 또 어려웠지만 해결 못한 문제는 각각의 Script 코드에서 중복된 코드가 너무 많아 Script 를 만들 때 Entity 단위가 아닌 Entity 가 가질 수 있는 Module 단위로 구현하며 복합적인 기능은 아래 기능부터 만든 후 상호작용하도록 만들어야 한다라는 생각을 했다. 마지막으로 피드백 부분에서는 유의미한 피드백을 고르기 위해서는 보기만 할 뿐만 아니라 실제로 구현하고 해보아야 된다는 것을 배웠다. 또한 게임에서 피드백이 추상적이지 않게 게임에서 게임 요소마다 용어로 정의하고 설명하는 것의 중요성을 알았다.