|  |  |
| --- | --- |
| 항목 | 내용 |
| 게임 제목 | 미정 |
| 개발 목표 | 빠른 템포와 전략적 전투 시스템을 제공하여 높은 몰입감을 구현하는 것이 목표입니다. 플레이어에게 강렬하고 즉각적인 피드백을 제공하여 몰입도를 높이기 위해, 빠른 전투 템포와 정밀한 조작이 필수적입니다. |
| 장르 | 3D TPS, 액션, 슈팅, 싱글 플레이 |
| 타겟 플랫폼 | PC를 우선 개발 플랫폼으로 설정하며, 향후 콘솔로의 확장 가능성도 검토합니다. |
| 타겟 사용자 | 18~35세 남성, FPS 및 액션 게임 팬 |
| 주요 경쟁작 | Warhammer 40,000: Space Marine 2, HELLDIVERS™ 2, DOOM ETERNAL. 이들 게임은 유사한 슈팅 및 액션 시스템을 제공하며, 특히 DOOM ETERNAL의 빠른 템포 전투 시스템이 참고할 가치가 있습니다. |
| 핵심 특징 | 빠른 템포 전투, 수직적 맵 디자인, 다양한 무기 시스템. 게임 제목은 아직 미정이지만, 이러한 핵심 특징을 반영할 필요가 있습니다 |

**1. 프로젝트 개요**

**Unity 구현**:

* 플랫폼 호환성을 고려하여 PC 개발을 우선적으로 진행하며, 필요 시 콘솔 버전 확장을 위한 컨트롤러 지원도 구현할 수 있습니다. PC와 콘솔의 차이를 고려한 조작 최적화가 필요합니다.

|  |  |
| --- | --- |
| 항목 | 내용 |
| 세계관 설정 | 현대적인 연구소가 무기화된 생명체 연구로 인해 파괴됩니다. |
| 스토리 요약 | 연구소에서 통제 불능의 실험체가 탈출하면서 주인공은 연구소를 탈출하려고 시도합니다. |
| 캐릭터 설정 | 주인공은 보안 책임자로서 생존과 임무 수행 능력이 뛰어난 인물입니다. 상황이 악화된 후 연구소를 탈출하기 위해 전문적인 기술을 사용합니다. |
| UX 목표 | 높은 난이도와 긴장감을 제공하는 전투와 탐험 시스템을 목표로 합니다. 플레이어는 스테이지별로 점점 더 도전적인 상황에 맞닥뜨리게 됩니다. |

**2. 게임 컨셉**

**Unity 구현**:

* Cinemachine을 이용해 긴장감 넘치는 연출을 극대화하며, 타임라인 기능으로 스토리 진행을 동적으로 연출합니다.

**3. 핵심 게임 시스템**

|  |  |  |
| --- | --- | --- |
| 기능 | 설명 | Unity 구현 방법 |
| 이동 | 캐릭터의 8방향 이동 및 카메라 기반 이동 | **CharacterController**로 물리 기반 이동 구현 |
| 공격 | 기본 무기 공격 및 특수 무기 사용 | **Raycasting**을 사용해 적과의 충돌 판정 처리 |
| 대쉬 | 짧은 거리 빠른 이동 및 회피 가능 | **Velocity** 조작으로 대쉬 구현, **Cooldown** 시스템 도입 |
| 더블 점프 | 특정 구역에서 더블 점프 가능 | **RigidBody** 조작을 통한 점프 구현 및 두 번째 점프 판단 코드 |
| 무기 전환 | 여러 무기를 사용하여 다양한 전투 스타일 구현 | **Inventory 시스템**을 사용해 무기 변경 시스템 구현 |

* **플레이어 시스템**
* **적 AI 시스템**

|  |  |  |
| --- | --- | --- |
| 기능 | 설명 | Unity 구현 방법 |
| 플레이어 추적 | 적이 플레이어를 인식하고 추적 | **NavMesh** 및 **AI 경로 탐색**을 통해 경로 추적 구현 |
| 공격 | 적이 일정 거리 내에 있을 경우 공격 | **Animator** 및 **Trigger**를 사용해 공격 애니메이션 구현 |
| 회피 및 방어 | 적이 일정 확률로 플레이어의 공격을 회피 | **\*\*Finite State Machine(FSM)\*\***을 사용해 다양한 상태 정의 |

|  |  |  |
| --- | --- | --- |
| 시스템 | 설명 | Unity 구현 방법 |
| 체력 및 방어구 | 플레이어와 적의 체력, 방어구 시스템 | **UI 체력 바, 방어구 시스템** 연동 |
| 탄약 관리 | 플레이어와 적의 탄약 제한 및 보급 시스템 | **Inventory 시스템**과 연동하여 탄약 소모 및 충전 구현 |
| 데미지 처리 | 각 무기별 데미지 계산 및 적 사망 처리 | **Collider**와 **Raycast**로 데미지 충돌 판정 |

* **게임 내부 시스템**

|  |  |  |
| --- | --- | --- |
| 기능 | 설명 | Unity 구현 방법 |
| 레벨 구성 | 던전형식으로 여러 방이 연결된 구조 | **ProBuilder**를 사용해 레벨 디자인 및 NavMesh 설정 |
| 스테이지 진행 | 적을 처치할 때마다 다음 방으로 진행 가능한 문이 열림 | **Trigger 시스템**으로 문 열림과 스테이지 클리어 조건 설정 |

* **레벨 디자인 및 진행 시스템**

**Unity 구현**:

* **플레이어 컨트롤 시스템**: **Animator**와 **CharacterController**를 사용해 자연스러운 움직임과 공격 애니메이션을 연동합니다.
* **적 AI 시스템**: **NavMesh Agent**와 **State Machine**을 통해 플레이어를 추적하고, 공격 시나리오를 다양하게 구현할 수 있습니다.

|  |  |
| --- | --- |
| 요구사항 | 설명 |
| 그래픽 요구사항 | 플레이어 및 적 모델링, 무기 텍스처, 환경 텍스처 등 고해상도 요구 |
| 사운드 요구사항 | 배경음악, 무기 효과음, 상호작용 효과음 등 세분화된 사운드 구현 |
| 플랫폼 요구사항 | 최소: i5 CPU, 8GB RAM, GTX 1060 / 권장: i7 CPU, 16GB RAM, RTX 3060 |

**4. 기술적 요구사항**

**추가 설명:**

* **그래픽 요구사항**은 각 오브젝트의 디테일한 텍스처와 모델링을 요구합니다. 이를 위해 고해상도 텍스처와 섬세한 조명이 필수적입니다.
* **사운드 요구사항**에서는 배경음악의 몰입감과 무기 효과음의 현실성을 중시하며, 플레이어의 상호작용에 따라 변화하는 음향 효과도 필요합니다.

**Unity 구현**:

* **HDRP**(고해상도 렌더 파이프라인)를 사용해 그래픽의 퀄리티를 높이고, **AudioMixer**를 통해 효과음과 배경음을 세분화하여 관리합니다.

**5. 스케줄 및 리소스**

|  |  |
| --- | --- |
| 항목 | 내용 |
| 개발 일정 | 3개월 동안 주별 목표 설정 후 진행 |
| 필요 인력 | 프로그래머 3명, 아티스트 3명, 기획 4명 |
| 개발 방법론 | Agile 방식의 Scrum 도입 |

|  |  |  |
| --- | --- | --- |
| 주차 | 주요 개발 목표 | 세부 내용 |
| 1주차 | 프로젝트 세팅 및 초기 컨셉 확정 | Unity 프로젝트 설정, 스토리 확정 |
| 2주차 | 기본 캐릭터 및 맵 구성 | 플레이어 이동 및 초기 적 설정 |
| 3주차 | 전투 시스템 개발 | 무기 시스템 구현 및 적 AI 설정 |

* **주간 일정표**

|  |  |  |
| --- | --- | --- |
| 위험 요소 | 설명 | 해결 방안 |
| 팀원 간 작업 충돌 | 각 팀원의 작업물이 통합되면서 충돌이 발생할 가능성 | \*\*버전 관리 도구(GitHub)\*\*를 사용하여 통합 및 충돌 방지 |
| 데이터베이스 관련 오류 | DB에서 데이터 이동 시 발생할 수 있는 오류 및 데이터 손실 | **SQL 쿼리 최적화**와 **에러 핸들링**을 통해 오류 최소화 |
| 일정 지연 | 예상보다 더 많은 작업 시간이 소요될 경우 | **Agile 방법론**을 통해 주기적으로 진행 상황 점검 및 조정 |
| 성능 문제 | 그래픽 및 AI 처리로 인해 성능 저하 발생 가능 | **프레임 최적화**와 **메모리 관리**를 통해 성능 향상 |

**6. 위험 요소 및 해결 방안**

**추가 설명:**

* **팀원 간 작업 충돌**: 여러 개발자와 아티스트가 협력하여 작업하는 과정에서, 버전 충돌이나 작업물 간 비호환성 문제가 발생할 수 있습니다. 이를 방지하기 위해 **GitHub**을 이용한 **분기(branch) 관리**와 주기적인 **코드 리뷰**를 진행하여 충돌을 조기에 발견하고 해결할 수 있도록 합니다.
* **데이터베이스 관련 오류**: 게임의 데이터 저장 및 관리가 중요하기 때문에, 데이터베이스 통신에서 발생할 수 있는 **데이터 손실**이나 **정합성 문제**를 대비하여 쿼리와 데이터 전송 과정을 세밀하게 검증합니다.

**Unity 구현**:

* GitHub과 같은 **버전 관리 시스템**을 사용하여 코드와 리소스 충돌을 최소화하고, **SQLite**를 이용하여 게임 데이터를 관리합니다. SQLite는 가볍고 빠르며, 싱글 플레이어 게임에 적합한 데이터베이스 솔루션입니다.

**7. 테스트 계획**

|  |  |  |
| --- | --- | --- |
| 테스트 종류 | 설명 | 테스트 기준 |
| 기능 테스트 | 모든 게임 기능이 의도대로 동작하는지 확인 | 버튼, UI, 전투, 이동 등의 작동 여부 확인 |
| 성능 테스트 | 게임이 원활하게 실행되는지, FPS가 일정하게 유지되는지 확인 | 60 FPS 이상 유지, 메모리 누수 여부 점검 |
| 프로그램 호환성 테스트 | 다양한 환경 및 플랫폼에서 게임이 실행되는지 테스트 | 최소 사양 및 권장 사양에서의 실행 여부 확인 |
| 사용자 피드백 테스트 | 실제 사용자의 피드백을 통해 UX 및 게임성 개선 | UX 만족도, 난이도 적정성 등 사용자 반응 측정 |

**테스트 계획 및 일정:**

* **1차 테스트**: 기능 테스트 및 초기 성능 테스트 (1개월 후)
* **2차 테스트**: 게임 중반부 기능 및 성능 최적화 테스트 (2개월 후)
* **최종 테스트**: 출시 전 호환성 및 사용자 피드백 반영 (프로젝트 종료 2주 전)

**추가 설명**:

* **성능 테스트**에서는 PC 최소 사양(i5, 8GB RAM, GTX 1060)과 권장 사양(i7, 16GB RAM, RTX 3060)에서 프레임 레이트와 CPU, GPU 사용량을 점검하여 최적화를 진행합니다.
* **사용자 피드백 테스트**는 베타 버전 테스트에서 실시간 피드백을 받아 UX와 난이도를 조정하는 과정입니다.

**Unity 구현**:

* **Profiler**를 통해 성능 문제를 분석하고, **PlayerPrefs**로 사용자의 피드백 및 테스트 결과를 저장하는 시스템을 구현합니다.

**8. 유지보수 계획**

|  |  |  |
| --- | --- | --- |
| 유지보수 항목 | 설명 | 대응 방안 |
| 기능 추가 | 추가적인 게임 모드나 스테이지 추가 | 업데이트 계획 수립 후 정기적인 업데이트 진행 |
| 버그 수정 | 사용자 피드백 및 발견된 버그 해결 | **버그 추적 시스템**을 통해 발견 즉시 패치 |
| 최적화 | 성능 개선 및 다양한 기기에서의 호환성 향상 | **GPU, CPU 사용량 최적화** 및 **메모리 사용량 개선** |

|  |  |
| --- | --- |
| 주기 | 항목 |
| 1개월 | 주요 버그 수정 및 작은 기능 업데이트 |
| 3개월 | 큰 기능 추가 및 확장팩 업데이트 |
| 6개월 | 새로운 모드, 스테이지 추가 및 성능 최적화 |

* **유지보수 주기**

**추가 설명:**

* **기능 추가**: 출시 후 새로운 무기, 캐릭터 스킨, 스테이지를 업데이트할 계획입니다. 이를 통해 유저들의 흥미를 유지하고 재방문율을 높일 수 있습니다.
* **버그 수정**은 사용자로부터 받은 피드백을 바탕으로, 버그를 추적하고 이를 빠르게 수정하여 게임의 완성도를 높이는 것이 목표입니다.
* **최적화**는 다양한 하드웨어 환경에서 안정적으로 게임을 실행할 수 있도록 **메모리 관리**와 **그래픽 최적화**를 중점으로 진행합니다.

**Unity 구현**:

* **Update Pipeline**을 통해 주기적인 업데이트를 계획하고, 각 패치에 대한 **버전 관리**를 철저히 진행하여 업데이트의 안정성을 확보합니다.

**9. 법적 및 윤리적 고려사항**

|  |  |  |
| --- | --- | --- |
| 항목 | 설명 | 대응 방안 |
| 저작권 문제 | 게임 내 사용된 모든 리소스에 대한 저작권 확보 필요 | 사운드, 텍스처, 모델링 리소스에 대해 정당한 라이선스 획득 |
| 개인정보 보호 | 개인정보를 수집하거나 저장하지 않음 | 개인정보 처리방침을 명시하여 사용자에게 투명하게 공개 |
| 사회적 영향 | 폭력적 콘텐츠에 대한 적절한 연령 제한 | 18세 이상 등급으로 분류하고 게임 내 폭력적 요소 조정 |

**추가 설명:**

* **저작권 문제**: 게임에서 사용되는 모든 그래픽, 사운드, 코드 라이브러리 등에 대해 라이선스를 확보하고, 타인의 저작물을 무단으로 사용하는 일이 없도록 주의합니다. 상업적 사용이 허가된 오픈소스 및 상용 리소스를 사용하는 것이 바람직합니다.
* **개인정보 보호**: 게임이 싱글 플레이 기반이므로 사용자의 개인정보를 수집하지 않지만, 추후 **온라인 기능** 추가 시 개인정보 처리 방침을 명확하게 작성하고 이를 사용자에게 공지할 계획입니다.
* **사회적 영향**: 게임 내 폭력적인 표현이나 잔인한 장면은 18세 이상 등급의 기준을 준수하도록 하고, 이러한 요소가 사회에 미치는 영향을 고려하여 연령 제한을 명확히 합니다.

**Unity 구현**:

* 상용 라이선스를 받은 **사운드와 텍스처**를 사용하고, **에셋 스토어**의 유료 에셋 사용 시에도 사용 범위를 철저히 확인합니다.

**10. 결론 및 추후 방향성**

**결론**

이 프로젝트는 3D 액션 슈팅 게임으로, 빠른 템포와 전략적인 전투가 주요 특징입니다. 전투 중 **다양한 무기 전환**과 **복잡한 맵 탐험**을 통해 플레이어에게 끊임없는 도전과 재미를 제공합니다. 또한, 클리어 타임을 비교함으로써 플레이어들에게 경쟁심과 도전욕구를 자극시킵니다. 그리고 지속적인 **업데이트와 유지보수 계획**을 통해 장기적으로 플레이어의 흥미를 유지하며, 게임의 퀄리티를 높일 수 있는 기반을 마련했습니다.

**추가 사항**

|  |  |  |
| --- | --- | --- |
| DLC | 새로운 스테이지 & 몬스터 | 새로운 무기 & 전투 메커니즘 |
| 지속적인 패치 | 버그 수정 | 밸런스 패치 |
| 멀티플레이 | PVP 모드 | PVE 협동 모드 |

* **추가 확장팩 개발**: 초기 출시 후 더 많은 스테이지와 몬스터, 그리고 새로운 전투 메커니즘을 추가하는 확장팩을 계획하여 플레이어들이 지속적으로 게임에 참여할 수 있도록 유도합니다.
* **커뮤니티 피드백 반영**: 플레이어들로부터 받은 피드백을 적극 반영하여 게임성 개선 및 신규 콘텐츠 추가를 고려해야 합니다. 이를 통해 사용자 만족도를 높이고 커뮤니티 형성에 기여할 수 있습니다.
* **온라인 기능 도입 검토**: 장기적으로 온라인 협동 모드나 PVP 모드를 추가하여 멀티플레이어로 확장할 가능성을 검토할 수 있습니다. 이는 게임의 수명 연장 및 플레이어 수 증가에 긍정적인 영향을 미칠 수 있습니다.