K-Sports 양궁 기획서

개요

해당 기획서는 K-Sports의 형태를 올림픽 체육과 같은 정형화된 체육 포맷에서 벗어나 스토리와 세계관을 유저가 플레이하고 있다는 느낌을 강조하는 기획이 될 것이다.

장르 설정은 액션 어드벤처 게임이 될 것이다.

주요 타깃은 액션 게임의 코드를 이해하고 액션의 쾌감을 즐기는 20대로 잡았다.

배경은 현대의 잠입액션(파크라이3, 크라이시스, 호크아이)을 기반으로 현대 도시에서 활과 다양한 특수화살을 이용한 게임이다.

1인칭 시점에서의 활을 이용한 슈터 장르이다.

기획의도

K-Sports와 XR 컨텐츠에 대한 관심 제고와 양질의 XR 컨텐츠 제작을 추진하기 위한 기획이다.

평소에 접할 기회가 적은 스포츠들을 콘텐츠화 함으로써, 사용자가 K-Sports를 친근하게 느낄 수 있게 한다.

개발방향

양궁이란 방향성은 유지한 채로 게임으로써의 다양한 경험을 제공한다.

게임으로써의 경험이란 스토리라인과 컨셉을 통해 플레이어가 세계관에 동화됨을 말한다.

팀프로젝트로서 팀장이 프로젝트의 필요에 의해 각 직군의 팀원에게 작업을 요청한다.

팀원들은 자신의 직군의 일이 너무 벅차다면 다른 팀원들에게 도움을 요청한다.

양궁 프리뷰

왼손잡이인지, 오른손잡이 인지 설정한 후 해당 손의 포지션을 UI로 안내한다.

화면과 사선으로 서 가슴, 어깨, 팔꿈치, 손의 위치가 잘 보이도록 한다.

활을 쏘는 자세를 잡은 후 활 시위를 잡은 손을 놓아서 활을 발사한다.

활을 잡은 손이 주먹을 쥐면 그 주먹의 방향을 통해 화살의 방향을 결정한다.

화살은 적의 체력을 감소시키는 역할을 함과 동시에 다양한 퍼즐요소를 특수화살을 통해 해결할 수 있도록 한다.

화살은 플레이어가 준 힘에 의해 일직선으로 날아가며 중력의 영향을 받는다.

전역 상수설정

난이도상수(비트플래그 enum으로 구현)

Easy = 1, Normal = 2(1 << 1), Hard = 4(1 << 2)

화살장전상태(enum으로 구현)

IDLE = 0, ArrowInHand = 1, LoadArrow = 2, DrawArrow = 3, ShootArrow = 4

적의 수(const int로 구현, 스테이지당 변수참조)

EnumyCount = 20(나중에 바뀌어도 됨)

프레임 수(const int로 구현, 최대 프레임 수 키넥트 인식 최대 프레임 수와 맞춤)

MaxFrameRate = 30

플레이어

배경설정

양궁선수였지만 국가대표에 지속적으로 뽑히지 못해 결국 군대를 갔다오고 나서 양궁계를 떠났다. 전에 같이 양궁을 했던 선배의 추천으로 해결사로 운호금융에 입사하게 된다.

애니메이션

애니메이션은 팔과 리깅하며 키넥트 위치에 팔 스켈레톤을 맞춘다.

부드러운 움직임을 위해서 화살장전상태 상수에 따라 팔의 움직임을 임의로 맞춘다.

모델링

1인칭 카메라 플레이어의 팔만 보임

전투 장갑 예시

1인칭 플레이인 만큼 플레이어의 모델링은 팔만 모델링하며 전투장갑을 낀 형태로 모델링한다.

오른손 버전 모델링을 만들고 미러링을 통해 왼손 버전을 구현한다.

적

적은 초반에 나오는 시큐리티 가드와 후반에 나오는 해결사(암살자)들로 나뉜다.

체력

시큐리티 가드: 8

해결사: 16

사용 무기

Sig P320


격발시 반동으로 슬라이드가 밀린다.


격발시 머즐 플래쉬

적들의 기본 사용 무기는 권총이며 모델은 Sig P320으로 한다.(총기 디자인은 저작권이 없고 총기 이름만 지적재산권이 인정되므로 베껴도 됨.)

권총의 데미지는 시큐리티 가드는 8, 해결사들은 16으로 한다.

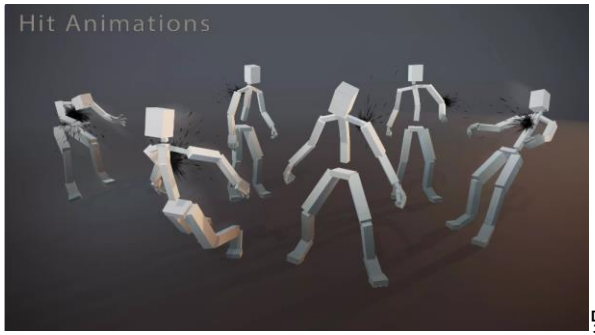
이펙트는 머즐 플래쉬 이펙트로 구현하며 격발시 슬라이드가 밀려야 한다.

애니메이션

적은 모델링된 전신을 리깅하며 총을 발사하는 애니메이션과 이동하는 애니메이션, 화살에 맞는 애니메이션과 쓰러지는 애니메이션을 구현한다.


총을 파지하는 모습

총을 발사하는 애니메이션은 총을 양손으로 파지하여 발사하는 식으로 동작한다.



맞는 애니메이션과 쓰러지는 에니메이션

피격 애니메이션은 적이 화살에 맞고 적이 뒤로 밀리는 식으로 동작한다.

쓰러지는 애니메이션은 적이 체력이 0과 같거나 떨어질 때 발생하게 앞으로 쓰러지는 식으로 동작한다.

이동하는 애니메이션


총을 파지하면서 이동하는 애니메이션

적이 이동하는 애니메이션은 위의 예시와 같이 달리고 난 뒤 플레이어의 부근에서 서서 총을 파지하거나 총을 파지한 상태로 이동하는 식으로 작동한다.

모델링

경비원 복장

해결사 정장

시큐리티 가드는 위의 예시된 경비원 복장을 입는다.

해결사는 위에 예시된 정장을 입는다.

게임 시작

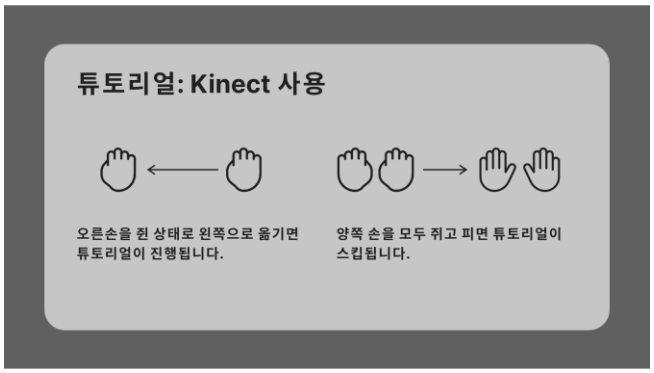
시퀀스

플레이어가 키넥트와 컴퓨터를 연결하고 게임을 켜면 인트로 화면이 등장하고나서 키넥트에 대한 가이드로 넘어가게 된다. 이때 플레이어가 어떤 식으로 게임의 UI와 상호작용할 수 있는지 알려주는 가이드가 등장하는데 키넥트를 통해 UI와 상호작용하는 법을 모르는 사람은 이 과정을 진행하며, 상호작용 방법을 아는 사람들은 이 과정을 스킵 모션을 통해 스킵할 수 있다.

키넥트 사용에 대한 가이드를 진행 혹은 스킵하면 주로 사용하는 손에 대한 설정 페이지로 넘어가게 되는데 이때 주로 사용하는 손을 번쩍 들면 이에 대한 설정이 된다.

UX/UI

인트로 화면 예시



키넥트 가이드 예시

주로 사용하는 손에 대한 설정

인트로 화면은 기본적으로 LEVIATHAN 로고에 글리치 효과를 주는 식으로 진행되며 키넥트에 대한 가이드는 어떻게 상호작용하는지에 대한 가이드로 진행된다.

키넥트 사용에 대한 가이드는 위의 방법으로 스킵 혹은 진행된다.

주로 사용하는 손에 대한 UI는 사람이 손을 드는 아이콘을 이용하여 가이드한다.

주로 사용하는 손에 대한 설정을 완료하면 게임 메뉴 씬으로 넘어간다.

게임 메뉴

시퀀스

게임 메뉴는 게임 시작 다음 씬으로 게임 시작과 게임 종료 버튼을 배치한다.

처음 게임 메뉴 씬에서 게임 시작을 선택한다면 난이도 설정 화면으로 넘어간다.

-> 게임 종료 버튼을 누른다면 게임이 종료된다.

난이도 설정 화면에서 난이도를 선택하면 튜토리얼 씬으로 넘어간다.

-> 뒤로 가기 버튼을 누른다면 게임 메뉴 씬으로 돌아간다.

UX/UI

메인메뉴

난이도 설정 화면

메인메뉴는 왼쪽에 로고를 배치하고 오른쪽에 START, SETTINGS, QUIT 버튼을 배치하며 폰트는 HACKED로 한다.

난이도 설정 화면에선 왼쪽에 Difficulty(폰트: 사라사고딕)라는 타이틀을 스트로크 효과로 표현하고 난이도 설정 버튼을 오른쪽에 배치한다. (폰트는 HACKED)

Pause

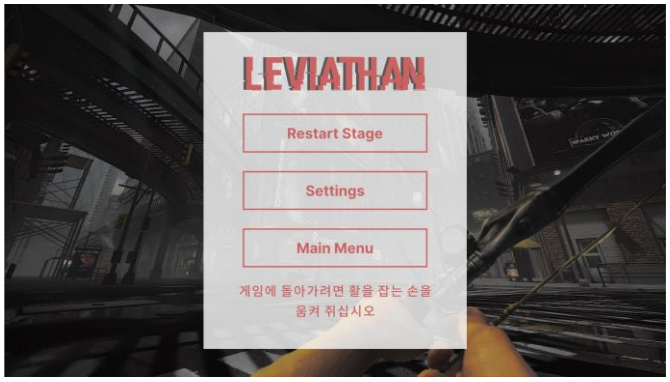
시퀀스

게임 중 활을 잡은 손(주로 사용하는 손의 반대 손)을 내려 놓으면 게임이 중단되며 Pause(일시정지) 화면이 출력된다.

-> 일시정지 화면에서 주로 사용하는 손으로 버튼을 선택하고 3초간 기다리면 해당 버튼의 기능이 실행된다.

-> 활을 잡은 손을 올리고 활을 잡는 자세를 취하면 다시 게임화면으로 돌아간다.

UX/UI



Pause 화면

Pause 화면에서는 게임 화면이 어두워지며 Pause UI가 출력된다.

Pause UI는 중앙에 배치되며 위에서 아래로 게임 로고, Restart Stage 버튼, Settings 버튼, Main Menu 버튼, Pause 화면에서 벗어날 수 있는 방법에 대한 가이드 순으로 출력한다.

활과 화살

화살 궤적

게임 내에서의 화살 궤적은 일직선으로 날아가며 활공 임계치에 도달하면 Destroy 함수가 작동하며 씬 내에서 사라진다.

활공 임계치 = 화살장전상태

조준 방법

1. 주로 사용하는 손을 화살을 잡는 손, 그 반대손을 활을 잡는 손으로 설정한다.
2. 화살을 잡는 손을 허리 옆에서 쥐어서 활을 잡는 손 근처에 두면 화살이 활에 장착된다.
3. 화살이 장착된 상태에서 화살을 잡는 손을 몸쪽으로 당기면 활이 당겨진다.
4. 활이 당겨진 상태에서 화살을 잡는 손을 놓으면 화살이 발사되며 활이 당겨진 정도에 따라 화살에 주는 힘을 설정한다.
5. 화살을 발사한 후 다시 장전이 가능하다.

화살 장전 상태(enum으로 구현)

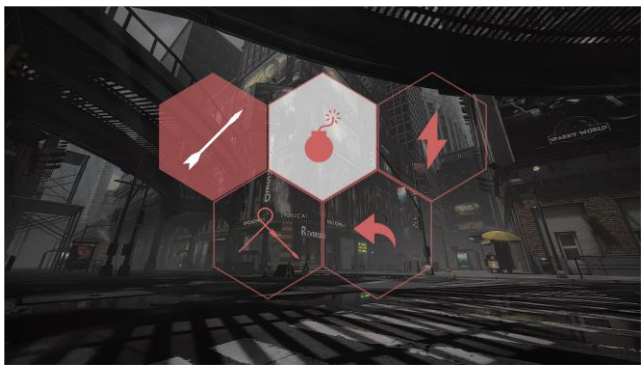
IDLE: 화살이 장전되지 않은 상태

Select: 화살을 선택한 상태

Draw: 화살을 활에 당긴 상태

Shoot: 화살을 잡는 손을 놓아서 발사하는 상태

화살 변경



화살 변경 장면

활을 잡는 손으로 활을 잡는 자세로 화살을 잡는 허리 옆에 두고 아래로 놓으면 게임 화면이 중

단되며 화살 변경 화면으로 넘어간다.

화살 변경 UI의 버튼 모양은 헥사곤이며 격자 무늬로 배치한다.

현재 쓰고 있는 화살 아이콘 격자는 투명 레드톤으로 채우고 선택된 화살 격자는 투명 화이트톤과 투명 레드톤 스트로크 다른 화살 아이콘 격자는 투명 레드톤 스트로크로 설정한다.

주로 사용하는 손으로 플레이어가 원하는 화살을 선택해서 손을 멈춘 상태로 2초를 기다리면 꽉 채워지는 애니메이션과 함께 해당 화살이 선택되며 게임 플레이 화면으로 돌아간다.

뒤로 가기 버튼을 선택하면 화살 변경 없이 게임 플레이 화면으로 돌아간다.

활의 모델링

컴파운드 보우

활은 컴파운드 보우를 베이스로 모델링된다.

활의 애니메이션

화살의 종류

일반 화살: 일반 화살은 기본적으로 제공되는 화살이다.

- ➔ 타격 데미지: 16 / 난이도상수
- ➔ 탄창: 무제한(Int32.MaxValue로 구현)

폭발 화살: 폭약과 뇌관이 화살촉에 장착된 화살로 닿으면 폭발하며 스플래쉬 데미지를 준다.

- ➔ 타격 데미지: 일반 화살 / 2
- ➔ 스플래쉬 데미지: 256 / 난이도상수

➔ 탄창: 32 / 난이도상수

전기 화살: 닿으면 전류를 발산하는 화살로 더 높은 타격데미지를 가진다..

➔ 타격 데미지: 일반 화살 * 2

➔ 탄창 32 / 난이도상수

화살의 모델링

senwick

일반 화살촉

화살 깃

폭발 화살촉

일반 화살 촉은 위의 사진과 같은 속이 비어 있게 디자인한다.

화살 깃은 위의 사진과 같이 돌고래 지느러미 모양으로 디자인한다.

폭발 화살촉은 위의 사진과 같이 뇌관과 장약이 돌출되게 디자인한다.

전기 화살촉은 일반화살에 전기 이펙트만 흐르만 방식으로 디자인한다.

화살은 각 화살촉을 배치하고 화살깃과 이어 붙히는 방식으로 제작한다.

In Game UI/UX

Game Play

게임플레이 화면

게임플레이 화면은 플레이어가 활을 든 모습을 기본으로 한다.

게임플레이 화면 중앙에는 조준선이 위치한다.

게임플레이 화면 왼쪽 상단에는 메인 미션과 서브 미션의 내용이 출력된다.

오른쪽 하단에 화살의 종류와 화살 개수를 표시한다.

Dialogue

Dialogue 예시

대사가 전달되는 방식은 하단에 텍스트를 출력하는 방식으로 진행된다.

폰트는 Sarasa Gothic이며 그림자 효과를 줘서 가시성을 높인다.

시나리오

튜토리얼

튜토리얼 화면

튜토리얼 씬은 난이도 설정 후에 시작된다.

장소는 운호금융의 훈련장이며 활 사용법, 특수 화살 사용법, 화살을 이용해 퍼즐 푸는 방법을 가이드한다.

튜토리얼 모든 화살은 무한으로 제공된다.

시퀀스

1. 튜토리얼 씬으로 넘어가며 1인칭 시점으로 유저 플레이가 진행된다.

2. 유저는 훈련교관의 지시에 따라 훈련사로에서 훈련한다.

-> 활과 화살에 대한 가이드

-> 특수화살에 대한 가이드

-> 퍼즐에 대한 가이드

3. 모든 튜토리얼을 끝마치면 스테이지 1에 진입하게 된다.

스테이지 1

스테이지 1 화면

스테이지 1 씬은 튜토리얼 씬이 모두 진행된 후 시작된다.

스테이지 1의 임무는 태산은행의 지부장을 제거하는 것이다.

시퀀스

1. 화면을 블러 처리하고 스테이지 1이라는 글자를 출력한다.

2. 스테이지 1의 임무를 진행한다.

-> 운호금융의 경쟁 은행인 "태산은행의 지부장을 제거하라"는 임무를 받으며 임무 목표를 왼쪽 상단에 출력한다.

-> 태산은행에 진입하며 앞에 있는 모든 적들을 활로 제거한다.

-> 지부장실에 진입하여 지부장을 제거한다.

-> 미션 클리어라는 메시지가 출력된다.

3. 스테이지 2로 넘어간다.

스테이지 2

스테이지 2 화면

스테이지 2 씬은 스테이지 1 씬이 모두 진행된 후 시작된다.

스테이지 2의 임무는 우신은행의 기밀을 훔쳐오는 것이다.

시퀀스

1. 화면을 블러 처리하고 스테이지 2라는 글자를 출력한다.

2. 스테이지 2의 임무를 진행한다.

-> "우신은행의 기밀을 훔쳐오라"는 임무를 받으며 임무 목표를 왼쪽 상단에 출력한다.

-> 우신은행 옆 건물에서 로프를 타고 우신은행에 진입한다.

-> 지하 서버실로 진입하는데 서버실 앞 보안문을 전기화살을 이용하여 망가뜨린 후 진입한다.

-> 무선 USB를 통해 태산은행 서버에 기밀을 전송한다.

-> 전송되는 동안 밀려오는 적과 싸우며 모든 적을 제거하면 미션 클리어라는 메시지가 출력된다.

3. 스테이지 3로 넘어간다.

스테이지 3

스테이지 3 화면

스테이지 3 씬은 스테이지 2 씬이 모두 진행된 후 시작된다.

스테이지 3의 임무는 금산분리 부활을 주장하는 국회의원을 제거하는 것이다.

시퀀스

1. 화면을 블러 처리하고 스테이지 3라는 글자를 출력한다.

2. 스테이지 3의 임무를 진행한다.

-> "금산분리 부활을 주장하는 국회의원을 동료와 함께 제거하라"는 임무를 받으며 목표를 왼쪽

상단에 출력한다.

-> 의원사무소에 진입하며 앞에 있는 모든 적들을 활로 제거한다.

-> 의원실에 진입하여 국회의원을 제거하고 나서 같이 온 동료가 총을 겨누며 "잘가 회사에서 고마웠다고..."라는 대사가 출력되면서 총을 쏜다.

-> 숨어있던 의원의 비서관이 나오고 어두운 화면으로 페이드 인된다.

3. 스테이지 4로 넘어간다.

스테이지 4(국회의원 비서관이 조력자)

스테이지 4의 장면

스테이지 4 씬은 스테이지 3 씬이 모두 진행된 후 시작된다.

스테이지 4의 임무는 배신한 회사에게 복수하는 것이다.

시퀀스

1. 화면을 블러 처리하고 "스테이지 4: 복수"라는 글자를 출력한다.

2. 스테이지 4의 임무를 진행한다.

-> "회사에게 복수하라"라는 목표를 왼쪽 상단에 출력한다.

-> 운호금융 건물에 진입하며 모든 적들을 제거한다.

-> 운호금융 회장실에 진입하여 회장을 제거한다.

-> 경찰이 들이닥치며 유저 아바타를 체포하려고 한다.

-> 유저 아바타는 순순히 투항하며 연행된다.

3. 엔딩으로 넘어간다.

엔딩(탈출 엔딩으로 변경)

엔딩 씬은 스테이지 4 씬이 모두 진행된 후 시작된다.

플레이어가 들이닥친 경찰에게 투항하는 것으로 엔딩 씬이 마무리된다.

시퀀스

1. 엔딩 크레딧이 출력된다.

2. 엔딩 크레딧이 모두 출력된 뒤 게임 메뉴 선택 씬으로 진입한다.

세팅

시퀀스

메인 메뉴나 Pause 화면에서 Settings 버튼을 선택하면 세팅 씬으로 넘어간다.

세팅에는 주로 사용하는 손에 대한 설정과 난이도 설정이 포함된다.

설정을 마치고 뒤로가기 버튼을 누르면 설정이 적용된다.

UI/UX

세팅 화면 UI/UX

세팅 화면 UI는 왼쪽 상단에 뒤로가기 버튼과 SETTINGS 텍스트가 출력되며 중앙에는 순서대로 주로 사용하는 손에 대한 설정과 난이도 설정의 설명 텍스트와 아이콘들이 배치된다.

Kinect 사용에 대한 가이드

시퀀스

1. 키넥트 사용에 대한 가이드 내용이 화면에 출력된다.

-> 본격적인 가이드 시작에 앞서 주로 사용하는 손에 대한 설정을 한다.

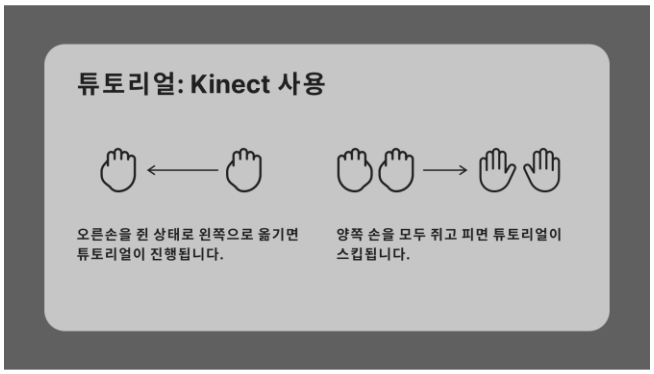
-> 가이드의 내용은 어떻게 UI 버튼을 작동하는지에 대한 것이다.

2. 플레이어는 키넥트로 UI와 상호작용하는 방법을 배운다.

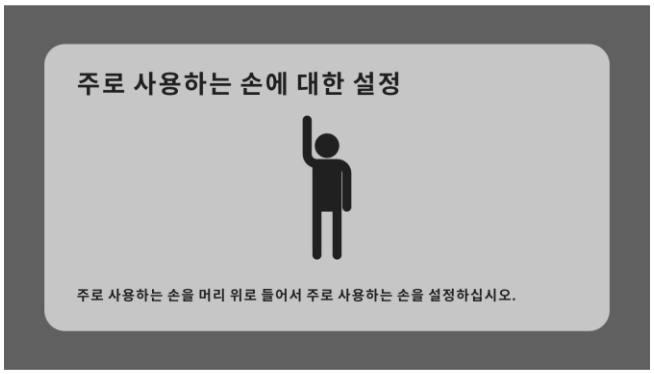
3. 모든 가이드를 배우며 상호작용을 숙지한다.

4. 게임 메뉴 선택으로 넘어간다.

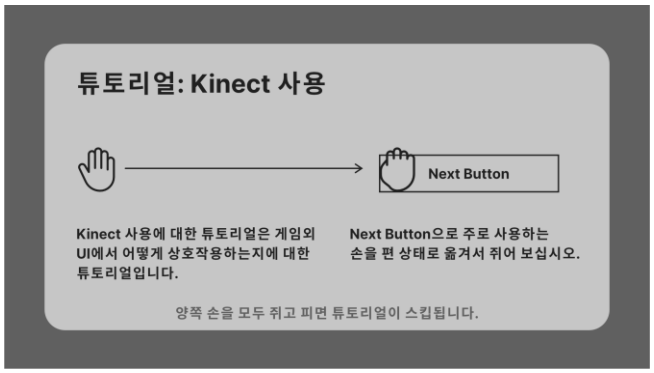
UI/UX



처음 키넥트 가이드 화면



주로 사용하는 손에 대한 설정



키넥트 가이드 화면

코딩가이드라인

코딩스탠다드

포프 킴의 C# 코딩 스탠다드를 따른다.

Ref: https://docs.popekim.com/ko/coding-standards/csharp

주의점

1. 하나의 enum 값으로 확인하는 분기라면 if문 보다 switch-case문을 쓴다.

Ref: https://www.youtube.com/watch?v=1Qg-dIh2qGQ

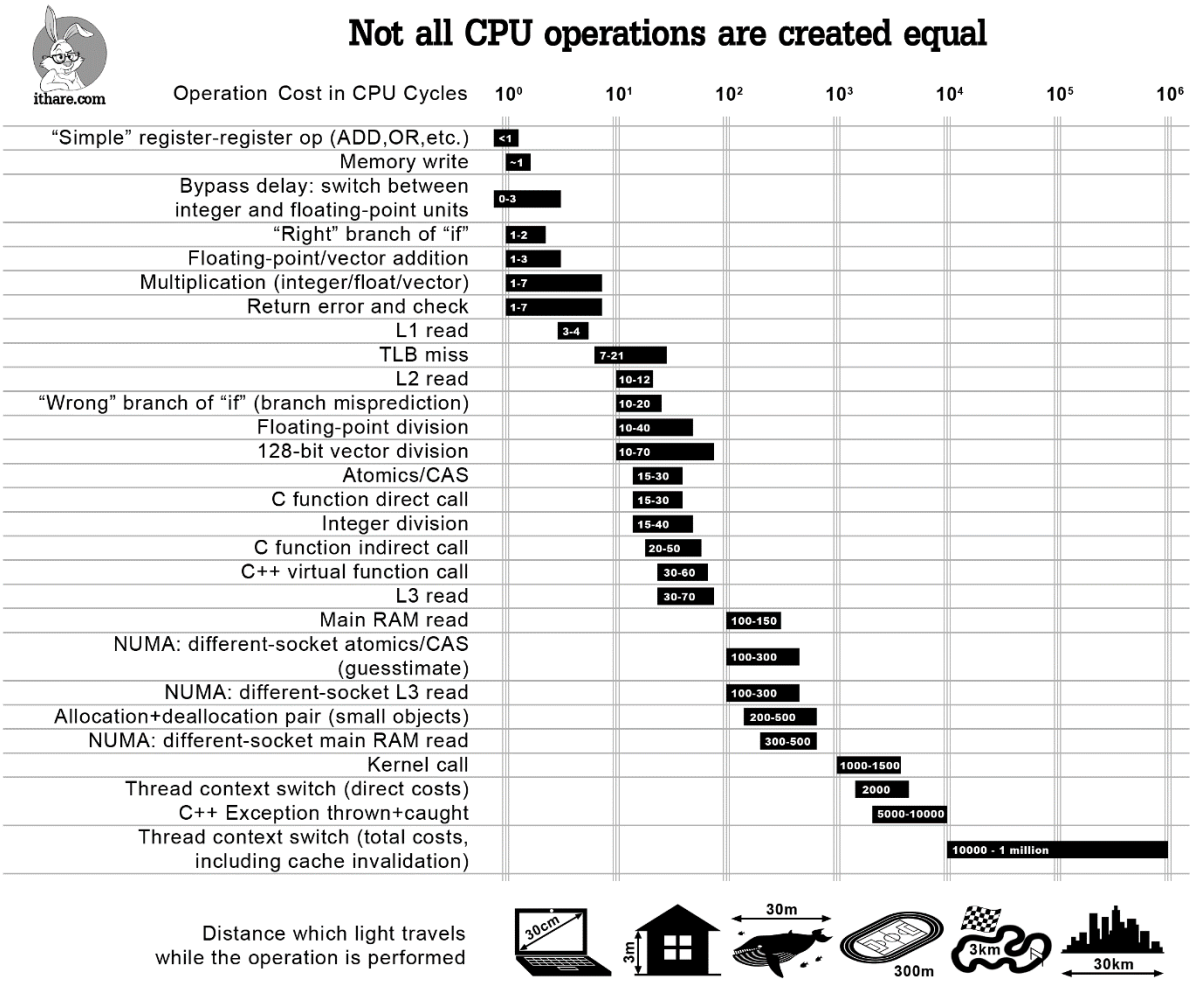
이유: 가독성이 높아지며 상태를 묶어서 관리가 편해진다. 그리고 컴파일러 최적화(Lookup table)을 switch-case가 더 잘 지원한다.

예시:

```
enum EBowState
{
    IDLE = 0,
    ArrowLoad,
    ArrowDraw,
    ArrowShoot,
}

private EBowState bowState = getBowState();

// unfavorite: if-statement
if (bowState is EBowState.IDLE)
{
    // IDLE Process
}
else if (bowState is EBowState.Draw)
{
    // Draw Process
}
else
{
    throw new NotImplementedException();
}

// favorite: switch-case statement
switch (bowState)
{
    case EBowState.IDLE:
        // IDLE Process
        break;
    case EBowState.Draw:
        // Draw Process
        break;
```

```
    default:
        throw new NotImplementedxception();
}
```

2. Exception Handling보다는 아예 에러가 발생할 상황을 막아라



이유: throw Exception 자체가 비용이 많이 드는 작업이기 때문이다. C++기준 5000-10000 정도의 CPU 사이클 정도의 시간이 드는데 이는 평상적인 경우 Thread Context Switch를 제외하면 가장 비용이 높은 작업이다.

대안: Exception을 쓸 수밖에 없다면 C#이나 Unity의 built-in Exception을 사용하고 임의로 Exception 클래스를 만들지 말자.

그리고 Exception보다는 Debug.Log를 활용해서 디버그하자.

Try-catch 문을 사용할 때는 이걸 사용할 필요가 있는지 다시 한번 생각하고 차라리 Assert나 Quit Log로 프로그램 자체를 나가고 에러 메시지 남기는 게 났다.

3. 최적화하겠다고 가독성을 포기하지 마라

이유: 간단하다. 비트 단위로 시스템 리소스를 관리하고 분기를 없애는 데에만 노력한다면 당연히 가독성이 떨어져서 유지보수가 힘들어질 수 있다. 그리고 이렇게 복잡한 코드는 컴파일러 최적화가 잘 안돼서 오히려 일반적인 코드보다 느려질 수 있다. 우리는 어셈블리어로 임베디드 OS 코드를 짜는 것이 아니다.

대안: 보기에 깔끔하고 어떤 식으로 컴퓨터에서 작동하는지 알 수 있는 코드 위주로 짠다. 가독성이 일단 첫번째 조건이고 그 다음이 성능이다.

4. LINQ에 Side-effect 있는 람다 함수를 넣는다면 한 번쯤 다시 생각해봐라

Ref: https://www.meziantou.net/don-t-use-method-with-side-effect-in-linq.htm

이유: LINQ(C#의 쿼리 라이브러리)는 Lazy Evaluation이기 때문에 Side-Effect가 있다면 변수를 사용할 때마다 변수가 변경되기 때문에 어떤 변수가 튀어나올지 모르는 상황이 발생하기 때문이다.

예외사항

추후 추가 바람