

목차

1. 애플리케이션 기획
   1. 게임 컨셉
   2. 게임 플레이
   3. 오브젝트
2. High Level Design
   1. 게임 플로우 차트
   2. 게임 네트워크 흐름도
   3. 데이터 송수신 설계
3. Low Level Design
   1. 객체
   2. Server 함수
   3. Client 함수
   4. 사용 프로토콜

# 팀원간 역할 분담

# 개발 환경

# 개발 일정

1. **애플리케이션 기획**

* 1.1 게임 컨셉
* 게임장르: 플래시 게임
* 게임설명: 대나무 숲에서 날아오는 수많은 화살들을 피하고 최대한 오래 살아남는 게임





* 1.2 게임 플레이
* 서버에 접속을 한 사람은 자신의 닉네임을 정한다.
* 접속이 된 플레이어는 대나무 숲(로비)에서 돌아다닐 수 있다.
* 로비에 접속한 플레이어가 두 명이 될 경우 카운트 다운 후 게임이 시작된다.
* 상 하 좌 우 (W S A D) 키를 이용하여 화면에서 날아오는 화살을 피해야 한다.
* 플레이어가 화살과 충돌할 경우 그 플레이어의 생존 시간이 기록된다.
* 플레이어가 모두 화살과 충돌하여 게임 오버 될 경우 두 플레이어의 생존시간을 화면에 출력해준다.
* 생존시간이 더 오래된 플레이어의 승리를 출력하며 게임이 종료된다.
* 1.3 오브젝트
* 기존 게임: 플레이어 방향으로 날아오는 화살 밖에 없음
* 구현 게임

-플레이어 방향으로 날아오는 화살

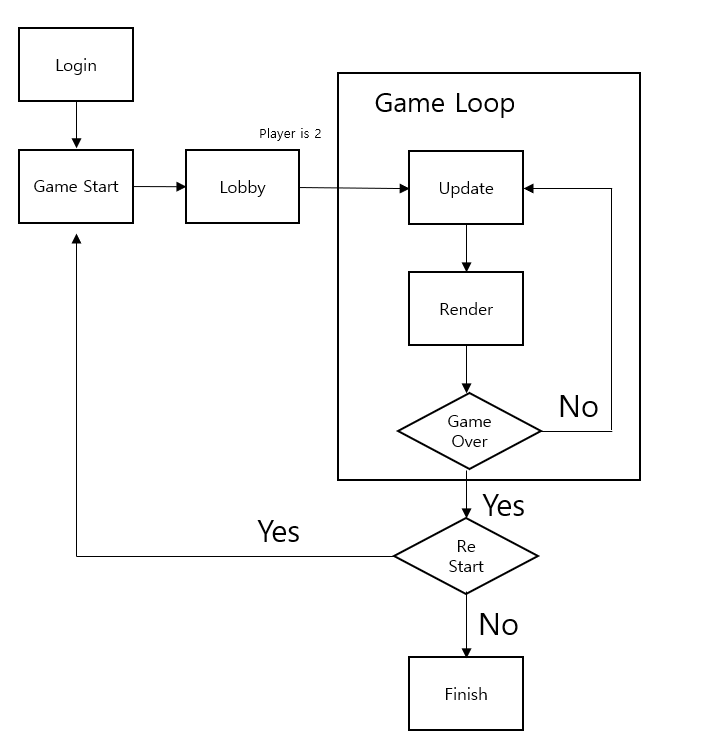
-플레이어를 계속 따라오는 화살

-일정 위치에서 퍼지는 가로 세로 방향의 레이저

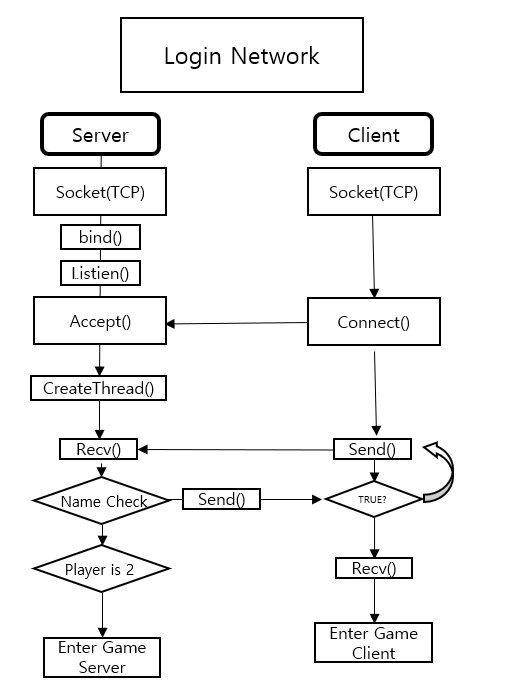
|  |  |  |  |
| --- | --- | --- | --- |
| 시간 경과 (초) | 10초 | 20초 | 30초 |
| 등장 패턴 (객체) | 일반 화살 | 유도 화살 | 레이저 |

1. **High Level Design**

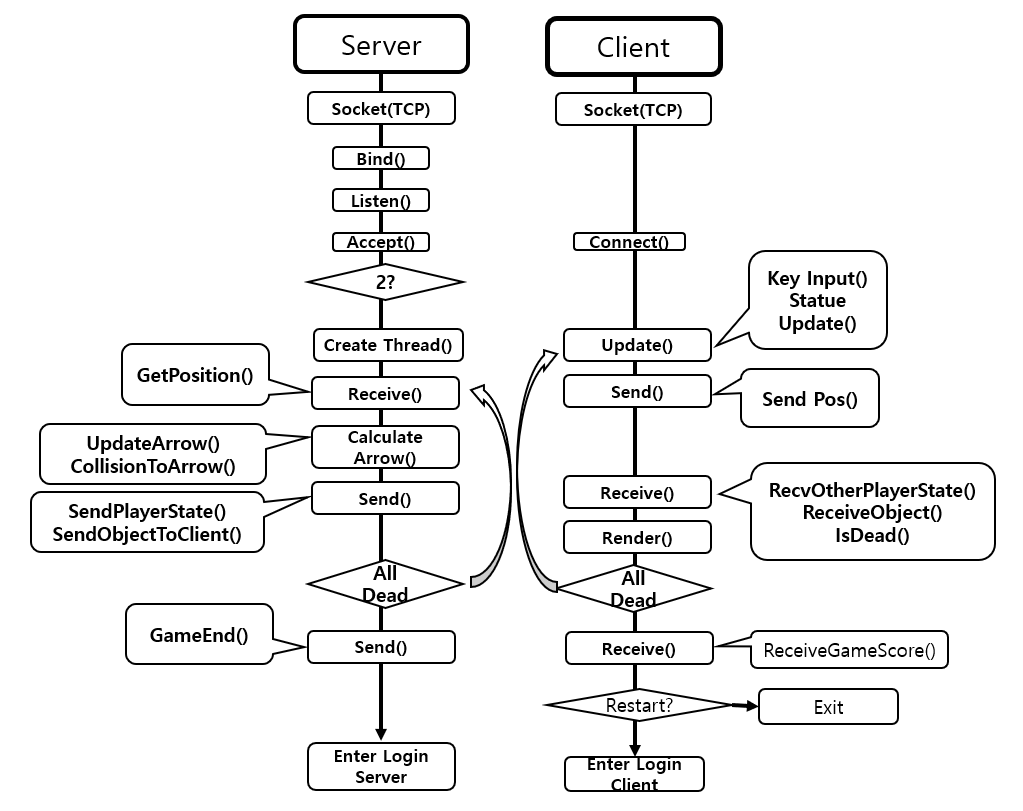
* 2.1 게임 플로우 차트



* 2.2 로그인 네트워크



* 2.3 인게임 네트워크



1. **Low Level Design**

* 3.1 객체

typedef struct tagTexture

{

LPDIRECT3DTEXTURE9 pTexture;

D3DXIMAGE\_INFO tImgInfo;

}TEXINFO;

typedef struct tagInfo

{

D3DXVECTOR3 vPos; // 위치벡터

D3DXVECTOR3 vDir; //방향벡터

D3DXVECTOR3 vStandard; // 각도선정의 기준이되는축 {-1,0,0}

D3DXMATRIX mat\_World; // 월드좌표

}INFO;

Object (BaseClass)

-RECT rect; (Collision Rect) //충돌처리사각형

-TEXINFO texInfo; // 객체들이 쓸 텍스쳐

-INFO Info;

-D3DXMATRIX Scale; // 크기행렬

-D3DXMATRIX Rotation; //회전행렬

-D3DXMATRIX Translate; //이동행렬

Normal Arrow

- float Angle; // 화살의 회전을 위한 각도

- float Speed; // 화살 스피드

Guide Arrow

- float Angle // 화살의 회전을 위한 각도

- Player \* Target // 화살이 따라갈 플레이어

- float Speed; // 화살 스피드

Player

- string PlayerTag; // 플레이어의 아이디

- bool PlayerDead; // 생존여부

- float Speed; // 플레이어 스피드

Lazer

-enum eLazerType{LagerX,LagerY};

- eLagerType LazerType; //가로 세로 판단

- DWORD LazerLifeTime; //레이저시간

Warning

* Enum WarnDir { X, Y};
* WarnDir LazerDir; //가로 세로 판단
* DWORD WarningTime;//표지판 시간
* 3.2 Server 함수
* Login

Void CheckNickName(const char \* NickName)

클라이언트의 닉네임을 받아 저장한다. 중복이 있을 경우 재 요청한다.

* Lobby

Void GameStart()

접속 클라이언트가 두개면 게임을 시작한다.

* InGame()

Void GetPosition(vector2 Pos)

플레이어의 위치를 받는다.

Void CollisionToArrow(vector2 Pos,Arrow \* ArrowLst)

화살과 플레이어의 충돌을 담당한다.

Void InitNormalArrow(vector2 Pos)

일반 화살의 회전, 위치정보를 정한다.

Void UpdateGuideArrow(vector2 Pos)

유도 화살의 회전, 위치정보를 매프레임 갱신한다.

Void SendObjectToClient(Object \* Obj)

오브젝트가 가진 정보들을 선택적으로 클라이언트에 송신한다.

Void SendWarningPos()

클라이언트에 경고판 위치를 전송해준다.

Void SendPlayerState()

각 클라이언트에게 모든 플레이어의 상태를 보내준다.

Void RecordScore()

플레이어가 모두 사망 시 플레이어의 기록을 Txt 파일에 기록한다.

Void GameEnd()

txt파일을 읽어들인 후 정렬을 하여 각 클라이언트가 몇 순위 인지, 몇 초 생존했는지, 누가 승리자인지 알려준다.

* 3.3 Client 함수

Void DrawPlayer(vector2 Otherpos)

자신의 위치와, 상대의 위치에 플레이어의 이미지를 그린다.

Void RecvOtherPlayState(bool isDead)

상대 플레이어의 생존여부를 받는다.

Void DrawObject(vector2 Otherpos, vector2 Rotate,int Objtag)

오브젝트의 위치와, 오브젝트의 회전정보, 오브젝트 번호를 받아 그린다.

Void KeyInput()

방향키로 플레이어를 움직인다.

Void SendPos()

서버에 자신의 위치를 보낸다.

Void StatueUpdate()

플레이어의 상태를 바꾼다.

Void ReceiveObject(Object \* Obj)

오브젝트의 정보를 서버로부터 받아온다.

Void IsDead(bool isDead)

플레이어의 생존정보를 받아온다

Void ReceiveGameScore(float Score)

플레이어의 기록을 수신 받는다

* 3.4 Protocol

#공통 상수 값

#define WINCX 1000

#define WINCY 1000

#define DEAD\_OBJ 1

#define PI 3.141592

#define PLAYERSPPED 100

\*로그인 서버

(클라이언트 ->서버)  
 char[100] NickName; // 닉네임

(서버->클라이언트)

Bool LoginSuccess; //로그인 성공여부

\*Ingame 서버

(클라이언트->서버)

vector2 Positon; //player 좌표

(서버->클라이언트)

Struct ObjectData //장애물 정보

{  
 vector<Object> Datalist;

Int ObjectCount;

}

Struct PlayerData //1p와2p의 생존여부

{

Bool PlayerDead;

Bool OtherPlayerDead;  
}

Struct Record //1p와2p의 생존시간

{  
float PlayerLifeTime;

float OtherPlayerLifeTime;

}

Struct RecordHistoryData

//역대 기록을 담을 구조체

{  
 Char[100] mostLifeLickName;

Float LifeTime;

}

**4. 개발 환경**

OS: Windows 10

개발 프로그램: Visual Studio 2019

사용 API: DirectX 9 / WSA / WinAPI

개발 언어: C/C++

**5. 팀원 역할 분담**

<Client>

|  |  |  |
| --- | --- | --- |
| 작업 | 설명 | 담당 |
| 경고판, 레이저 클래스 | 오브젝트 로직 및 클래스 디자인 | 김희준 |
| 화살, 유도 화살 클래스 | 오브젝트 로직 및 클래스 디자인 | 박준오 |
| ID입력 화면추가 | 폰트를 이용한 ID 입력 UI를 추가 | 박준오 |
| 플레이어 이벤트 처리 | 인풋값으로 플레이어가 움직이도록 구현 | 김희준 |
| Send, Recv담당 클래스 생성 | 네트워크 통신을 위한 클래스 디자인 | 박준오 & 김희준 |

<Server>

|  |  |  |
| --- | --- | --- |
| 작업 | 설명 | 담당 |
| 서버 기본 틀 제작 | 기본적인 데이터를 주고받는 서버 구현 | 박준오 |
| 로그인 프로토콜 | 클라이언트의 ID를 받아 로그인 로직 구현 | 김희준 |
| 로비 프로토콜 | 클라이언트의 접속을 허용하고 로비상태로 변경시키는 로직 구현 | 김희준 |
| 인게임 프로토콜 | 인게임에서의 데이터 송수신, 충돌처리 로직 구현 | 박준오 |

**6. 개발 일정**

<10월>

|  |  |  |
| --- | --- | --- |
| 날짜 | 박준오 | 김희준 |
| 10/23 | 기획서 작성  서버 – 클라이언트 틀 설계 | |
| 10/24 |
| 10/25 |
| 10/26 |
| 10/27 |
| 10/28 |
| 10/29 |
| 10/30 |
| 10/31 |

<11 월>

|  |  |  |
| --- | --- | --- |
| 날짜 | 박준오 | 김희준 |
| 11/1 |  | |
| 11/2 |
| 11/3 | 보고서 검수 | |
| 11/4 | Git Repository 생성 | |
| 11/5 | Server 기본 틀 제작 | 클라이언트에 텍스처 띄우기 |
| 11/6 | Client – Server 데이터 주고받는 것 구현 | Client – Server 데이터 통신을 위한 Client 수정 |
| 11/7 |
| 11/8 |  |  |
| 11/9 |  |
| 11/10 |  |
| 11/11 | Login을 위한Client Font 구현 | Login Server 구현 |
| 11/12 |  |  |
| 11/13 | 1차 중간 검토 | |
| 11/14 |  | |
| 11/15 |
| 11/16 | Class design(경고판 ,레이저) | Class design (화살,유도화살) |
| 11/17 | Class design (플레이어) |  |
| 11/18 |  | Lobby server 구현 |
| 11/19 | Ingame server구현 |  |
| 11/20 |  |  |
| 11/21 | 2차 중간 검토 | |
| 11/22 |  |  |
| 11/23 | Server Collision 구현 | Client to Server  Collision Data 통신 |
| 11/24 |  |  |
| 11/25 |  |  |
| 11/26 | 게임 데모 1차 시연 | |
| 11/27 | 피드백 및 오류 검토 | |
| 11/28 | Server-Client 파일 입출력 | 클라이언트 최고기록 점수 구현 |
| 11/29 |  |  |
| 11/30 |  |  |

<12 월>

|  |  |  |
| --- | --- | --- |
| 날짜 | 박준오 | 김희준 |
| 12/1 | 최종 데모를 위한 게임 테스트 및 오류 수정 | |
| 12/2 |
| 12/3 | 최종 데모 | |