1. 개요

* 설계에 대한 전체적인 개요 정리

1. 개발일정별 진도

* 계획서 작성 시의 개발일정과 현재 진도 정도를 일목요연하게 정리
* 필요 시 개정이력 정리

프로젝트 계획

자료조사

* 클라: 유니티 사용,
* 서버: Github 생성, C# 명명법 결정,
  + https://rogerdudler.github.io/git-guide/index.ko.html
  + <http://www.dotnetspider.com/tutorials/CodingStandards.doc>
* 네트워크
  + TCP/UDP, 동기/비동기
    - <http://www.ktword.co.kr/abbr_view.php?nav=1&opt=&m_temp1=347&id=428>
    - https://www.ibm.com/developerworks/linux/library/l-async/
  + TCPListener / SocketAsyncEventArgs
    - https://msdn.microsoft.com/ko-kr/library/system.net.sockets.tcplistener(v=vs.110).aspx
    - https://msdn.microsoft.com/ko-kr/library/system.net.sockets.socketasynceventargs(v=vs.110).aspx
    - 추상화 레벨이 너무 높아 SocketAsyncEventArgs로 결정
  + UTF-8/UTF-16
    - <https://msdn.microsoft.com/ko-kr/library/system.text.encoding.utf8(v=vs.110).aspx>
    - <https://msdn.microsoft.com/ko-kr/library/system.text.encoding.unicode(v=vs.110).aspx>
    - http://pickykang.tistory.com/13
  + 패킷 통신을 위한 프로토콜
    - ppt
* 서버
  + 성능
    - 시간 당 처리량, 평균 응답시간, 네트워크 통신량, 동시 사용자
  + 동시성, 안정성, 확장성, 분산, 보안, 로그
  + 더미 클라이언트
* 데이터베이스, GUI
  + MySQL
  + WinForm, WPF

5주: 시스템 설계, 구현

6주: 시스템 설계, 구현

7주: 구현, 테스트

8주: 구현, 테스트

9주: 구현, 테스트, 보완 및 추가 작업

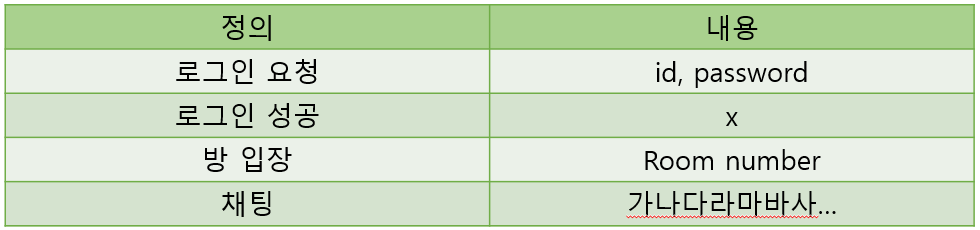
10주: 구현, 테스트, 보완 및 추가 작업

1. 현재 진행 상황

* 현재 진행된 사항을 그림 및 글로 정리

~ 03. 26

* 1. 네트워크 연결 클래스 생성
     1. TcpListener를 테스트위해 에코서버와 에코클라이언트 생성
     2. 추상화 수준이 너무 높아 커스터마이징이 힘들고 여러 버그로 SocketAsyncEventArgs을 사용하기로 결정
     3. 네트워크 서비스를 제공하는 클래스 생성(NetworkService)
        1. 비동기 소켓을 작업하는 SocketAsyncEventArgs 생성
           1. 송신, 수신에 쓰이는 SocketAsyncEventArgs객체 생성
           2. SocketAsyncEventArgs에 통신에 사용하는 버퍼 설정(크기)
           3. 각 송수신 SocketAsyncEventArgs에 빈 유저 슬롯을 전달하여 유저를 관리하는 클래스 연결(UserToken)
     4. 각각 유저를 받기 위한 리스너 생성(Listener) ⬄ 로그인, 채팅 등 서버를 나타냄
        1. 소켓 생성
        2. 소켓 Bind
        3. 소켓 Listen
        4. 크라이언트 접속 시작
           1. 클라이언트 접속 성공 시 클라이언트 소켓 생성
           2. 송수신 객체 SocketAsyncEventArgs의 빈 유저 슬롯에 연결된 유저를 저장
           3. 해당 유저와 통신을 시작.
  2. 네트워크 메시지 생성
     1. 서버와 클라이언트가 연락하기 위한 패킷 구축(Packet)
        1. 패킷의 최대크기는 버퍼의 크기와 같다.  
           패킷은 정의부분과 내용부분으로 나뉘어져 있다.  
           어떤 패킷인지 알려주기 위해 정의부분은 크기가 고정되어 있다.  
           내용은 해당 패킷의 내용을 가진다



* + - 1. 패킷이 버퍼보다 클 경우 2개 이상으로 분할되기 때문에 나뉘었는지 확인할 수 있게 제일 앞 부분의 패킷의 전체크기를 입력한다.
    1. 패킷 중에서 분할된 패킷을 결합 (MessageTranslator)
  1. 유저들과 통신하는 클래스 생성(UserToken)
     1. 해당 유저의 패킷을 송수신하는 클래스이다.
     2. 수신한 메시지를 처리부분은 IPeer에게 넘긴다.
  2. 수신메시지를 해석 및 처리를 역할을 규정 (IPeer Interface)
     1. 서버와 클라이언트의 처리방식이 각각 다르기 때문에 인터페이스로 생성한다.
  3. 서버에서 해당 유저의 수신패킷을 처리 (GameUser : IPeer)
     1. 로그인 요청, 회원가입, 방 접속 등
  4. 클라에서 서버에게 받은 패킷을 처리하는 클래스 생성(RemoteServerPeer : IPeer)
     1. 로그인 성공/실패, 회원가입 성공/실패 등
  5. 런타임 시 유저들의 동시접속으로 인한 인스턴스의 생성과 소멸로 병목발생 예상
     1. 초기 구동시 빈 객체를 생성하고 재활용하는 풀링 기법 차용
     + 리스트에 정해진 개수만큼 빈 객체를 생성하고(new) 필요 시 꺼내서 값을 넣어둔 후 폐기시 초기화하고 다시 리스트에 집어넣는 방법
     + 서버 초기를 제외한 실시간 중에 new/delete로 인한 속도 저하를 예방할 수 있다.
       1. SocketAsyncEventArgsPool을 생성하고 관리 (SocketAsyncEventArgsPool)
       2. SocketAsyncEventArgsPool에 쓰이는 버퍼를 관리 (BufferManager)
       3. 서버와 클라이언트가 연락할 때 마다 사용하는 패킷 관리 (PacketBufferManager)
  6. 유저들을 관리하기 위한 데이터베이스 연결
     1. 쿼리문을 코드로 적용하는 과정에서 쿼리문에 따른 분할 생성  
        일반 서버들은 MySqlCommand만 사용  
        디비서버는 DataSet으로 변환할 예정
     2. MySqlAdapter
        1. MySQL에 접속하고 통신하는 역할
           1. 접속: Connect()
           2. DB선택: SelectDataBase()
           3. 쿼리 전송: SendQueryNoData(), SendQueryDataSet(), SendQueryList()
     3. DataBaseInventory
        1. MySQL에 있는 데이터베이스 목록을 보관
           1. 접속: OpenDataBase( DB이름 )
        2. DataBase
           1. MySQL의 디비에 존재하는 테이블 목록을 보관

다른 DB의 테이블들을 접근하려면 DataBaseInventory의 OpenDataBase()를 이용해야 한다.

테이블 관련 쿼리 가능

* + - 1. Table
         1. MySQL의 테이블에서 속성(Attribute) 목록을 보관

컬럼과 필드 관련 쿼리 가능

03. 27 ~ 03. 31

1. 더미 클라이언트 생성
2. 1:1 접속에서 다중 접속으로 변경
3. 로그인 프로토콜 생성
4. 리펙터링

04. 01 ~ 04. 07

1. 클라이언트와 서버개발 독립적으로 가능하도록 설정
2. 유니티에서 씬(Scene) 전환시 네트워크가 유지되게 설정
3. 데이터베이스 관련 클래스 생성
4. 통신 종류 프로토콜에 정의(Single, Multi)
5. 각 캐릭터마다 식별방법 필요
6. 리펙터링

04. 08 ~ 04. 14

1. 접속한 유저목록을 관리하는 싱글클래스 생성(UserList)
2. 리펙터링
   1. MessageTranslator에서 콜백 제거
3. 서버 개발 다음 일정 추가
4. 서버 분산시 방법 정의
5. 캐릭터 이동 프로토콜 추가

04. 15 ~ 04. 21

1. 리펙터링
   1. 디렉터리 정리
   2. 중복 코드 제거
   3. 불필요한 주석 제거

04. 22 ~ 04. 28

1. 리펙터링
   1. 데이터베이스 연동 클래스와 데이터에베이스 클래스 분리

04. 29 ~ 05. 05

1. 버그 수정
   1. 전송 패킷 생성 오류 수정
   2. 호환성 문제 해결
   3. 키를 입력할 때 여러 캐릭터가 동시에 움직이는 문제 수정
2. 캐릭터연동을 위해 고유의Id를 생성하는 클래스 생성(NumberingWaitObj)
3. 콘솔이 아닌 GUI로 모니터링을 하기로 결정(WPF)
4. 여러 캐릭터 생성

05. 06 ~ 05. 12

1. 버그 수정
   1. 카메라 고정문제 수정
2. 모니터링에 사용할 CPU, RAM 이용률 추가
3. 콘솔과 GUI코드의 혼재로 코드 분리

05. 13 ~ 05. 19

1. 버그 수정
   1. 먼저 접속한 유저가 안보이던 현상 수정
2. UI 레이아웃 설정
3. CPU, RAM 이용률 GUI에 추가
4. 서버 접속 시 서로 캐릭터 생성
5. 깃헙 다이어트
6. 설계 진행의 문제점 및 해결방안

* 현재 진행에서의 문제점과 해결방안 요약
* 두 캐릭터간 충돌결과를 누가 서버로 보낼것인지. 만약 둘다 보낼경우 어떻게 한번만 처리할것인지.
* 클라이언트 종료시 서버에 제대로 반영 안되고 있음
* 때리기, 물건습득, 각종 스킬등 이동과 로그인외에는 아직 프로토콜 정의 부재
* 사설ip, 공용ip…공용ip를 통해 접속하는것에 대한 처리가 아직 미흡…
* 서버와 디비를 물리적으로 분산하려 했으나 시간이 부족할것같음…
* 여러 예외사항에 대한 처리 부족….(안하게 될 것 같은건 말 안하는게 나을려나)

1. 향후 일정

* 중간 보고 이후 개발 범위 및 기간 등을 정리

1. 유니티 내에 Assets폴더 정리 -

2. 게임매니저 만들어서 게임매니저가 전체적인 게임 관장하도록

3. 플레이어 객체 하이라키에 등록해놓지 말고 게임매니저의 스크립트에서 코드로

생성되도록...(지금상태는 여러 사용자와 연결되었을때 내가 다른 플레이어까지

조작하게됨...)

4. 플레이어 코드에 불변수 줘서 내가 조종하는 캐릭터일때랑 다른사용자일때 구분하던가

아니면 아에 따로 만들던가...

====================================================

01. 각각의 네트워크 접속자에게 아이디 부여

- 들어온 피어에게 들어온 순서대로 넘버링 부여

- 넘버링은 서버에서 부여

...각 유저에게 고유 id주는것 말고도 어떠한 객체가 생성될떄 서버에게 생성요청을 하도록하고

각 객체에게 고유번호를 주어야 npc도 네트워크 통신이 가능할것 같다...

유저의 고유id는 유저클래스에 저장하면 그만이지만 객체의 id는 어따가 저장하지?

02. 서버에서 클라캐릭터의 위치를 출력해보는 동기화

- 프로토콜 추가

구현방법 : 1) 반다이남코 네트워크 책 참고

2) 위치 전송(주기적으로 전송)

3) 키이벤트 전송(누르거나 뗄때마다)

씬에 진입하면 플레이어가 생성되도록 코드로 조정해야함....---넘버링 부여 순서때문에

플레이어랑 똑같은데 입력만 안받는 캐릭터와 스크립트를 장착한게 있어야함 -- 네트워크 플레이어용

///////////////////////// ~ 5월 3일 ////////////////////////////////////////////

03. 2명이상의 클라이언트 캐릭터 동기화

- 프로토콜 추가

구현방법 : 1) 반다이남코 네트워크 책 참고

///////////////////////// 5월 5일 ~ 5월 10일 ////////////////////////////////////////////

04. 100명 이상의 클라이언트 캐릭터 동기화(더미만들어서 테스트)

05. 때리는것도 동기화

06. 그외 프로토콜 추가

1 ) 물건습득 알림

2 ) 각종 스킬에 대한 알림

07. 로그매니저(서버 상태부터 시작해서 패킷송수신에 대한 로그 디비에 저장)

///////////////////////// 5월 10일 ~ 5월 17일 ////////////////////////////////////////////

08. 유저 데이터베이스 속성추가

1 ) ID, password, 성, 이름, 이메일, 생년월일, 승패

09. GUI

1 ) 윈폼으로 제작

10. 서버 및 디비 분산(물리적 분산 포함)

1 )

///////////////////////// 5월 18일 ~ 5월 28일 ////////////////////////////////////////////

// ~ 6월 16일

11. 프로젝트 마무리

보고서 작성

코드 정리

1. 공학설계 구성요소 및 제한요소 검토의견(서식 4-1참조)

공학설계 구성요소 검토 항목

|  |  |
| --- | --- |
| 항목 | 내용 |
| 목표설정 | - 본 프로젝트는 온라인 게임제작이다. 그 제작과정에서 필요한 클라이언트와 서버 개발 및 네트워크 연동하는 방법을 학습한다.  - 다수의 인원이 접속하여 도둑은 정해진 물건을 훔치고 경찰을 이 도둑을 잡는 게임을 제작하는 것이 목표이다.  - 추가적으로 스킬이나 다양한 컨셉을 추가하여 게임의 난이도 조절과 흥미를 높인다. |
| 분석 | - 본 프로젝트는 서버와 클라이언트로 구성된 게임이다. 게임이 실행될 수 있는 기기의 최소 사양과 게임제작에 필요한 자료가 필요하다.  - 초기에는 단순한 경찰과 도둑 게임이었지만, 팀원들과 토론을 진행하면서 새롭게 스킬이나 룰을 추가하여 발전시켰다. |
| 설계 | - 초기 테스트 버전 제작 후 문제점을 발견 및 수정하여 최종 결과물을 도출하는데 사용한다. |
| 구현 | - 게임 개발에 필요한 유니티 지식과 C# 학습을 진행한다.  - IOCP를 랩핑하여 클래스화 시킨 C# 소켓 라이브러리를 이용하여 서버 구현 한다. |
| 시험 및 평가 | - 제작된 게임이 기기마다 다른 결과가 나올 수 있기 때문에 다수의 기기에서 클라이언트를 실행 후 서버에 접속하여 게임을 플레이한다. |

공학설계 구성요소 항목별 검토의견

공학설계 제한요소 검토 항목

|  |  |
| --- | --- |
| 경제성 | - 본 프로젝트는 학생들로만 이루어져 진행되기 때문에 전문적인 수익모델을 구성하는 것은 상당히 어렵다.  - 다만, 구글 애드센스의 경우 사용이 상당히 간편하기 때문에 이용자가 어느정도 확보될 경우 이용자에 비례하는 수익을 기대할수 있다.  - 최종결과물이 가격경쟁력을 가지고 있는가? |
| 윤리성 | - 정보 수집과정에서 정보제공을 꺼려하는 점주가 있을수도 있지만 아주 극소수 일것으로 판단된다.  - 최종결과물이 윤리적으로 문제를 일으킬 소지는 없는가? |
| 신뢰성 | - 과장되지 않은 계획, 구체적인 계획, 개발에 필요한 지식을 가지고 있다.  최종결과물이 안정적으로 실행되는 등 신뢰성을 충분히 만족하는가? |
| 실행가능성 | - 관련 전공을 3년이상 공부한 학생들로 구성되어 있다. 이전에 해보지 않은 새로운 프로젝트이지만 빠르게 학습하고 진행할수 있다.  - 현실적으로 개발이 가능하여 실행할 수 있는가? |
| 사회성 | - 이러한 플랫폼에 대한 수요는 계속 있었지만 아직 대표적인 플랫폼이 명확하지 않기 때문에 결과물에 따라 학부 프로젝트보다 더 높은곳까지 바라볼수 있다.  - 최종결과물이 사회에 미치는 영향에 대하여 적절하게 대처하였는가? |

공학설계 제한요소 항목별 검토의견

1. 기타

* 위 내용 이외에 기타 다른 사항 정리

1. 참고자료 및 출처

**C# Coding Standards and Best Programming Practices** (http://www.dotnetspider.com/tutorials/CodingStandards.doc)