**이 력 서**

**개인신상**

성 명 : 최 동 현 (한자 : 䙑東鉉 / 영문 :Choi Dong Hyeon )



생년월일 : 1994년 01월 10일

주 소 : 서울 송파구 성내천로6길 1-6 (오금동)

핸 드 폰 : 010 - 6745 - 4317

E-Mail : [MicrosoftMainuser@gmail.com](mailto:MicrosoftMainuser@gmail.com)

병역사항 :군필(해군 병장) 2013. 1 ~ 2015. 1

**학력사항**

2009~2012 : 상일미디어고등학교 [서울] **졸업**

**Unity 프로젝트**

|  |  |
| --- | --- |
| **기간** | **내용** |
| 2023.09.01  ~  2023.09.15 | **– Potfolio 소개.** |
| - **Class Logic 분석.** |
| - **JsonData FireBase DB 동기화. Login Auth인증 규칙 설정.** |
| - **Code 리뷰.** |
| - **.GooglePlayer 서비스 출시.** |
| -  **느낀 점과 깨달은 점.** |

**자격사항**

2009. 10. 03. 정보처리기능사 (한국산업인력공단) **최종합격**

2011. 07. 06. 정보기기운용기능사 (한국산업인력공단)  **최종합격**

2019. 10. 18. 1종보통운전면허 (경찰청(운전면허시험관리단)  **최종합격**

**교육과정**

2023. 01 ~ 2023. 08 SBS 게임아카데미 [유니티 프로그래머 실무 교육과정] **수료**

2023. 02 ~ 2023. 05 인프런 [유니티 클라이언트 기초 과정] **수료**

2023. 03 ~ 2023. 05 인프런 [C#프로그래밍 기초 과정] **수료**

|  |  |  |
| --- | --- | --- |
| **기간** | **동아리명** | **학교** |
| 2009~2012 | 여울컴 | 상일미디어고등학교 |

**자기소개서**

**[개발 직무 지원동기]**

수업 중 웹사이트를 만드는 경험을 통해, 평소 하던 게임이 어떻게 동작하고 만드는지 호기심이 생겼습니다.

게임은 프로그래밍 언어로 구성된다는 것을 알게 되었고 학원을 등록해 C언어를 학습했습니다.

콘솔 창에 테트리스를 만드는 것이 목표가 되어 꾸준한 학습 과 도전 끝에 완성하여 성취감을 느껴보았습니다.

고교 진학 때 스스로 주변 고등학교 중에 프로그래밍 수업이 있는 학교를 찾아, IT 미디어과에 진학했습니다.

평소 프로그래밍 수업에 열심히 하여 생활기록부 에 좋은 평가를 받았습니다.

**[몰입하여 이뤄낸 성과]**

방과후 컴퓨터 공부에 몰입하여 정보처리기능사,정보기기운용사 자격증을 단기간에 취득하는 성과를 이뤘습니다.

**[성장하는 게임을 통한 경험]**

학원을 수료하고 여러가지 게임을 만들었습니다. 재밋는 요소를 추가하고 싶어서 꾸준히 공부했습니다.

게임이 완성 된 후에 리팩토링 작업을 하며 스스로 발전한 경험을 했습니다.

여러 게임을 만들어 보며 실무에 대한 준비를 했습니다.

**[본포지션 을 선택한 이유]**

어릴 적부터 게임에 대한 관심과 애정을 갖고 있습니다 게임을 플레이하는 것뿐만 아니라

그 속에서 어떻게 동작하는지,어떻게 만들어지는지 알고싶은 갈증을 느꼈습니다

본포지션에 대한 학습이 갈증을 해소 시켜주었습니다.

**[즐거웠던 경험]**

처음접한 게임 일랜시아 는, 레벨업을 통해 강해지는 즐거움도 있지만 유저들과 파티플레이를 하면서 친근한 관계가 되어,소소한 일상대화를 했던 것이 즐거움 이였습니다. 앞으로 즐거운 경험을 주는 게임을 만들고 싶습니다.

**[본인의 장점 + 채용해야 하는 이유]**

우선 아르바이트를 하며 사회생활을 경험해 봄으로 위계질서가 있는 어느 조직에 있어도 화합이 됩니다.

주어진 업무에 책임감을 가지고 해나가는 장점이 있으며, 업무를 해낸 후 얻는 성취감에 대해 알고 있습니다.

지시사항에 대해 여러번 고민해 보고 목표하는 결과물에 최대한 근접하려 노력합니다.

일손이 모자라 어려운 상황에 동료를 위해 휴가 반납을 한 경험이 있습니다.

꾸준한 자기계발 을 하며 긍정적인 사고를 가지고 있습니다.

**[경험으로 보이는 본인의 장점]**

미용실에서 아르바이트를 한 적이 있습니다 원장님은 매출이 저조하여 걱정이 있으셨습니다.

오피스텔 상권 으로,퇴근후 방문을 원하는 요구를 파악해 마감시간을 2시간 늦춰 보는 것이 어떨지 제안했습니다.

흔쾌히 그러자 하셨고 월 매출 15% 상승을 기록했습니다.

개선하여 더 좋은 결과를 만들고 실천하는 것이 저의 장점입니다.

**감사합니다.**

**포트폴리오**

**개인 프로젝트**

|  |  |  |
| --- | --- | --- |
| **개발 기간** | 2023.09.01 ~ 2023.09.15 | |
| **개발 환경** | Unity 2021.3.21f1 / GitHub / VisualStudio 2022 | |
| **개발 인원** | 1) 최동현 | |
| **요 약** | MMORPG | |
| **로그인 씬** | | **로비 씬** |
|  | |  |
| **●게임 제목**  **-위험한 숲의 모험**  **●콘셉트**  **-포트폴리오**  **●장르**  **-MMORPG**  **●주요 플랫폼**  **-안드로이드 모바일**  **●비즈니스 모델**  **-와우/리니지/로스트아크/마비노기**  **[ 피그마 툴을 이용해 필요한 기능과 로직 분석 ]**  **싱글톤 매니저 클래스 입니다.**    **매니저들의 로직 입니다.**    **[ FireBase DB / Login Auth 규칙설정 과 동기화]**  **- Authentication / Realtime Database -**   |  |  | | --- | --- | | **Code** | **해설** | |  | **● 로그인 사용자만 데이터에**  **접근할 수 있는 규칙 설정.** | |  | **● 로그인 정보를 DB에 저장.** | |  | **● 리얼타임 데이터베이스 구조.**  **● Json데이터 실시간 동기화.** |   **[ State패턴 으로 Animation 상황에 맞는 메서드 선언 ]**  **- BaseController : MonoBehaviour -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 기본클래스의 맴버변수 선언 | |  | **●State패턴 으로**  **Animation상황에 맞는 메서드 선언.** | |  | State패턴에 사용할 가상메서드 정의 | |  | 초기화 하는 부분 입니다. |   **피그마 로직 여기다가**  **[ State 상황에 맞는 메서드 구현 ] (Player)**  **- PlayeControllerr : BaseController -**     |  |  |  | | --- | --- | --- | | **Code** | **해설** | | |  | 맴버변수 선언 | | |  | 기본클래스의 메서드를 재정의하여  싱글톤의 InputManager클래스의  대리자를 인스턴스화 합니다. | | |  | | | | 레이케스팅하여 충돌된 물체가 레이어에 따라 구분 | | | |  | | 애니메이터의 공격 모션에 맞춰  호출되어 실행되는 공격 메서드 입니다.  ●공격모션에 카메라 모드가 변경됩니다.  ●체력이 0이되면 죽습니다. | |  | | 기본클래스에서 정의한  가상메서드를재정의하여 상태에따른 기능을 정의했습니다.  ●가상메서드를 재정의하여 상태기능 구현. |   **[ State 상황에 맞는 메서드 구현 ] (Monster)**  **- MonsterControllerr : BaseController -**     |  |  | | --- | --- | | **Code** | **해설** | |  |  | |  |  | |  |  | |  |  | |  |  |   **[ FireBase DB 에 동기화 시킬 JsonData ]**  **- JsonData -**   |  |  | | --- | --- | | **Code** | **해설** | | **Player** | | |  | 플레이어의 스탯 을  Json으로 따로 관리 하고 있습니다.  ●플레이어의 기본 스탯. | | **Monster** | | |  |  |   **//피그마 도식화 여기다가**  **[ JsonData를 Dictionary 형태로 저장 ](Player)**  **- PlayerStat -**       |  |  | | --- | --- | | **Code** | **해설** | |  | 직렬화하여  Json의 맴버변수를 선언 했습니다. |        |  |  | | --- | --- | | **Code** | **해설** | |  | Json파일을 딕셔너리에  키값[Level] 값[Stat] 을 저장.  ●Json파일 => List => Dictionary |   **[ JsonData를 Dictionary 형태로 저장 ](Monster)**  **- MonsterStat -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 직렬화하여  Json의 맴버변수를 선언 했습니다. | |  | Json파일을 딕셔너리에  키값[Level] 값[Stat] 을 저장.  ●Json파일 => List => Dictionary |   **[ 공통적인 스탯속성 데이터 선언 ]**  **BaseStat : MonoBehaviour**     |  |  | | --- | --- | | **Code** | **해설** | |  | 기본클래스의 맴버변수 선언  ●모든스탯의 부모가 될 클래스. | |  | 기본 클래스에서 스탯을 초기화 합니다. | |  | 맴버변수를 초기화하는 가상메서드 |   **[ 공통적인 데이터를 상속받아 스탯속성 구현 ](Player)**  **- PlayerStat -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 맴버변수 선언과  플레이어의 EXP에 값이 변동 된 것을  확인하여 레벨업을 진행 합니다. | |  | Json에 있는 스텟 데이터를 맴버변수에 대입 |   **[ 공통적인 데이터를 상속받아 스탯속성 구현 ](Monster)**  **- MonsterStat -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 맴버변수 선언과  플레이어의 EXP에 값이 변동 된 것을  확인하여 레벨업을 진행 합니다. | |  | Json에 있는 스텟 데이터를 맴버변수에 대입 |   **[ SingleTon 패턴으로 전역에서 사용을 위한 여러 메서드 인스턴스화 ]**  **- Managers : MonoBehaviour -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 싱글톤패턴을 사용하여  필요한 메서드들을 한곳에 전부 등록하여  호출 했으며 단 한번 데이터메모리를 차지함으로 메모리 절약하는 효과를 봤습니다. | |  | 싱글톤패턴을 사용하기 위해  인스턴스 변수가 비어있으면 게임오브젝트에서  찾아보고 없다면 새로운 오브젝트를 생성하고  Manager스크립트를 추가한후 인스턴스 변수에  할당합니다.    UGUI의 이벤트사용을 위한 EventSystem을  찾아보고 없다면 프리팹으로 만들어놓은 @EventSystem을 생성합니다.  이둘은 게임이 종료되기전까지 필요하기에  DonDestroyOnLoad로 파괴되지 않습니다. | |  | 전역에서 사용할 메서드들을 싱글톤 클래스에 인스턴스화 후 프로퍼티로 접근제한을 설정후  싱글톤Managers클래스를 통해 접근합니다. | |  | @Managers오브젝트를 찾고 없으면 새로 생성하여 Managers스크립트를 추가 합니다. | |  | 대리자 Action에 구독신청 되어있는  콜백메서드를 실행합니다. |   **[JsonFile 파싱 / Dictionary에 저장]**  **- DataManager -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 인터페이스를 상속받은 클래스는  딕셔너리를 정의 해야 합니다. | |  | JSon파일을 파싱 해오는 메서드 입니다.  인터페이스로 각기다른 데이터들을  새로운 딕셔너리에 매핑 합니다. | |  | | | 런타임 시작시 필요한 스탯을 Dictionary에 저장 합니다. | | |  |  |     **[ GameObject의 객체 생성 / 삭제 ]**  **- ResourceManager -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 한번 생성했던 오브젝트는  딕셔너리에 저장해놓아 또다시  필요로할때 딕셔너리에 값을 찾아 반환 합니다. | |  | Object.Instantiate하는 과정은 성능상  느린 결과를 초래하기 때문에  한번 생성했던 오브젝트는  딕셔너리에 저장해놓아 또다시  필요로할때 딕셔너리에 값을 찾아 반환 합니다. | |  | 오브젝트 삭제시 추가해야할  기능이 생길꺼 같아 만들어 놨습니다. |   **[ Listener 패턴 으로 구독된 메서드 CallBack 호출 ]**  **- InputManager -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 맴버변수 선언 | |  | 입력에 대한 모든 처리를 관리 하기위해  리스너 패턴을 사용했습니다.  대리자Action에 구독 신청이 있고  마우스 왼쪽클릭이 있다면 등록된 메서드를  Managers클래스에서 실행시킵니다. | | **- PlayerController : BaseController -** | | |  | 구독  콜백호출 | | **- Managers : MonoBehaviour -** | | |  |  |     **[씬 전환시 불필요한 데이터 초기화]**  **- SceneManagerEx -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 씬 전환시 호출시킬 메서드 입니다.  씬 전환과 동시에 불필요한 데이터들을  정리할 수 있도록 했습니다 |   **[ 씬에 필요한 리소스 호출 / 불필요한 데이터 Clear ]**  **- BaseScene : MonoBehaviour -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 기본클래스가 씬 이동에  필요한 EventSystem을 찾고  불필요한 자원을 클리어 할 메서드를 정의  하였습니다. | | **- LoginScene : BaseScene -** | | |  |  | | **- GameScene : BaseScene -** | | |  |  |     **[ BGM / Effect 로 구분 하고 Dictionary에 저장하여 호출]**  **- SoundManager -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 사운드 배열을 만들어 상황에 따른  효과음 또는 BGM공간을 만들 었습니다.  딕셔너리는 자주 발생시키는 효과음을  저장 하여 반환 합니다. | |  | @Sound오브젝트를 산하에 사운드관리를  하기위해Effect,BGM등 필요사운드 오브젝트 를 생성하여 기본초기값을 설정 합니다. | |  | BGM인지 Effect인지 구별하는 파라미터를 받아 그에 맞는 사운드를 실행시키고  파마리터 문자열 인자로  실행시킬 클립명을 입력 받습니다. | |  | 씬이동시 불필요한 데이터를 제거 합니다. |   **[ UI를 팝업 / 고정 UI 를 구분 하여 생성 ]**  **- UIManager -**   |  |  | | --- | --- | | **Code** | **해설** | |  | 팝업으로 사용할 UI프리팹을  소팅오더 순서 관리를 하여 순서대로 띄우고 삭제 할 수있습니다.  씬 메서드는 소팅오더를 카운팅하지않고  고정되어 있는 UI를 뛰우기 위해 만들었습니다. | |  | 팝업으로 띄운UI를 저장할때  자료구조Stack을 이용했습니다.  팝업으로 사용할 컴포넌트는 UI\_Popup을  상속 받습니다.  프리팹이름과 스크립트의 이름을 동일하게 네이밍 하여 오브젝트를 생성 했습니다. | |  | 팝업UI를 제거하며  소팅오더와 Stack에서 해제시킵니다. |   **[ UI의 원하는 Componenet를 찾아 매핑 ]**  **- UI\_Base : MonoBehaviour -**     |  |  | | --- | --- | | **Code** | **해설** | |  | 기본클래스가 되는 UI클래스 입니다.  Canvas의 계층구조에 원하는 UI를  이름으로 오브젝트를 찾아서  Dictionary 값에 매핑 하고  그 값을 찾기위한 키 값을 ComponentType으로 매핑 합니다. | |  | Dictionary에  매핑되어 있는 오브젝트를 반환 받습니다. | | **- UI\_Popup : UI\_Base -** | | |  | UI\_Base의 파생클래스  팝업으로 띄울때 소팅오더를 관리 합니다. | | **- UI\_Scene : UI\_Base -** | | |  | UI\_Base의 파생클래스  씬으로 띄울 UI는 소팅오더를 하지 않습니다. |          |  |  | | --- | --- | | **Code** | **해설** | | **\** | @댈리게이트 선언  각각의 이벤트를 호출시킬 대리자를  따로 두어 각각의 기능이 모듈화가 되어  관리하기 용이 합니다. | |  | @댈리게이트로  EventSystem의 입력처리를  익명메서드로 정의한후  콜백방식으로 호출 합니다.  . | | | |
| **프로젝트를 진행하면서 어려웠던 점** | **1. 서버를 통한 통신을 위한 서버에 대한 개념을 학습하는데 어려움이 있었습니다.**  **2. 반복되는 코드가 많아지면서 가독성도 떨어졌으며 코드 변경 시 불편함이 있었습니다.** | |
| **깨달은 점** | **1. 어떤 개념이든 가장 기본 베이스부터 차근차근 베이스를 탄탄히 쌓으면 그 위에 생기는**  **개념들을 쉽게 습득할 수 있다는 것이었습니다.**  **2. 리팩토링으로 반복되는 코드를 재사용이 가능한 모듈화된 메서드로 해결할 수 있었습니다.** | |

**감사합니다**