C++ Team Project 결과보고서

Civilization 『Hanpotamia』
한포타미아 문명 : 부족 전쟁

console game

https://github.com/ProtossDragoon/Hanpotamia

Title Arial 26
Subtitle Arial 22
topic Arial 15
Text Arial 11
Title - Text indent 줄바꿈 1
Text - Title indent 줄바꿈 2

분업

이름	역할	클래스 담당	기타
18 / 박태정	팀장	플레이어 클래스	git 사용 안내, 코드 리뷰, 발표
18 / 이장후		마스터 클래스, main()	git 사용 안내, merge 담당, 문서작업
19 / 김태헌		지역 클래스	UML, 그래프 구조 구현
19 / 임동재		병력 클래스, 자원 클래스	발표 자료 준비

배운 내용의 활용

type	skills	class
polymorphism	operator / function overloading	다양한 클래스
	default parameter	player class
	template	player class
	inheritance	unit class
	function overriding	unit, map class
C++ characteristic	use new / delete instead	모든 클래스
	exception handling (beta)	player class
keyword	static, const	master class, and so on

목차

스토리와 개요

직접 만든 게임 룰을 적용한 전략 시뮬레이션 게임입니다. 이 게임은 턴제를 기본 원칙으로 하며, 2인~4인의 플레이어들이 전략적으로 경쟁할 수 있도록 하는 것이 목표입니다.
포타미아는 '강' 이라는 뜻을 가지고 있습니다. 한강을 따라 서울시의 25개 지역구를 놓고 원시 부족들이 다투는 내용을 모티프로 합니다.

게임을 구성하는 요소는 다음과 같습니다.

1. 게임 심판
2. 플레이어
3. 자원, 병력, 지역(맵)

아웃라인

- 지역을 소유하면, 매턴마다 들어오는 자원이 증가합니다.
- 자원을 이용해 병력을 생산합니다.
- 나의 턴마다, 조작할 수 있는 횟수가 주어집니다.

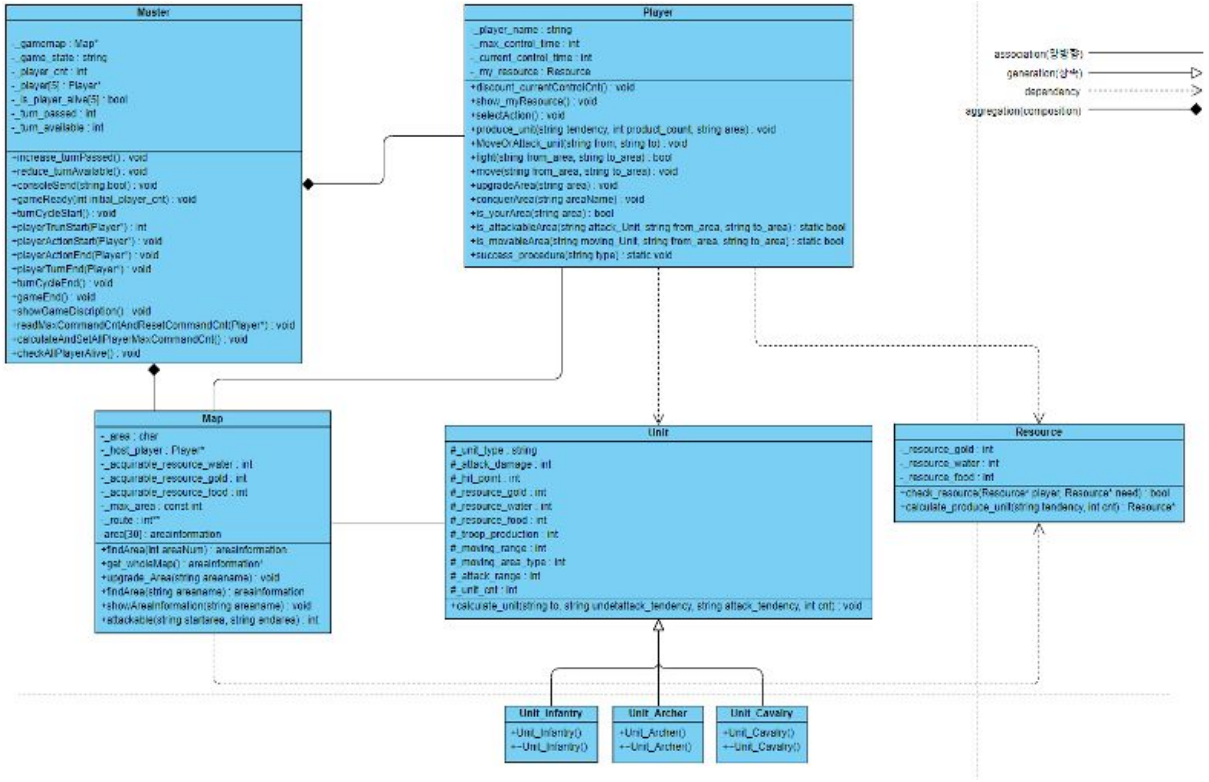
승리 조건은 다음과 같습니다.

- 병력을 이용해 상대 플레이어의 모든 지역을 점령합니다.
- 게임의 모든 턴이 종료되었을 때 점수가 가장 높습니다.

패배 조건은 다음과 같습니다.

- 나의 모든 지역이 함락당하는 경우
- 게임의 모든 턴이 종료되었을 때 점수가 낮다.

전체 클래스 구조도


Master
_gamemap : Map*
_game_state : string
_player_cnt : int
_player[5] : Player*
_is_player_alive[5] : bool
_turn_passed : int
_turn_available : int
+increase_turnPassed() : void
+reduce_turnAvailable() : void
+consoleSend(string,bool) : void
+gameReady(int initial_player_cnt) : void
+turnCycleStart() : void
+playerTurnStart(Player*) : int
+playerActionStart(Player*) : void
+playerActionEnd(Player*) : void
+playerTurnEnd(Player*) : void
+turnCycleEnd() : void
+gameEnd() : void
+showGameDiscription() : void
+readMaxCommandCntAndResetCommandCnt(Player*) : void
+calculateAndSetAllPlayerMaxCommandCnt() : void
+checkAllPlayerAlive() : void
Player
_player_name : string
_max_control_time : int
_current_control_time : int
_my_resource : Resource
+discount_currentControlCnt() : void
+show_myResource() : void
+selectAction() : void
+produce_unit(string tendency, int product_count, string area) : void
+MoveOrAttack_unit(string from, string to) : void
+fight(string from_area, string to_area) : bool
+move(string from_area, string to_area) : void
+upgradeArea(string area) : void
+concuerArea(string areaName) : void
+is_yourArea(string area) : bool
+is_attackableArea(string attack_Unit, string from_area, string to_area) : static bool
+is_movableArea(string moving_Unit, string from_area, string to_area) : static bool
+success_procedure(string type) : static void
association(참조함)
generation(상속)
dependency
aggregation(composition)
Map
-_area : char
-_host_player : Player*
-_acquirable_resource_water : int
-_acquirable_resource_gold : int
-_acquirable_resource_food : int
-_max_area : const int
-_route : int**
area[30] : areainformation
+findArea(int areaNum) : areainformation
+get_wholeMap() : areainformation*
+upgrade_Area(string areaname) : void
+findArea(string areaname) : areainformation
+showAreaInformation(string areaname) : void
+attackable(string startarea, string endarea) : int
Unit
#_unit_type : string
#_attack_damage : int
#_hit_point : int
#_resource_gold : int
#_resource_water : int
#_resource_food : int
#_troop_production : int
#_moving_range : int
#_moving_area_type : int
#_attack_range : int
#_unit_cnt : int
+calculate_unit(string to, string undefattack_tendency, string attack_tendency, int cnt) : void
Resource
_resource_gold : int
_resource_water : int
_resource_food : int
+check_resource(Resource* player_Resource* need) : bool
+calculate_produce_unit(string tendency, int cnt) : Resource*
Unit_Infantry
+Unit_Infantry()
+~Unit_Infantry()
Unit_Archer
+Unit_Archer()
+~Unit_Archer()
Unit_Cavalry
+Unit_Cavalry()
+~Unit_Cavalry()

클래스 설계 : 마스터 클래스 설계

마스터 클래스 설계 원칙

마스터 클래스는 전역 클래스 객체로 모든 클래스들이 정보를 조회하는 용도로 사용할 수 있도록 만들었습니다. 메인 함수에서 돌아갈 큰 동작 원칙을 정의하는 것이 마스터 클래스의 가장 큰 역할입니다. 이러한 구조는 교수님께서 소개해 주신 “스테이트 패턴” 에서 하나의 상황마다 클래스를 만드는 것에 영감을 받았습니다. 하나의 상태를 클래스 하나로 담지 못한 점은 아쉽지만, main 함수에서 명확하게 함수 이름에 따라 동작 상황을 적을 수 있게 보여줄 수 있다는 점에서 의의가 있다고 생각합니다.

마스터 클래스에서 구현한 내용

게임의 상태에 따른 동작 정의 함수
게임의 상태에 대한 정보

마스터 클래스 멤버 변수 및 멤버 함수

주요 변수	Map	**Map *_gamemap**	Map 클래스 객체는 마스터 클래스에서 생성합니다.
	보조	int _turn_passed int _turn_available string _game_state	
	Player	**Player *_player[5]**	Player 클래스 객체는 마스터 클래스에서 생성합니다.
	보조	int *_player_score int _player_cnt bool _is_player_alive[5]	
주요 함수	getter	getter 함수입니다.	
	setter	setter 함수입니다.	
	게임 흐름 제어	**gameReady()** **turnCycleStart()** **playerTrunStart()** **playerActionStart()** **playerTurnEnd()**	게임의 흐름에 따라 적절히 실행되는 함수입니다.

		turnCycleEnd() **gameEnd()**	
	보조	showGameDiscription()	간단한 상황판을 콘솔 출력합니다.
	게임 규칙 제어	readMaxCommandCntAndResetCommandCnt() calculateAndSetAllPlayersMaxCommandCnt() checkAllplayersAlive()	게임의 흐름을 담당하는 함수 내에서 실행되는 함수입니다.

클래스 설계 : 플레이어 클래스 설계

플레이어 클래스 설계

다른 클래스를 참조하여 사용 할 클래스가 많았던 만큼, 코드 재사용에 신경을 썼습니다. 그 대표적인 예로 공격 가능 범위와, 이동 가능 거리에 관해 Map 클래스에서 넘겨받는 정보입니다. 거리에 관해서는 단일 지역간의 최대 2의 거리 정수형으로 넘겨받게 됩니다. 이때 사거리와 이동가능 거리에 대해 따로 코드를 구현하지 않고 같은 함수를 사용하여 재사용 했습니다.

플레이어 클래스에서 구현한 내용

플레이어 클래스는 실제 Player 가 이용하게 될 기능들을 구현 했습니다. 지역 점령, 유닛 생산, 이동 및 공격, 지역 업그레이드, 그리고 각 자원들과 정보 조회에 관한 기능을 Master 클래스에서 충분히 이용가능하도록 구현하였고, Player 클래스 내부에서도 Unit, Map ,Resource 클래스를 참조하여 게임의 구체적인 기능을 사용할 수 있게 하였습니다.

플레이어 클래스 멤버 변수 및 멤버 함수

주요 변수	_player_name	string 자료 형으로 , Player 의 이름을 저장합니다.
	_max_control_time	각 플레이어에게 한 턴에 주어지는 최대 조작 횟수 입니다.
	_current_control_time	현재 플레이어가 사용할 수 있는 조작 횟수 입니다.
	_my_resource	Resource 객체, 자신이 보유하고 있는 자원을 의미합니다.
주요 함수	getter	getter 함수입니다.
	setter	setter 함수입니다.
	selectAction()	Player 가 게임을 진행하기 위해 기능을 선택합니다.
	MoveOrAttack()	이동 및 공격을 수행합니다. 이동 대상 지역에 상대방 병력이 주둔하면 Attack 함수로, 그렇지 않으면 move 함수로 작동하게 됩니다.
	produce_Unit	유닛을 생산하여 배치하게 됩니다.
	show_myWholePlace	자신이 HOST 인 지역을 모두 Display 합니다.

클래스 설계 : 유닛 클래스 설계

유닛 클래스 설계 원칙

유닛 클래스는 게임에서 사용할 수 있는 4가지 병종의 능력치 및 수를 Unit클래스(기반 클래스)에 멤버변수로 두었고, 그 멤버 변수를 사용하는 4개의 파생클래스를 두어 각각의 능력치와 수를 저장하여 사용하게 설계하였습니다.

유닛 클래스에서 구현한 내용

유닛클래스의 멤버변수는 총11개로, 병종, 공격력, 체력, 생산하는데 필요한 자원들, 한번에 생산할 수 있는 병력수, 한턴 이동 범위, 이동할 수 있는 지역특성, 사거리, 병력 수를 가지고 있다. 멤버 함수로는 전투시에 자신의 병종, 수와 상대방의 병종, 수를 통해 계산하여 상대방의 남은 병종, 수를 계산하여 적용하는 함수입니다.

유닛 클래스 멤버 변수 및 멤버 함수

주요 변수	_unit_type	string으로 병종을 담고 있는 멤버 변수입니다.
	_resource_water _resource_gold _resource_food	한 병력을 생산하는데 필요한 자원의 양을 담고 있는 멤버변수입니다.
	_attack_range	전투시 맵의 최단거리와 함께 필요한 사거리를 담는 멤버 변수입니다
주요 함수	getter	getter 함수입니다.
	setter	setter 함수입니다.
	calculate_unit	공격을 수행시 계산하여 상대의 남은 병력 수를 계산하는 멤버 함수입니다.

클래스 설계 : 맵 클래스 설계

맵 클래스 설계 원칙

맵 클래스는 게임을 설계할 때 지역에 병력에 대한 정보(병종, 수, 그 병력을 보유한 플레이어)를 담도록 했기 때문에

맵 클래스에서 구현한 내용

내용을 입력하세요

맵 클래스 멤버 변수 및 멤버 함수

주요 변수	_route	각 지역의 연결여부를 보여주는 인접행렬 변수
	area[30]	각 지역의 정보를 담고있는 구조체 배열
주요 함수	getter	getter 함수입니다.
	setter	setter 함수입니다.
	attackable	지역간에 공격할 수 있는 범위에 있는지 계산
	findArea	이름을 인자로 지역을 찾아 반환하는 함수
	get_wholeMap	플레이어의 전체 소유지역을 알려주는 함수
	showAreaInformation	지역정보를 보여주는 함수

클래스 설계 : 자원 클래스 설계

자원 클래스 설계 원칙

자원 클래스는 멤버 변수로 물, 식량, 금 등을 만들어 플레이어들의 보유자원과 여러 함수에서 필요로 하는 자원들의 양등을 계산할 수 있게 설계하였습니다. 또, 어느 행동을 수행할 때 필요한 자원들을 계산하고 플레이어들의 보유 자원과 비교하여 행동을 수행 할 수 있는지 검사하는 멤버 함수를

자원 클래스에서 구현한 내용

자원 클래스에서는 어떤 행동을 수행 할 때 필요한 자원의 양을 계산하고, 플레이이들의 보유 자원과 비교하여 행동을 수행할 수 있는지 확인하는 멤버함수를 가지고 있다. 이를 위해 필요한 멤버 변수도 자원인 물, 식량, 금 따로 구현하였습니다.

자원 클래스 멤버 변수 및 멤버 함수

주요 변수	_resource_gold	자원 금의 양을 다루는 int형 멤버변수
	_resource_water	자원 물의 양을 다루는 int형 멤버변수
	_resource_food	자원 음식 의 양을 다루는 int형 멤버변수
주요 함수	getter	getter 함수입니다.
	setter	setter 함수입니다.
	calculate_produce_unit	유닛을 생산하기 위해 필요한 자원의 양을 계산하는 함수입니다
	check_resource	필요한 자원의 양과 플레이어의 보유 자원을 비교하여 행동을 수행할 수 있는지 확인하는 함수입니다

발표 질의응답 정리

Q. 왜 기간이 끝났는데도 개발을 하시나요?
A. 게임을 설계할 때 “이왕 하는거, 우리가 진짜 보드게임 대신해서 할 만한 게임을 만들어 보자” 라는 것이 목표였습니다. 하지만 그러기 위해서 학기중만으로는 시간이 현실적으로 부족했습니다. GUI 등을 구현하는 것이 최종 목표이지만, C++ 과목 프로젝트 제안서로 제출했던 것과는 별개로, 학기중의 시간만으로는 만족할 만한 결과를 내지 못했습니다. 그래서 있는 결과를 낼수 있게 개발을 계속 진행하는 것입니다.

사족을 붙이면, 우선 지금 소스코드를 보아도 아시겠지만, 설계패턴부터 UI까지, 프로그램 유지보수성이나 클래스 멤버 접근 등 서투른 것 투성이라는 것을 알 수 있습니다. 이렇게 클래스를 이용해 설계하는 것과, 서로가 서로를 많이 참조하는 것을 충분히 생하지 못한 채로 충돌의 늪에서 허우적대는 그 경험만으로도 값진 경험이라고 생각합니다.

말이 길었지만 무엇보다 컴퓨터리스트들은 방학에도 개발을 해야 삽니다.

시나리오 및 실행 화면

<최종 버전과 UI 가 다를 수 있습니다. >

Persona (Ver . 동재)

종강 후 심심하던 동재와 태헌은 Hanpotamia 를 플레이 한다. 둘은 프로그램이 배치해 준 초기 지역에서 부터 서울 전역을 점령하기 위해 전투한다. 하지만 게임을 진행하는 사회자 프로그램(Master Class) 은 그가 사용하는 고유의 언어(Hanpotamian) 가 있어 그가 원하는 입력 외에는 처리하지 않거나, 플레이어를 프로세스 밖으로 퇴출 시킨다.

동재와 태헌은 이 룰을 숙지하고, 원시 서울을 점령하기 위해 Hanpotamia 로 향한다.

1. 플레이어 1, 플레이어 2 가 게임에 참여합니다.

2. 플레이어의 이름을 설정합니다.

```
player 수 입력 (2~4) : 2
game start
player1name : DongJae
player2name : TaeHeon
```

3. ‘부족’장 동재는 강남구를 초기 지역으로 설정 받았으며, 2번의 조작 기회가 있다.
   Action) 동재는 강남구 지역에 병력을 생산 하여 , 근처를 점령 할 계획이다.

```
========== 부족장 상세 ==========
부족장 DongJae >> 남은 동작 횟수 : 2/2
========== 점령지 목록 ==========
DongJae 가 소유하는 지역을 조회합니다....
DongJae 님이 소유하는 지역
강남구
========== 자원   목록 ==========
금  : 200
음식 : 200
물  : 200
========== 동작을 선택하세요 ==========
0. [동작횟수 -1] 지역점령
1. [동작횟수 -1] 유닛생산
2. [동작횟수 -1] 이동 및 공격
3. [동작횟수 -1] 지역 업그레이드
4. 보유 지역 조회
5. 보유 자원 조회
6. 보유 병력 조회
9. [동작횟수 -1] 실행 1회 넘기기
```

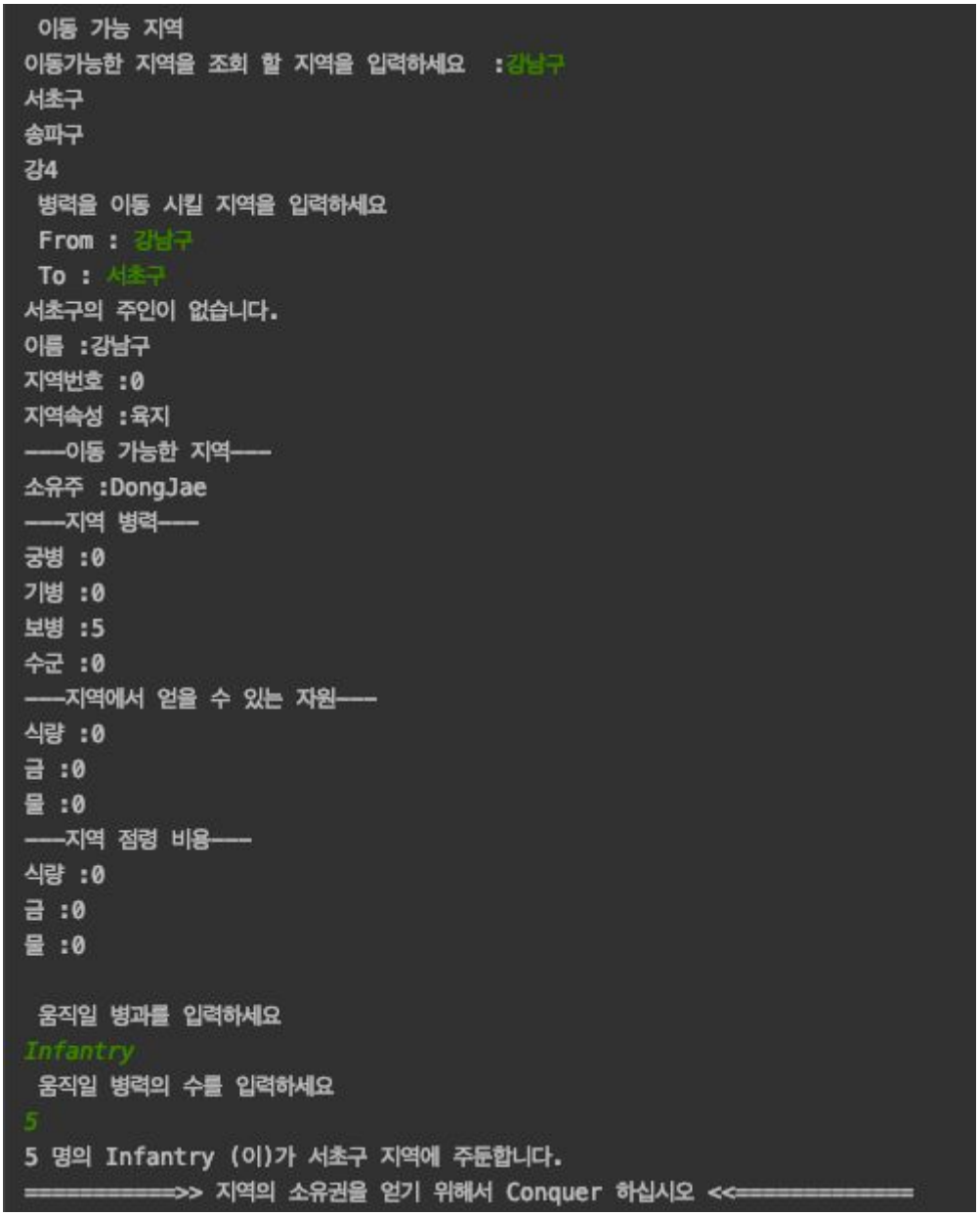
4. 병력을 생산한다.

```
동작 입력 ::: 1

DongJae님이 소유하고 있는 지역 목록 ( 유닛 생산 가능 지역 목록 )
 ═══════════════════════════════════════════════════════════════════
DongJae 가 소유하는 지역을 조회합니다....
DongJae 님이 소유하는 지역
강남구
 병과를 선택하세요
 보병 : Infantry  궁병 : Archer  기병 : Cavalry 해병 : Navy
Infantry
 생산할 병력의 수를 입력하세요
5
 배치 할 지역을 입력하세요
강남구
유닛생산완료 작업을 성공적으로 수행 했습니다.
```

```
═══════════ 부족장 상세 ═══════════
부족장 DongJae >> 남은 동작 횟수 : 1/2
═══════════ 점령지 목록 ═══════════
DongJae 가 소유하는 지역을 조회합니다....
DongJae 님이 소유하는 지역
강남구
═══════════ 자원   목록 ═══════════
금  : 150
음식 : 150
물  : 150
═══════════ 동작을 선택하세요 ═══════════
0. [동작횟수 -1] 지역점령
1. [동작횟수 -1] 유닛생산
2. [동작횟수 -1] 이동 및 공격
3. [동작횟수 -1] 지역 업그레이드
4. 보유 지역 조회
5. 보유 자원 조회
6. 보유 병력 조회
9. [동작횟수 -1] 실행 1회 넘기기

동작 입력 :::
```

동작횟수가 줄어 들고, 병력을 생산한 만큼 자원 또한 줄어들게 된다.

5. 병력을 이동 시킨다.

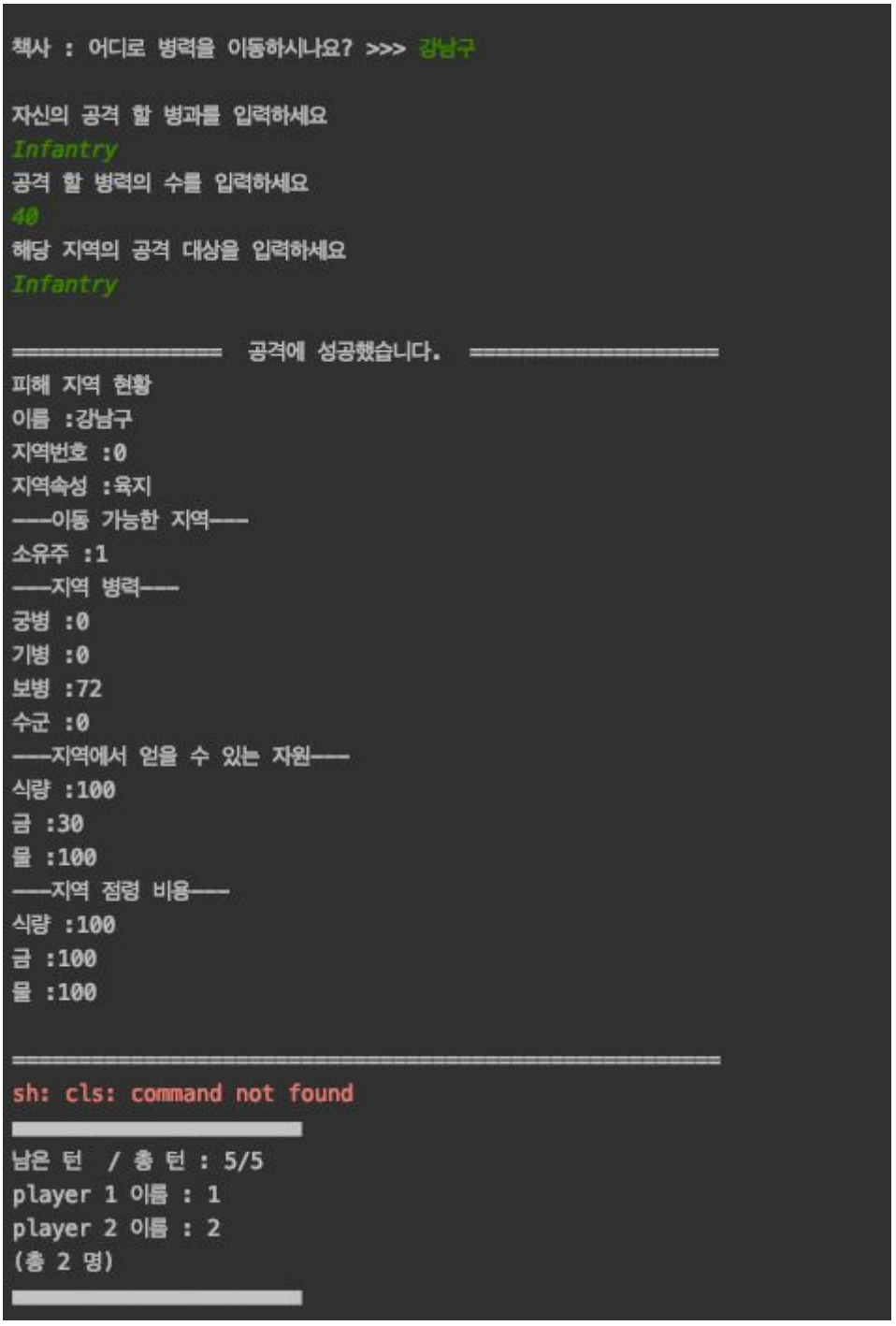


강남구에서 이동가능한 곳 (최대 거리 2) 는 서초구,송파구, 한강지역4 이다. 동재는 강남구에서 서초구로 보병을 5기 이동 시킨다.

서초구를 장악하고 있는 병력과, 플레이어가 없기에, 싸움 없이 서초구에 병력을 주둔시키게 된다.

동작횟수가 줄어든다.

6. 다른 플레이어와 fight



4, 5 동작을 반복하다 태헌의 병력이 있는 지역을 만나게 되면 FIGHT , MOVE 반복

7. 서로의 지역이 사라질때까지 , 혹은 최종 턴이 끝날 때 최종 점수가 높은 사람이 승리하게 됨.

■■■■■■■■■■■■■■■■■■■■■■
게임 종료!
■■■■■■■■■■■■■■■■■■■■■■

■■■■■■■■■■■■■■■■■■■■■■

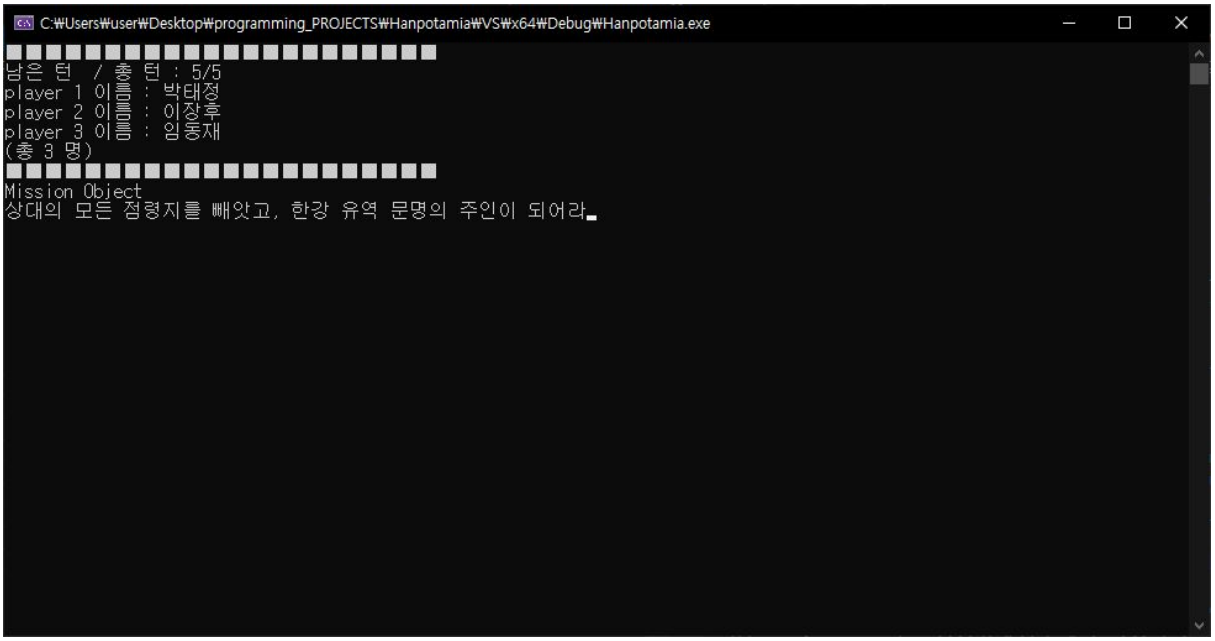
5점 :: Dongjae 의 최종 보유 성 : 5개 *

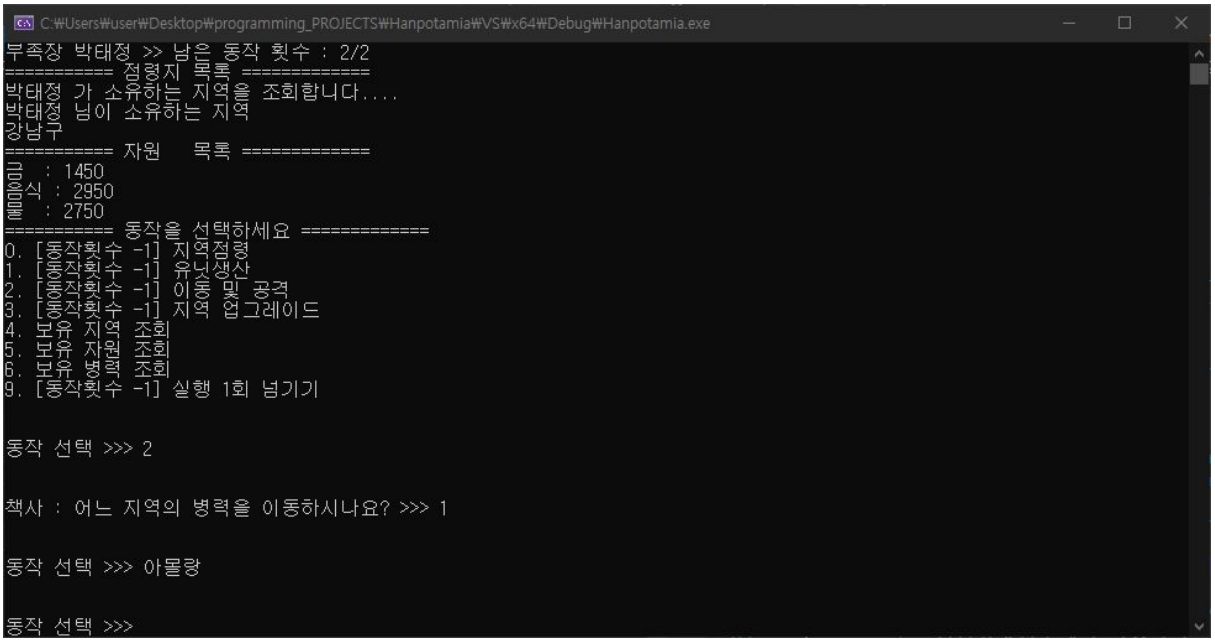
■■■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■■■■■■■■■■■■■■

0점 :: Taehun 의 최종 보유 성 : 0개 *

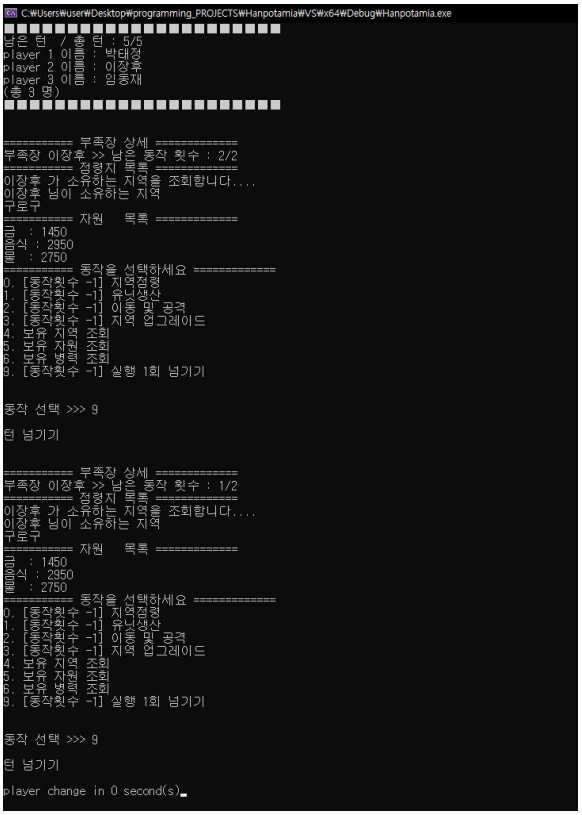
■■■■■■■■■■■■■■■■■■■■■■

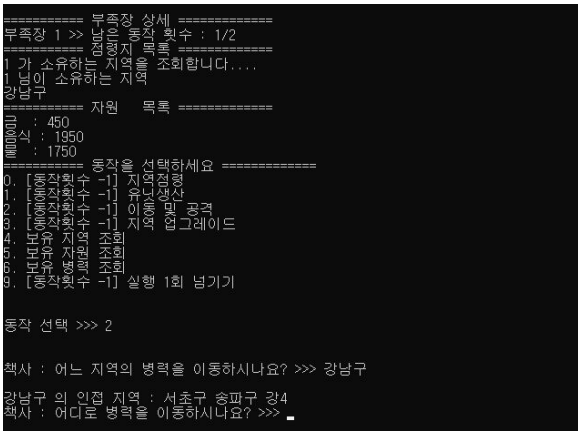
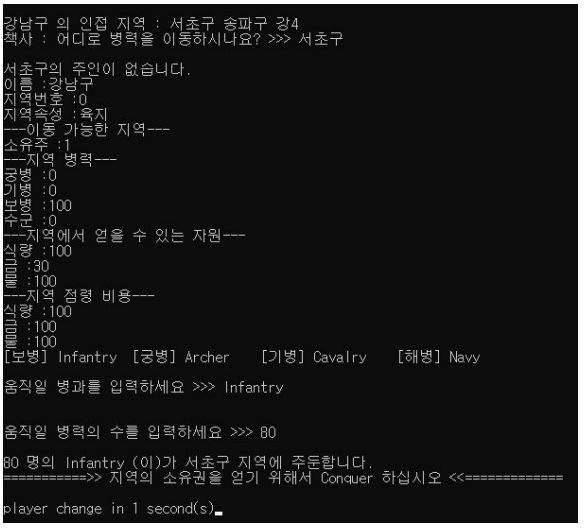
Windows

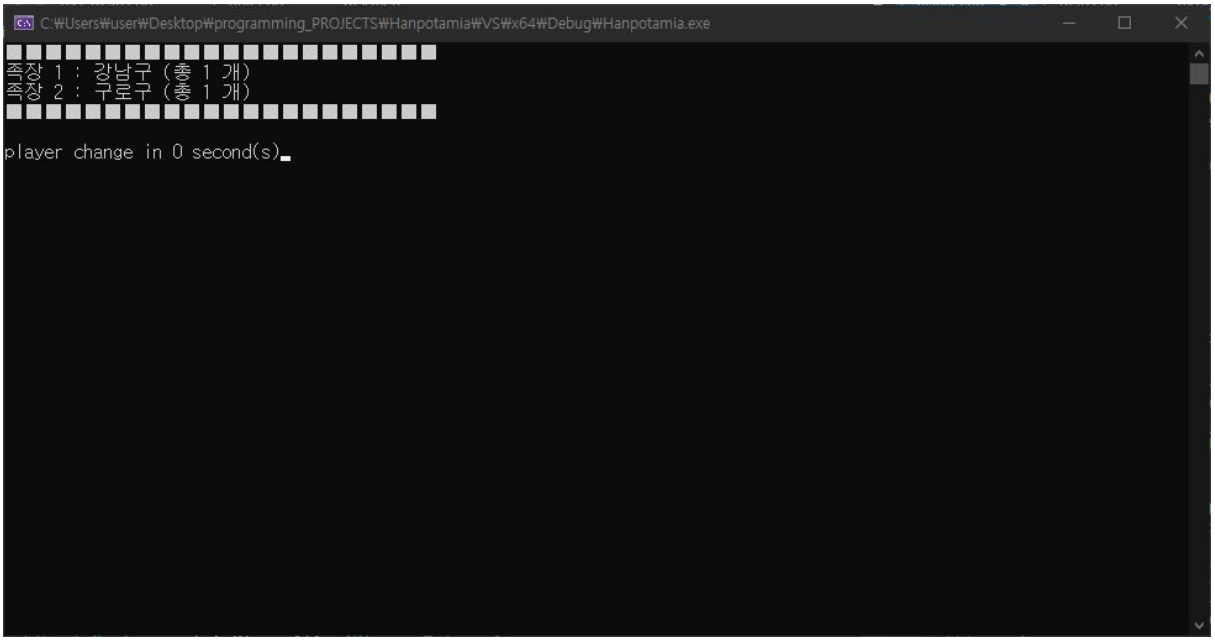


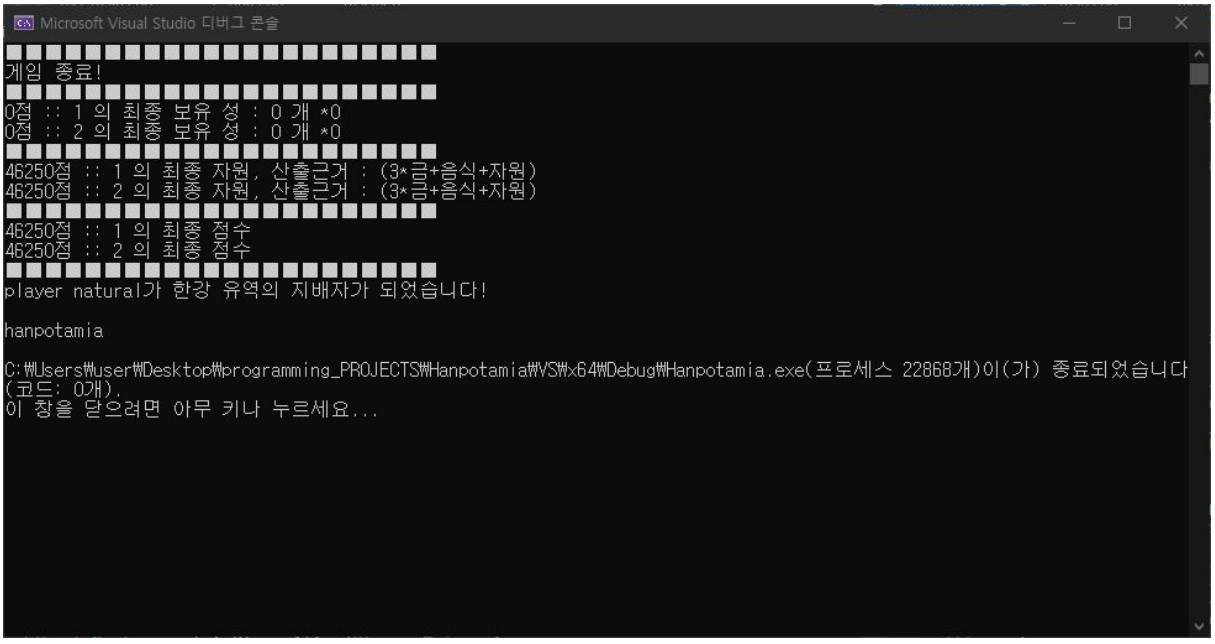


꽤 높은 수준의 exception handling 자랑

exception catch 1 example	exception catch 2 example
	

실행 동작 example 1 (유닛 생성)	실행 동작 example 2 (유닛 이동)
	
그래프에 저장되어 있는 인접지역을 표시해 줍니다.	console 에서 최대한 이용자에게 많은 정보를 보여주기 위해 노력한 부분입니다.





턴 넘기기만 한 경우의 결과물 (보여드리기 위한 자료입니다.)

사용 환경 및 도구

협업 환경

google docs

가장 간단하면서, 최신 동기화 상태를 유지한 상태로 함께 문서 작업을 할 수 있는 최고의 플랫폼이기에 채택하여 사용하였습니다.

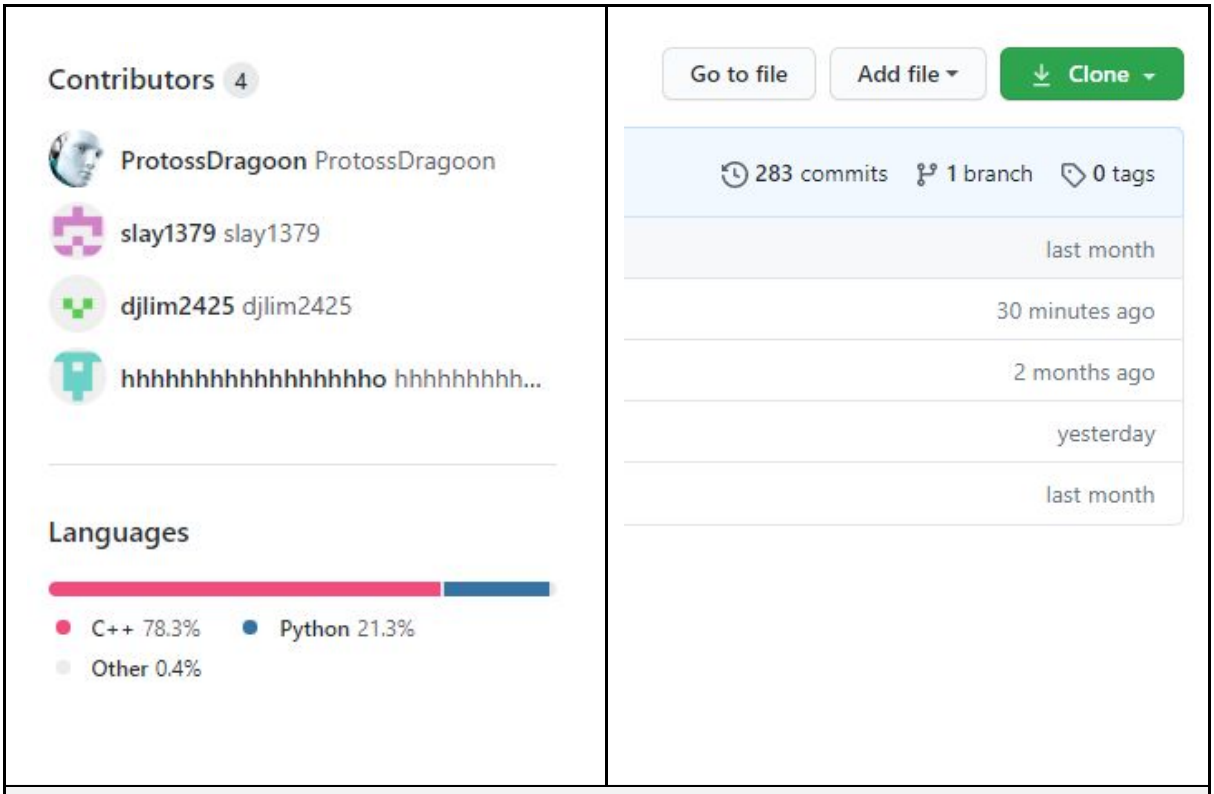
github

저희는 프로젝트의 소스 코드가 적지 않을 것이고, 서로가 서로를 많이 참조해야 하는 상황이 발생할 것 같다고 느꼈기 때문에, 조금 더 좋은 코드 공유 솔루션이 필요했습니다. 둘은 git 의 사용이 처음이었고, 당연히 코드가 증발하는 등 많은 우여곡절이 있었지만, git 은 저희에게 조금 더 빠르게 코드를 비교해서 서로의 소스코드를 반영할 수 있게 도와 주었습니다.

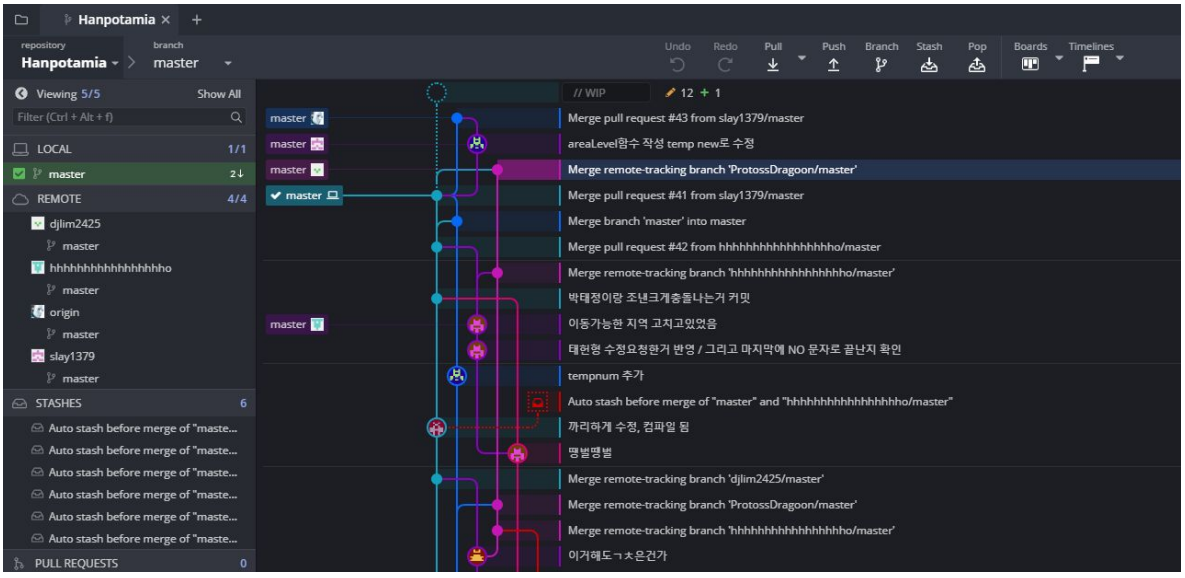
https://github.com/ProtossDragoon/Hanpotamia

gitkraken

git 을 완전히 처음 다루어 보는 친구들이 있었기에, git GUI 는 좋은 콘솔의 대안이었습니다. 서로가 어느 정도까지 작업을 했는지 한 눈에 확인을 할 수 있었기에 서로를 구박하는 용도로도 잘 활용할 수 있었습니다.



Hanpotamia Github



Gitkraken GUI

개인 개발환경

MAC : C Lion Default
Windows 10 64 bit : Visual Studio 2019 Default

소스코드 작성 규칙

절대적인 것은 아니지만, 서로가 서로의 소스 코드를 쉽게 읽을 수 있기 위해 아래 규칙들을 최대한 지키기를 약속했습니다.

참조

- NHN/C++ 코딩 규칙, Google C++ Style Guide 中 쉬운거
- 안용학 교수님이 수업에서 언급했던 규칙

파일명

- 지금 하고 있는 프로젝트의 컨벤션, 안용학교수님의 컨벤션에 따른다.
- 클래스를 설계하는 경우, 파일 앞에 대문자 C 를 붙인이고, 대쉬 (-) 를 붙이고, 대표 class 이름으로 사용하며, C 를 제외하고는 소문자와 언더바를 사용한다. ex : C-rect_base
- 테스트용 (실행 파일) 의 경우, Test- 를 가장 처음에 포함한다. ex : Test-rect_base
- 파일 이름에 대쉬(-) 를 두 개 이상 사용하지 않는다.

함수명

- 일반적인 함수는 소문자로 시작하며, 각 새로운 단어마다 대문자를 사용한다. 언더라인은 사용하지 않는다. ex : myExcitingFunction()
- 접근자와 수정자(get, set)는 변수 이름과 일치시킨다. ex : set_myExcitingMemberVariable()
- True/False 값을 return 하는 경우, 함수 이름은 is 혹은 has 로 시작한다. ex : isHungry()
- private 함수 이름은 언더바(_) 로 시작한다. ex : _dontTouchMe()

타입명

- 타입명은 대문자로 시작하며, 각 새로운 단어마다 대문자를 갖으며 언더라인을 사용하지 않는다. ex : MyRectangle

변수 및 상수명

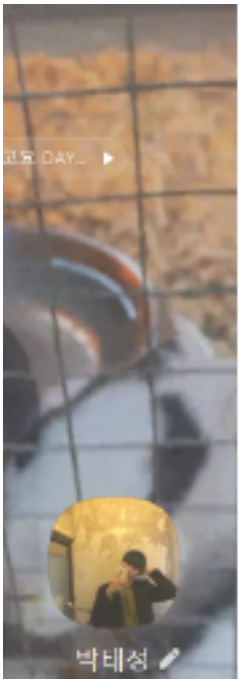
- 변수명은 소문자로 시작하며, 소문자와 언더바만 사용한다.
- static 멤버 변수의 경우 's_' 으로 시작한다. ex : \s_my_exciting_static_variable
- const 멤버 상수는 'k_' 로 시작하며 대소문자를 섞어서 사용한다. ex : \k_days_in_a_week
- 그 외 private 멤버 변수 및 private 함수는 '_' 로 시작한다. ex : _my_private_variable

- 이름은 가능한 설명적으로 짓는다. 공간 절약이 중요한 게 아니라, 코드를 즉시 보고 이해할 수 있어야 한다. ex : num_completed_connections
- 모호한 약어나 의미를 알 수 없는 임의의 문자를 사용하지 않는다. ex : nerr (?)
- 구조체의 데이터 멤버는 일반적인 변수처럼 이름을 짓는다. 클래스처럼 언더라인으로 끝나지 않는다.
- 전역 변수는 특별한 요구사항이 없으며, 거의 사용을 하지 않는다. 만약 사용한다면, 'g_'로 시작하거나 로컬 변수와 구별되는 표시를 한다.

기타

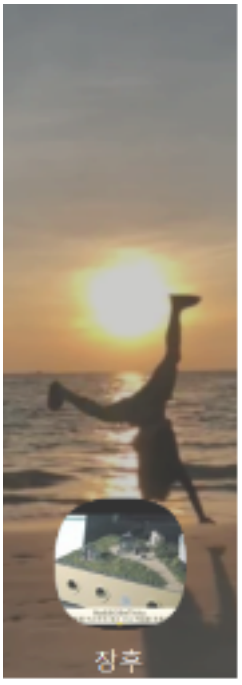
- 들여쓰기는 Tab 을 사용한다.
- 간단한 생성자 초기화는 콜론 초기화로 한다.
- 이항 연산자 (=, >, <, 등..) 앞과 뒤에 공백을 제공한다. ex : a = b + c
- 단항 연산자 앞과 뒤에 공백을 제공하나, (A++), [--BB], {--KK}와 같이 사용할 때는 공백이 없어도 좋다.
- 일부 연산자(“ , “, “ ; “)는 연산자 뒤에 공백을 제공해야 한다. ex : for(i = 0; i < 3; i++)
- brace({)는 분리된 라인에 작성한다.

프로젝트에 대한 개인적 의견



게임을 만든다는 계획을 세우고, 스토리를 만들고, 룰을 만드는 것처럼 어떤 게임을 만들지를 생각하는 것 보다 힘든 것이 처음 설계한대로 개발을 마무리 하는 것이라고 생각했습니다. 하루면 끝날 줄 알았던 작업도 어설픈 설계 때문에 밤을 세워도 끝나지 않아 팀원들이 고생했던 경우도 있었고, 한줄로 끝났어야 할 코드들이 열 줄이 되는 경우도 있었기에 설계 실력의 부족을 실감했습니다.
하지만 그와 별개로 개발 과정에서 얻은 교훈이 있습니다. 하향식의 개발을 할 때 팀 리더의 소양에 관한 것 입니다. 하향식의 소프트웨어 개발에서의 장점은 상위단의 개발자가 청사진을 그려놓으면 하위단의 기능들이 뚝딱뚝딱 만들어지기 때문에 초반에 고려할 부분이 적다는 것인데, 미숙한 설계의 종단에는 수정되어야 할 부분들이 많아지며 그 구조를 바꾸기가 힘들어지는 경우가 발생합니다. 이에 상위단의 개발자는 하위단의 코드를 이해하지 않기 때문에, 결국 하위단의 개발자와 상위단의 개발자가 서로의 개발 영역을 침범하며 불필요한 오버헤드가 발생하게 된다는 것 입니다. 따라서 시범적이거나, 불완전한 기획 일 경우에는 때에 따라 하향식의 코딩 방법을 지양하는것이 좋을 것 같다는 생각을 했습니다.
힘들었지만 얻은 것이 분명 더 많다고 생각합니다. 상기의 의미에서 이번 수업은 온라인 수업임에도 불구하고 아주 유익한 수업이었습니다.

박태정 18011549, 컴퓨터공학과 3학년

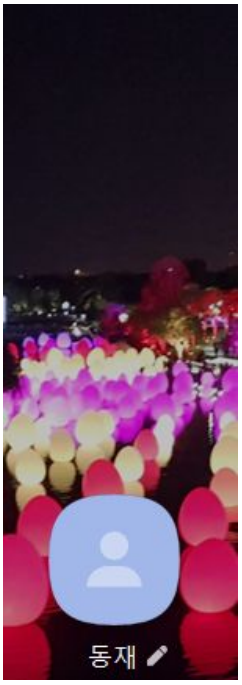


간단한 게임을 만들려고 했던 것이었지만, 이마저도 굉장히 신경쓸 것이 많았습니다. 특히, 클래스를 설계하는 데 있어 다양한 설계패턴을 미리 숙지하지 못한 점과 그에 익숙하지 못했다는 점은 아름다운 코드를 작성하는 것을 못하게 막았습니다..
클래스 설계 원칙을 잘 알았더라도, 그것을 내가 만드려는 프로그램에 잘 적용하는 것은 또 다른 문제였을 것이라는 생각이 듭니다.
클래스기반의 프로그래밍을 하면서 가장 많이 드는 생각은, “클래스를 이렇게 설계하고 이를 객체로 여기고 이를 포함하는 새로운 클래스를 만들었으면 어땠을까” 와 같은 설계를 다시 하고싶다는 마음들이었을 만큼, 클래스 설계는 어려운 부분이었습니다. 특히, 협업하기 좋은 클래스, 서로 독립성을 유지하는 클래스를 잘 설계하는 것은 엄청난 재능이 될 것이라고 생각합니다.
마음이 많이 앞서 다른 프로세스와 동기화를 시킬 때 어떻게 해야 하는가, 이를 고려해서는 어떻게 설계해야 하는가 등 생각을 하다가, 결국 시간에 맞추어 “우선 계획한 부분까지 돌아가게 하자” 라는 마인드로 임하게 되었던 것이 아쉽습니다.
그래도 직접 설계한 논리 기반으로 만든 콘솔 게임을 보니 뿌듯합니다.
그리고 수업을 2학년 둘에게 3학점 수업을 5학점 뺨치는 프로젝트 강도로 만들어서 미안합니다.

이장후 18011573, 컴퓨터공학과 3학년

김태헌 19, 컴퓨터공학과

프로젝트를 비교적 간단한 걸로 하고 끝낼 수도 있었지만, 저희는 한 달 이상 잡고 있을 프로젝튼데 이왕이면 재밌는걸로 하기로 해서 기존의 게임들에는 못 미치지만 게임을 만들기로 했습니다. 이런 규모의 프로젝트는 해본적도 없고 팀으로 다른 사람들과 코드를 짜 합쳐서 프로그램을 만들기는 처음이라 저에 대한 걱정도 됐지만 또 한편으로는 그에 대해 기대도 되고 또 하나를 배운다는 것으로 기대가 앞섰습니다. 하지만 프로젝트를 진행해 보니 코드를 짜는 과정에서 다른 클래스를 더 신경쓰지 못해 생각과는 다르게 제 코드에서 오류들이 발생해 팀원들이 고생을 많이 하기도 했습니다. 이런저런 점에서 많이 힘들었지만 수업과 질문에 친절히 답변해주시는 교수님 18선배들 두 명에게서 너무 많은 것을 배워 너무 유익한 수업이였다고 생각합니다.



임동재 19011471, 컴퓨터공학과

프로젝트를 시작하게 되었을때 그래도 재미있게 설계할 만한 게임을 고르게 되어 기대감이 컸습니다. 설계초반에는 미숙한 코딩실력과 얼마 배우지 않은 c++을 가지고 설계를 시작했습니다. 하지만 설계를 계속하면서 게임의 규모도 생각보다 크고, 필요로 할 만한 설계패턴들이 많다는 것을 알게 되었습니다. 이러한 설계패턴들을 공부했지만 충분히 이해하지 못해 사용하지 못한 점이 아쉬웠습니다. 그래서 방학동안 공부하여 다음에 이러한 프로젝트를 다시 하게 된다면 쉽게 사용할 수 있게 목표를 세웠습니다.
고급c프로그래밍과 이 수업을 병행하며 들었는데 고c에서 배우자마자 프로젝트에서 바로 사용하여 기억에 잘 남고 사용법을 확실하게 알 수 있어서 뜻깊고 유익한 수업이었습니다.