CS-112 Project Proposal: JRPG Combat Simulation and League Manager

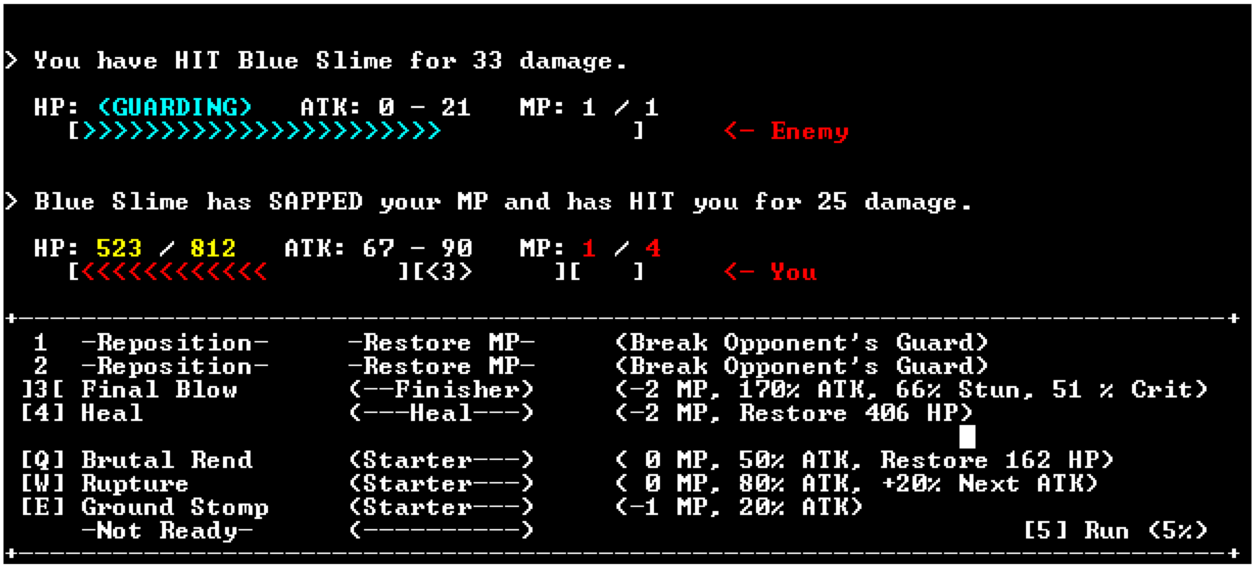
CS-112 Project

Yazdan Ali Khan 2024665, Muhammed Hammad Shahid 2024389 [FEE Section E]

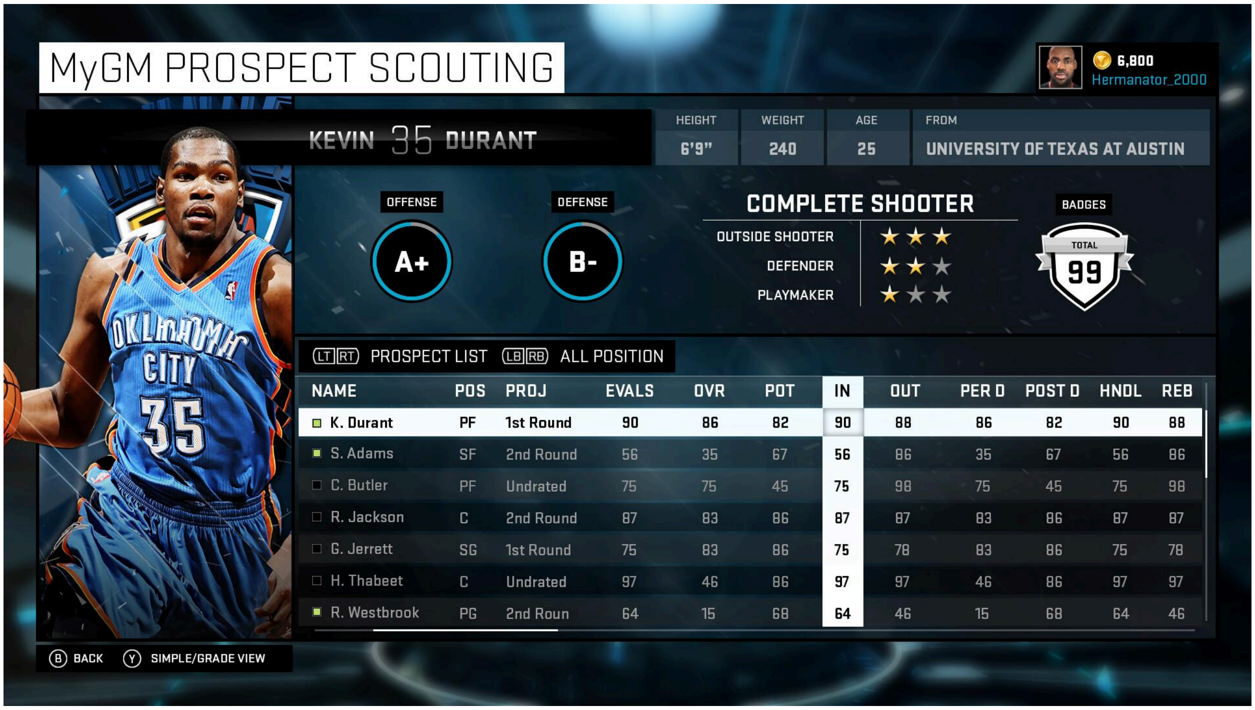
Overview

This project blends the turn based combat of a text based Japanese Role-Playing Game (JRPG) with the strategy and structure of a sports league manager. Built in C++, the simulator showcases core object-oriented programming (OOP) concepts—encapsulation, inheritance, polymorphism, and abstraction. It sets up a competitive environment where AI- and player-controlled teams, made up of unique characters with varying randomly-generated stats, face off in turn-based battles until every team plays each other once after which the top 2 teams face in a final game that decides the winner.

Below are some images of reference projects that we took inspiration from



```
> You have HIT Blue Slime for 33 damage.

  HP: <GUARDING>    ATK: 0 - 21    MP: 1 / 1
    [>>>>>>>>>>>>>>>>>>>>>>>                ]      <- Enemy

> Blue Slime has SAPPED your MP and has HIT you for 25 damage.

  HP: 523 / 812   ATK: 67 - 90    MP: 1 / 4
    [<<<<<<<<<<<<             ][<3>     ][    ]      <- You

+-----------------------------------------------------------------------------+
  1   -Reposition-      -Restore MP-      (Break Opponent's Guard)
  2   -Reposition-      -Restore MP-      (Break Opponent's Guard)
 ]3[ Final Blow         (--Finisher)      (-2 MP, 170% ATK, 66% Stun, 51 % Crit)
 [4] Heal               (---Heal---)      (-2 MP, Restore 406 HP)

 [Q] Brutal Rend        (Starter---)      ( 0 MP, 50% ATK, Restore 162 HP)
 [W] Rupture            (Starter---)      ( 0 MP, 80% ATK, +20% Next ATK)
 [E] Ground Stomp       (Starter---)      (-1 MP, 20% ATK)
     -Not Ready-        (----------)                              [5] Run (5%)
+-----------------------------------------------------------------------------+
```



MyGM PROSPECT SCOUTING

6,800 Hermanator_2000

KEVIN 35 DURANT

HEIGHT	WEIGHT	AGE	FROM
6'9"	240	25	UNIVERSITY OF TEXAS AT AUSTIN

OFFENSE A+

DEFENSE B-

COMPLETE SHOOTER

OUTSIDE SHOOTER	★★★
DEFENDER	★★☆
PLAYMAKER	★☆☆

BADGES TOTAL 99

LT RT PROSPECT LIST LB RB ALL POSITION

NAME	POS	PROJ	EVALS	OVR	POT	IN	OUT	PER D	POST D	HNDL	REB
K. Durant	PF	1st Round	90	86	82	90	88	86	82	90	88
S. Adams	SF	2nd Round	56	35	67	56	86	35	67	56	86
C. Butler	PF	Undrated	75	75	45	75	98	75	45	75	98
R. Jackson	C	2nd Round	87	83	86	87	87	83	86	87	87
G. Jerrett	SG	1st Round	75	83	86	75	78	83	86	75	78
H. Thabeet	C	Undrated	97	46	86	97	97	46	86	97	97
R. Westbrook	PG	2nd Roun	64	15	68	64	46	15	68	64	46

:-Please dont pay attention to the fancy graphics or images in this picture and instead try to consider how the information is arranged and what kind of information is considered useful by sports sims such as these

Objectives

- Recreate turn-based JRPG combat using OOP design.
- Build a team management system with randomized character stats.
- Simulate a full season: scheduled matches, stat tracking, and a final Battle
- Enable user input during battles (like attacks, buffs, and healing).
- Apply and demonstrate OOP fundamentals through structured class design and modular methods.

Key-Features

- **Entity System:** At the heart is a base class, `Entity`, with derived-class `Player` defining specific behaviors and overriding functions
- **Team System:** Two main team types—AI-controlled `Team` and user-driven `PlayerTeam`— Contain Entities and Player class objects.
- **Battle Manager:** Orchestrates combat flow, Both runs simulated (CPU) matches and player driven matches triggers player and enemy actions, and checks win/loss conditions.
- **Schedule System:** Handles matchups, tracks days, and ensures seasonal consistency. Will be stored in the Game Manager Class
- **Game Manager:** Ties everything together—manages the season, trading, and the climactic final match.
- **Console UI:** A simple but engaging color-coded console interface using `windows.h`.

Object Oriented Design

- **Inheritance:** For instance, `Player` derives from `Entity`, and `PlayerTeam` from `Team`.
- **Polymorphism:** Methods like `Actions()` and `Choose_Target()` behave differently depending on which class calls them.
- **Encapsulation:** Attributes are kept secure and accessed through getters and setters.
- **Abstraction:** Complex operations like resolving combat or generating stats are tucked into manager classes, keeping them separate from other systems.

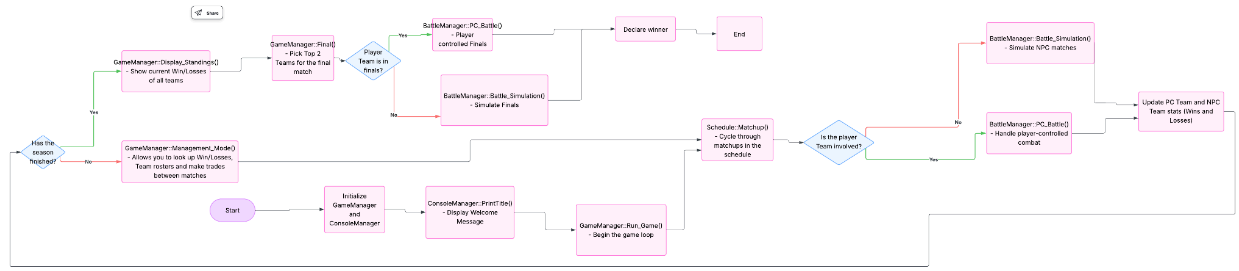
Tools

Language: C++

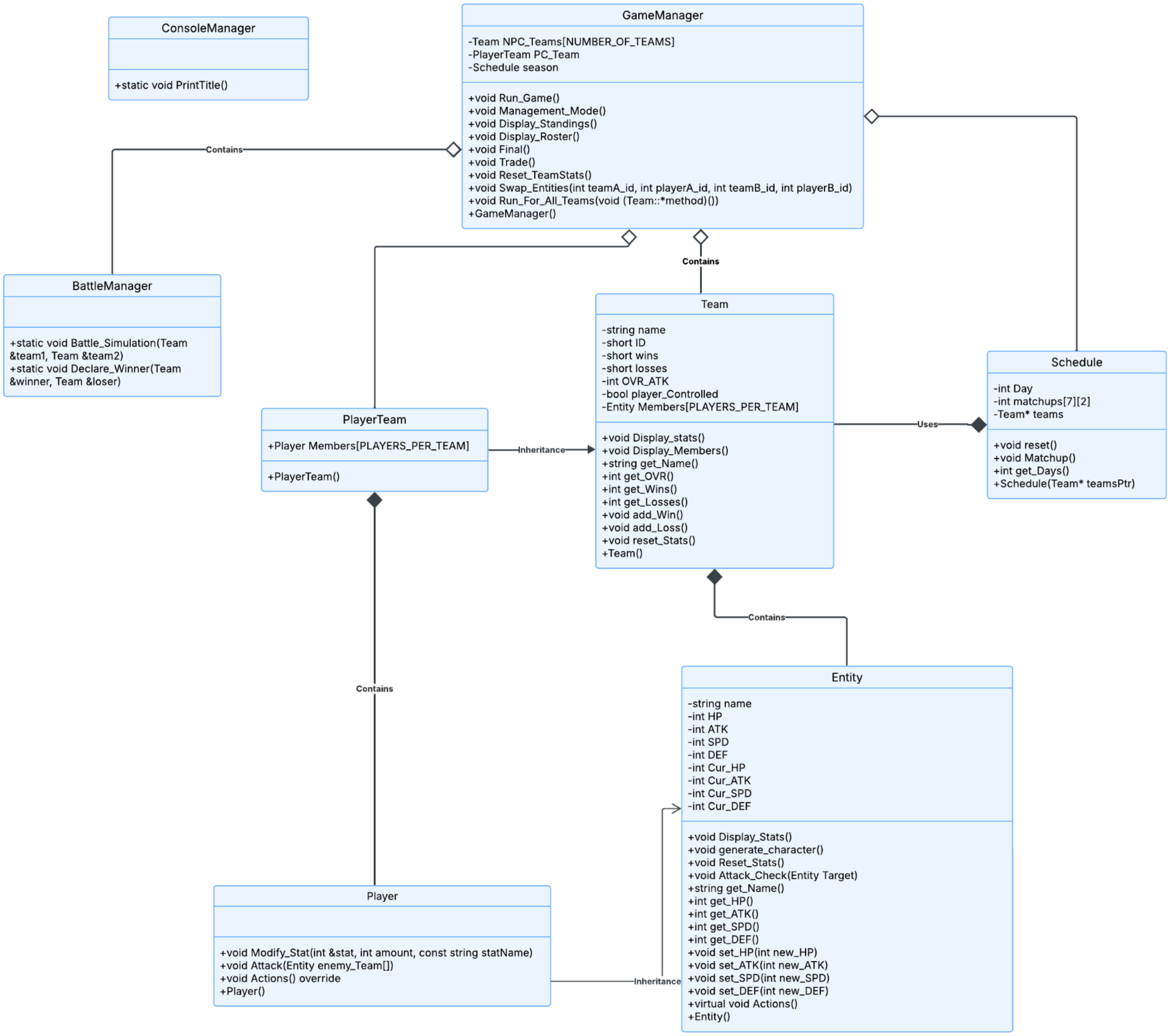
Libraries Used: `<iostream>`, `<vector>`, `<cstdlib>`, `<ctime>`, `<windows.h>`, `<string>`

Platform: Windows (due to use of `windows.h` and `SetConsoleTextAttribute` for UI coloring)

Flowchart and Diagrams


Share
Start
Initialize GameManager and ConsoleManager
ConsoleManager::PrintTitle() - Display Welcome Message
GameManager::Run_Game() - Begin the game loop
Schedule::Matchup() - Cycle through matchups in the schedule
Is the player Team involved?
No
Yes
BattleManager::Battle_Simulation() - Simulate NPC matches
BattleManager::PC_Battle() - Handle player-controlled combat
Update PC Team and NPC Team stats (Wins and Losses)
Has the season finished?
Yes
No
GameManager::Display_Standings() - Show current Win/Losses of all teams
GameManager::Management_Mode() - Allows you to look up Win/Losses, Team rosters and make trades between matches
GameManager::Final() - Pick Top 2 Teams for the final match
Player Team is in finals?
Yes
No
BattleManager::PC_Battle() - Player controlled Finals
BattleManager::Battle_Simulation() - Simulate Finals
Declare winner
End

JRPG Manager



Conclusion

This project successfully demonstrates how object-oriented programming can be used to model complex, interactive systems like a game simulation. It showcases an integration of core OOP concepts in a creative and engaging way, making it not only functional but also a solid demonstration of real-world application of C++.