Team FDR: Final Report

Ryan O'Leary, Debbie Wong, Fernando Mercado

Features Built-Out;

- Stats Plugin: We store the statistics for each playable character in a json file, and using our plugin we are able to read in the statistics and display them in the UI. We utilize the built in json deserializer for Unity for the file parsing.
- Character Selection screen: We implemented a screen where the user can select a character and the statistics and picture of the character is displayed side by side, similar to the character select screen from Super Smash Bros.
- Transition from Character Selection screen to Battle Scene: We implemented a button that transitions the user from the character select screen to the battle scene once they are ready to battle.
- Battle Screen: The player can select items to boost stats of their character and select attacks from their respective menus. There is a dialogue screen that shows which attack the AI uses and how much damage each character deals. The rounds continue back and forth until one character passes out. Once a character passes out, we are taken back to the character select screen.
- Title Menu: a simple menu that you can press start to go to the character select screen or exit to quit the game.
- Sound effects: buttons and background music have distinct sounds.
- AI decision tree: We designed a decision tree for the AI to determine what move to use next based on the enemy state and the AI's state. The AI will use the most optimal move every turn.

Stat Plugin (loads character stats from each respective Json):

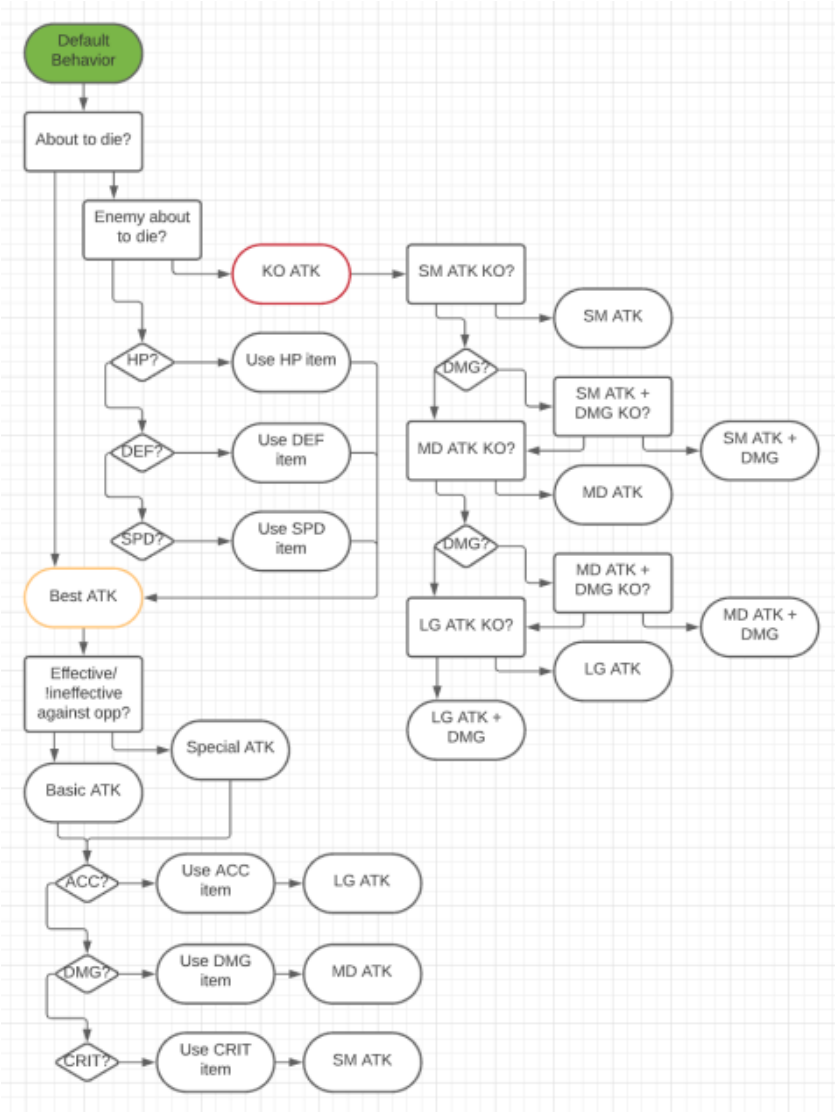
```
public Character LoadJson(string name)
{
    //string name = "Squirtle";
    using (StreamReader r = new StreamReader("Assets/Characters/" + name + ".json"))
    {
        string json = r.ReadToEnd();
        Character character = JsonUtility.FromJson<Character>(json);

        return character;
    }
}

[System.Serializable]
public class Character
{
    public int HP;
    public int ATK;
    public int DEF;
    public int SPD;
    public string TYPE;
    public List<string> basicAttackNames;
    public List<int> basicAttackDMG;
    public List<int> basicAttackACC;
    public List<string> specialAttackNames;
    public List<int> specialAttackDMG;
    public List<int> specialAttackACC;
}
```

*changed path to Application.streamingAssetsPath + "Resources/Data/StreamingAssets/" to have the files be included in the packaged game

Example Json File (we used pokemon as placeholders but plan to create our own characters):

```
{
  "HP": 190,
  "ATK": 48,
  "DEF": 65,
  "SPD": 43,
  "TYPE": "Water",
  "basicAttackNames": ["Tackle","Tail Whip", "Shell Spin"],
  "basicAttackDMG": [40,45,47],
  "basicAttackACC": [100,85,65],
  "specialAttackNames": ["Brine", "Waterfall",  "Hydro Pump"],
  "specialAttackDMG": [45,48,52],
  "specialAttackACC": [100,80,50]
}
```


AI Decision Tree:



Division of Labor:
We all did equal amounts of work since we planned out and wrote all the code together in person.

Setbacks:
We had a significant setback with trying to get the plugin to work. At first we tried to build a native plugin using C++, which involved creating DLL files and importing them into our Unity C# files. However, we tried for many days and for some reason we were never able to get Unity to find the entry point for our main function (it seemed to never be exported correctly). Eventually, we decided that trying to get the DLL file to work was not worth it as we were a bit behind schedule at that point, so we scrapped the C++ stats plugin and the DLL file approach and ended up rewriting the stats plugin in C# and using json files, since Unity contained a very useful JsonUtility library. Besides the plugin, we had some trouble getting our animations to play correctly as well as having the canvas for each scene scale to the screen size once exported.

Game Trailer Link:
https://youtu.be/SHDwahXuU3E