GAM111 Assessment 2
Pet Battle Game
Due: Week 12
Weighting: 35%

Brief

You will create a pet battle game similar to the battle system in games like Poke'mon or Outernauts. As a battle pet trainer, the game sees the player commanding pets against an AI opponent in turn based combat. Each trainer will command their battle pets to attack the opponent, until one of the trainers has no pets remaining. Using a combination of damaging, defensive and healing abilities, the trainers will compete to see who is the battle pet champion.

Details

Each battle pet trainer will have three unique battle pets, but only one battle pet will be active for each trainer at any given time. A battle pet must have an element type, and a series of abilities. Abilities can be attack, defend or apply status effects to itself of its opponent. Each attack ability will inflict elemental damage, with some elemental effects being weaker or stronger against others.

Each turn, the trainers will have, at minimum, the following options:

- Use a pet ability
- Change active pets

Once each trainer chooses their actions, you must resolve those choices. You must design this system yourself, and must deal with all possible situations - eg the first pet attack kills the other, or a pet changed before the other has attacked.

The game can be in 2D or 3D, and can use free art. The game assets must reflect the state of the game, including attack and damage animations. This can be simple (eg shaking a sprite when it takes damage), but must be present to provide additional information and feedback to the player.

The game must make appropriate use of audio. Battle pets should play sound when summoned, hurt, knocked out, etc. UI element should have sounds (button clicks).

One battle pet trainer will be controlled by the player, and the other will be controlled by a simple AI.

Requirements

- You must have the following UI components using a UI Framework (UnityUI/nGUI)
    - Title Screen
    - Game Screen
        - A visual representation of each active pet
        - The status of each active pet must be visible (health/status effects applied)
        - A command window for choosing player input (command pet/change pet)
        - A status window for displaying game state updates (pet attacks/player summons new pet/pet dies)
- Design a combat system
    - Each trainer will choose an action, then the actions will be resolved. You must decide how to resolve these actions.
- Pet config and elemental strength/weakness table must be abstracted using an external file solution (eg XML, Spreadsheet and CSV), and/or Scriptable Objects. There must be at least six unique pet types.
- An elemental system to give each battle pet a series of strengths and weaknesses.
    - Each pet must be of at least one element type
    - Each pet must be able to deal damage of a certain elemental type
    - There must be at least three element types, with each being strong and weak vs at least one other.
- Each pet must have at least four abilities. Each of these abilities must also have a cooldown before it can be used again. At least one ability must have a cooldown of one turn to ensure the player always has an option.
    - An attack with elemental damage
    - An attack with normal damage
    - An attack that applies a status effect to the enemy
    - An defensive ability (eg defense buff/healing)
- There must be at least four types of status effects that can be applied. These status effects must be removed from a pet it is replaced with a new active pet.
    - A damage over time
    - A heal over time
    - A stat buff
    - A stat debuff
- Each pet must use at least these four stats
    - Strength - A stat that affects how much damage is dealt
    - Defense - A stat that affects how much damage is mitigated
    - Health - how many hitpoints your pet will have
    - Speed - how quickly your pet will act

Examples of Optional Extras

- Additional pet stats (eg critical hits) and status effects (eg sleep)
- Loading and Saving game state
- A roster screen for selecting your pets before battle
- Pet levelling up and unlocking new abilities
- Make effective use of source control and assignment submission

Submission Requirements

- Upload complete Unity project and executable to either Campus Online, or a third party cloud service such as Drop Box with a link submitted to Campus Online.
- Include a short readme file outlining extra features you have added, any known bugs, and some brief instructions on how to play.

Estimated Complexity

Requirement	Complexity	Time
UI	Low	Medium
Combat System	High	High
Externalised Pet Data	Medium	Low
Elemental System	Medium	Low
Ability System and Pet Use	High	High
Pet Stat System	Low	Low

Marking Guide

Criteria	Weighting	Details
Brief Requirements	30%	If you have met the brief requirements, and how well each requirement has been met. Requirements have been implemented using techniques as demonstrated in the unit.
Sophistication & Elegance	30%	The quality of the implementation. Code is well written, no magic numbers, and code reuse is minimal. How bug free the code is, and use of appropriate coding techniques.
Code Formatting	20%	The quality of your code layout. Comments are well clear, and white space is well formatted. Appropriate file name, variable names and class names have been used. Project files are well laid out and easy to navigate.
Extra Features	20%	Additional features added that are not included in the brief. This may be a single complex addition, or several smaller features.