INFO1113 Assignment Report

Author: Shaoqing Ni (Steven)

Object-Oriented Programming Decisions

For the structure and design of the game, I employed various object-oriented programming principles to achieve modular and maintainable code. Here's a breakdown of the key decisions:

Package Organization

I categorized different classes into distinct packages based on their specific functions. Examples of these packages include:

- `BackGround`
- `Buttons`
- `DFS`
- `Monsters`
- `Tower`
- `WizardTD` (default package)

Each package serves a unique role within the game.

Button Interfaces and Inheritance

For game buttons, I introduced two interfaces: `Button` and `UpgradeButton`. I also developed a base class named `BaseButton`, which encapsulates common attributes and methods such as button position, `isClicked`, and `ButtonClicked`. Other button classes inherit from `BaseButton` and override its methods as required, promoting code reuse and polymorphism.

Data Management Classes

Two distinct classes were created for data management:

- **`KeyValuePair`** : Manages related data pairs, ensuring all operations related to key-value pairs remain encapsulated within this class.
- **`DrawingData`** : Focuses on data essential for GUI drawings. By compartmentalizing the data used for drawing away from algorithmic logic, the code remains cleaner, with each class focusing solely on its primary function.

Enumeration for Game States

The game's various states are defined and managed using the `GameState` enumeration.

`GameState` Enumeration

This enumeration offers a lucid and type-safe method to signify the potential states in the game:

- `RUNNING` : Active game state where all the logic is processed.
- `WIN` : State signifying the player's victory.
- `LOSE` : State indicating the player's loss.
- `PAUSED` : Temporarily halts the game, usually by the player's action.

Utilizing an enumeration for the game states provides numerous advantages:

1. **Clarity**: The possible game states are readily apparent to any code reader.
2. **Type Safety**: The use of enums assures the assignment of only valid states, thus minimizing the potential for errors.
3. **Ease of Maintenance**: Adjusting or adding states in the future can be effortlessly done in one central place.

Implementation of the IceBallTower Extension

The primary goal of this extension was to diversify the gameplay and offer the player an additional strategic layer by introducing a new type of tower: the `IceBallTower`. Here's a detailed explanation of its implementation:

1. IceBallTower Characteristics:

- **Type**: Unlike the initial single tower type (`fireball tower`), this new tower is called `IceBallTower`.
- **Properties**: The `IceBallTower` has predetermined properties:
  - **Range**: It has a fixed shooting range.
  - **Damage**: It inflicts a set amount of damage to monsters.
  - **Fire Frequency**: It shoots iceballs at a consistent rate.

2. Unique Mechanic: Slowing Effect:

The primary distinction of the `IceBallTower` is its unique ability to slow down monsters.

- **Iceball Impact**: Whenever an iceball from this tower hits a monster, it temporarily reduces the monster's speed. For instance, a monster with an initial speed of `10` will have its speed reduced to `8` upon being hit by an iceball. This slow effect is represented by a `freeze` factor of `0.8`.

3. Tower Upgrades:

Differentiating it further from the `fireball tower`, the `IceBallTower` has a specialized upgrade system focusing on its slowing capability.

- **Available Upgrades**: Unlike the `fireball tower` which has upgrades for damage, speed, and range, the `IceBallTower` only offers a single type of upgrade - `freeze`.
- **Upgrade Mechanism**:
  - **Freeze Level**: The `freeze` upgrade has multiple levels, maxing out at level `5`.

- **Intensity**: With each upgrade, the intensity of the slowing effect increases. At the maximum freeze level (level 5), the slowdown effect doubles. For example, if a monster with an initial speed of `10` and been shot by a maximum freeze level iceball tower, its speed would be slowed down to `4`.

4. Integration with Gameplay:

With the inclusion of the `IceBallTower`, players now have more strategic decisions to make. They can choose between using the `fireball tower` for direct damage or the `IceBallTower` to control the pace of monsters. This choice becomes especially crucial when players have to manage hordes of monsters or deal with particularly fast ones.

In conclusion, the addition of the `IceBallTower` enriches the gameplay by introducing new tactical possibilities. Whether players prioritize damage or control, the game now caters to a broader range of playstyles, making it more engaging and replayable.