Documentation - Nimblest

Jan Nekarda

Intoduction

Nimblest is a turn based 2D chaser. Map is split into tiles, some of them have special effects including:

- Teleporting the character staying on it
- Freezing the character for some time
- Breaking down after being used

Goal of the player is to get to target tile(s). There are characters chasing the player, that player has to avoid. Touching any of them means failure.

User interface

Game has simple graphical interface and it is controlled by a keyboard.

Key features:

- Escaping enemy creatures with simple AI
- Using enviroment to your advantage
- Simple graphical interface

Program architecture

`GameData` is a class that manages all game resources that need to be shared and keeps track of game state. It has main loop method which starts after the game initializes and runs until player chooses to quit. It holds pointer to `LevelData` which holds data that depend on current level.

`LevelData` holds the map and all non-player sprites. They are loaded from text files when needed.

There are two main game entities: sprites and tiles. All sprites are a child of abstract class `BaseSprite`. It's children are `Player` and `Enemy`.

`Player`'s movement is controlled by player input and enemy simply goes to the tile that puts him closest to the player(which is deterministic and player can quickly learn to use it to their advantage).

Map is a two dimensional array of `BaseTile`s. `BaseTile` is used as a root in tile hierarchy and also as a normal tile.

Technical documentation

In `main` initialization of loaded data is executed. `GameData` class provides `MainLoop` method that goes on until the game is over. Based on `GameState state` which holds the phase game is in a function that takes care of the state is executed(some states just change image so they share the function). It also has metod for loading level and drawing UI images.

`BaseTile` holds coordinates and data required for drawing one tile of given type along with data shared by all tiles such as size multiplier and offsets(to make sure isometric draw calls puts the tiles on the right spot). It has method `SpriteEntered` in which the effect of a tile is executed.

`LevelData` store two dimesional array of `BaseTile` pointers and vector of `BaseSprites`(Enemies). Map and sprites are loaded from file using `LoadMap` function in it's constructor.

`BaseSprite` holds it's coordinates, current freeze duration, direction in which the sprite is going and static data for drawing. It has methods for moving the sprite and function that resolves death of the sprite. `Player` has also input processing and `Enemy` has function `TurnToPlayer` to set correct direction before moving.