DAU 2024 - Game Project Documentation

Author: Jiayu Hu

Overview

My game project is organized into a Game object, which includes a state machine, with two main states: Start State and Play State. In GameLoop.cpp, Game is initalized, updated, rendered and exited.

Note: In Visual Studio, to display the code structure more clearly, please use the "Show All File" button in Solution Explorer to show the project's folders instead of filters.

Pre-Programming Code Acknowlegement

The following system/engine blocks are pre-programmed before Jan 12, 2024 by myself:

- Tile system (Tile class and TileMap class)
- State machine class
- Entity class
- Game Level class
- Player class

Main Game

The main game's state machine manages the overall flow of the game and transitions between the following states:

1. **Start State**: Initial state when the game is launched.
2. **Play State**: The main state where the game is actively played.

Game Components

Play State

The Play State's variables include:

- **LevelMaker**: Makes a Game Level randomly using algorithms.
- **GameLevel**: Represents the game level, including map, entities and game objects.
- **Player**: Represents the character that could be controlled by the player.

The Play State's functions include:

- **Enter**, **Update**, **Render**, **Exit** for game loop.
- **SpawnEnemies**: Add enemy entities (slime, bat) using random algorithms to the game level.

Level Maker

The Level Maker's variables include:

- **Tilemap**: Represents the layout of the game world.

    - **Tile**: Represents the Tile of the Tilemap.
        - The Tile's unique ID, which is the frame number on the sprite sheet.
        - The Tile's position, dimension and sprite.
- **MapData**: A 2D vector including ID representation of the tilemap. Every ID in the vector represents a Tile.
- **NewLevel**: A new GameLevel that is generated and will be passed to the Play State.

The Level Maker's functions include:

- **GenerateData**: Returns mapData by looping through the whole map and generate different tile's ID at the 2D vector randomly.
- **GenerateDecor**: Adds decorations (rock, grass, tree, etc) as Game Objects by looping through the whole map and adding them to different locations randomly.
- **Generate**: Based on the return values of GenerateData and GenerateDecor during runtime, generates a new Game Level and return it.

Game Level

The Game Level's variables include:

- **Tilemap**: Represents the layout of the game world.
- **Entity**: Represents living entities present in the game, including Slime and Bat.
    - The Entity's position, dimension and sprite.
    - The Entity's own StateMachine, direction, and boolean value representing it is dead or not.
- **GameObject**: Represents inanimate objects present in the game, including rock, tree, and grass, etc.
    - The GameObject's position, dimension and sprite.

State Machine

The State Machine's variables include:

- A map including pairs of State names and objects.
- A pointer representing the current state.

The State Machine's functions include:

- **AddState**: Adds a new State object it the map.
- **ChangeState**: Change the current state to another state specified.

Base State

The Base State is an abstract base class for all State classes. It includes virtual funtions that can be overridden by specific states.

Player Entity

The Player entity's variables include:

- **State Machine**: Manages different states such as Idle, Walking, Jump, Falling, Dead, Win, etc.
- **score**: Represents the score player gets from elimilating Slime or Bat.
- **isWin**: Represents if player wins the current level or not.

Slime Entity

The Slime entity's variables include:

- **State Machine**: Manages states like Moving, Chasing, Dead, etc.

Bat Entity

The Bat entity's variables include:

- **State Machine**: Manages states like Flying, Dead, etc.