Overview

The game engine HaxeFlixel and the high-level, open-source programming language Haxe were used to create this project. It has several states, including menu, play, and start screens, in addition to a unique class for controlling squares within the game environment. Below is a description of each file in the project, its purpose, and important functions.

1. AssetPaths.hx

Purpose:

The routes to different assets, including audio, music, and graphics, that the game requires are managed and defined using AssetPaths.hx. Asset paths can be updated or reused more easily across the project by centralizing the routes.

Key Features:

- **Asset path management:** Helps in managing file paths for images, music, and other game-related assets.
- No specific methods but is vital for keeping the asset references organized.

2. Main.hx

Purpose:

The project's entry point is Main.hx. It launches the game's starting state, initializes the game, and configures the window's parameters.

Key Features:

- **Main function:** This file contains the main function that sets up the game window and frame rate.
- **State management:** This file likely manages the transition between game states.

3. MenuState.hx

Purpose:

The main menu of the game, handled by MenuState.hx, lets players go to various sections including "Play" and "Settings."

Key Features:

- **Menu items:** The file sets up clickable buttons such as “Play” to allow navigation to the gameplay.
- **Graphic assets:** Loads images for buttons and menu backgrounds.
- **Sound integration:** Integrates button click sounds and background music.
- **State transitions:** Switches to PlayState when the player clicks “Play.”

4. PlayState.hx

Purpose:

The primary gameplay mechanics are defined in PlayState.hx. It has logic for managing player actions, grid layouts, game events, and the depiction of squares and other game elements.

Key Features:

- **Grid management:** Contains methods for generating different grids (like grid6, grid7, etc.) using Square objects.
- **Square objects:** Manages and positions Square objects on the screen to represent various gameplay elements.
- **Gameplay logic:** Implements logic for interaction between game objects and player actions.

5. Square.hx

Purpose:

Square.hx defines a class used to represent grid elements in the game. These could be game tiles or objects that form the game world.

Key Features:

- **Attributes:** Each Square object can be defined with specific attributes such as type (e.g., "Terminus," "Right," "Intersection") and coordinates.
- **Grid positioning:** Responsible for assigning positions to squares on a grid and initializing their properties.

6. StartState.hx

Purpose:

StartState.hx handles the initial start screen, which displays options like "Play" to the user. It sets up the introductory interface when the game is launched.

Key Features:

- **Background setup:** Sets up the initial background and elements for the start screen.
- **Sound effects:** Plays background music and button click sounds.
- **Button interaction:** Handles interaction with the "Play" button, transitioning the game to MenuState.

USAGE NOTES

1. **Asset Management:** Be sure to update the asset paths in AssetPaths.hx when adding new graphics or sounds to the project.

2. **State Transitions:** The project uses a state-based approach where each state like MenuState or PlayState has distinct functionality. Ensure smooth transitions between states by using FlxG.switchState().

3. **Sound Integration:** The project integrates sound via FlxSound and uses external assets from websites like Freesound. Make sure to attribute any external resources appropriately.

4. **Grid Management:** The grids used in the gameplay are defined dynamically in PlayState.hx. You can adjust grid size and square types by modifying the methods that create these grids.

REFERENCES

- HaxeFlixel documentation: https://haxeflixel.com/documentation/
- Button sound: Vilkas_Sound on Freesound
- Cheer sound: GregorQuendel on Freesound
- Background music: Timbre on Freesound