Program Class Documentation

Table of Contents

- Global Summary/External Dependencies: Page 2

- Global Constants Section: Page 2

- Outside Game Methods: Page 3

- Main Game Methods: Page 4

- Level Editor Methods and Fields: Page 5-6

Global Summary: Provides and calls all of the methods needed to run a game where the user controls a character and dodges waves of enemies. The user can make their own levels and play them using a dual-mode level editor.

External Dependencies: Coin.cs, Enemy,cs, GameArea.cs, Level.cs, LevelBuilder.cs, Player.cs

Global Constants Section:

ConsoleKey HOW\_TO\_PLAY: The input that calls the Instructions() method on the title screen.

ConsoleKey NEW\_GAME: The input that allows the NewGame() method to be called from the first round of the game.

ConsoleKey CURRENT\_ROUND: The input that allows the NewGame() method to be called from the round the user last exited from.

ConsoleKey EDITOR\_MODE: The input that calls the LevelEditorMode() method directly from the title screen.

ConsoleKey EXIT\_TO\_DESKTOP: The input that calls the Environment.Exit() method directly from the title screen.

ConsoleKey SIMPLE\_EDITOR: The input that calls the SimpleEditor() method from the GetEditorMode() method.

ConsoleKey ADVANCED\_EDITOR: The input that calls the AdvancedEditor() method from the GetEditorMode() method.

ConsoleKey EXIT\_EDITOR: The input that calls the “return” keyword from the GetEditorMode() method.

ConsoleKey CHOICE\_1 and ConsoleKey CHOICE\_2: The inputs corresponding to the two choices in the TwoKeyChoices() validator method.

ConsoleKey PAUSE\_RESTART: The input that allows the current round to be restarted from the Pause() method.

ConsoleKey PAUSE\_QUIT\_TO\_MENU: The input that ends the current round and returns to the main menu from the Pause() method.

ConsoleKey PAUSE\_QUIT\_TO\_DESKTOP: The input that calls the Environment.Exit() method from the Pause() method.

ConsoleKey WINSCREEN\_EXIT: The input that terminates the win screen and returns the user to the main menu.

ConsoleColor EDITOR\_COLOR: The color that the top text appears in when the user is editing their level.

Int EDITOR\_X: The limits imposed on the level editor (See Program.cs).

Static void Main() summary: Starts the program, gets the levels that will be generated, and runs the main game loops. Returns void. Try/Catch implemented (see DisplayCrashInfo(Exception ex))

Outside Game Methods

Private static void Maximize() summary: Takes no arguments, maximizes the console window size to fit the size of the screen the user is using, and returns void.

Static void TitleScreen(ref int currentLevelIndex, ref bool gameInProgress) summary: Takes two parameters, a ref int corresponding to the INDEX of the current level in the levels array, and a ref bool determining whether the program is currently running a game. Displays the title, the options, and the credits, and returns void

Static void Instructions(int menuX, int menuY) summary: Takes two integers corresponding to the X and Y positions of the instructions. Tells the user how to play the game and returns void.

Static void DisplayCrashInfo(Exception ex) summary: Takes an Exception and displays in red the stack trace of the Exception as well as its corresponding ex.Message. Returns void.

Main Game Methods

Static void MoveToNextRoundScreen(int currentLevelIndex, int numberOfLevels) summary: Takes two integers, the first corresponding to the current INDEX in the levels array, and the second corresponding to the number of levels in the game. Clears the console screen and displays a countdown to the next level with a beep after each second in the countdown. Returns void

Static void WinScreen() summary: Takes no arguments, displays a victory message and returns void when the user presses the WINSCREEN\_EXIT key (see Global Constants Section).

Static void DeathScreen(ref bool gameInProgress) summary: Takes a ref bool corresponding to whether the game should be running after the method ends. Lets the user know they have died and gives them the choice to restart the current round, exit to menu, or exit to desktop. Returns void as the game’s state is controlled by the reference bool gameInProgress and not by a return value at this point.

Static ConsoleKey Pause(GameArea area) summary: Takes the current game area as an argument, and uses its dimensions to display a centered pause message at the bottom of the play area. Gives the user the choice between restarting the current round, qutting to menu, or quitting to desktop. Returns a ConsoleKey based on the choice the user makes.

Static void PrepareLevel(Level current) summary: Takes the current Level class instance as an argument and renders the play area, the coins, the player and the enemies. Begins playing the level’s specific music as well. Returns void.

NewGame(Level current, ref int currentLevelIndex, int numberOfLevels, ref bool gameInProgress) summary: Takes the current Level class instance, the index of the current level in the levels array, the total number of levels (levels.Length) and the gameInProgress bool which is used to stop or start the game wherever necessary. Contains the main game loop which calls all of the methods necessary to get inputs, move the characters, and draw what is happening on screen. Also calls the methods that check collision. When the game loop is broken out of, it then checks to see if it was because the player won or lost and displays the correct screens based on this. Returns void.

Level Editor Methods and Fields

Public static readonly<int, string> Dictionary editorSoundtracks summary: Uses an integer inputted by the user in order to locate the correct file path (string) of the level soundtrack the user wishes to play in the level that they are designing.

Public static readonly Dictionary<int, ConsoleColor> colors summary: Uses an integer inputted by the user in order to return a ConsoleColor used to control the colors of the level elements when the user is designing their level.

Static void DisplayColors() summary: Takes no arguments, loops through the colors dictionary displaying the numbers and their corresponding ConsoleColor values in their respective colors. Black will have a background color of white so it is visible to the user on the black console. Returns void

Static void DisplaySoundtrack() summary: Takes no arguments. Displays the soundtrack options that are NOT directly linked to the editorSoundtracks dictionary itself (see editorSoundtracks). Returns void

Static void LevelEditorMode() summary: Takes no arguments, calls the correct editor mode based on the user’s choice (see Program.GetEditorMode()). Runs a game level based entirely on the parameters defined by the user (see Program.NewCustomGame(custom, ref editorRunning)). Returns void.

Static ConsoleKey GetEditorMode() summary: Takes no arguments. Clears the console and displays the menu with the two editor modes and the option to exit to main menu. Gets the user’s input and if it corresponds to one of the menu options it will return the ConsoleKey that the user pressed.

Static Level SimpleEditor() summary: Takes no arguments. Allows user to design their own level with basic control over the level’s width/height, number of coins/enemies, and the difficulty setting. Returns a new Level using the Simple Editor Constructor (see Level.cs) based on these parameters inputted by the user.

static Level AdvancedEditor() summary: Takes no arguments. Allows user to design their own level with full control over every aspect of the level including the characters and colors used to build the level, enemies, and players, full control over the level size, and everything mentioned in the SimpleEditor(). Returns a new Level() using the Advanced Editor Constructor based on these parameters inputted by the user.

Static void NewCustomGame(Level custom, ref bool editorRunning) summary: Takes the custom level created by the user (see SimpleEditor(), AdvancedEditor()), and the reference boolean determining whether the game is running or not. Does the same thing as the NewGame() method, except has no form of level progression system as the user is creating only one level. Returns void.

Static bool EditorWinOption(gameArea area) summary: Takes the custom level’s game area and uses it to display a centered menu underneath it, giving the user the option to restart the current level or to return to editor menu. Gets user key based on the menu options (see TwoKeyChoices()) Returns true if the user wishes to restart or false if the user wishes to return to the menu.

Input Handling Methods:

Static ConsoleKey TwoKeyChoices() summary: Takes no arguments. Method will keep reading the user’s key until one of two valid keys are pressed (see Global Constants Section, CHOICE\_1/CHOICE\_2). Returns the ConsoleKey the user pressed.

static int ValInt(int minValue, int maxValue, string errorMessage) summary: USED FROM MY OWN LIBRARY. Takes three parameters: Two integers defining the minimum and maximum value that will be used to validate the user’s input, and the error message it will display if the user enters an invalid input. Reads the user’s input, if it is a valid int within the min and max value, it will return that int. Otherwise it displays the error message.

Static bool ConfirmKeyPress() summary: Takes no arguments, keeps reading the user’s key presses until the user enters Y (for yes) or N (for no). Returns true if user inputted Y or false if user inputted N.