Player Class Documentation

Table of Contents

- Global Summary/External Dependencies: Page 2

- Backing Fields/Properties: Page 2-3

- Constructors: Page 4

- External Methods: Page 5-6

- Internal Methods: Page 7

Global Summary: Player class defines a user-controlled player in a console game that can detect collision with enemies and moved using the assigned keyboard buttons

External Dependencies: Enemy.cs, Coin.cs, GameArea.cs

Backing Fields/Properties

Private const char \_playerChar (get/set): Defines the ASCII character that the player is represented by on screen.

Default public constant: DEFAULT\_PLAYER\_CHAR

Private const int \_speed (get/set (value cannot be negative)): Defines the speed in console spaces the player moves at.

Default public constant: DEFAULT\_SPEED

Private const int \_xPos (get/set (value cannot be negative)): Defines the horizontal position of the player on the screen.

Default public constant: DEFAULT\_X\_POS

Private const int \_yPos (get/set (value cannot be negative)): Defines the vertical position of the player on the screen.

Default public constant: DEFAULT\_Y\_POS

Private int \_oldXPos (get/set (value cannot be negative)): The horizontal position of the player that is recorded before the player moves so the previous player icon can be erased.

Private int \_oldYPos (get/set (value cannot be negative)): The vertical position of the player that is recorded before the player moves so the previous player icon can be erased.

Private int \_startingLives (get/set (value cannot be negative)): The amount of lives that the player starts with.

Default public constant: DEFAULT\_STARTING\_LIVES

Private int \_livesRemaining (get/set (value cannot be negative)): The amount of lives that the player has left.

Private int \_coinsCollected (get/set (value cannot be negative)): The amount of coins the player has collected.

Private ConsoleColor \_playerForeColor (get/set): The foreground color of the ASCII character used to represent the player.

Default public constant: DEFAULT\_FORE\_COLOR

Private ConsoleColor \_playerBackColor (get/set): The background color of the ASCII character used to represent the player.

Default public constant: DEFAULT\_BACK\_COLOR

Private ConsoleKeys \_keyUp, \_keyDown, \_keyLeft, \_keyRight (get/set): The keys that are used to move the player around on the screen.

Default public constants: DEFAULT\_KEY\_UP, DEFAULT\_KEY\_DOWN, DEFAULT\_KEY\_LEFT, DEFAULT\_KEY\_RIGHT

Constructors

Default/Zero-Parameter constructor() summary: Sets PlayerChar, Speed, XPos, YPos, StartingLives, ForeColor, BackColor, KeyUp, KeyDown, KeyLeft, and KeyRight to their default public constants. LivesRemaining is set equal to StartingLives and CoinsCollected is set to 0.

Advanced Editor/Five-Parameter constructor (char playerChar\_, int startingLives\_, ConsoleColor playerForeColor\_, ConsoleColor playerBackColor\_) summary: Speed, XPos, YPos, KeyUp, KeyDown, KeyLeft, and KeyRight are set to their default public constants. PlayerChar, StartingLives, ForeColor and BackColor are set to their respective parameters. LivesRemaining is set equal to StartingLives and CoinsCollected is set to 0.

External Methods

Public void Move(ConsoleKey userKeyPress, GameArea area) summary: Takes the user’s key press and the current game area as arguments, and moves the player by its speed in the direction corresponding to the key press. Collision with the outer game area limits is checked here too. Returns void.

Public void Draw(GameArea area) summary: Takes the current game area as an argument. Erases the old player icon after moving and draws the new one with its current X and Y positions. Returns void.

Public void DrawDirectly() summary: Takes no arguments. Draws the player in its current X and Y pos (no erasing any previous icons). Returns void.

Public void InitializePosition(GameArea area) summary: Takes current game area as an argument and uses its dimensions to initialize the player’s position in the center of the play area. Returns void.

Public void LoseLife() summary: Takes no arguments. Reduces player’s LivesRemaining by 1 and returns void.

Public void ResetStats() summary: Takes no arguments. Resets player’s LivesRemaining back to its StartingLives property and resets its CoinsCollected property back to 0.

Public void HitTest(Enemy[] enemies, Player player, ref bool cycleCollision) summary: Takes an array of enemies, a Player class instance, and a cycleCollision boolean that is used externally to check if the player has had an enemy collision for that run of the game loop. If the player’s lives are greater than 0, draw the player directly (so it doesn’t get hidden by the enemy), and check if the player’s coordinates match those of any enemies in the array. If so, the player loses a life (see LoseLife()). Returns void as collision check is done internally with a reference boolean (see CheckEnemyCollision(Enemy[] enemies)).

Public void CheckCoinCollision(Coin[] coins) summary: Checks if the player’s position matches that of any coins in the game. If so, that coin’s Collect method is called (see Coin.Collect(Player player)). Returns void as coin stat modification is done internally through the Coin.Collect(Player player) method.

Internal Methods

Private bool CheckEnemyCollision(Enemy[] enemies) summary: Takes an enemy array as an argument, and checks to see if the player’s position matches that of any enemies. If so, console beeps and returns true. If not, returns false.