Developer’s Guide

When the program starts, execution begins with the big-bang function, which serves as the main entry point for the Snake game application. This function initializes the game environment by creating an initial world state, represented by the `initial-world` structure. The initial world is configured in menu mode, providing a configuration screen where the player can adjust game options such as speed, snake color, board size, number of fruits, and game mode. Menu navigation is controlled using the WASD keys, and the game starts when the player presses the spacebar.

All keypresses, whether in menu mode or in-game, are handled by the `handle-key-unified` function. This function takes the current world state and the key pressed as input and returns an updated world reflecting any changes triggered by that keypress. In menu mode, pressing `W` or `S` moves the menu selector up or down to choose a different setting, while `A` and `D` modify the value of the selected setting. When the spacebar is pressed in menu mode, the game is initialized with the selected options. This includes generating the snake, food positions, obstacles, and outer walls if necessary, and it switches the world mode to `'game`. In game mode, `handle-key-unified` is responsible for snake movement. WASD keys control the direction of the snake, but movement is restricted so the snake cannot move in the exact opposite direction of its current movement. Special keys such as `L` can be used to trigger a forced win condition, useful for debugging.

Mouse input is handled by the `handle-mouse-unified` function. This function takes the current world state, the mouse coordinates, and the type of mouse event, typically `"button-down"`. It checks whether the click occurred over any interactive UI elements, such as the Restart or Pause buttons. If the click occurs over the Restart button, the world state resets to menu mode with default values, effectively starting a new game. If the click occurs over the Pause button, the world toggles the `world-paused?` flag, pausing or resuming gameplay. If the click is outside these button regions or the game is not in the appropriate mode, the world remains unchanged.

The central logic of the game is controlled by the `update-unified` function. This function is called automatically every tick, as configured in the big-bang call. If the game is paused or in menu mode, `update-unified` does not modify the world. Otherwise, it calls `update-game`, which advances the game by a single step. `update-game` handles snake movement, food consumption, collision detection, and win/loss conditions. It first checks whether the tick counter has reached a threshold determined by the selected speed. If not, it increments the counter and returns the world unchanged. Once the tick threshold is reached, `update-game` computes the new head position of the snake using `compute-new-head`. `update-game` then checks whether the new head position coincides with any food using `snake-ate?`. If food is eaten, `handle-eat` is called, which increases the score, updates the record if necessary, grows the snake, generates new food in an unoccupied position, and checks if the player has won. If no food is eaten, `handle-move` is called, shrinking the tail to simulate movement and checking for collisions against walls, obstacles, or the snake itself. Collisions result in the game being marked as over.

Finally, the `render-unified` function is responsible for drawing the current state of the world on the screen. If the world is in menu mode, it calls `render-menu` to draw the configuration interface. If the game is over, `render-unified` checks the specific outcome: if `world-game-over?` is `'win` calls `render-win`, displaying a congratulatory message along with the score, if world-game-over? is #t it calls render-game-over, showing the final score and record. If the game is active and not over, render-game draws the snake, food, obstacles, and score, ensuring the player sees an accurate, real-time representation of the game state.