Additional Diagrams

Table of Contents

Extra Diagrams

None.

Extra Documentation

Directory Structure

/A_Blocky_Start/

- /userdata/
    - AliceLiddell_userdata.csv
    - BruceLee_userdata.csv
    - Developer_userdata.csv *shared developer account*
    - JaneDoe_userdata.csv
    - JohnDoe_userdata.csv
    - Teacher_userdata.csv *shared teacher account*
- /mazedata/
    - *stage1_mazedata.csv*
    - *stage2_mazedata.csv*
    - *stage3_mazedata.csv*
    - *stage4_mazedata.csv*
    - *stage5_mazedata.csv*
- game_highscoredata.csv
- game_settingsdata.csv
- game_errorlog.txt
- game.exe *compiled game executable*

Action Blocks

Action blocks are encoded as strings, which are then stored in an action chain in CSV format. Function arguments are passed in as values separated with "_".

Name	Functional Representation	String Encoding	Description
Start	Start()	"Start"	(i.e. Entry in Assembly language) (i.e. Begin in Pascal language) Must be present at the start of the action chain (program), i.e. always at line 1. Any jumping to a line before "Start" crashes the action chain.
End	End()	"End"	(i.e. End in Assembly/ Pascal language.) Must be present at the end of the action chain (program), i.e. always at line N. Any jumping to a line behind "End" crashes the action chain.
Forward	Forward()	"Forward"	(i.e. Move forward) Move the player 1 cell forward (following the current direction)
Back	Back()	"Back"	(i.e. Move back) Move the player 1 cell back (following the current direction)

Left	Left()	"Left"	(i.e. Turn left) Rotate the player anti-clockwise and update the direction.
Right	Right()	"Right"	(i.e. Turn right) Rotate the player clockwise and update the direction.
Goto	Goto(line = Z)	"Goto_Z"	(i.e. Always Branch (unconditional jump) in Assembly language) (i.e. Goto in C/ C++ language) Make the instruction counter jump to line Z, i.e. execute line Z instead of the next line.
Loop	Loop(end = X, start = Y, line = Z)	"Loop_X_Y_Z"	(i.e. Branch Equal or Negative (conditional jump) in Assembly language) Initialize the loop counter = X when creating the action block. If the loop counter <= Y, then make the instruction counter jump to line Z; (i.e. execute line Z instead of the next line) else if the loop counter > Y, then decrement it by 1 every time this instruction is executed. (i.e. loop X - Y times in total) When a jump occurs, reset the loop counter = X. (i.e. in nested loops, the inner loop resets itself when finished) Notes: • Loop_0_0_Z is functionally identical to Goto_Z. • The Loop block can be paired with a Goto block (at some later line) to form a working for-loop.

Action Chains

Action chains store action blocks (in ascending order based on line number) in a linear CSV format.

Action Block 1	**Action Block 2**	**Action Block 3**	**...**

Example:

Start	*Left*	*Forward*	*End*

Basic Instruction(s)

CSV format: Start, Forward, End

Line	Action Chain	Java Code (comparison)
1	Start	{
2	Forward	Forward();
3	End	}

Repeat-Forever Loop

CSV format: Start, Left, Forward, Goto_3, End

Line	Action Chain	Java Code (comparison)
1	Start	{
2	Left	Left();
		. . do {
3	Forward	. . Forward();
4	Goto_3	. . } while (TRUE);
5	End	}

Repeat-Until Loop

CSV format: Start, Left, Forward, Loop_10_0_3, Goto_3, End

Line	Action Chain	Java Code (comparison)
1	Start	{
2	Left	Left();
		int i = 10;
		. . do {
		. . i--;
3	Forward	. . Forward();
4	Loop_10_0_6	. . if (i <= 0) break;
5	Goto_3	. . } while (TRUE);
6	End	}

For-Loop

CSV format: Start, Loop_5_3_8, Forward, Left, Forward, Right, Goto_2, Back, End

Line	Action Chain	Java Code (comparison)
1	Start	{
2	Loop_5_3_8	for (int i = 5, i > 3, i--) {
3	Forward	. . Forward();
4	Left	. . Left();
5	Forward	. . Forward();
6	Right	. . Right();
7	Goto_2	. . }
8	Back	Back();
9	End	}

Nested For-Loops

CSV format: Start, Loop_5_3_8, Loop_2_0_6, Left, Goto_3, Right, Goto_2, Back, End

Line	Action Chain	Java Code (comparison)
1	Start	{
2	Loop_5_3_8	for (int i = 5, i > 3, i--) {
3	Loop_2_0_6	. . for (int j = 2, j > 0, j--) {
4	Left	. . . . Left();
5	Goto_3	. . . . }
6	Right	. . Right();
7	Goto_2	. . }
8	Back	Back();
9	End	}