Test Cases **Software Engineering II - Group 6 - Homework 4**

Bauhaus-Universität Weimar

Black-Box Test Cases

These test cases are defined based on the available documentation and execution of the program. The code was not inspected.

#	Test case (very brief description)	Preconditions (any required setup)	Test steps (steps executed during testing)	Expectation	Observation ("pass" or failure description)
1	running program	none	run App.main() file	game is loading without problems	parser cannot read files, because default run directory is different. fixed problem with path-strings
2	try to complete one match	none	run App.main() file, place ships on grid, shoot on other grid till a win is achieved	game works like in the presentation	ships can be placed, but cannot shoot/select point on grid of opponent, needed to look at code for the controls
3	placing ships on grid with drag and drop	start App.main() file	click and drag each ship with left mouse click and place on left player grid. find a way	fight should start after placing all the ships. ship is set, if you place it on the grid. ships are not allowed to touch each other	pass, ships can be dragged and dropped. important: fight can only start if ships are set while dragging and pressing 'E'. Ships can move/switch if you drag another through them. Sadly, we did not find a way to correct that. Visual clipping of ships can be seen while dragging. Ships cannot be moved after the fight started.
4	shoot on enemy field	all ships are placed with pressing 'E' while dragging ship, fight has started	click on enemy field you wish to	get result if you hit ship or not	pass
5	if ship is hit, get another move	you hit ship of AI	hit a ship, try to shoot again right after hitting	get another move	pass
6	start screen requirement	none	run App.main() file and start screen should appear	start screen appears	not implemented
7	rotate ship	run App, place at least one ship on grid	click on ship (not implemented), press 'R' on each ship to rotate while dragging with left mouse to see immediate result	if you click on ship, it rotates 90 degrees	pass, but requirement was to rotate ship by clicking with mouse click and not by typing 'R'

8	start fight by clicking on button	all ships are placed	try to find button or any feedback to know when fight starts	button with ‘Start fight’ is visible	not implemented
9	shots displayed	all ships are placed, fight has started	click on field	if you hit ship field turns red. if you hit nothing, you get other colour	pass
10	positioning ships without placing	picking up ship model	pick up ship and move it	ship is tethered to player location, not moving beyond arbitrary x number of units from player	increasing aim on z-axis while holding ship models will proportionally scale model distance from player location
11	collision with items	none	move to solid-appearing object	player movement should be prohibited when reaching “walls” or solid objects	player movement is not inhibited
12	placing ships with erratic rotation	pick up ship model	Press “R” repeatedly and “E” repeatedly simultaneously	ships will only place at valid location	ships can clip into each other

White-Box Test Cases

These additional test cases were defined during inspection of the code.

#	Test case (very brief description)	Preconditions (any required setup)	Test steps (steps executed during testing)	Expectation	Observation (“pass” or failure description)
1	getLeft() once with hasShip true and false	2 cells with coordinates and hasShip bools	make testCell2.hasShip true and place cells adjacent, at border of cell [1,y],[0,y]; see if index out of bounds can be forced	The code should catch and return null or return index in bounds	pass with null cell or valid index
2	getRight() once with hasShip true and false	-||-	-||-	-||-	-||-
3	getUp() with true bool and obscure indices	x value of testCell set to MaxValue of int, y value set to -0	compare any two cells, see how code reacts to obtuse or irregular values	The code should catch and return null or return index in bounds	-||-
4	getUp() with true bool and obscure indices	-||-	-||-	-||-	-||-

Reflection on Testing:

Much of the code is entwined in a way that white-box testing would be very difficult. In effect, one would have to hard-code an entire game scenario to be able to test the functions of the game. As a rule of thumb, game products are better suited for black-box testing seeing as that the user very rarely has any direct input options, and is rather confined to the interactions defined by the game devs. White-box testing in this scenario was thus kept to a minimum, and black-box testing was done to ensure that vital features were working as intended. In addition, we performed several black-box tests to see if we could push the bounds of the game. These were much more successful than any white-box attempt, in the sense of uncovering bugs.