Unit Testing

Team name: Endeavour (Team 28)

Team Members: Sarah Berry, Finley Brown, Yupeng Di, William Griffiths, Kurtas Joksas, Fraser Masson

Test Cases

ID	Class	Method	External Condition	Valid Classes	Invalid Classes
1.1	Utils	randomFloatInRange()	Upper bound and lower bound.	UB ≥ LB(#1)	UB < LB(#2)
1.2		randomIntInRange()	Upper bound and lower bound.	UB ≥ LB(#1)	UB < LB(#2)
1.3		randomListItem()	List length	>0 (#1)	=0 (#2)
2.1	Navigation Mesh	setCell(int x, int y, boolean value)	x coordinate	0≥x>(width/tileWidth) (#1)	<0 (#2) , >(width/tileWidth) (#3)
			y coordinate	0≥y>(height/tileHeight) (#4)	<0 (#5), >(height/tileHeight) (#6)
			value	true (#7), false(#8)	-
2.2		cellAccessible(int x, int y)	x coordinate	0≥x>(width/tileWidth) (#1)	<0 (#2) , >(width/tileWidth) (#3)
			y coordinate	0≥y>(height/tileHeight) (#4)	<0 (#5), >(height/tileHeight) (#6)
			Cell accessibility	accessible (#7), false(#8)	-
2.3		getWorldCoordinates (int x, int y)	x coordinate	0≥x>(width/tileWidth) (#1)	<0 (#2) , >(width/tileWidth) (#3)
			y coordinate	0≥y>(height/tileHeight) (#4)	<0 (#5), >(height/tileHeight) (#6)
2.4		getSuccessorNodes(PathNode node, int[] destination)	AccessibleCells	None adjacent (#1) Directly left (#2) Directly above (#3) Directly below (#4) Directly right (#5) Diagonal (#6)	-
2.5		generateTilemapPathToPoint()	Path between start and destination	Direct path (#2) Diagonal path(#3) Indirect path (#4)	No path (#1)

2.6		getFurthestPointFromEntity(GameEntity entity)	Entity x coordinate	0≥x>width (#1)	<0 (#2) , >width (#3)
			Entity y coordinate	0≥y>height(#4)	<0 (#5), >height (#6)
2.7		getEuclidianDistance(float[], float[])	firstPoint x coordinate	≥0(#1) <0(#2)	-
			firstPoint y coordinate	≥0(#3) <0(#4)	-
			secondPoint x coordinate	≥0(#5) <0(#6)	-
			secondPoint y coordinate	≥0(#7) <0(#8)	-
2.8		getEuclidianDistance(int[], int[])	firstPoint x coordinate	≥0(#1) <0(#2)	-
			firstPoint y coordinate	≥0(#3) <0(#4)	-
			secondPoint x coordinate	≥0(#5) <0(#6)	-
			secondPoint y coordinate	≥0(#7) <0(#8)	-
3.1	PathNode	toString()	x coordinate	All (#1)	-
			y coordinate	All (#2)	-
3.2		equals(object)	Classes of objects	Both objects are PathNodes(#1) One object is not a PathNode(#2)	-
			Obj1 coordinate, obj2 coordinates	Obj1 coords = obj2 coords (#3) Obj1 coords ≠ obj2 coords (#4)	-
4.1	GameEntity	getCenterX()	X position of sprite of game entity	All	-
4.2		getCenterY()	Y position of sprite of game entity	All	-

4.3		getCenter	XY position of sprite of game entity	All	-
5.1	Player	powerStopInfiltratorPower()	Infiltrator power (blinded,confused,slowerd)	All	-
5.2		powerOn()	Number of infiltrators caught	=1 =2 =3 =4 =5	-
5.3		respawn(float x float y)	Player confused/slowed	All	-
5.4		heal(float rate)	Health	<maxHealth(#1) =maxHealth(#2)	-
			rate	>0(#3)	-

ID	Possible test cases
1.1	● UB > LB (#1) ● UB = LB (low boundary #1) ● UB < LB (#2)
1.2	● UB > LB (#1) ● UB = LB (low boundary #1) ● UB < LB (#2)
1.3	● Length > 0 (#1) ● Length = 1 (#1 lower boundary) ● Empty list (#2)
2.1	● (all test cases use false for value unless stated, therefore belonging to (#7)) ● (x < (width/tileWidth), y<(height/tileHeight) for all test cases unless stated) ● x>0, y>0 (#1,#4) ● x=0,y=0(#1 low boundary, #4 low boundary) ● X = (width/tileWidth), y = (height/tileHeight) (#1 high boundary, #4 high boundary) ● x<0, y>0 (#2,#4) ● x>0,y<0 (#1,#5) ● x > (width/tileWidth) , y>0 (#3,#4) ● x>0, y > (height/tileHeight) (#1,#6) ● x>0,y>0,value = false (#1,#4,#8)
2.2	● (all test cases use accessible cells unless stated, therefore belonging to (#7)) ● (x < (width/tileWidth), y<(height/tileHeight) for all test cases unless stated) ● x>0, y>0 (#1,#4) ● x=0,y=0(#1 low boundary, #4 low boundary) ● X = (width/tileWidth), y = (height/tileHeight) (#1 high boundary, #4 high boundary) ● x<0, y>0 (#2,#4)

	• x>0,y<0 (#1,#5) • x > (width/tileWidth) , y>0 (#3,#4) • x>0, y > (height/tileHeight) (#1,#6) • x>0,y>0,inaccessible cell (#1,#4,#8)
2.3	• (x < (width/tileWidth), y<(height/tileHeight) for all test cases unless stated) • x>0, y>0 (#1,#4) • x=0,y=0(#1 low boundary, #4 low boundary) • X = (width/tileWidth), y = (height/tileHeight) (#1 high boundary, #4 high boundary) • x<0, y>0 (#2,#4) • x>0,y<0 (#1,#5) • x > (width/tileWidth) , y>0 (#3,#4) • x>0, y > (height/tileHeight) (#1,#6)
2.4	• No adjacent accessible cells (#1) • Left adjacent accessible cell (#2) • Up adjacent accessible cell (#3) • Down adjacent accessible cell (#4) • Right adjacent accessible cell (#5) • Left, right and left right diagonal adjacent (#6) • All adjacent cells accessible (#1-6)
2.5	• No path between start and destination (#1) • Destination to the right of start with a direct path between them (#2) • Destination on the left,down diagonal to start, with a direct path between them(#3) • Destination to the down and right of start, with an indirect path between them (#4)
2.6	• (x < width, y<height for all test cases unless stated) • x>0, y>0 (#1,#4) • x=0,y=0(#1 low boundary, #4 low boundary) • X = width, y = height (#1 high boundary, #4 high boundary) • x<0, y>0 (#2,#4) • x>0,y<0 (#1,#5) • x > width , y>0 (#3,#4) • x>0, y > height (#1,#6)
2.7	• x1>0, y1>0, x2>0, y2>0 (#1,#3,#5,#7) • x1=0,y1=0,x2=0,y2=0 (#1, #3, #5, #7 low boundaries) • x1<0, y1<0, x2<0, y2<0 (#2,#4,#6,#8) • x1>0, y1>0, x2>0, y2>0, x1=x2, y1=y2 (#1,#3,#5,#7)
2.8	• x1>0, y1>0, x2>0, y2>0 (#1,#3,#5,#7) • x1=0,y1=0,x2=0,y2=0 (#1, #3, #5, #7 low boundaries) • x1<0, y1<0, x2<0, y2<0 (#2,#4,#6,#8) • x1>0, y1>0, x2>0, y2>0, x1=x2, y1=y2 (#1,#3,#5,#7)
3.1	• Positive x, positive y (#1,#2) • Negative x, positive y(#1 low boundary,#2) • Positive x, negative y (#1,#2 low boundary) • Negative x, negative y(#1 low boundary,#2 low boundary)
3.2	• Both objects are PathNodes with equal coordinates (#1,#3) • Both objects are PathNodes with unequal coordinates (#1,#4) • Obj 2 is an Integer, Obj1 is a pathnode (#2)

5.4	• health<max health, rate>0 (#1.#3) • health=max health , rate>0 (#2,#3)