Test Documentation

This assessment (4):

Anything added will be highlighted in green like so: EXAMPLE

Anything removed will be crossed out and highlighted in yellow like so: ~~EXAMPLE~~

Assessment 3 group:

Anything added will be highlighted in yellow like so: EXAMPLE

Anything removed will be crossed out and highlighted in yellow like so: ~~EXAMPLE~~

Assessment 1 & 2:

No colours

Play Testing

Test ID	Description	Requirement ID	Description	Logic Test Result
1	GameShouldRunTest	SCR_RUNNABLE	The game should run without crashes	Pass
2	FireTruckShouldAttackIfInRangeTest	SFR_FORTRESS_DESTROY	Detecting an ET fortress in the firetruck's range should trigger the firetruck to start attacking it with a water jet	Pass
3	FortressShouldGetDestroyedTest	SFR_FORTRESS_DESTROY	After fatally damaging an ET fortress, it should be marked as 'destroyed'	Pass
4	FortressShouldAttackIfInRangeTest	SFR_FORTRESS_ATTACK UR_FORTRESS UR_FUN	Entering the range of an ET fortress should trigger the fortress to start attacking	Pass
5	FireTrucksShouldHaveDifferentStatsTest	UR_FIRETRUCKS_UNIQUE_SPEC	Each firetruck of the four should each have a specific statistic that differs it from the other three	Pass
6	ETShouldHaveUniqueSpecsTest	UR_ET_UNIQUE_SPEC	Each ET fortress should have unique statistics that make it different from other fortresses	~~Fail~~ Pass
7	TruckWaterTankShouldRefill	UR_FIRETRUCKS_REFILL, SFR_ALLOWED_TO_REFILL, SFR_CANCEL_REFILL, SFR_REFILL_OVER_TIME, SFR_REFILL_CONSTANT	Entering the range of the fire station should trigger the water refilling, assuming the water tank is not full	Pass
8	TruckHealthShouldRepairTest	UR_FIRETRUCK_REPAIR, SFR_ALLOWED_TO_REPAIR, SFR_CANCEL_REPAIR	Entering the range of the fire station should trigger the repairing, assuming the health bar is not full	Pass
9	ETPatrolsShouldDestroyFireStationTest	UR_ET_DESTROYS_STATION, UR_GAME_TIMER SFR_PATROL_FIRESTATION SFR_PATROL_DIFFICULTY	After ~~15~~ 5 minutes of gameplay, the ET patrols should destroy the fire station	~~Fail - Not Implemented~~ Pass

10	GameShouldGetToGameOverScreenTest	UR_WIN_CONDITION, UR_LOSS_CONDITION SFR_ENDSCREEN	After destroying all ET fortresses or losing all four lives, the game should automatically reach the Game Over screen	Pass

11	GameShouldGetToGameOverScreenTest	SFR_MOVE_WHILE_DAMAGED	Getting hit by a bullet should not ~~empair~~ impair the truck's movement abilities	Pass
12	FireTruckShouldMoveWhileWaterTankEmptyTest	SFR_MOVE_WHILE_EMPTY	The fire truck should be able to move even when the water tank is empty	Pass
~~13~~	~~FireTruckShouldBeSelectedBeforeGameTest~~	~~SFR_FIRETRUCKS_STATS, SFR_FIRETRUCKS_SELECTION UR_FIRETRUCK_MIN_START~~	~~Before a new game is initiated, the user should be prompted with a fire truck selection screen~~	~~Pass~~
14	ScreenShouldSwitchTest	UR_MINIGAME, UR_DIFFICULTY_LEVEL, UR_CONTROLLER, UR_INSTRUCTIONS, UR_COLOUR_ACCESSIBILITY	The user should be able to move between ~~differnt~~ different screens without system bugs or crashes	Pass
15	FireTruckShouldNotDriveOnBuildingsTest	SFR_BUILDINGS UR_DRIVE	The firetruck should not be able to drive over buildings tiles	Pass
16	FireTruckShouldNotDriveOnRiversTest	SFR_RIVERS UR_DRIVE	The firetruck should not be able to drive over rivers tiles	Pass
17	HealthBarShouldAlwaysBeVisibleTest	SFR_HEALTH_BAR	The health bar should be visible at all point int time during gameplay	Pass
18	WaterBarShouldAlwaysBeVisibleTest	SFR_WATER_SUPPLY_BAR	The water bar should be visible at all point int time during gameplay	Pass
~~19~~	~~DifficultyHarder~~	~~SFR_PATROL_HEALTH SFR_PATROL_DIFFICULTY SFR_PATROL_DAMAGE UR_PATROL UR_ET_IMPROVEMENT~~	~~The game should become harder over time as the fortresses become more difficult to flood and the number of ET Patrols increase.~~	~~Pass~~
20	userShouldNotChangeFortressLocation	SFR_ET_LOCATIONS_NOT_CHANGEABLE UR_ET_MIN_START	The game should not allow the user to change locations of the fortresses.	Pass
21	FireTruckCounterShouldAccuratelyShowAmountAlive	SFR_DESTROYED_TRUCKS	After a truck has been destroyed, there should be one less life on the heads-up display. ~~This means the truck cannot be used again.~~ This should reset with powerup of respawn.	Pass
~~22~~	~~MiniGameOption~~	~~SFR_MINIGAME~~	~~On the main menu of the game you can click the minigame option and start playing the mini game.~~	~~Pass~~
23	ArrowKeysShouldControlTruckInCorrectDirection	SFR_ARROWKEYS	The user should be able to move the fire truck with Arrow keys.	Pass
24	FlappyMiniGame	UR_MINIGAME	There should be a fully functional mini game based on flappy bird.	Pass

25	SavingShouldSaveFireTrucksWithFullHealth	UR_SAVING SFR_HEALTH_BAR	The user should be able to save a game when all fire trucks are at full health and load it up with the same health	Pass
26	SavingShouldSaveFireTrucksWithDifferentHealth	UR_SAVING SFR_HEALTH_BAR	The user should be able to save a game when fire trucks are at differing health and load it up with the same differing health	Pass
27	SavingShouldNotLoadDeadFireTrucks	UR_SAVING	The user should not be able to reload a dead firetruck that was dead at the time of saving	Pass
28	SavingShouldSaveTheFullWaterTank	UR_SAVING SFR_WATER_SUPPLY_BAR	The user should be able to save when fire trucks have full water tanks and load them up with the same amount	Pass
29	SavingShouldSaveWaterWithDifferingWaterTanks	UR_SAVING SFR_WATER_SUPPLY_BAR	The user should be able to save when fire trucks have different water levels and load them with the same difference (this including empty)	Pass
30	SavingShouldLoadTheCorrectDifficulty	UR_SAVING SFR_DIFFICULTY UR_DIFFICULTY_LEVEL	The user should keep the same difficulty (and changes in stats) when loading in a game	Pass
31	SelectDifficultyBeforeGame	SFR_DIFFICULTY UR_DIFFICULTY_LEVEL	The user should be able to select between 3 difficulties before entering the game. Easy, medium, and hard	Pass
32	EasyModeShouldOnlyIncludeEasyStats	SFR_DIFFICULTY UR_DIFFICULTY_LEVEL	The easy mode should only load 2 UFOS, have twice the health and water tank, and twice the spawn rate of UFOs (60 seconds)	Pass
33	MediumModeShouldOnlyIncludeMediumStats	SFR_DIFFICULTY UR_DIFFICULTY_LEVEL	The easy mode should only load 4 UFOS, have standard health and water tank amounts, and standard spawn rate of UFOs (30 seconds)	Pass
34	HardModeShouldOnlyIncludeHardStats	SFR_DIFFICULTY UR_DIFFICULTY_LEVEL	The hard mode should only load 8 UFOS, have half the health and water tank, and half the spawn rate of UFOs (15 seconds)	Pass
35	MinigameShouldBePlayable	UR_MINIGAME	The minigame should be playable once triggered and has no bugs.	Pass

36	UserShouldBeAbleToGetOver30InMinigame	UR_MINIGAME	The user should be able to achieve over the threshold for the maximum powerup	Pass
37	MinigameScoreBelowRangeGivesNoPowerUp	UR_MINIGAME UR_POWER_UPS SFR_POWER_UPS	After the player achieves a score in a specific range and then losses in the minigame, if the score is below 3 they should receive no power-up	Pass
38	UnlimitedWaterPowerUpShouldChangeStatAndIcon	UR_POWER_UPS SFR_POWER_UPS	When the user has the Unlimited Water power-up there should be no decrease in water level and the water icon should appear	Pass
39	ShieldPowerUpShouldChangeStatAndIcon	UR_POWER_UPS SFR_POWER_UPS	When the user has the Shield power-up there should be no decrease in health and the shield icon should appear	Pass
40	FreezeEnemyPatrolShouldChangeStatsAndIcon	UR_POWER_UPS SFR_POWER_UPS	When the user has the Freeze Enemy Patrol power-up it should stop the UFOs from moving on their patrols and attacking, and the icon should appear	Pass
41	RestoreTimeShouldAlterStatsAndIcon	UR_POWER_UPS SFR_POWER_UPS	When the user has the Restore Time power-up it should reset their time and score but not their progress and the icon should appear	Pass
42	ResurrectDeadTruckShouldRestoreATruck	UR_POWER_UPS SFR_POWER_UPS	When the user has the Resurrect Dead Truck power-up it should restore one of the dead fire trucks and no icon should appear	Pass
43	ResurrectDeadTruckShouldNotResurrectIfTrucksAreAlive	UR_POWER_UPS SFR_POWER_UPS	When the user has the Resurrect Dead Truck power-up it should not restore a truck if all are alive and no icon should appear	Pass
44	RainDanceShouldKillAllPatrols	UR_POWER_UPS SFR_POWER_UPS	When the user has the Rain Dance power-up it should ‘rain’ and all patrols will die and respawn in their elected time.	Pass
45	PowerUpBoxesAreRandomAndReachable		The power-up boxes in the game should be spawned at random in points that are always reachable	Pass

Junit tests

Fire Station Test (Run with JUnit FireStationTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUFS1	fireStationShouldInitializeCorrectly()	getCentre()	Returns location of the fire station.	Pass	Checks if the location of the fire station is the correct location.
JUFS2	dieShouldChangeTheTextureOfTheFirestation()	die()	Kills the fire station.	Pass	Checks if the fire station can be destroyed.
JUFS3	updateOnFireStationShouldReplenishWater()	replenish()	Repairs the fire trucks health and refills its water supply.	Pass	Checks if fire station can repair and refill a fire truck.

Fire Truck Test (Run with JUnit FireTruckTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUFT1	Hitbox()	getHitbox()	Returns the hitbox of the fire truck.	Pass	This is a test to check if the hitbox generated is the right size.
JUFT2	movementTest()	getDirection()	Returns the direction the fire truck is facing.	Pass	This is a test to check if the directions of the fire truck work properly.
JUFT3	testInitialisation()	getHealthPoints()	Returns the health of the fire truck.	Pass	This is a test to see if the fire truck spawns with correct amount of health.
JUFT4	testRefill()	getHealthPoints() getCurrentWater()	Returns the health of the fire truck. Returns the water levels of the fire truck.	Pass	This is a test to see if the value of health and the value of water supply is correct after being repaired and refilled.

Fortress Test (Run with JUnit FortressTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUF1	takeDamageShouldResultInCorrectDecreaseInHealth()	damage()	Lowers the health of a fortress by the amount within the brackets.	Pass	This test checks if damage to a fortress lowers the health by a correct amount.
JUF2	deathShouldChangeDisplayableTheFortress()	death()	Removes a fortress from being active and	Pass	This test destroys a fortress and checks if it

			displays it as a destroyed state.		is displayed as a destroyed state.
JUF3	setPositionShouldSetNewPositionAndReturnCorrectCentre()	getCentre()	Returns the location of the fortress.	Pass	This test checks if the fortress is in the right location.
JUF4	setPositionShouldSetTheCorrectPosition()	getPosition()	Returns the position of the fortress.	Pass	The test checks if the fortress is in the right position. Was combined with JUF3 but separated since testing different functions

Goose Test (Run with JUnit GooseTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUG1	movementTest()	getY()	This returns the y value of the goose.	Pass	~~The first part of the test checks if the gravity works. The y value should become lower as time goes due to gravity.~~ ~~The second part of the test checks if the jumping function works, this time the y value should be greater to represent the spaceship going up.~~ This checks that the gravity works, the Y value should become lower as time goes due to gravity.
JUG2	hitboxShouldBeToScaleOfGoose()	getHibox()	Returns the value of the hitbox.	Pass	This checks if the size of the hitbox generated is correct.
JUD3	movementShouldChangeWithUpdatesJumping()	getY()	This returns the y value of the goose.		This was part of JUG1 but since they test partly different aspects it is cleaner if they are apart. This check that the jumping function works with the Y value increasing if the goose is going up

Pipe Test (Run with JUnit PipeTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUP1	movementTest()	getX()	This function returns the x value of the pipe.	Pass	This test is to check if the movement and the gravity works within the minigame.

JUP2	testIsRemove()	isRemove()	Returns true if the pipe can be removed.	Pass	This is a test to check if the pipe has been removed.
JUP3	testGetHitboxes()	getHitboxes()	Returns the hitboxes of the pipe.	Pass	This is a test to see if the hitbox is the correct size.
JUP4	testGameEnd()	gameEnd()	Returns whether the goose collides with the pipe.	Pass	This is a test to see whether the minigame has finished.

GameObject Tests (Run with Junit GameObjectTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUGO1	initializationShouldSetCorrectValuesToGameObject()	GameObject() and getPosition()	Returns a value that is used in initialized compared to the same value passed	Pass	This test to ensure the initialization of this class has no problems.
JUGO2	changePositionShouldChangeToCorrectNewPosition()	.getPosition and changePosition()	Returns the position and check against what it should be changed to	Pass	To ensure, at the highest level, that changePosition() and getPosition work with positive values. All relative to current position
JUGO3	changePositionShouldChangeToNegativePosition()	.getPosition and changePosition()	Returns the position and check against what it should be changed to	Pass	To ensure, at the highest level, that changePosition() and getPosition work with negative values (needed for rendering tricks) all relative to current position
JUGO4	getCentreWillGiveCorrectCentreWithStandardPosition()	.getCentre	Returns the calculated center value to pre-calculated values	Pass	To ensure that the calculation method of getting the center of an object works with a standard position (both x and y are positive)
JUGO5	getCentreWithNegativePosition()	.getCentre and .getPosition	Returns the calculated center value to pre-calculated values	Pass	To ensure that the calculation method of getting the center of an object works with an irregular position (with negatives). It first ensures the position did changed with a assertEquals of Pos (so if this test fails the developers can see it's the position not center)
JUGO6	setRotationShouldSetRotation()	getRotation and setRotation	Returns the value of rotation	Pass	To ensure, at the highest level, that the rotational setting is not compromised.
JUGO7	setPositionShouldSetVectorToExactInput()	setPosition	Returns the value of the setPosition	Pass	To ensure, at the highest level, that setPositon is working as intended. (Not the same as changePosition)

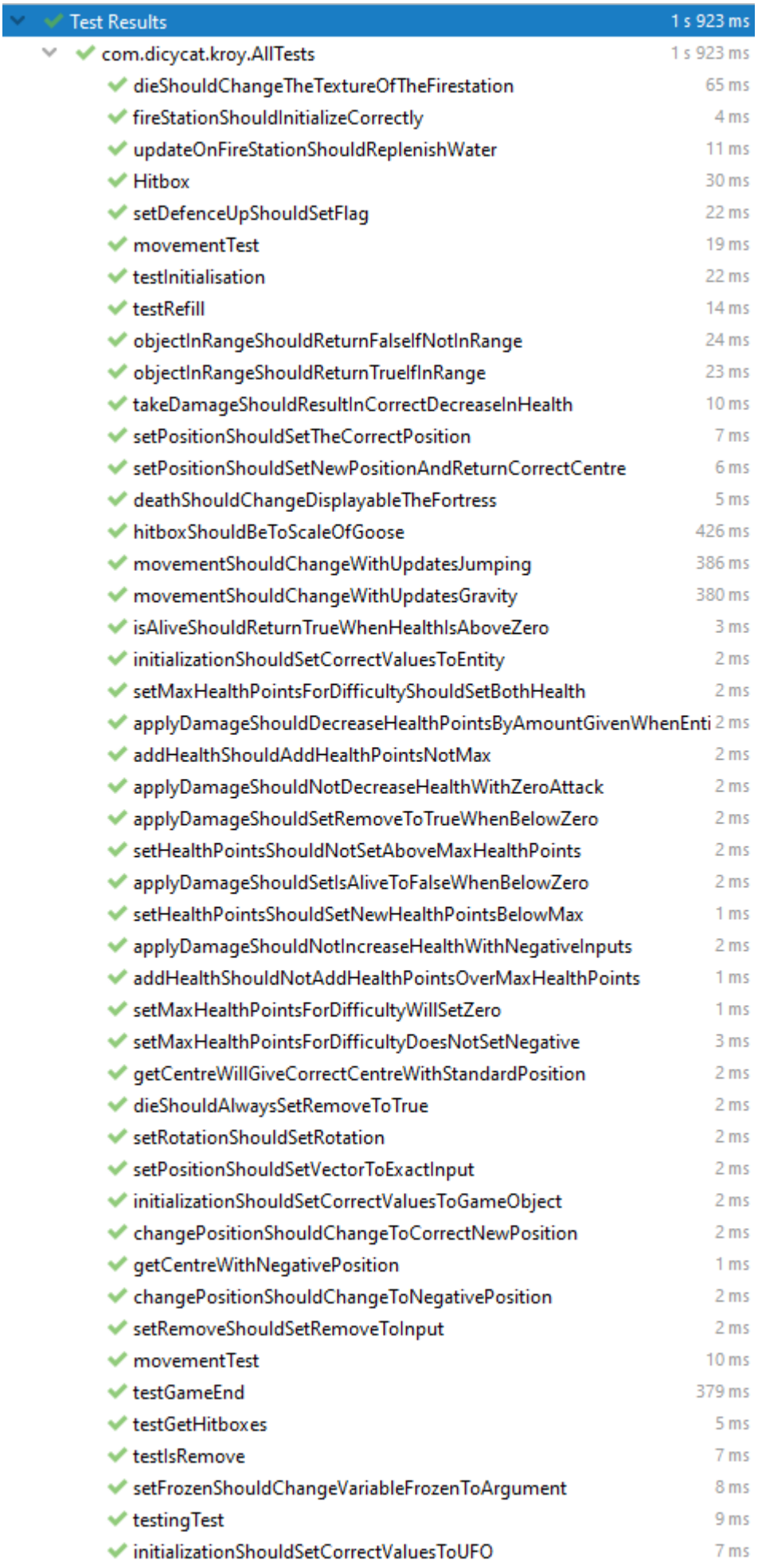
JUGO8	setRemoveShouldSetRemoveToInput()	isRemove() and setRemove	Returns the value of removed which should be the set value	Pass	To ensure that, at the highest level, setRemove() works as intended with no compromise.
JUGO9	dieShouldAlwaysSetRemoveToTrue()	isRemove and die()	Returns false if die() works	Pass	To ensure, at the highest level, that die() works and will set remove to true

Entity Tests (Run with Junit EntityTest)

Test ID	Test function name	Function tested	Function Use	Result of test	Test description
JUE1	initializationShouldSetCorrectValuesToEntity()	Entity() and getMaxHealthPoints()	Returns a value that is used in initialized compared to the same value passed	Pass	This test to ensure the initialization of this class has no problems.
JUE2	isAliveShouldReturnTrueWhenHealthIsAboveZero() ()	isAlive	Returns a Boolean if the entity has health above 0	Pass	Test that isAlive(), which should 'true' if the Entity has greater than 0 health points, returns true when the testEntity has its initialized health
JUE3	*applyDamageShouldDecreaseHealthPointsByAmountGivenWhenEntityIsAlive()*	isAlive, applyDamage, getHealthPoints	applyDamage minuses the health points by the given int	Pass	Test that applyDamage(), which (after checking the entity is alive) applies damage to an entity
JUE4	applyDamageShouldNotIncreaseHealthWithNegativeInputs()	applyDamage, getHealthPoints	applyDamage minuses the health points by the given int	Pass	Test that applyDamage(), does not take negative damage that would heal it.
JUE5	applyDamageShouldNotDecreaseHealthWithZeroAttack()	applyDamage, getHealthPoints	applyDamage minuses the health points by the given int	Pass	Test that applyDamage() allows for 0 damage and does not remove HP.
JUE6	setMaxHealthPointsForDifficultyShouldSetBothHealth()	setMaxHealthPointsForDifficulty	It sets both the maxHealthPoints variable and then the current HealthPoints of that Entity	Pass	Test that setMaxHealthPointsForDifficulty() will set both the healthPoints (current) and themaxHealthPoints which is needed for difficulty

JUE7	setMaxHealthPointsForDifficultyDoesNotSetNegative()	setMaxHealthPointsForDifficulty	It sets both the maxHealthPoints variable and then the current HealthPoints of that Entity	Pass	Test that setMaxHealthPointsForDifficulty will not set negative health points and will keep the current values
JUE8	setMaxHealthPointsForDifficultyWillSetZero()	setMaxHealthPointsForDifficulty	It sets both the maxHealthPoints variable and then the current HealthPoints of that Entity	Pass	Test that setMaxHealthPointsForDifficulty will set the value of maxHealth and healthPoints to zero
JUE9	addHealthShouldAddHealthPointsNotMax()	applyDamage, getHealthPoints, addHealth	addHealth will add a value to the current healthPoints	Pass	Test that addHealth() adds health to damaged entity - not hitting the max healthPoints
JUE10	addHealthShouldNotAddHealthPointsOverMaxHealthPoints()	applyDamage, getHealthPoints, addHealth	addHealth will add a value to the current healthPoints	Pass	Test that addHealth() add health but caps at maxHealthValue
JUE11	setHealthPointsShouldSetNewHealthPointsBelowMax()	getHealthPoints, setHealthPoints	setHealthPoints will set the current healthPoints to the passed value	Pass	Test that setHealthPoints() sets a new health points that are below max
JUE12	setHealthPointsShouldNotSetAboveMaxHealthPoints()	getMaxHealthPoints, setHealthPoints, getHealthPoints	setHealthPoints will set the current healthPoints to the passed value	Pass	Test that setHealthPoints() does not set new health points to above max health points
JUE13	applyDamageShouldSetRemoveToTrueWhenBelowZero()	isRemove, applyDamage	applyDamage minuses the health points by the given int	Pass	Test that applyDamage() sets the entity to be dead to then be removed
JUE14	applyDamageShouldSetIsAliveToFalseWhenBelowZero()	applyDamage	applyDamage minuses the health points by the given int	Pass	Test that applyDamage() sets entity to be not alive

All the tests passing


Test Results 1 s 923 ms
com.dicycat.kroy.AllTests 1 s 923 ms
dieShouldChangeTheTextureOfTheFirestation 65 ms
fireStationShouldInitializeCorrectly 4 ms
updateOnFireStationShouldReplenishWater 11 ms
Hitbox 30 ms
setDefenceUpShouldSetFlag 22 ms
movementTest 19 ms
testInitialisation 22 ms
testRefill 14 ms
objectInRangeShouldReturnFalseIfNotInRange 24 ms
objectInRangeShouldReturnTrueIfInRange 23 ms
takeDamageShouldResultInCorrectDecreaseInHealth 10 ms
setPositionShouldSetTheCorrectPosition 7 ms
setPositionShouldSetNewPositionAndReturnCorrectCentre 6 ms
deathShouldChangeDisplayableTheFortress 5 ms
hitboxShouldBeToScaleOfGoose 426 ms
movementShouldChangeWithUpdatesJumping 386 ms
movementShouldChangeWithUpdatesGravity 380 ms
isAliveShouldReturnTrueWhenHealthIsAboveZero 3 ms
initializationShouldSetCorrectValuesToEntity 2 ms
setMaxHealthPointsForDifficultyShouldSetBothHealth 2 ms
applyDamageShouldDecreaseHealthPointsByAmountGivenWhenEnti 2 ms
addHealthShouldAddHealthPointsNotMax 2 ms
applyDamageShouldNotDecreaseHealthWithZeroAttack 2 ms
applyDamageShouldSetRemoveToTrueWhenBelowZero 2 ms
setHealthPointsShouldNotSetAboveMaxHealthPoints 2 ms
applyDamageShouldSetIsAliveToFalseWhenBelowZero 2 ms
setHealthPointsShouldSetNewHealthPointsBelowMax 1 ms
applyDamageShouldNotIncreaseHealthWithNegativeInputs 2 ms
addHealthShouldNotAddHealthPointsOverMaxHealthPoints 1 ms
setMaxHealthPointsForDifficultyWillSetZero 1 ms
setMaxHealthPointsForDifficultyDoesNotSetNegative 3 ms
getCentreWillGiveCorrectCentreWithStandardPosition 2 ms
dieShouldAlwaysSetRemoveToTrue 2 ms
setRotationShouldSetRotation 2 ms
setPositionShouldSetVectorToExactInput 2 ms
initializationShouldSetCorrectValuesToGameObject 2 ms
changePositionShouldChangeToCorrectNewPosition 2 ms
getCentreWithNegativePosition 1 ms
changePositionShouldChangeToNegativePosition 2 ms
setRemoveShouldSetRemoveToInput 2 ms
movementTest 10 ms
testGameEnd 379 ms
testGetHitboxes 5 ms
testIsRemove 7 ms
setFrozenShouldChangeVariableFrozenToArgument 8 ms
testingTest 9 ms
initializationShouldSetCorrectValuesToUFO 7 ms

Acceptance Testing

TEST ID	REQUIREMENT ID	FIT CRITERION	RESULT	EVIDENCE
A_1	SNFR_INSTRUCTIONS	Instructions should cover all features of the game and how they work.	Pass	The game has a manual.
A_2	SNFR_TARGET_AUDIENCE	Game should be based on easy to understand rules, fast-paced and with relatively wide range of bullets' patterns difficulties	Pass	The game has simple controls, you only have to use the arrow keys to play the game. The map is simple and the shooting is automatic.
A_3	SNFR_JARGON	All user-facing messages shall be in plain English and will not use technical videogames jargon	Pass	The game tries to use the least amount of words as it possibly can so it is easy to understand.
A_4	SNFR_HIGHSCORES	The game should have a local record of the top high scores.	Pass	The game does not currently have a record of high scores.
A_5	SNFR_ACCESSIBILITY	There should be a way to modify the colour scheme in the for people who may be colour-blind.	Fail	No colour blind mode implemented
A_6	SNFR_MOBILE	The game should use an engine which allows you to easily transfer from pc to mobile.	Pass	The game is programmed on LIBGDX which can be easily transferred to android.

A_7	SNFR_TIME	You should be able to finish the game in under 5 minutes.	Pass	The game on average took 3 minutes to play for each user in our group.
A_8	SNFR_SIMPLE	The game should use arrow keys for the controls and the water cannons should be automatic.	Pass	The game uses up down left right arrow keys to control the game and if you are in close proximity to a fortress it will attack the fortress.
A_9	SNFR_FORTRESS	You are able to destroy all the fortresses in the game.	Pass	All the fortresses can be destroyed if the fire station is not destroyed.
A_10	SNFR_SAVING	User being able to save their game in its current state in one of 3 slots	Pass	The game state is saved (position, health and water of firetruck, etc) and can be loaded from the same slot it was saved into