Unit Test Results

ImageActor tests (Run with JUnit class ImageActorTests):

Test ID	Name of Test Function	Function(s) Tested	Function Use	Result	Details
Image_Con	testImageActor()	ImageActor()	Initialises ImageActor	Pass	Tests that position and dimensions are zero, there is no image and the bounding polygon is set
Image_ConString	testImageActorString()	ImageActor(String url)	Initialises ImageActor with image stored at url	Pass	Tests that position and dimensions are zero, there is an image and the bounding polygon is set
Image_sImage	testSetImage(String url)	setImage(String url)	Sets ImageActor's image to image at url	Pass	Tests that the image stored is not null.
Image_gImage	testGetImage()	getImage()	Gets ImageActor's image	Pass	Tests that null is returned if the ImageActor has no image and returns the image if it does have an image
Image_Clone	testCloneImageActor()	clone(ImageActor a)	Makes the image actor a copy of a	Pass	Tests that after cloning, an image actor that previously had no image now has the same image. Also tests position and dimensions.
Image_Origin	testSetOrigin()	setOrigin()	Sets ImageActor origin to centre of the image	Pass	Tests that the origin is half the width and half the height.
Image_Rect	testSetRectangleBoundary()	setRectangleBoundary()	Sets the bounding polygon as a rectangle with the same dimensions as the ImageActor	Pass	Tests that the bounding polygon's vertices are in the exact expected areas relative to the ImageActor
Image_Ellipse	testSetEllipseBoundary()	setEllipseBoundary()	Sets the bounding polygon as	Pass	Tests that a bounding polygon is set with 8 vertices

			an ellipse with 8 vertices		
Image_Poly	testSetBoundingPolygon()	setBoundingPolygon(float[] vertices)	Sets the bounding polygon to a polygon with vertices corresponding to those given	Pass	Tests that the vertices of the bounding polygon are exactly equal to the vertices given
Image_gBounds	testGetBoundingPolygon()	getBoundingPolygon()	Sets the bounding polygon's position and rotation to equal the actor and then returns the poly	Pass	Tests that return value isn't null if a polygon is set
Image_Overlaps	testOverlaps()	overlaps(ImageActor o, boolean resolve)	Returns if the 2 image actors are overlapping and separates them if resolve is true	Pass	Tests that 2 image actors that are overlapping return true and separate when overlaps with resolve true is run.

AnimatedActor tests (Run with JUnit class AnimatedActorTests):

Test ID	**Name of Test Function**	**Function(s) Tested**	**Function Use**	**Result**	**Details**
Anim_Con	testAnimatedActor()	AnimatedActor()	Initialises AnimatedActor.	Pass	Tests that timing starts as 0 and currentAnim is null
Anim_sFrame	testSetFrameSize()	setFrameSize(float x, float y)	Sets width as x and height as y.	Pass	Tests that width and height are correct after setting
Anim_gFrame	testGetFrameSize()	getFrameSize()	Returns a vector of	Pass	Tests that the correct width and height are returned

			(width, height)		
Anim_storeAnim	testStoreAnim()	storeAnim(String url, String name, float width, float fr)	Stores the animation stored at the url and uses the width as the frame width and fr as the framerate. Uses name to identify it.	Pass	Tests that the animation store isn't empty after an animation has been stored
Anim_getAnim	testGetAnims()	getAnims()	Returns all the animations stored in the actor	Pass	Tests that null is returned if the store is empty and the store is returned if it's not empty
Anim_setAnim	testSetAnim()	setAnim(String name)	Sets the current animation to the animation corresponding to name	Pass	Tests that currentAnim is not null after setAnim
Anim_Clone	testCloneAnimatedActor()	clone(AnimatedActor a)	Makes this AnimatedActor a copy of a	Pass	Tests that the animStore is no longer empty after cloning
Anim_Act	testAct()	act(float dt)	Updates the timing of the animation	Pass	Tests that the timing is no longer 0

MovingActor tests (Run with JUnit class MovingActorTests):

Test ID	**Name of Test Function**	**Function(s) Tested**	**Function Use**	**Result**	**Details**
Moving_moving	test()	setAcceleration(float accx, float accy), setVelocity(float velx, float vely),	Sets all of the variables required for the movement and gets	Pass	Tests that all the variables are correctly set and then tests that they modify the position appropriately given a certain jump in time.

		setAngularVel(float a), setMoving(boolean m), getVelocity(), getAcceleration(), getAngularVel()	them		

Entity tests (Run with JUnit class EntityTests):

Test ID	Name of Test Function	Function(s) Tested	Function Use	Result	Details
Entity_Con	testEntity()	Entity()	Initisalises Entity	Pass	Tests that the stats are set to 0 and the type is default
Entity_Con2	testEntityStringResourceManager()	Entity(String type, ResourceManager r)	Sets the type to the type given and uses the resource manager to load in the associated animations and speed	Pass	Tests that the stats are equal to the ones stored in the Entity of that type in the ResourceManager
Entity_sSpeed	testSetSpeed()	setSpeed(float s)	Sets the movement speed	Pass	Tests that the speed is set to the value given
Entity_gSpeed	testGetSpeed()	getSpeed()	Gets the movement speed	Pass	Tests that the speed returned is correct
Entity_sAngle	testSetAngle()	setAngle(float a)	Sets the angle of movement	Pass	Tests that the angle is set to the value given
Entity_gAngle	testGetAngle()	getAngle()	Gets the angle	Pass	Tests that the angle returned is correct
Entity_sMaxHealth	testSetMaxHealth()	setMaxHealth(float h)	Sets the max health	Pass	Tests that the max health is set to the value given
Entity_gMaxHealth	testGetMaxHealth()	getMaxHealth()	Gets the max health	Pass	Tests that the max health returned is correct
Entity_aHealth	testAddHealth()	addHealth(flo	Adds a onto	Pass	Tests that the health is correct

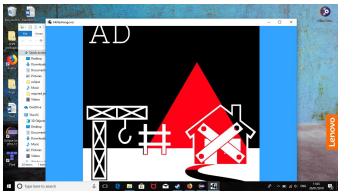
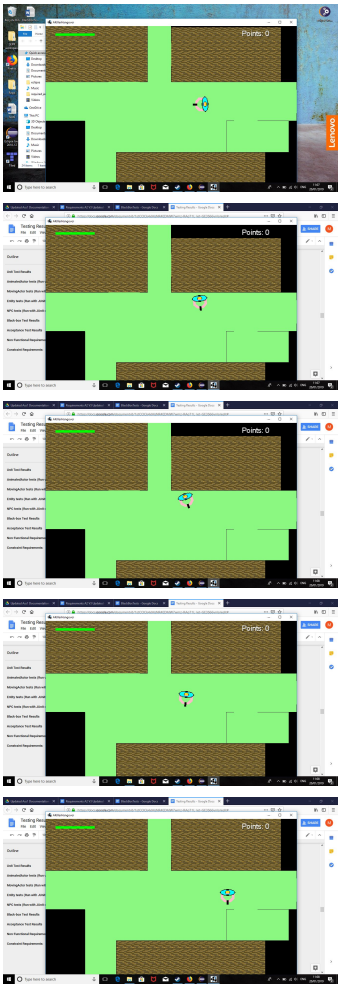
		at h)	the health up to the max.		after the addition.
Entity_tHealth	testTakeHealth()	takeHealth(float t)	Takes t from the health down to 0	Pass	Tests that the health is correct after the subtraction
Entity_gHealth	testGetHealth()	getHealth()	Gets the health	Pass	Tests that the health returned is correct
Entity_sType	testSetType()	setType(String t)	Sets the movement speed	Pass	Tests that the type is set to the value given
Entity_gType	testGetType()	getType()	Gets the movement speed	Pass	Tests that the type returned is correct
Entity_Clone	testCloneEntity()	clone(Entity e)	Sets this entity's stats and type equal to entity's	Pass	Tests that the entities have equal stats after cloning
Entity_VfromA	testSetVelocityFromAngle()	setVelocityFromAngle()	Uses the speed and angle to calculate velx and vely	Pass	Tests that velx and vely are correct
Entity_Act	testAct()	act(float dt)	Recalculates the velx and vely	Pass	Tests that the entity has moved correctly

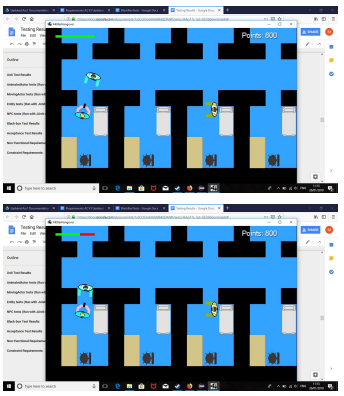
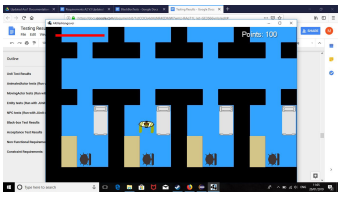
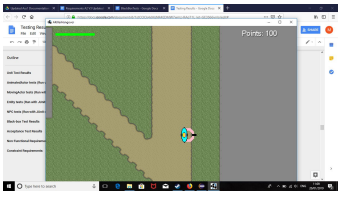
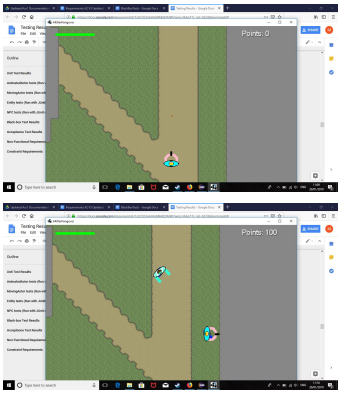
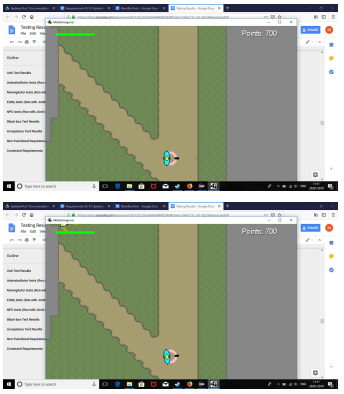
NPC tests (Run with JUnit class NPCTests):

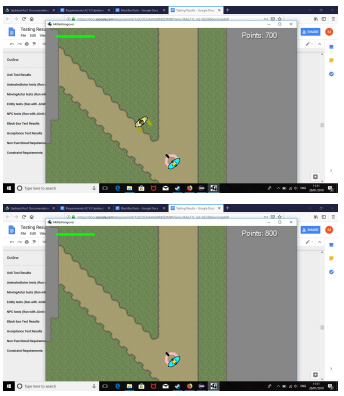
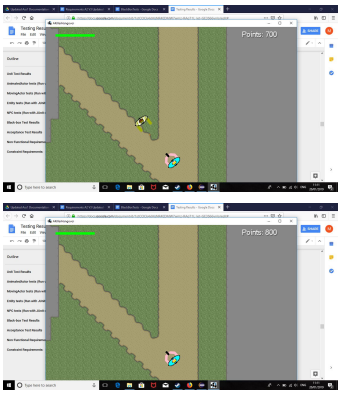
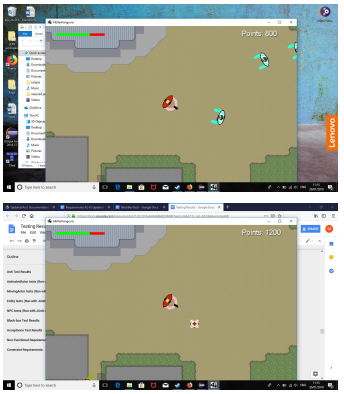
Test ID	**Name of Test Function**	**Function(s) Tested**	**Function Use**	**Result**	**Details**
NPC_Con	testNPC()	NPC()	Initialises the NPC	Pass	Tests that NPC is friendly by default
NPC_sFriendly	testSetFriendly()	setFriendly(boolean f)	Sets whether the NPC is friendly	Pass	Tests that friendly is set to value given
NPC_gFriendly	testGetFriendly()	getFriendly()	Returns if the NPC is friendly	Pass	Tests that value returned is what is stored
NPC_Chase	testSimpleChasePlay	simpleChase	Uses the	Pass	Tests that the NPC will have

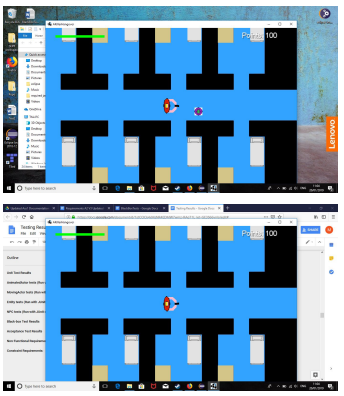
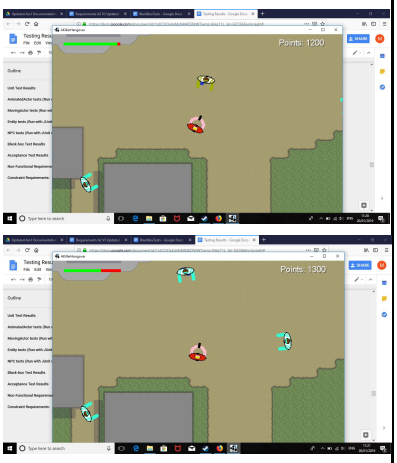
	er()	Player(float px, float py)	player's coordinates (px and py) to make the NPC look at and follow the player		the correct velocity based on the player's position

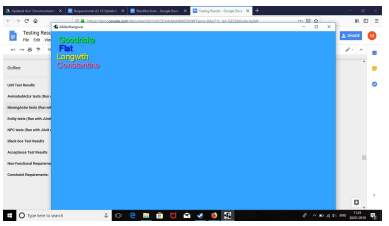
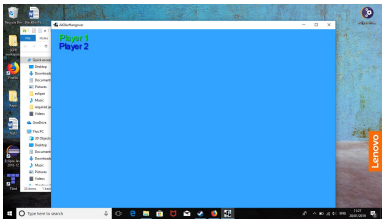
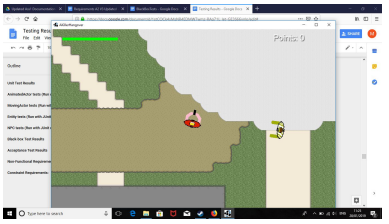
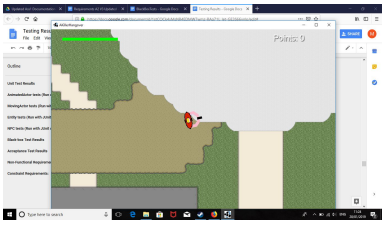
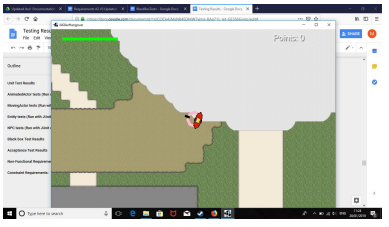
Black-box Test Results

Test ID	Test	Relevant Requirements	Expected Result	Actual Result	Overall Result	Evidence
BB_Start	Game successfully starts up	Con.Run	Game correctly starts up to show the group logo page	Game correctly starts up to show the group logo page	Pass	
BB_Control	Player moves when WASD keys pressed	Non.Play, Func.Input	Character moves up when W pressed, down when A pressed, left when S pressed and right when D pressed	Character moves up when W pressed, down when A pressed, left when S pressed and right when D pressed	Pass	

BB_Health	Player loses health when touched by a zombie	Func.UI, Func.Vary	When a zombie collides with the player, the amount of green in the health bar decreases to show health lost	When a zombie collides with the player, the amount of green in the health bar decreases to show health lost	Pass	
BB_KO	Player dies if all health is lost	Non.Play, Non.Enjoy	When the health bar is fully red, the player character is no longer playable	When the health bar is fully red, the player character is no longer playable	Pass	
BB_Collide	Entities collide with walls instead of passing through	Non.Play, Non.Enjoy	Entities collide with walls instead of passing through	Entities collide with walls instead of passing through	Pass	
BB_Shoot	Left clicking causes the player to shoot	Non.Play, Non.Enjoy	Left clicking causes the player to shoot	Left clicking causes the player to shoot	Pass	
BB_WallHit	Bullets disappear when hitting a wall	Non.Play, Non.Enjoy	Bullets disappear when hitting a wall	Bullets disappear when hitting a wall	Pass	

BB_EnemyHit	Bullets disappear when hitting an enemy	Non.Play, Non.Enjoy	Bullets disappear when hitting an enemy	Bullets disappear when hitting an enemy	Pass	
BB_EnemyKO	Enemies die when shot	Non.Play, Non.Enjoy	Enemies cease to move towards or hurt the player when shot	Enemies cease to move towards or hurt the player when shot	Pass	
BB_PowerDrop	Enemies randomly drop powerups or powerdowns	Non.Play, Non.Enjoy, Func.Powers	Killing an enemy occasionally causes a powerup or powerdown to appear at the location of the zombie's death	Killing an enemy occasionally causes a powerup or powerdown to appear at the location of the zombie's death	Pass	
BB_PowerEffect	Powerups affect gameplay	Non.Play, Non.Enjoy, Func.Powers	Powerups affect gameplay	Powerups affect gameplay	Pass	

BB_PowerDedpawn	Powerups not picked up despawn after a certain amount of time	Non.Play, Non.Enjoy, Func.Powers, Non.FPS	Powerups not picked up despawn after a certain amount of time	Powerups not picked up despawn after a certain amount of time	Pass	
BB_HitGain	Gain points when zombie killed	Non.Play, Non.Enjoy, Func.Points	Gain points when zombie killed	Gain points when zombie killed	Pass	
BB_SafeGain	Gain points when entering a safe area for the first time	Non.Play, Non.Enjoy, Func.Points	Gain points for entering a safe area for the first time	No points gained for entering a safe area	Fail	
BB_MiniGain	Gain points when completing a minigame	Non.Play, Non.Enjoy, Func.Points	Gain points when completing a minigame	No minigame available to gain the points	Fail	
BB_AvoidGain	Gain points when avoiding zombies	Non.Play, Non.Enjoy, Func.Points	Gain points when not being attacked by zombies	No points gained for avoiding zombies	Fail	

BB_Navigate	Provide some ability to navigate from one place to another	Func.Input, Func.Area, Func.Safe, Non.Enjoy, Non.Play,	Provide some ability to navigate from one place to another	Map screen pops up when player reaches the edge of a screen	Pass	
BB_Choose	The player can select a type of character to use	Func.Input, Func.Char, Non.Enjoy, Non.Play	When the game starts, the player can choose what type of character to use	The game starts, the character selection screen appears, the player can choose between the options	Pass	
BB_Zombie	The game is zombie themed	Con.Theme	Enemies that spawn in the game are zombies	The only enemies that spawned in the game were zombies	Pass	
BB_Turn	The player character must turn to face the mouse cursor	Func.Input, Non.Play,	When the mouse cursor moves, the player character remains facing the mouse cursor at all times	The mouse cursor is moved. As it moves, the player character faces the cursor at all times.	Pass	 
BB_Follow	Enemies follow the player	Func.Vary, Non.Enjoy, Non.Play	As the player moves, enemies turn and move to follow them	The player moves around, enemies turn and move to follow them	Pass	

						
BB_Mini					Fail	

Acceptance Test Results

Functional Requirements:

Test ID	Requirement ID	Fit Criterion	Result	Evidence
Acc_UI	Func.UI	All information user requires should be available through GUI	Pass	A GUI is provided for the player and this GUI shows the player how many points they have currently got and their current health.
Acc_Input	Func.Input	Game will take both keyboard and mouse input	Pass	As described in the user manual, the game is played using both the mouse and the keyboard.
Acc_Points	Func.Points	The game will have a points system within which the player gains points for avoiding enemies, defeating enemies, reaching safe areas and winning minigames	Fail	No minigames currently available to gain points and player doesn't gain points for avoiding the zombies.
Acc_Char	Func.Char	The game will have number of characters that differ from one another in both the values of their attributes (Strength, Speed, etc) and in appearance	Partial Pass	We fulfil the fit criteria written here but we do not fulfil the statement in the Requirement which demands we have 3 different playable characters. We currently only have 2.
Acc_Area	Func.Area	There will be a number of different areas in the game that are both major landmarks of the university and undeniably separate places in reality	Partial Pass	We fulfil the fit criteria written here but we do not fulfil the statement in the requirements which demands we have 6 different locations. We currently only have 5.

Acc_Safe	Func.Safe	There will be some areas which increase the difficulty of the game and grant points when reached. These areas must be totally safe and failure of the game must be impossible in these areas.	Pass	We currently have 1 safe area in the game which grants points, does not contain any enemies to kill the player and increases the spawning rate of the zombies.
Acc_Mini	Func.Mini	There will be at least 1 game within the main game that is functionally distinct from the main game and undeniably shorter	Fail	We currently have not implemented a minigame.
Acc_Boss	Func.Boss	Will contain at least 2 enemies in the game that are distinctly more difficult to combat than other enemies in the game and also visibly different from other enemies	Fail	We currently have not implemented these.
Acc_Vary	Func.Vary	Will contain at least 2 enemies that are not as difficult as 'boss enemies' but are still distinct from one another	Pass	We have 2 different kinds of zombies in the game which both are different colours to distinguish them from one another and also vary in attributes.
Acc_Powers	Func.Powers	Will contain at least 5 powerups and powerdowns that are distinct from one another in what they do to help or hinder the user	Partial Pass	We currently have 3 powerups in the game. We do not fulfil the requirement for 5.

Non-Functional Requirements:

Test ID	**Requirement ID**	**Fit Criterion**	**Result**	**Evidence**
Acc_Enjoy	Non.Enjoy	The game will be tested on the SEPR cohort during development to ensure both proper balancing of game difficulty and enjoyability of	Fail	We have not yet conducted this test.

		gameplay		
Acc_Play	Non.Play	The game will be tested on the SEPR cohort to ensure that the game's controls are not too complicated and the UI conveys the information the player needs appropriately	Fail	Not yet tested.
Acc_FPS	Non.FPS	Game will be tested on the Computer Science department computers regularly to ensure they can run the game at 60 FPS	Fail	Not yet tested.

Constraint Requirements:

Test ID	**Requirement ID**	**Fit Criterion**	**Result**	**Evidence**
Acc_PG	Con.PG	We will have an initial tutorial area that serves as a completely 'PG' area that can be shown to any and all prospective students.	Partial Pass	We do not currently have a tutorial area but we do have completely PG areas as the entire game is PG in its current incarnation.
Acc_Theme	Con.Theme	The game will be shown to both the customer and the SEPR cohort regularly to ensure the requirement of a zombie theme is met	Pass	The majority of the characters in the game are zombies.
Acc_Run	Con.Run	The system will be tested regularly on those computers to ensure they are capable of running the system	Pass	They have proven capable of running the system
Acc_Java	Con.Java	In order to ensure this, the system will be created using the	Pass	The entire game is coded in Java. There is no part of it coded in another language.

		Java-based game library LibGDX		
Acc_Sell	Con.Sell	In order to ensure this, we will avoid the use of any open-source software or anything else that could cause licensing issues.	Pass	Our core system only uses LibGDX and therefore is not restricted by licensing.