Assignment 2 Task 2: Application Design

By Dr Derek Peacock

Task 2 P6: Use appropriate tools to design a solution to a defined requirement

Requirements Summary

In this game the player moves a crab around the sand-scape trying to catch and eat worms that live in the sand and pop up from time to time from their burrows in the sand. Each time a worm is eaten the crab's energy levels increases. Each time the crab moves its energy levels decreases. When its energy levels fall below a certain value, the crabs speed of movement decreases.

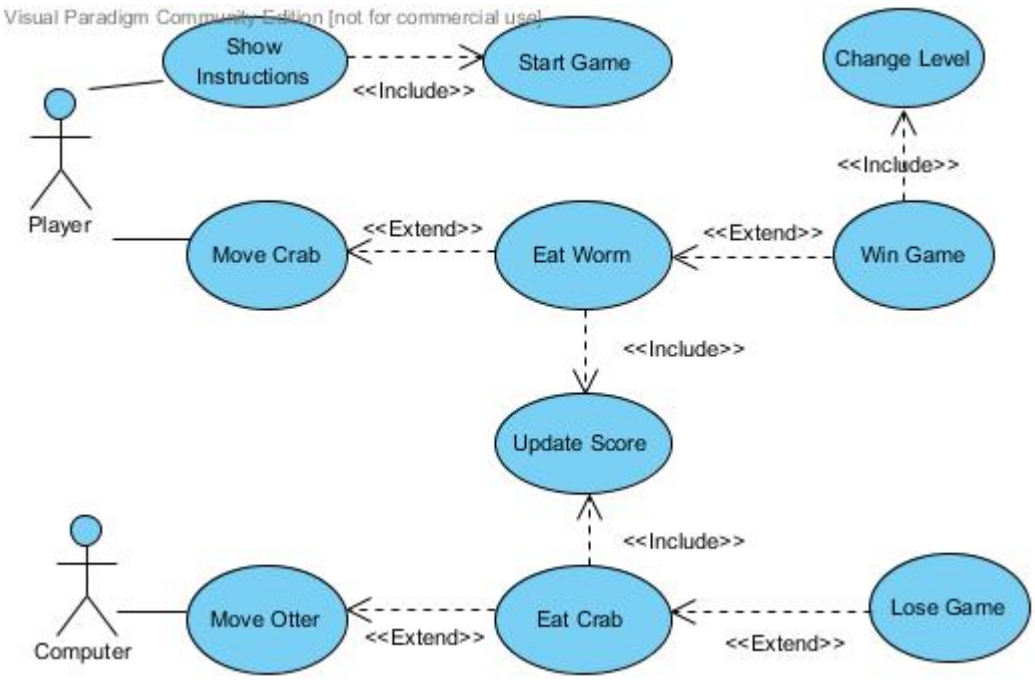
The purpose of the game is to progress to the end of the third level without being eaten by the otter. The otter loves to eat crabs, and searches for them largely using line of sight. It the otter sees a crab it will give chase. The crab's only defence is to hide behind rocks, until the otter is distracted and loses interest.

The crab successfully completes a level if it eats all the worms in that level before being caught by the otter. The quicker the crab eats the worms, the higher the score. If the crab is eaten, it can be re-generated three times per level, but its score is decreased each time it is eaten.

Hardware Platforms

It is planned that this game shall run on Android platforms on tablets with a minimum resolution of 1200 x 800 pixels.

Use Case Diagram

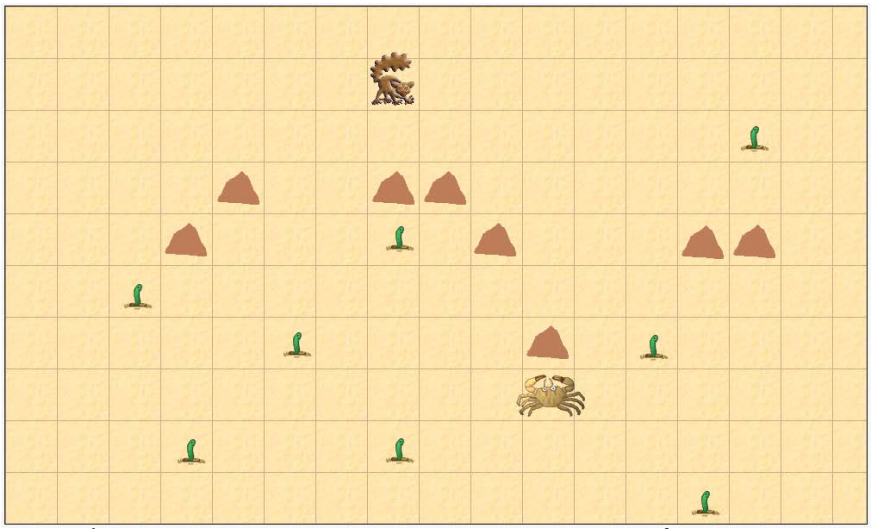


Use Case Specifications

Eat Crab Use Case

Use Case Name:	Eat Crab	**Author**	Derek Peacock
Description: **Pre-conditions**	When the Otter touches the crab, the crab is eaten and the player has lives left the crab is re-generated and the level re-starts. If there are no lives left, the game ends. The otter touches the crab		

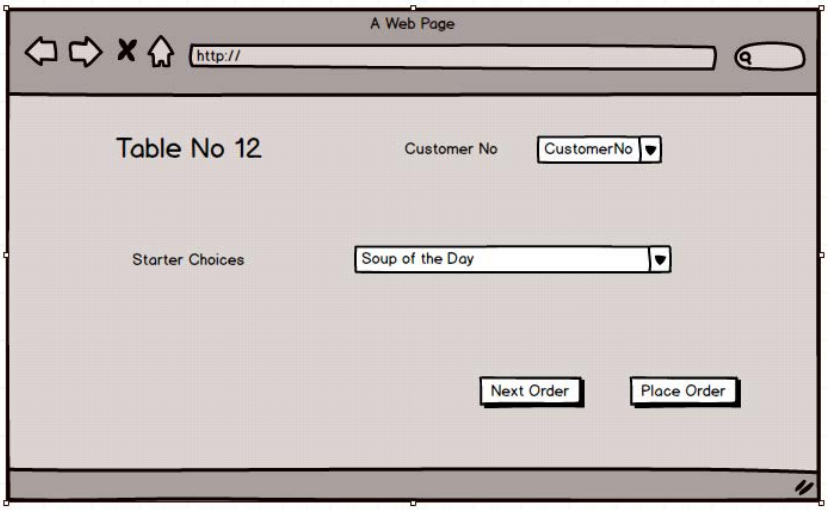
Post-conditions
 	
Normal Events
	The level re-starts or the game ends
	
	• The score is reduced
	• If crab has more lives
	  • Lives are decreased by 1
	  • All points gained in that level are lost
	  • The level is re-started		

Alternative Events	• If crab has no more lives the game ends		
Screen Layout	 Actors objects are approximately 60 x 60 pixels, the GameWorld is 1200 x 800 pixels. Missing score board!		

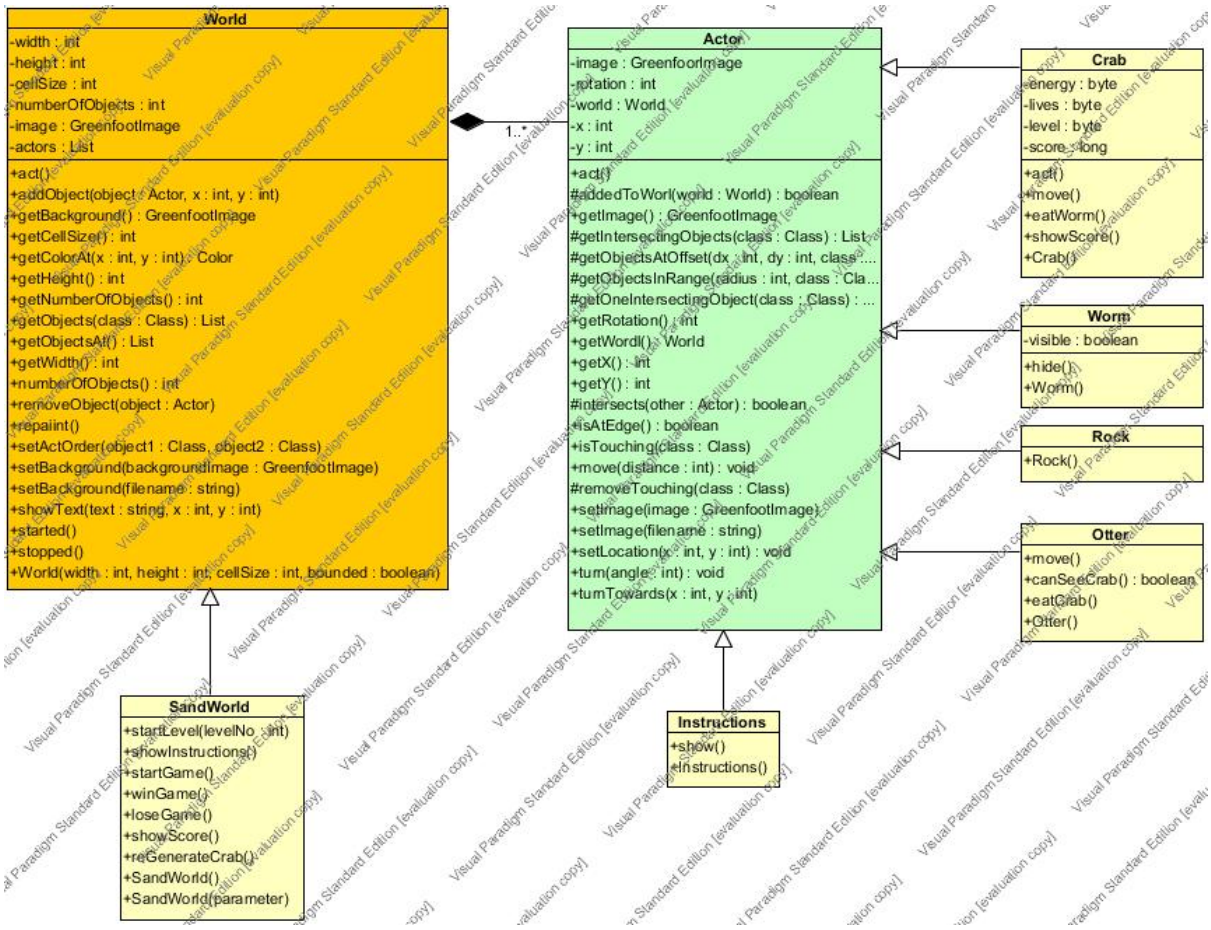
Order Meal Use Case

Use Case Name:	Order Meal	Author	Derek Peacock
Description:	The customer selects one of the pizzas from the menu and indicates their preferences.		
Pre-conditions	The table number has been recorded		
Post-conditions	The chosen starters and mains have been recorded for one customer		

Normal Events	• The waiter enters the customers selected starter
	• The waiter makes a note of any variations
	• The waiter enters the customer's selected main course
	• If a appropriate meat course the waiter asks for the cooking instructions
	  • Rare
	  • Medium
	  • Well done
	• The waiter notes any customer variations		

Alternative Events			
Screen Layout			

Class Diagram



M2: Justify the choice of data types and software structures used in a design solution

Data Types

Class	Attribute	Max range of Values in the game	Justification
Crab	energy: int	0..100	
	lives: int	0..3	
	scorel: int	0..20,000	In java there are no data types for large whole numbers that are only positive in value. So int with a range of values -- 2,147,483,648 .. 2,147,483,647 is the best choice.
SandWorld	level: int	1..3	
Worm	Visible: boolean	Yes or no	

Software Structures

The Greenfoot system offers two basic classes which together can be sued to form a simple 2D game, the World class and the Actor class. The purpose of the World class is to provide a background and container for many instances (objects) of the Actor class . An Actor is a class that does something in the game, and usually but not always moves around the world.

The **SandWord** class is needed to …

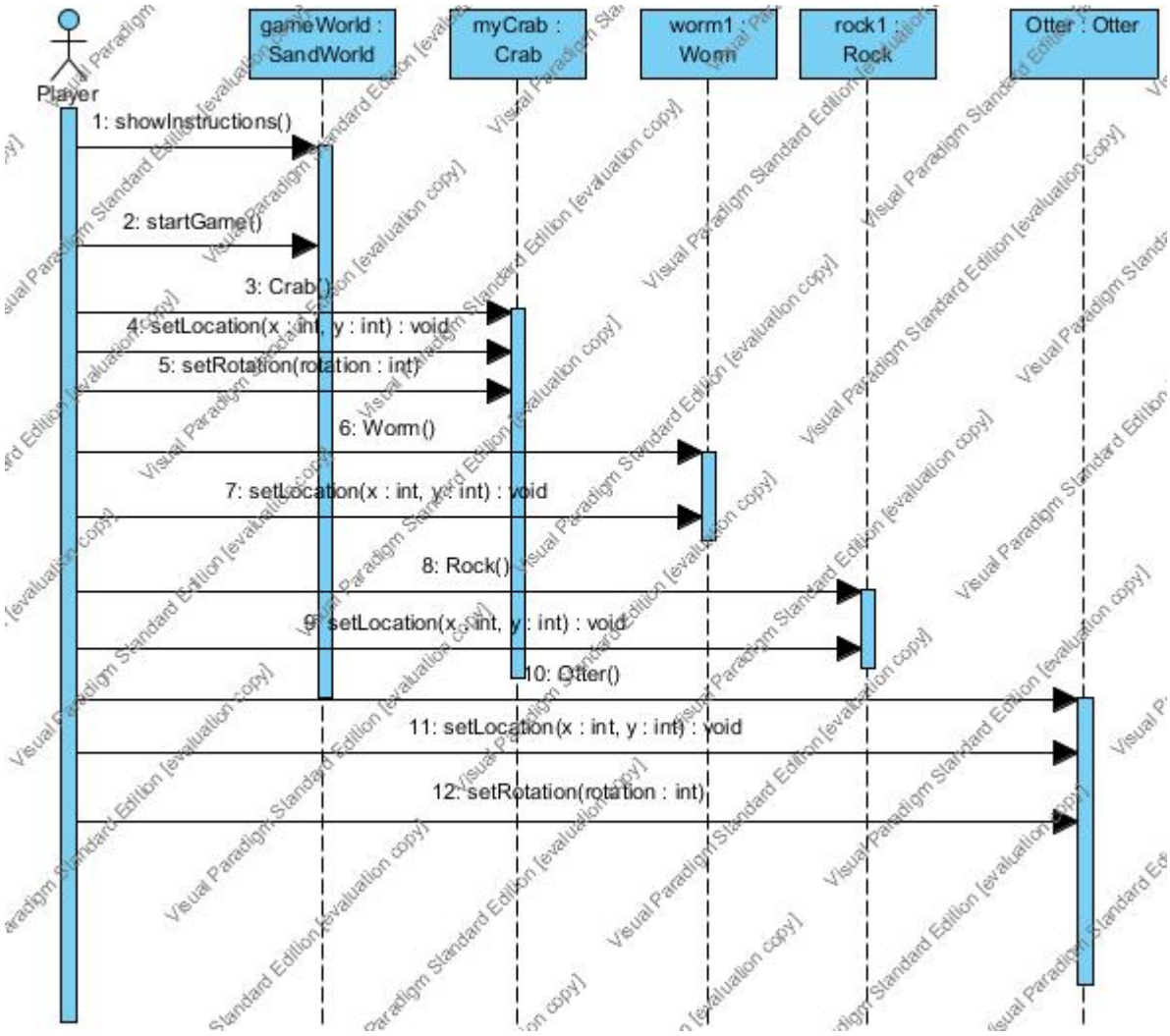
The **Otter Class** is needed to …

The **Crab Class** is needed to …

The **Worm Class** is needed to …

The **Rock Class** is needed to…

Sequence Diagram



D2 Develop algorithms to represent a design solution algorithms eg using pseudo code

The Otter is going to hunt crabs by line of sight. The Otter's vision is quite good, and the otter can see to a distance around 60% of world width. However if the Scot the Crab is behind a rock then the Otter will not be able to see the Scot.

Moving the Otter will be based on checking along a direct line between the otter's position and the crab's position. If there are no rocks in between then the otter will move directly towards the crab, otherwise the otter will move in one of four directions at random.