Written by: Santosh Gurung

Table of Contents

1. Introduction

mrCrab-v1 is crab world game which is written in java Greenfoot.

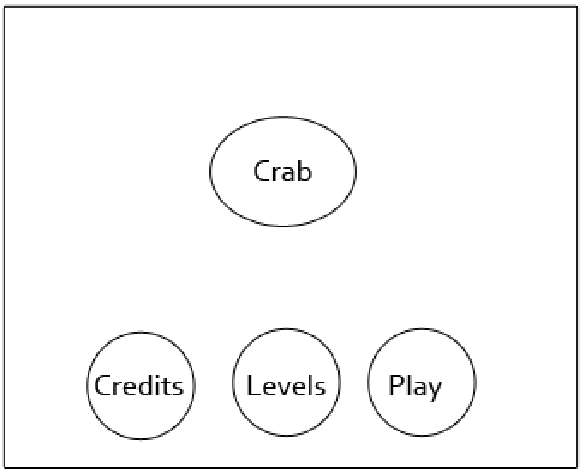
In this game crab has to eat worms to complete a level and if the lobster catches the crab game over.

Software use:

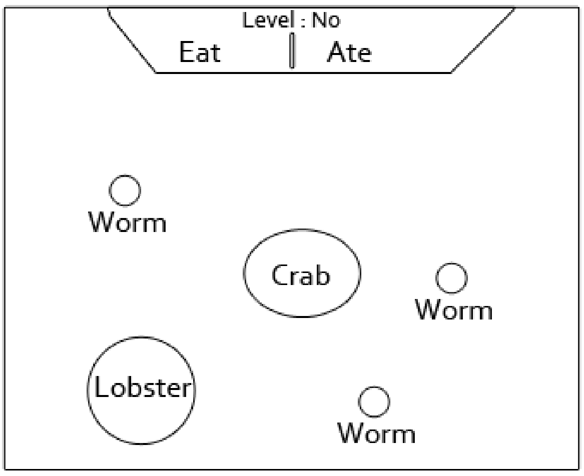
i. Programming language: Greenfoot (JAVA)
ii. Graphics (Adobe Photoshop CS5)

2. GUI Design (Menu Design)

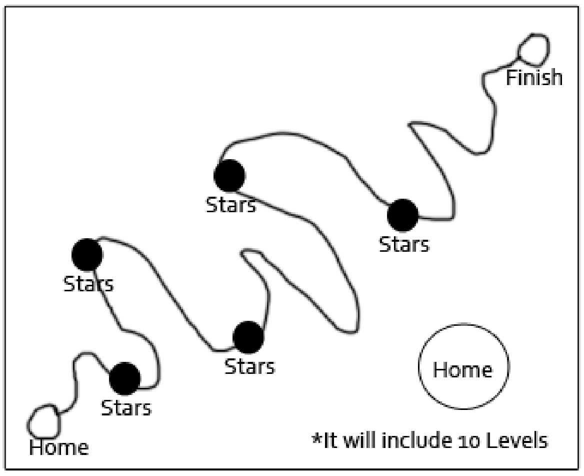
Home or main menu design



Game play window design

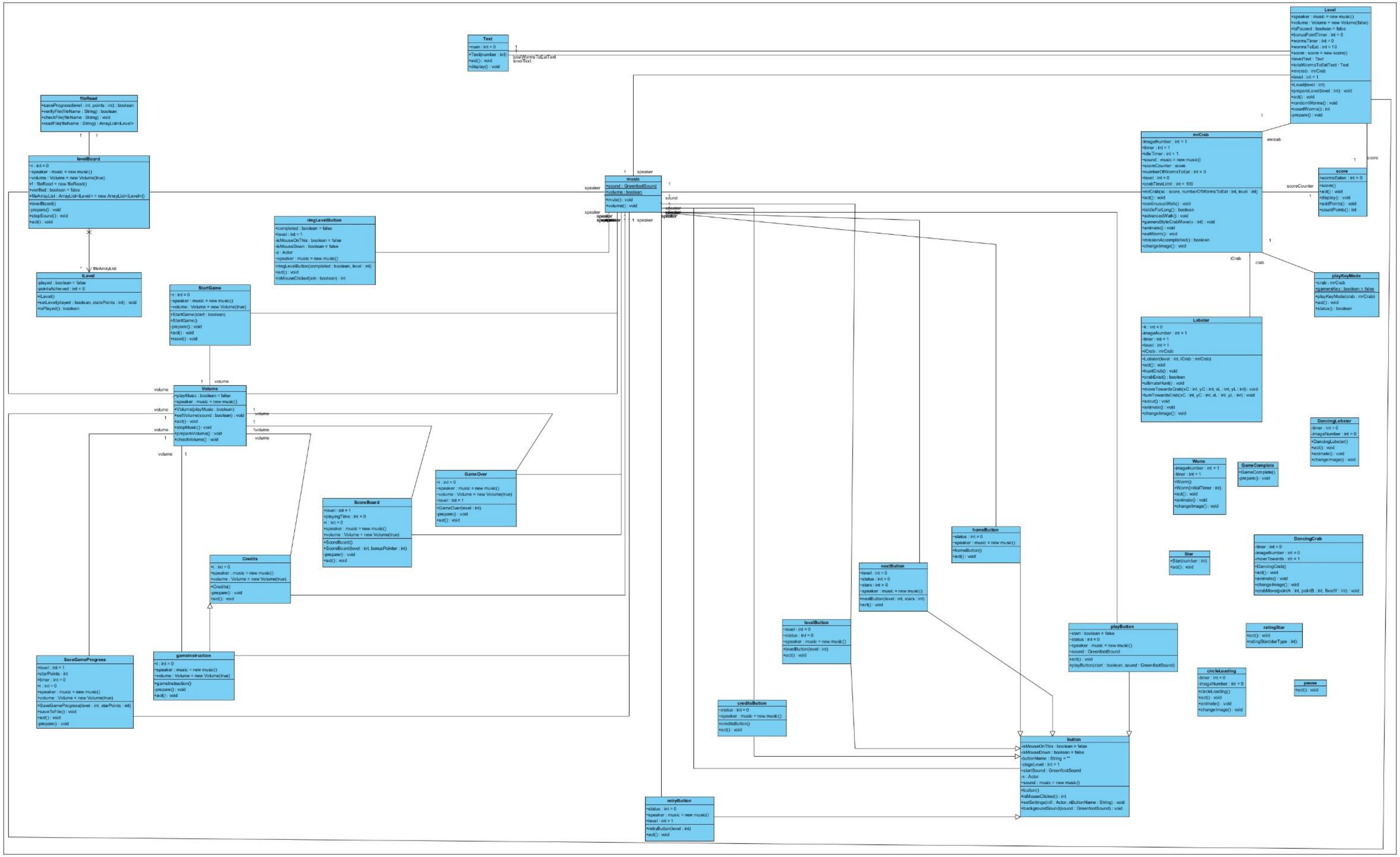


Level board window design.



3. Game design in paper

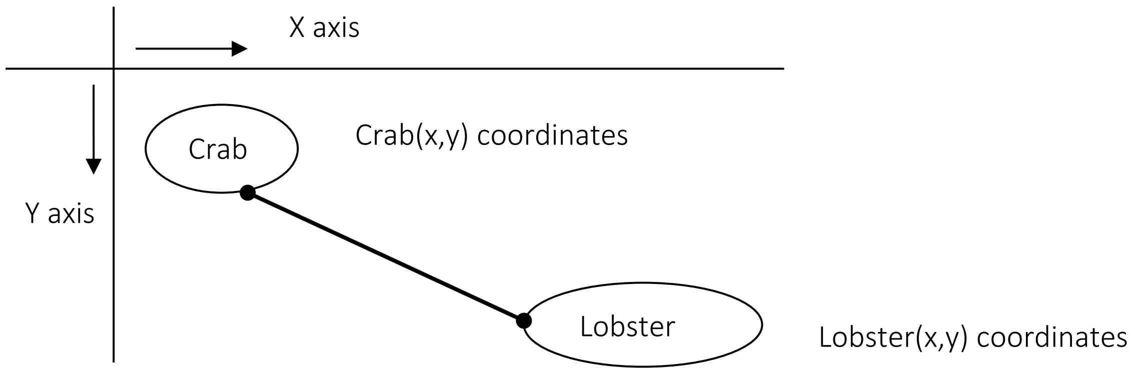
3. Class Diagram



5. Algorithms & Coding for mrCrab-v1

5.A Intelligent Lobster attack mode

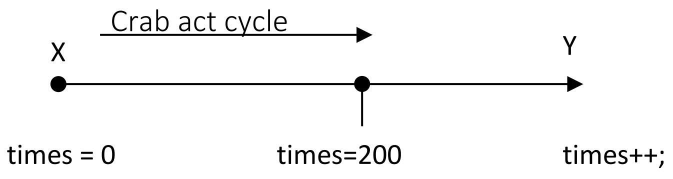
In version 1.0 of the Crabworld-x the movement for lobster was in random movement (A prototype version I programmed, *Included in CD), however it lacked intelligent attack. Hence, to achieve intelligent attack mode, I have thought to use maths to accomplish this task. Here is the scenario,



Scenario 1.a: Crab and Lobster coordinates

i. When to launch attack mode

To achieve intelligent lobster attack mode and to use it in different levels, I have thought of checking idle time for crab which means the number of act() cycles for the crab which increments the times++; for example,



Scenario: 1.b Showing idle time of crab.

Idle state for crab is when the user does not press right as left key. When the user press one of those key times is set to 0 i.e, times=0 and when no key is pressed 0 times increments.

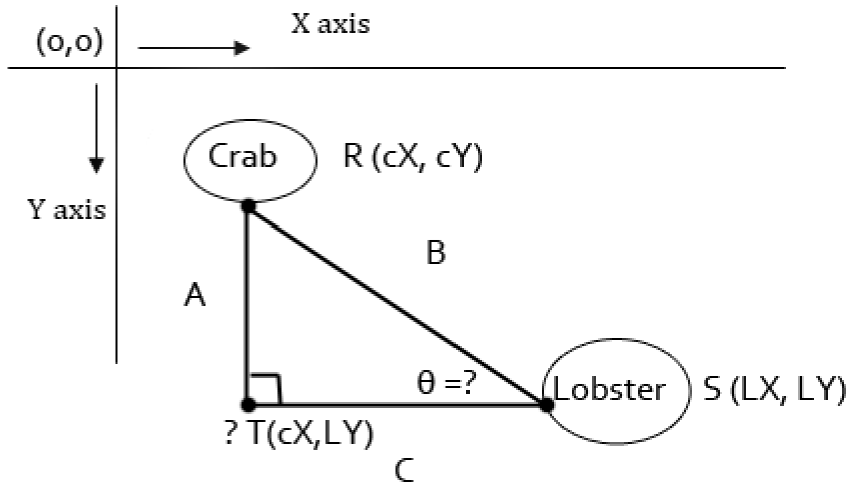


Fig: 1.c Co-ordinates and the angle of inclination of lobster

ii. How to carry out attack mode using maths

To initiate ultimate lobster attack mode I have thought of moving the lobster towards crab to eat it. To move lobster to crab in a natural way i have thought of using maths which is to get angle θ to rotate lobster towards crab.

To get the angle θ, we need to know,

a. T (X,Y) which is X value of crab and Y value of lobster. Hence, it is T(cX, eY).
b. Value of A, B and C

Hence we have formula,

C = cX – eX (As C is in X axis)
A = cY – eY

We can use Pythagorean theorem.
$B = \sqrt{(A^2 + C^2)}$

Now we can get the value of θ using Sin, Cos and Tan,

$$\boxed{\Theta = \tan^{-1}(A/C)}$$

Once we get the value of θ, we know the angle to face the lobster towards the crab.

In greenfoot, we can use built in method setRotation (int rotation).

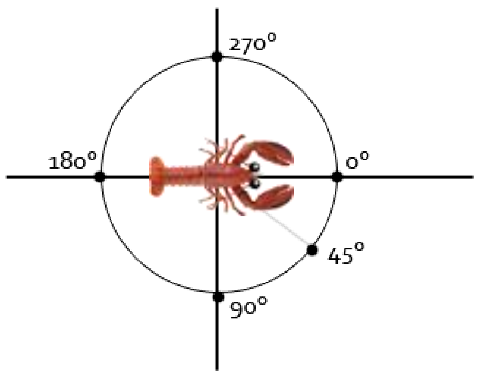


Fig 1.d setRotation(value) Rotation.

To make the lobster face towards 45^0, we need to use *this.setRotation(45);*
However, while testing the formula:

$\Theta = \mathrm{Tan}^{-1}(A/C)$

I have noticed the value of θ as below:

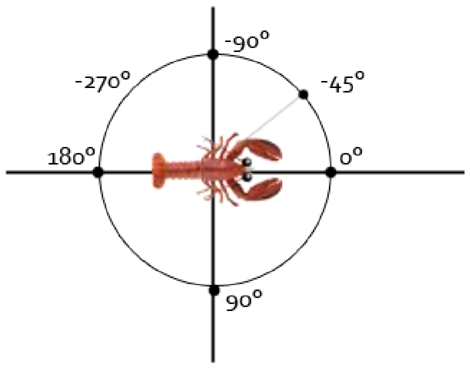


Fig 1.e Lobster angle using the formula

As we can see in fig 1.e we cannot use those negative values as the setRotation(int) function in Greenfoot requires 0^0-360^0 as shown in fig 1.d .

Hence we can use the following code,

```
If(angle < 0)
{
        angle=360-angle;
}
```

If angle is -270^0 then,

angle=360-(-270)
=630

As 630 isn't the angle expected and -270 is negative value we can fix it by two methods,

a) angle=360+(-270)
b) angle=math.abs(-270)

math.abs(x) is a method in java to make negative value to positive value.
We can make lobster ultimate attack to by setting the value of idleTime different for each level. For example,

```
if (level==1) idleTime=400;
if (level==2) idleTime=100;
if (level==3) idleTime=10;
```

Lowest value of idleTime means the attack mode is activated frequently.

Final code for the lobster is in Appendix A

5.B Random appearance of worms

In test version of crabworld-x which was written as a prototype had random appearance of worms in the window.

```
Int x = Greenfoot.getRandomNumber(580);
int y = greenfoot.getRandomNumber(430);

addObject(new worm (); X,Y);
```

The problem was some of the worms appeared on top and didn't showed their head and same for the left corner as shown in the screenshot below:

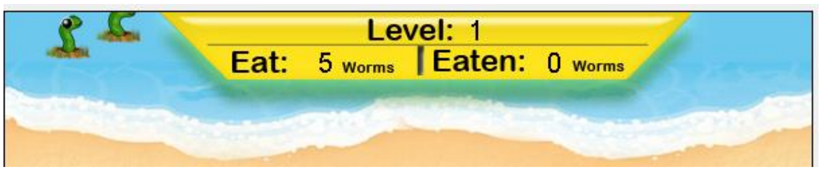


Fig 1.f: Worm outside the window.

Sometimes random number generated makes the worm to appear in such manner. To solve this problem, I thought to add eX to any random generated number X and eY to any Random generated number Y.

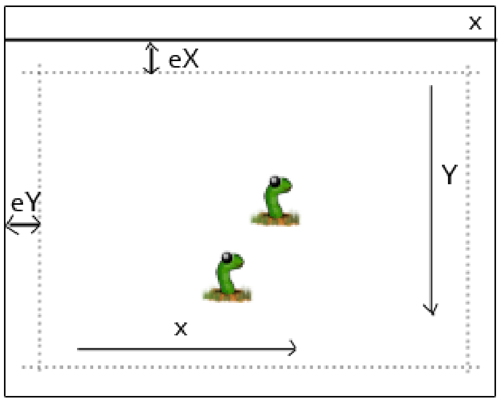


Fig 2.a Show eX & eY

It solved coordinates of worms as problem of worm appearing outside viewing window, however random generation of worm still persist. To solve that a time variable is set to 0 as times = 0; and in each act cycle of worm i.e,

```
public void act()
{
  times ++;
}
```

Times++ makes the times to increment by 1, hence we can use it as a time interval for worm to appear i.e,

```
If(Wormstimes%97==0)
{
          //add new worm
          //code omitted
}
```

In the above code if wormsTimes was 50, then 50 divide by 97 and the remainder wil not be 0. If the Wromstimes was 97 and the remainder would be 0 and a new worm will be created.

Below is the random worms method:

```
public void randomWorms()
{
   if(!this.isPaused){
      if(countWorms()<10){
         if(wormsTimer%97==0){
            int x=Greenfoot.getRandomNumber(580);
            int y=Greenfoot.getRandomNumber(430);
            //It ensures the worms isnt in the left hand corner and is visible area.
            if(x<65) x=83;
            if(x>554) x=550-(x-540);
            if(y<77) y=72;
            if(y>410) y=410-(y-410);
            addObject( new Worm(), x, y );
         }
         wormsTimer++;
      }
   }
}
```

2.C Implementation of Pause button:

To accomplish a pause button was a bit challenging as there is not any built in Greenfoot method. There is stop() which stops the whole application or stops the execution of act() method in all World and Actor class. That means we cannot use the same pause button to play as the act() method of pause button is stopped as well.

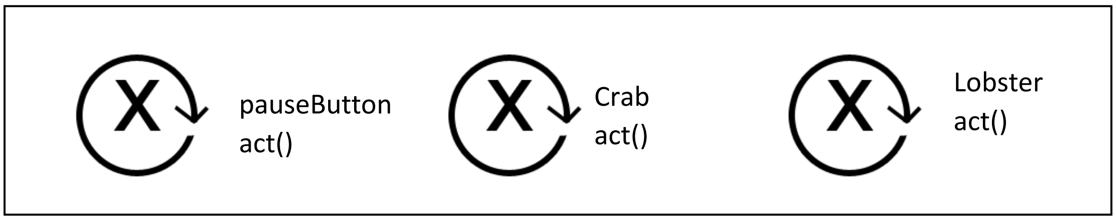
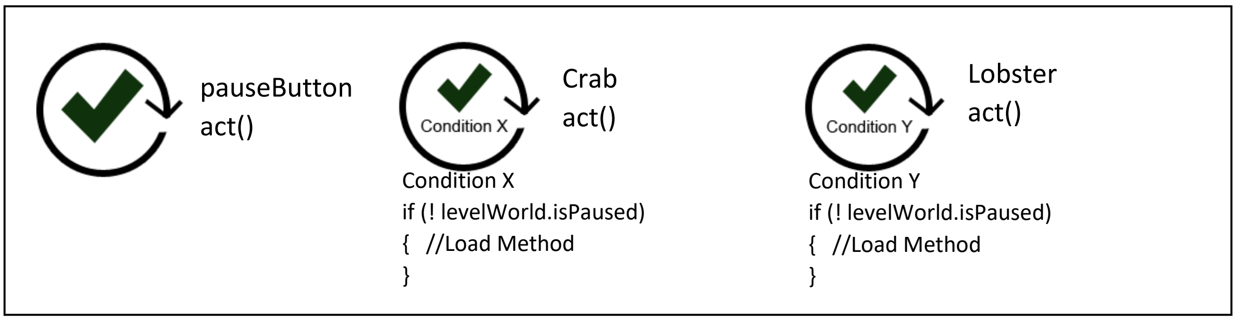


Fig 3.a Act method stopped in pauseButton, crab & lobster.

X represents the statements inside the act() method.

To solve it, a Boolean variable was used is paused in level window and when there is a mouse click is pauseButton set isPaused = True; and also check status of pauseButton. So now pauseButton act() method will be executed. However, crab and lobster act() method will be executed but we can use is paused to check if game is paused and stop execution of crab and lobster act() method itself as shown in code below and figure 3.b.



```
Condition X
if (! levelWorld.isPaused)
{  //Load Method
}
```

```
Condition Y
if (! levelWorld.isPaused)
{  //Load Method
}
```

Fig 3.b Act methods runs but crab and lobster has condition X and Y.

In the above, levelWorld is the object of level class and ifConditionX is met which means if a game is not paused, then only crab will run its methods like movement same for lobster.

5.D Smart level board and game progress

The idea behind level board which shows the game progress for example, if the user completes level 1,2 with 3 stars (★★★) and level 3 with 2 stars (★★) and the rest of the levels did not attempt then show not attempted sign.

To accomplish this task, there are different method. However, they must store it and save it for the next application execution suitable methods,

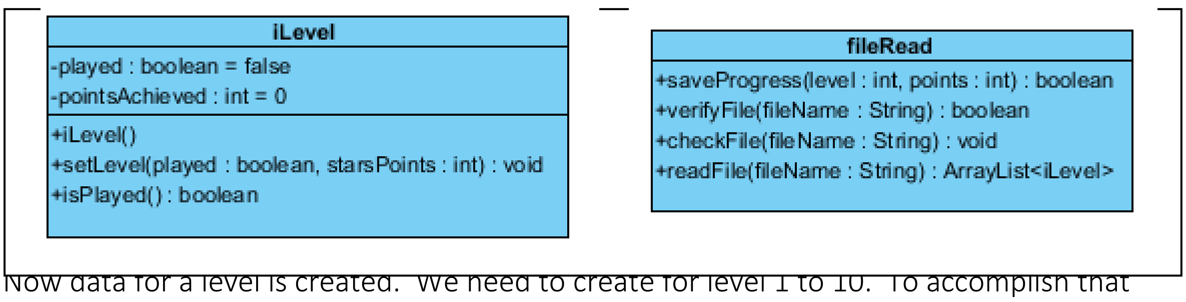
a. Use database to store level details
b. Use text file to store level details
c. Use Greenfoot userInfo library

Greenfoot userInfo does not allow us to solve complex data such as the mentioned above. Database would require user to follow tedious steps to make database connection and run it. Hence, for a small such as the game text file is appropriate.

5.E Serializable class

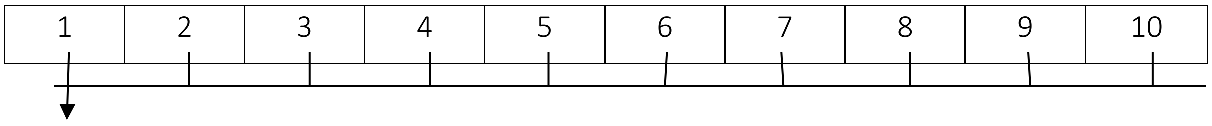
To store a level data a serializable class called ilevel is created which has:

Played variable is a Boolean value which states if the level is played and points Achieved would be set 0 – 3 which indicates 0 as 0 stars and 3 as 3 stars.


iLevel
-played : boolean = false
-pointsAchieved : int = 0
+iLevel()
+setLevel(played : boolean, starsPoints : int) : void
+isPlayed() : boolean
fileRead
+saveProgress(level : int, points : int) : boolean
+verifyFile(fileName : String) : boolean
+checkFile(fileName : String) : void
+readFile(fileName : String) : ArrayList<iLevel>

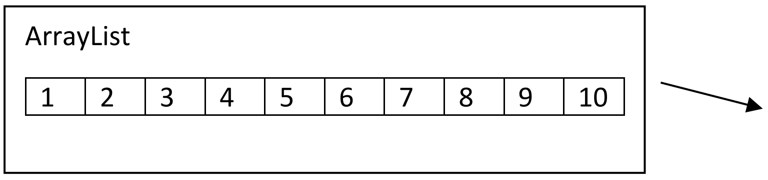
Now data for a level is created. We need to create for level 1 to 10. To accomplish that Arraylist could be used.

Arraylist



Add(Object of class iLevel)

→ Each Object has its own data level 1-10 is created, we can store the ArrayList in the file itself. As the class iLevel is serializable we can store it in file as follows:



Note:

In actual file it is written differently, this is for illustration only.

Fig 5.a ArrayList stored in file.

5.F Animation of crab, worm and Lobster

To animate the actor, the best way would be to create “Animation” actor and create crab, worm and lobster.

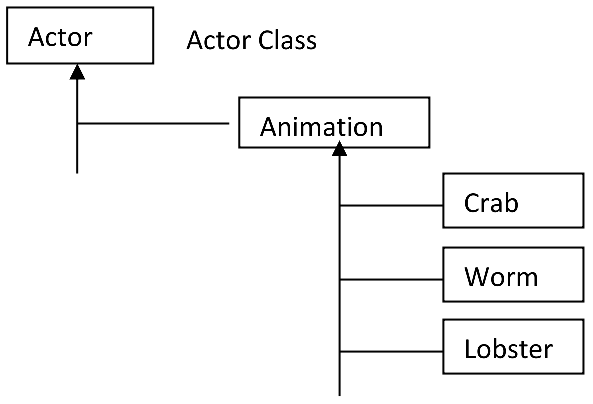


Fig 6.a Showing hierarchy of class

The method used is very basics of movie making as making animation which is to run frames simultaneously repeated.

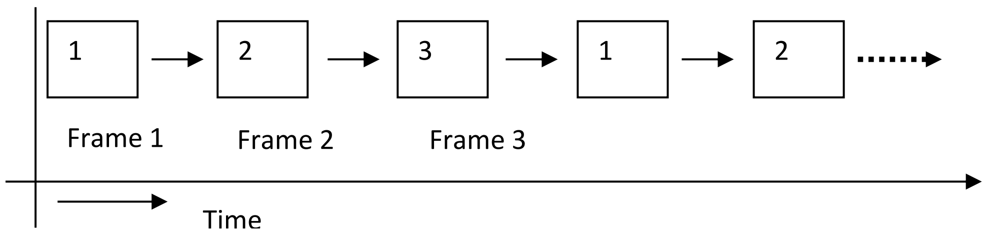
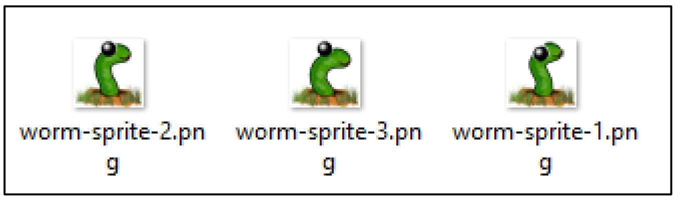


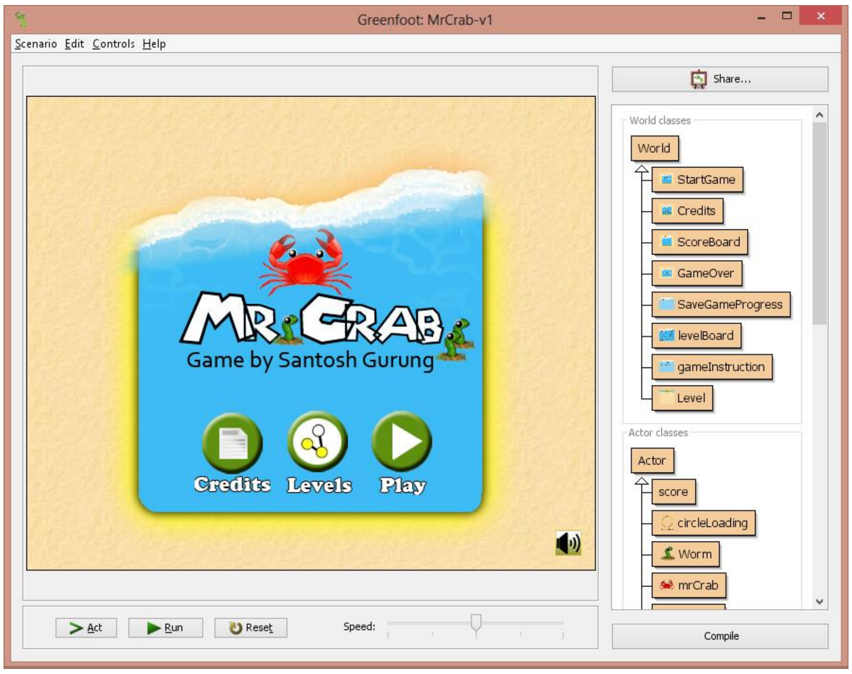
Fig 5.6 Showing Frames for continuous animation. Firstly Frame 1 would be shown followed by Frame 2 and Frame 3 and so on. It creates optical illusion which makes moving character.



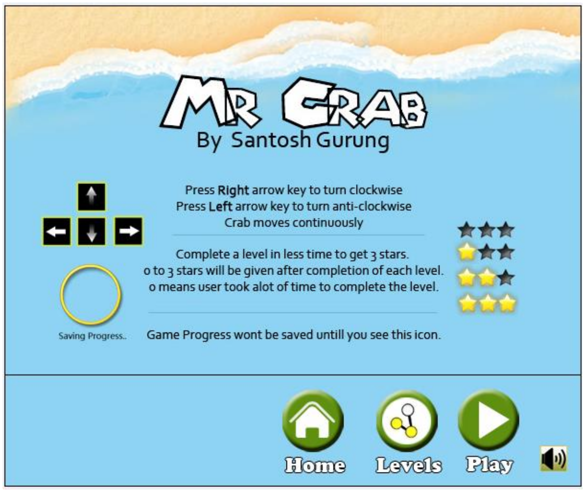
The above image files has 1, 2, 3 in file name which would be used to animate worm.

6. Game Screenshots:-

i) MrCrab-V1 in Greenfoot development environment. Also main screen (Home) of the game.



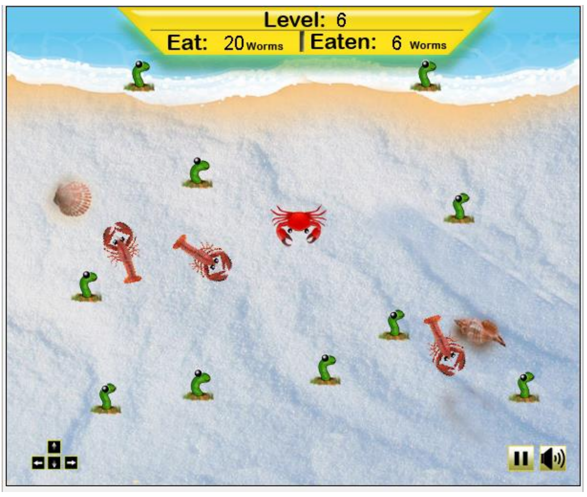
ii) Game Instruction and Game play screenshots



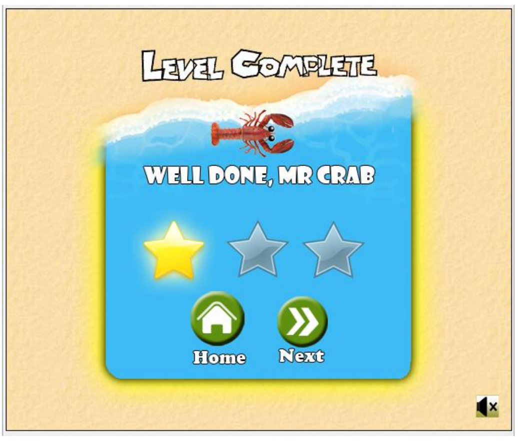
Press **Right** arrow key to turn clockwise
Press **Left** arrow key to turn anti-clockwise
Crab moves continuously

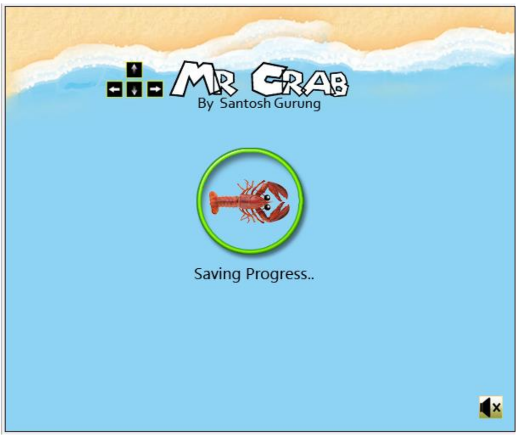
Complete a level in less time to get 3 stars.
0 to 3 stars will be given after completion of each level.
0 means user took alot of time to complete the level.

Game Progress wont be saved untill you see this icon.

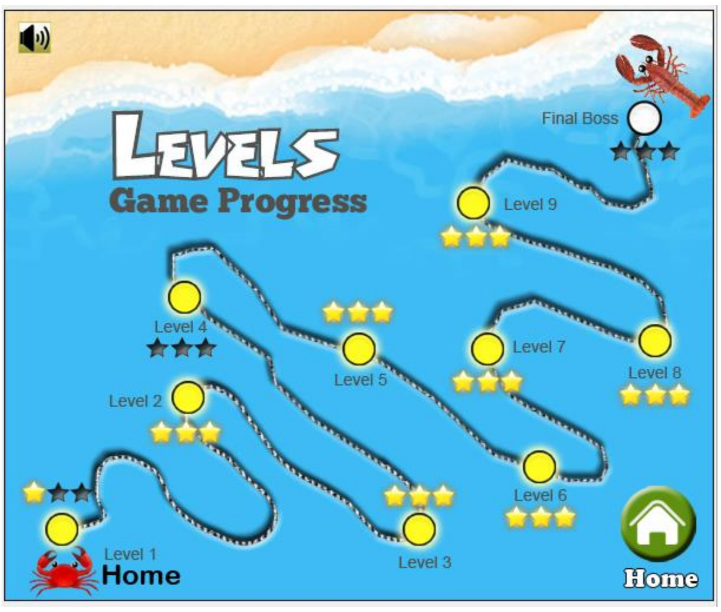


iii) Level Complete screen with scores (0-3 Stars) , Right image shows saving the game progress.

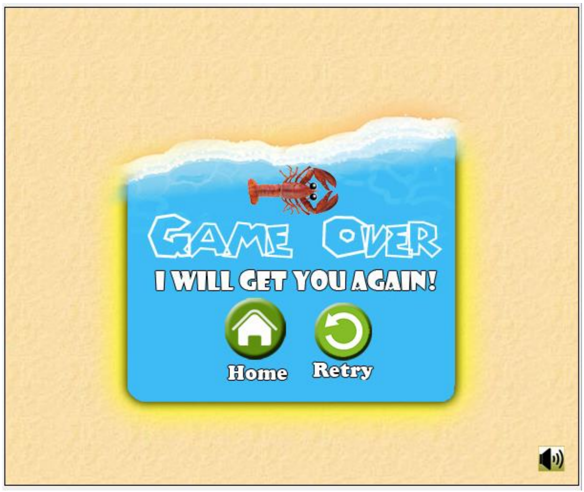


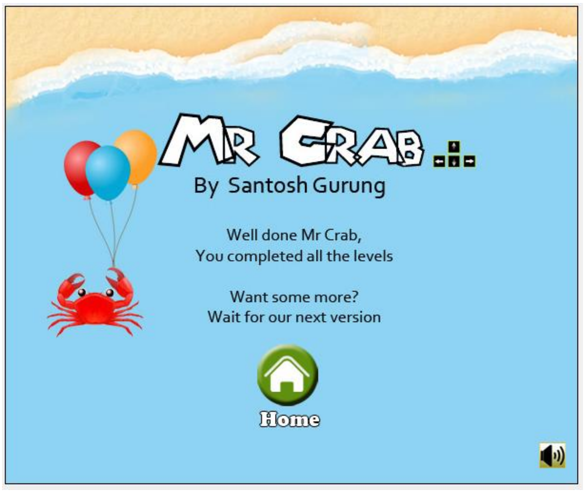


iv) Levels window shows level 1-10 along with the stars achieved. (Data are retrieved from file)



v) Game over screen is shown when the lobster catches crab and on right image, when user completes all levels 1-10 well done message is shown.





7. Testing

All levels are tested thoroughly, and a testing checklist is completed.

Task Name	Brief description/ Expected	Result
Button	Test Play, levels, home button	✓
Test Crabs movement	i) If gamers key is on then crab can be controlled with up, down, right &left key ii) But if off crab will move continuously and controlled with right and left key only.	✓
Test Lobster ultimate hunt attack	If crab is idle for specific time mentioned for each level the lobster will move towards the crab in shortest path. Idle means when right , left , up & down key is not pressed a counter is incremented to count the idle time. Comments: Because the setRotation(int) is in integer value and the angle generated through algorithm gives decimal values lobster is slightly inclines away from the actual angle.	✓ Acceptable
Remove worms	i. If crab catches or eats the worm remove the particular worm from the world. More specifically if the lobster and worm are in same near co-ordinates remove the worm.	✓
Test Volume button or check sound during the play time	i. When the game starts sound must start ii. If the volume button is 🔊 image then there must be sound if it's not then no sound must be generated from any of the actor or world. iii. Users choice to mute or sound must be remembered throughout the game. i.e. If sound is disabled in a world it must be disabled in other world (Greenfoot) or window.	✓
Next level if worms to eat and worms eaten is same	i. If the crab eats the number of worms to be eaten then score board is shown	✓
Game over	i. If lobster catches crab then game over window to shown.	✓
Save level progress	i. After the score board is shown game progress or the score of that particular level must be stored in file. ii. If the data file is stored as read only game progress can't be saved and a message must be shown	✓
Stars shown for each level in level board	i. Stars ★★★ shown in level board for particular levels. Ii. Level 1-10 must show its own stars	✓
Play a level which has been played before	i. If status of a level is true or has been played must show yellow button ● and white for not played ○ ii. If it is yellow or the level has been played load the level, if not show message locked.	✓
Animation	Worm, Crab & Lobster animation	✓
Pause button test (With Mouse click)	i. If user clicks pause button and the image is ⏸ then it lobster and crab must stop movement. ii. If user clicks pause button and the image is ▶ then allow movement of lobster and crab.	✓
Pause button test (With space key)	i. Functionality similar to mouse click however it is controlled with space key (Keyboard) Comments: Test shows pause with space wasn't working however it could be achieved, even YouTube player suffers from it.	✗ Partial

8. User/End user Instruction

The game is similar to the android/iOS games (2013) so it is very easy to play.

A. How to play the game?

   i. After the game loads up click play button.
   ii. Then you will be greeted with game instruction window. It will show you how to control the crab.
   iii. Click play button.
   iv. Use right, left, up & down key to move the crab.
   v. To complete the level eat the number of worms to eat shown in top bar.

B. How to move the crab?

   There are two keyboard style to play the game or to move the crab. To change it click on the keyboard button on left in game play.

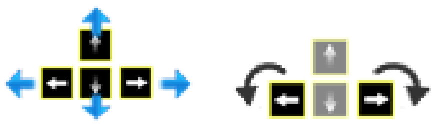


Fig: Showing two keyboard styles.

In above figure, straight arrow means move towards the direction. Circular arrow means rotate towards the direction.