Games Programming With Greenfoot

Project 1 - Wombats

Learn about -

- Opening a project
- Adding objects to the world
- Running programs
- Viewing code
- Calling methods
- Using documentation

Tasks -

1. **Opening a project**
   Download, unzip and run the Wombat project

2. **Adding objects to the world**
   Right click the **Wombat** and choose **new Wombat()**.
   Click on the game world to place it.
   Hold shift while you click to place more than one.

3. Place a leaf somewhere on the edge of the screen.

4. **Running programs**
   Use **Act** and **Play** to see how the program runs.

5. **Viewing code**
   Double click on the **Wombat** actor class to see the program code.
   The important bit is where it says **public void act()**.

6. **Calling methods**
   The line that says **turnLeft();** is a **method**. Further down there are (complicated) instructions on just how to do this - but we don't need to worry about that right now.

   Try changing this line to say **turnRandom();** (note the capitals and the semi-colon).

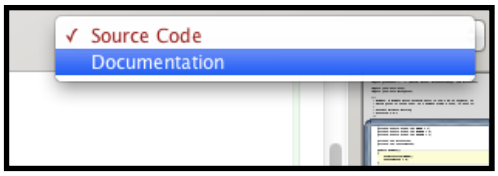
```
/**
 * Do whatever the wombat likes to to just now.
 */
public void act()
{
    if(foundLeaf()) {
        // Do something
    }
    else if(canMove()) {
        move();
    }
    else {
        turnLeft();
    }

    Wombat wombat = new Wombat();

}
```

7. Close the code window and click **Compile** in the main **Greenfoot** window. Create a new wombat and test the program.

8. **Documentation**
   At the minute the wombat stops if it touches a leaf.



   Go back to the Wombat code window and in the top right, click on **Source Code**, then change it to **Documentation**.

   There are 9 methods to choose from. Decide, just from the names, which method we want to run when the wombat touches a leaf.

   Go back to the **Source Code** view.

   Work out which part of the **public void act()** method is set up to deal with what happens when the wombat finds a leaf. Try writing the function call there.

   Test the program out and see if it works.

Have you have learned about?

- Opening a project
- Adding objects to the world
- Running programs
- Viewing code
- Calling methods
- Using documentation

Games Programming With Greenfoot

Project 2 - Frog

Learn about -

- If statements
- Keyboard controls
- Changing images

Tasks -

1. Download, unzip and run the Frog project

2. Run the program and try pressing the left and right arrow keys. What happens?

3. **If statements & Keyboard controls**
   Look at the **Frog** code and see if you can find 4 lines of code that make the frog move left.

   Copy and paste these 4 lines of code and try altering them to let the frog move right as well. Test the program.

4. **Changing images**
   At the top of the **Frog** code is a line that uses a picture file called **frog-sitting.png**.

   This line tells the program to create a **GreenfootImage** called **sitting** and to use the file **frog-sitting.png**.

   Copy and paste this code again, immediately underneath. Change the new line to that it creates a **GreenfootImage** called **jumping** that uses the file **frog-jumping.png**.

   In the **public void act()** method is a test to see if the frog is on the ground. If the frog is on the ground, the image is set to **sitting** and the **verticalSpeed** is set to 0.

   If the frog isn't on the ground, add one more line of code to change the image there as well.

   Test the finished program to see if it works.

Have you have learned about?

- If statements
- Keyboard controls
- Changing images

Games Programming With Greenfoot

Project 3 - Cats. And lasers.

Learn about -

- Mouse input
- Rotating objects
- Adding objects at runtime

Tasks -

1. Download, unzip and run the Cat project

2. **Mouse input**
   Go into the **Pointer** code and find the following line in **public void act()**:

   ```
   MouseInfo info = Greenfoot.getMouseInfo();
   ```

   This line creates an object called **info**, of type **MouseInfo** and runs the Greenfoot method **getMouseInfo()** to give it a value.

   As long as info exists - `if (info != null)` - then pointer needs to set its location to the same position as the mouse pointer, so add this line inside the if statement:

   ```
   setLocation(info.getX(),info.getY());
   ```

   Go back to the main window and create a new **Pointer** object. Run the program and see if it works.

3. **Rotating objects**
   Go into the Cat code and add 4 lines of code in public void act():

   i. Set up MouseInfo using the same line that was in the **pointer** code.

   ii. Only do the next bit `if (info != null)`
   (remember to put in curly braces - `{` and `}`)

   iii. Use the existing **angleTo** method to get the direction we want to point it
   `target = angleTo(info.getX(),info.getY());`

   iv. Use the **setRotation()** method to point towards the target.

   v. Use the **move()** method to move forwards.

   Test the program to see if it works.

4. **Adding objects at runtime**
   By editing the **Floor** code we can tell Greenfoot to add objects to the game world at the start automatically. Add the following code to the **private void prepare()** method.

   The first line will create a new object of type **Cat** (upper case C), called **cat** (lower case) by creating a new **Cat** (upper case).

   The second line will add the object to the world in the X and Y positions provided.

```
Cat cat = new Cat();
addObject(cat,100,100);
```

   Test that this works, and then add two more lines of code to add a pointer at (300,300).

Have you have learned about?

- Mouse input
- Rotating objects
- Adding objects at runtime

Games Programming With Greenfoot

Project 4 - Frog v2

Learn about -

- Creating objects at runtime
- Moving objects automatically
- Respawning objects
- Random numbers
- Stopping Greenfoot

Tasks -

1. Download, unzip and run the Frog v2 project

2. **Creating objects at runtime**
   Using what you learned with the cat project, edit the **FrogWorld** code to add 2 logs to the game, at positions (700,250) and (300,320).

   Check that the game works so far and that the frog can jump from one log to the other.

3. **Moving objects automatically**
   We want the logs to automatically scroll to the left (like many good platform games).

   There is already a variable called **speed** that we can use within the **Log** code. Inside **public void act()**, use the **setLocation()** method to make the log slide to the left.

   Hint: you can use getX() and getY() to get the log's current location.

   As always, check that this works. It should work for both logs, although they will get stuck at the edge of the screen.

4. **Respawning objects**
   Once an object reaches the edge of the screen we want it to disappear and respawn at the right hand side.

   To do this to each log, simply check if getX() is less than 25 (ie. its position is far to the left) and set its location to (700,300) to make it appear on the right, around halfway up.

5. **Random numbers**
   Having the logs always come back at exactly the same height is a bit boring. By changing the second co-ordinate to a random number we can make each log start at a random height.

   Try replacing the number **300** with `Greenfoot.getRandomNumber(450)` - which will create a random number between 1 and 350.

   To make it even better, try to work out how to have the logs go a bit lower, and never scrape along the ceiling.

6. **Stopping Greenfoot**
   At the minute the game never ends.

   Look in the **Frog** code to see what happens if the frog hits the very bottom of the world. Replace the contents of the if statement with the line `Greenfoot.stop();` and see if it works.

   Hint: It's always best to comment a line out using // rather than deleting it, just in case you want to get it back later.

Have you learned about?

- Creating objects at runtime
- Moving objects automatically
- Respawning objects
- Random numbers
- Stopping Greenfoot

Games Programming With Greenfoot

Project 5 - Crabs

Learn about -

- Rotation (recap)
- Random numbers (recap)
- Generating objects at runtime (recap)
- Collision detection
- Object removal

Tasks -

1. Download, unzip and run the Crab project

2. Drop a crab into the world, run the game and the crab should walk to the right and get stuck.

3. **Rotation (recap)**
   Edit the **Crab** code so that it can turn left or right.

   Hint 1: Look at previous programs to see how to check for a key press
   *Hint 2: Look at the **Actor** class to see how to set the rotation*
   *Hint 3: Look at the **Actor** class to see how to get the original rotation*

4. ***Random numbers (recap)***
   Edit the **Lobster** code so that it will always walk forward and turn **randomly**.

   Hint 1: Generate a random number using Greenfoot.getRandomNumber(20)
   Hint 2: Think about how to randomly turn left OR right.
   Hint 3: The simplest solution will have the lobster turning up to 10 degrees at a time.

5. **Generating objects at runtime** (recap)
   Edit the **CrabWorld** code to add 1 crab and 4 lobsters to the world.

   Hint 1: Create a crab by saying Crab crab = new Crab();
   Hint 2: Add the object to the world using addObject(, ,);
   Hint 3: Give each lobster a slightly different name.

6. **Collision detection**
   It's easiest to think of one object destroying another. The crab is going to eat the lobsters, so this code goes in the **Crab** code.

   We want the crab to detect a lobster that it is touching (intersecting with):

   ```
   Lobster deadLobster =
   (Lobster)getOneIntersectingObject(Lobster.class);
   ```

   While the crab is not touching the lobster, there is no intersecting object - and so there is no deadLobster. In Java terms, deadLobster == null.

   When we **do** touch a lobster then deadLobster != null.

   If we do touch a lobster then use Greenfoot.stop(); to end the game. This will let us know if it has worked.

7. **Object removal**
   To remove an object from the world, use:

   ```
   getWorld().removeObject(deadLobster);
   ```

8. **Extension**
   If you've coped well so far, then in the **Crab** code use the **lobstersEaten** variable to count how many lobsters have been. Well. Eaten.

   If all of the lobsters have been eaten then end the game.

Have you learned about?

- Rotation (recap)
- Random numbers (recap)
- Generating objects at runtime (recap)
- Collision detection
- Object removal