Лабораторная работа 2
Изучение рабочей среды “GreenFoot”, Сценарий Вомбаты (WOMBATS)

Цель

Изучить основы объектно-ориентированного программирования на примере рабочей среды “GreenFoot”.

Задание на лабораторную работу

1. Изучить рабочую среду GreenFoot. Ознакомится с объектами базового мира ***«Wobmats».***
2. Добавить в Мир «***Wobmats***» новый класс ***«Конечная точка».*** Изменить поведение класса ***Wombat*** так, что бы он двигался по кратчайшему пути до конечной точки, минуя препятствия (***Rocks***).
3. Добавить в Мир ***«Wobmats»*** новый класс ***«Хищник»***. ***Хищник*** должен охотится за Вомбатами.

Рабочая среда GreenFoot

- Открытие проекта
- Создание объектов в мире
- Активация объектов
- Запуск сценария
- Вызов методов напрямую
- Пересоздание мира
- Вызов методов мира
- Смена поведение объекта
- Компиляция проекта
- Смена картинок
- Информация о классах Greenfoot
- Просмотр объекта
- Создание нового класса
- Создание собственного сценария

Открытие проекта

Запустите ***Greenfoot***, если он еще не запущен. Для этого необходимо выполнить командный файл ‘**greenfoot’** в папке установки. Если после запуска ***Greenfoot*** сценарий ***Wombats*** не открылся автоматически, нажмите пункт ‘***Open’*** меню ‘***Scenario’*** и выберите сценарий ‘**wombats’** из папки ***scenarios Greenfoot***.

На экране должно появиться что-то похожее на рисунке 1., за исключением пустого желтого поля.

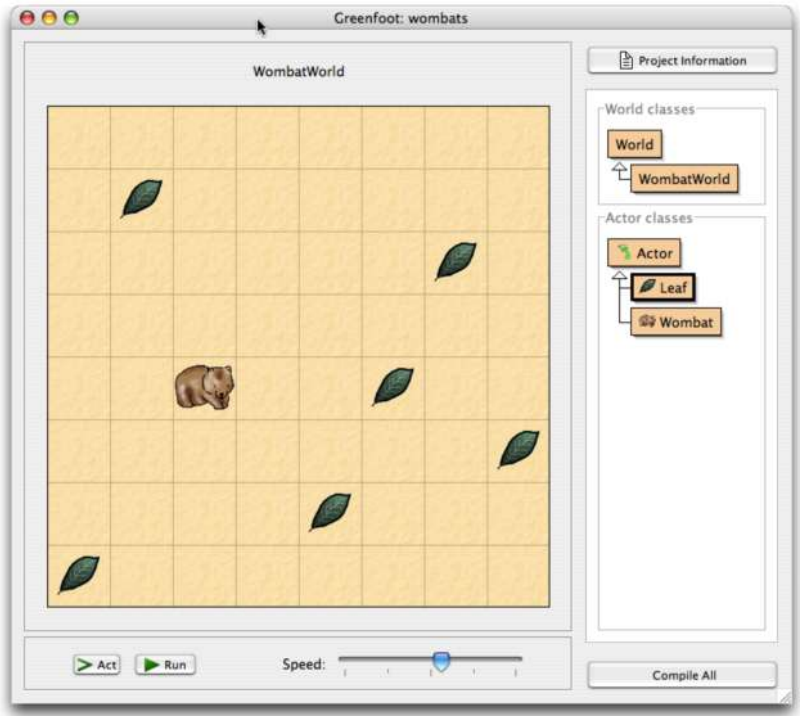

Рис. 2.1. Сценарий Wombat в Greenfoot

Большое желтое поле с сеткой называется “ ***World*** ” - Мир. Поскольку в сценарии участвуют Вомбаты, это – мир вомбатов.

Справа от мира находится дерево классов. В нем показаны все классы Java, задействованные в сценарии. Классы “***World***” (Мир) и “***Actor***” (Актер) присутствуют всегда, так как являются системными классами ***Greenfoot***. Остальные зависят от сценария.

Под миром расположены Элементы Управления (Кнопки ‘***Act’*** (выполнить 1 действие) ‘**Run’** (запустить выполнение действий) и движок скорости).

Информация о сценарии

В правом верхнем углу находится кнопка “***Scenario Information***” - Описание сценария. Обычно там содержится цель сценария, способ запуска, возможные действия и пути модификации.

Создание объектов в мире

Щелкните правой кнопкой мыши по классу ***Wombat*** в дереве классов слева. В появившемся меню нажмите левой кнопкой мыши ‘***New Wombat()***’ - Создать Вомбата.

Теперь кликните левой кнопкой мыши в мире. Вы создали новое существо, вомбата, - новый объект класса Java, - и выпустили его в мир.

Вомбаты едят листья, поэтому давайте создадим немного. Для этого кликните правой кнопкой мыши по классу ***Leaf***, выберите ‘***New Leaf()***’ и кликните левой кнопкой в мир. Если несколько раз кликнуть левой кнопкой с зажатой клавишей ***Shift*** в мир, каждый раз будет создаваться объект выбранного класса.

Активация объектов

Кликните по кнопке ‘***Act’*** внизу окна. Каждый объект выполнит свое действие - то, что он хочет. Что хочет объект зависит от самого объекта.

В нашем примере листья хотят лежать на месте, вомбаты перемещаются вперед. Попробуйте добавить пару вомбатов в мир и нажать кнопку ***Act***. Оба переместятся.

Вомбаты хотят есть листья. Если на пути окажется лист, вомбат съест его.

Запуск сценария

Нажмите левой клавишей мыши на кнопку '***Run'*** - запуск. Такой же эффект дает постоянные быстрые нажатия на ***Act***. Заметьте, что кнопка ***Run*** стала '***Pause'*** - стоп. Теперь нажатие на нее остановит весь мир.

Движок слева от кнопок ***Act*** и ***Run*** задает скорость. Попробуйте нажать ***Run*** и подвигать его.

Вызов методов напрямую

Вместо запуска сценария можно вызывать методы отдельных объектов. Метод - это одно из возможных действий объекта.

Создайте вомбата в мире и остановите сценарий, если Вы этого еще не сделали. Нажмите правой кнопкой мыши на созданного вомбата и Вы увидите всплывающее меню объекта (рис 2.).

Рис. 2.2. Всплывающее меню объекта

Вы можете кликнуть на любой из методов, попросив вомбата что-то сделать. Попробуйте, например***, turnLeft().*** Вомбат повернет налево. Попробуйте ***move*****().** Некоторые методы возвращают результат. Например, ***getLeavesEaten*****()** покажет, сколько листьев вомбат съел.

Обратите внимание на метод **'*act*()**'. Он вызывается всякий раз, когда Вы нажимаете кнопку ***Act***. Если Вы хотите, чтобы действовал только один объект, а не все, что есть в мире, Вы можете вызвать метод ***act*** этого объекта напрямую.

Пересоздание мира

Если у Вас накопилось множество ненужных объектов в мире, и Вы хотите начать заново, Вы можете уничтожить мир и воссоздать его девственно чистым.

Для этого предназначена кнопка ***'Reset'*** внизу экрана. После ее нажатия Вы получите пустой новый мир, а старый будет стерт вместе со всеми объектами.

Вызов методов мира

Мы видели, что можно вызвать метод любого объекта в мире с помощью всплывающего меню. Сам мир - тоже объект и тоже содержит методы. Над полем мира (будьте внимательны: не в дереве классов справа) находится заголовок, где написано название – "***WombatWorld***" в нашем случае. Нажмите на него правой кнопкой мыши, и Вы увидите меню мира.

Один из методов - '***populate*()**'. Попробуйте выбрать его - в мире появятся несколько листьев и вомбатов. Теперь можно запускать сценарий.

Другой метод - '***randomLeaves(int howMany)***'. Он создает в мире несколько листьев в случайных местах. Обратите внимание, что в скобках содержатся слова: ***int howMany*** . Это - параметр метода. Он означает, что Вы должны указать дополнительную информацию при вызове метода. Слово '***int'*** говорит, что указать надо целое число, а '***howMany'*** предполагает, что число определит, сколько листьев Вы хотите.

Вызовите этот метод. Появится новое диалоговое окно, в котором можно ввести значение параметра. Введите число например, 12, и нажмите ***Ok***. Если сосчитать появившиеся листья, может получиться меньше 12 - это оттого, что в одной клетке лежат несколько листьев.

Смена поведение объекта

Вы можете программировать собственные объекты - вомбатов или кого-то еще - с помощью программы на Java, включая ее в класс объекта.

Дважды кликните левой кнопкой мыши на классе ***Wombat*** в диаграмме классов справа. Откроется окно текстового редактора с исходным Java кодом класса ***Wombat***. Во-первых, давайте сделаем так, чтобы вомбат поворачивал не налево, а в случайную сторону, когда встречает препятствие.

Добавим '***turnRandom*()**' метод внутрь класса ***Wombat***. Следующие строчки нужно поместить перед последней закрывающей фигурной скобкой } - она обозначает конец класса вомбатов:

```
/**
 * Turn in a random direction.
 */
public void turnRandom()
{
   // get a random number between 0 and 3...
   int turns = Greenfoot.getRandomNumber(4);
   // ...an turn left that many times.
   for (int i=0; i<turns; i++) {
      turnLeft();
   }
```

```
}
```

Теперь найдите метод ‘act()’. Он должен выглядеть примерно так:

```
public void act()
 {
    if(foundLeaf()) {
       eatLeaf();
    }
    else if(canMove()) {
       move();
    }
    else {
       turnLeft();
    }
 }
```

Замените вызов ‘***turnLeft*()**’ в конце перед двумя скобками } } на вызов ***‘turnRandom*()**’ без кавычек и не трогая ; после (). Получится примерно так:

```
    else {
       turnRandom();
    }
```

Нажмите левой клавишей мыши на кнопку ‘**Compile’** в верхней части окна редактора. Если будет показан список ошибок, исправьте их и нажмите снова. Закройте редактор.

Компиляция проекта

Перед запуском сценария необходимо скомпилировать проект. Это можно сделать из редактора или из главного окна ***Greenfoot’s***, нажав на кнопку в правом нижнем углу ***Compile All.***

Если компиляция успешна, Вы можете создавать объекты, поскольку после успешной компиляции мир будет разрушен и создан вновь чистым.

Смена картинок

Есть два способа сменить картинку объекта: сменить ее у всего класса, или запрограммировать смену картинки у одного объекта. Во втором случае объект может сам менять свою картинку когда угодно.

Для смены картинки класса выберите ‘***Set Image***’ из выпадающего меню класса. Попробуйте это сделать с классом ***Leaf***. Поставьте вместо листа яблоко или банан - и вомбат будет их есть.

Greenfoot содержит встроенную библиотеку картинок, которые можно использовать. Можно также создать свое изображение и поместить в папку ‘***images’*** внутри папки сценария (‘***wombats’***).

Другим способом смены картинки является программирование объекта. Он может вызвать метод ‘setImage’, полученный по наследству от своего класса-родителя **Actor**, как одно из своих действий. Есть два варианта ‘***setImage’***: у первого параметр - объект класса ***GreenfootImage***, у второго - имя файла картинки. Второй при этом читает картинку, создает объект ***GreenfootImage*** и вызывает первый.

Мы будем использовать простой. Наша цель - чтобы вомбаты не ходили вверх тормашками влево. В проект ‘wombats’ уже входит картинка ‘wombat-left.gif’ в папке ‘images’. На ней вомбат нарисован идущим влево.

Для смены картинки на левостороннюю в программе надо написать:

setImage("wombat-left.gif");

Напишем это в том месте, где вомбат меняет направление. Это - метод **‘*setDirection(int Direction)***’. Найдите его в исходном коде класса.

Добавим несколько строк для установки правильной картинки и угла поворота при установке направления. Когда направление ***WEST*** (запад, налево) и ***NORTH*** (север, вверх) будем использовать ‘wombat-left’, и исходное ‘***wombat’***, если ***EAST*** (восток, направо) и ***SOUTH*** (юг, вниз). Заметьте, что картинка ‘***wombat-left*’** направлена налево (на запад) изначально и ее не нужно вращать, когда вомбат направлен туда.

Вот так должен выглядеть обновленный метод ‘***setDirection’***:

```
    /**
  * Sets the direction we're facing.
     */
    public void setDirection(int direction)
    {
        this.direction = direction;
        switch(direction) {
            case SOUTH :
                setImage("wombat.gif");
                setRotation(90);
                break;
            case EAST :
                setImage("wombat.gif");
                setRotation(0);
                break;
            case NORTH :
                setImage("wombat-left.gif");
                setRotation(90);
                break;
            case WEST :
                setImage("wombat-left.gif");
                setRotation(0);
```

```
                break;
            default :
                break;
        }
    }
```

При более серьезном подходе необходимо создать два объекта класса ***GreenfootImage*** с соответствующими картинками в конструкторе класса ***Wombat***, и затем использовать их для вызова первого варианта метода ***setImage*()**.

В дальнейшем можно добавить отдельные картинки для верхнего и нижнего направлений.

Информация о классах Greenfoot

Для изменения поведения объектов часто требуется использование базовых классов ***Greenfoot***. Таких классов четыре: ***World, Actor, GreenfootImage*** и ***Greenfoot***.

Первые два видны в дереве классов главного окна и являются родительскими классами для классов мира и объектов в сценарии. Класс ***GreenfootImage*** используется для работы с картинками, а ***Greenfoot*** предоставляет доступ к глобальным методам среды ***Greenfoot***.

Простейший способ узнать больше об этих классах - это прочитать документацию по адресу: ***http://www.greenfoot.org/programming/***

Там расположена онлайн версия описания класса ***Greenfoot ("Greenfoot API").*** Ее можно скачать в одном документе, пригодном для печати.

Просмотр объекта

Вызовите метод '***Inspect'*** из всплывающего меню любого из вомбатов в мира. Он покажет внутреннее состояние объекта - значение его свойств. Это может потребоваться при тестировании классов в процессе создания.

Заметьте, что некоторые свойства определены в классе ***Wombat*** (например, ***'leavesEaten'*** - съеденные листья), а другие - нет. Дополнительные свойства (например, **x, y** и ***rotation*** - поворот) унаследованы от родительского класса ***Actor*** и присутствуют во всех объектах в мире.

Если в качестве значения нарисована стрелка, значит свойство содержит ссылку на другой объект, который тоже можно просмотреть путем нажатия на нем правой кнопкой мыши и выбора метода '***Inspect'***.

Создание нового класса

Настало время устроить нашим вомбатам веселую жизнь: время рассыпать камни! Наши камни будут настолько огромными, что ни один вомбат не сможет через них перебраться.

Для этого создадим новый класс объектов. Во всплывающем меню класса ***Actor*** в дереве классов выберите ***'New subclass'***. Мы получим новый класс - потомок класса ***Actor***. Введите слово '***Rock'*** - скала, камень - в качестве имени нового класса.

Откроется окно выбора картинки для него. Вы можете найти или нарисовать или сфотографировать любое изображение и поместить его в папку ‘***images’*** проекта перед созданием нового класса. Тогда оно появится в этом диалоге выбора. В данном случае там уже лежит картинка ***‘rock.gif’*** - можете ей воспользоваться.

Выберите картинку, нажмите кнопку ***Ok***, и новый класс ***Rock*** (камень) готов. Откройте редактор исходного кода класса - Вы увидите, что пустой скелет класса создан автоматически. Мы не будем ничего сейчас писать, поскольку поведение у камней, обычно, отсутствует. Закройте редактор, скомпилируйте проект, попробуйте создать камень.

Теперь наполните мир вомбатами и запустите сценарий. Заметьте, что вомбаты проходят сквозь камень, как по ровному месту.

Создание собственного сценария

Для создания новых сценариев или значительного изменения старых Вы можете прочитать ***Greenfoot Programmer’s Manual,*** или посмотреть, как сделаны другие сценарии. Вам также понадобится описание классов ***Greenfoot (Greenfoot API),*** о котором говорили раньше.

Последнее изменение в этом сценарии - запретить вомбатам проходить по камням. Откройте редактор класса ***Wombat*** и найдите метод ‘***canMove’*** (можно ли двигаться). Сейчас он лишь проверяет, не дошли ли мы до границы мира и возвращает ***false*** (ложь), если дошли. Надо добавить проверку, нет ли впереди камня, и тоже возвращать ***false,*** если есть.

Найдите последнюю строчку перед закрывающей скобкой } метода. Она выглядит так:

return true;

на следующий текст:

```
 List rocks = myWorld.getObjectsAt(x, y, Rock.class);
if(rocks.isEmpty()) {
   return true;
}
else {
   return false;
 }
```

В этом куске кода мы находим все объекты типа ***Rock.class*** - то есть, камни, - лежащие в ячейке с координатами ***x, y,*** и помещаем их в объект ***rocks*** класса ***List*** - список. Координаты мы вычислили в начале метода. Если список ***rocks*** пуст (метод ***isEmpty()*** вернет истину), мы пишем ***return true*** (вернуть истину тому, кто вызвал метод ***canMove()***, что означает "движение разрешено"). В противном случае - там, где стоит ***else,*** - в ячейке лежат камни и идти туда нельзя. Тогда возвращаем ***false*** - ложь.

После всех изменений сценарий должен будет выглядеть и вести себя как сценарий ‘wombats2’.