程序说明
version 1.0.0

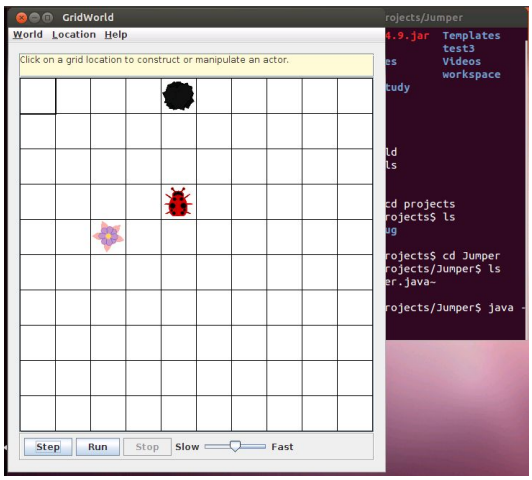
修订历史

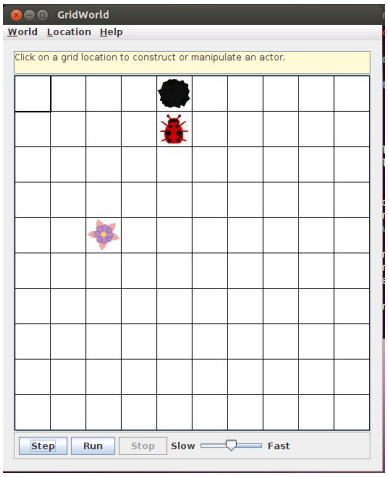
日期	版本	作者	描述
2015/8/21	1.0.0	13331224　耍燕萍	
		13331018　陈炜建	
		13331011　陈广灿	
		13331122　李绍焜	

目录

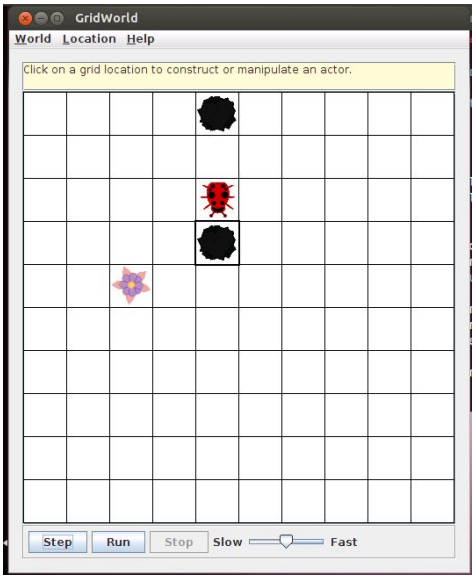
1 程序功能

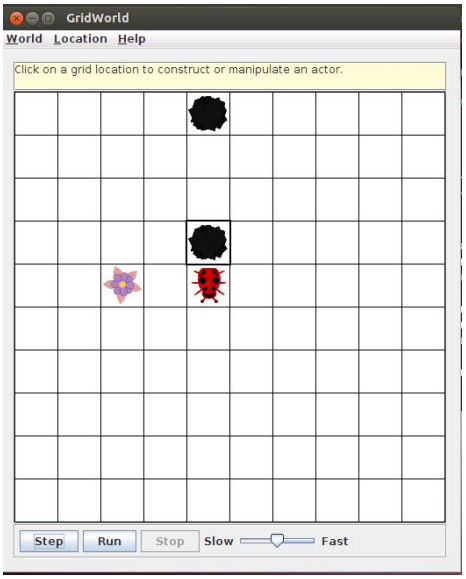
(一)实现一个 Jumper 的类，Jumper 移动的时候并不会留下花(flowers)。如果 Jumper 前面没有障碍物或者遇到边界时，Jumper 会 Jump()，移动两格。



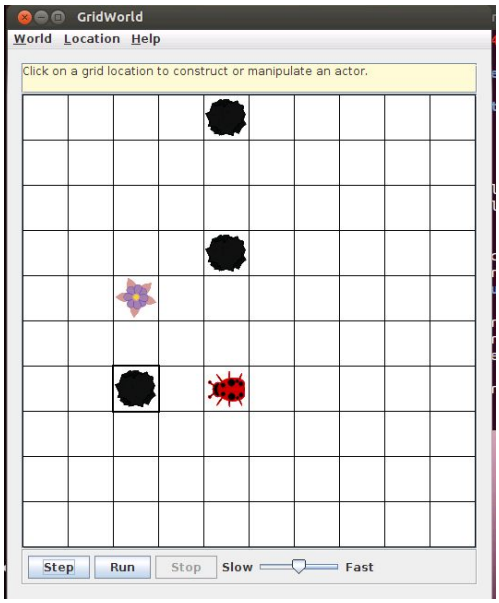


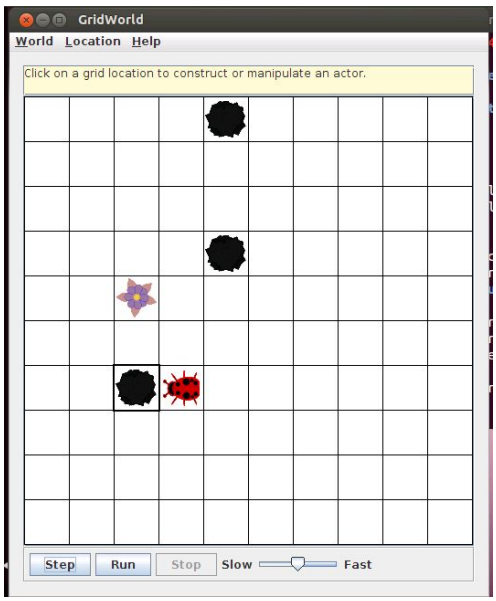
（二）如果 Jumper 移动方向的正前方一格有障碍物（如：actor,bug,rock）,而正前方两格没有障碍物同时也不是边界的时候，可以跳过（Jump）障碍物。



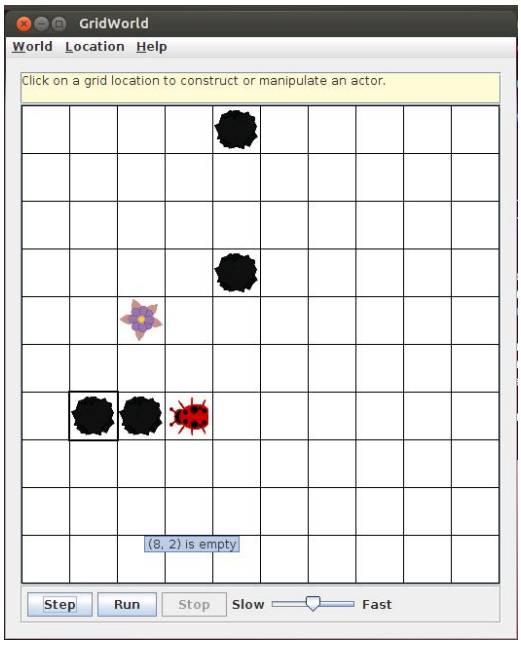


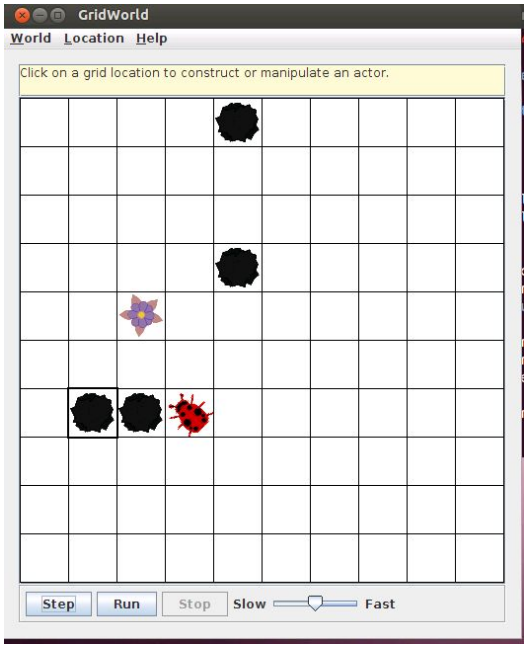
（三）如果 Jumper 移动方向的正前方第二个格有障碍物，而正前方一格没有障碍物的时候，可以向前移动（move()）一格。





(四)当 Jumper 遇到正前方两格都有障碍物的时候，Jumper 会向右转 45 度（turn()）,直到找到能继续前进的方向。





简单的来说就是能跳的时候就跳，不能跳就走，不能走就转，直到转到能跳或者能走的方向。

2 实现过程

Jumper 类是 Actor 类的子类。Bug 类也是 Actor 类的子类。所以 Jumper 类的大部分方法跟 Bug 类的方法基本相同。不同点在于，Jumper 类增加了 Jump()和 canJump()方法。Jump()

方法也跟 move()方法类似。Move()方法是判断如果下一格能走，就往下一个走。而 Jump()方法是如果下下格能走（没有障碍物），就可以跳过去，无视下格的障碍物。CanJump()方法也跟 canMove()方法类似。CanMove()是判断下格是否为障碍物或者边界，是就返回 false,如果是花或者空格子，就返回 true。canJump()则是先判断下个是否存在，不存在表面已经是边界，下下格肯定不存在(返回 false)，如果下格存在，则再判断下下格是否存在。不存在就返回 false。如果存在，则判断是否为障碍物（又回到 canMove()一样的方法），是障碍物就返回 false,是花或者空格子就返回 true。

```
public void jump()
    {
        Grid<Actor> gr = getGrid();
        if (gr == null)
    {
            return;
    }
        Location loc = getLocation();
        Location next = loc.getAdjacentLocation(getDirection());
        if (gr.isValid(next)) {
       Location next2 = next.getAdjacentLocation(getDirection());
            if (gr.isValid(next2))
        {
                moveTo(next2);
        }
            else
        {
                removeSelfFromGrid();
        }
    }
     }

     /**
      * Tests whether this bug can jmp forward into a location that is empty or
      * contains a flower.
      * @return true if this bug can jump.
      */
     public boolean canJump()
     {
        Grid<Actor> gr = getGrid();
        if (gr == null)
    {
            return false;
    }
```

```
        Location loc = getLocation();
        Location next = loc.getAdjacentLocation(getDirection());

if (!gr.isValid(next))
{
        return false;
}
Location next2 = next.getAdjacentLocation(getDirection());
        if (!gr.isValid(next2))
{
        return false;
}
        Actor neighbor = gr.get(next2);
        return (neighbor == null) || (neighbor instanceof Flower);
        // ok to move into empty location or onto flower
        // not ok to move onto any other actor
 }
```

3 总结体会

在认真看完网页上给的阶段二的 part3 的文档后，理解 Location，Grid 和 Actor 这几个类的时候便容易了很多。当要求我们写一个 Jumper 类的时候，我们自然而然的就想到了继承 Actor 类。其实 Jumper 类跟 Bug 类很相似， 我们只需要在 Bug 类的基础上修改，再加上我们需要的函数，Jumper 就基本已经实现了。有些东西看起来很难，其实只要认真的分析，一步一步的去做，其实都是很容易的，就像 Jumper。