

A decorative graphic in the top-left corner consisting of two overlapping parallelograms. The front one is blue and the back one is a light mint green. They are set against a dark navy blue background with faint, larger-scale geometric patterns.

Eat-Bugs

By Nick Bowley and Piers Watson

Introduction

Eat-Bugs is a game about a frog (the player) eating bugs while avoiding a lizard. It's not very complicated, but it doesn't need to be. The player wins when they get 10 points from eating Flies and Fireflies, but if the player is caught by the Lizard, Game Over.






Design Considerations: Classes

- Adding actors to the game at the start and over time.
- Sub-Classes = Player, Fly, Lizard classes, with Firefly as a subclass of the Fly.
- Reduction of code use using ACT method in MyWorld.

The Player




- The Player has code for Jumping forward, turning left and right, and collision detection for eating bugs.
- Jump Delay and animation using two pictures.
- The collision detection is done using greenfoot's 'isTouching' method, the MyWorld class is imported and told that the 'score' variable has increased. Any detected bugs are also removed from the world.



Code for Jumping seen below, and collision detection seen on the right.

```
//this if condition allows the jump delay to tick down.  
//It also helpfully shows the 'jumping' image when the player cannot jump yet.  
if(jumpDelay > 0)  
{  
    --jumpDelay;  
  
    if(jumpDelay >= 1)  
    {  
        this.setImage("Frog2.png");  
    } else {  
        this.setImage("Frog1.png");  
    }  
}
```

```
public void checkCollision()  
{  
    if(isTouching(Firefly.class))  
    {  
        removeTouching(Firefly.class);  
  
        MyWorld mw = (MyWorld)getWorld();  
        fireFliesEaten +=2;  
        mw.increaseScore(2);  
    }  
  
    if(isTouching(Fly.class))  
    {  
        removeTouching(Fly.class);  
  
        flysEaten +=1;  
  
        MyWorld mw = (MyWorld)getWorld();  
        mw.increaseScore(+1);  
    }  
}
```




The Bugs

(the winged kind, not code bugs)



- The Fly and Firefly classes both move forward constantly, while occasionally turning to a random direction, to simulate the erratic way real insects fly.
- They share almost identical code for movement, but Firefly has a slightly higher move speed to make them more difficult for the player to catch.
- They have a variable 'anim' dedicated to switching their images back and forth every 4 game ticks, to simulate the flapping of their wings.



Code for Movement on the right, code for animation below.

```
if (Greenfoot.getRandomNumber(100) < 10)
{
    turn(Greenfoot.getRandomNumber(90) - 45);
}

if (getX() <= 5 || getX() >= getWorld().getWidth() - 5)
{
    turn(180);
}

if (getY() <= 5 || getY() >= getWorld().getHeight() - 5)
{
    turn(180);
}
```

```
public void act()
{
    move(2);
    anim -= 1;
    //Derek, if you're reading this, the numbers seen in the next two 'if' statements are just how many game ticks it takes to switch images.
    //I'm not defining more than one integer for that.
    if(anim == 0)
    {
        this.setImage("Fly2.png");
        anim = 4;
    }
    if (anim <= 2)
    {
        this.setImage("Fly1.png");
    }
}
```

The Lizard



- Move toward the player, and cause a Game Over if it touches the player.
- The Game Over part is done using Greenfoot's inbuilt image drawing code, which changes the lizard's image to the text "GAME OVER". It then removes the Player object, and stops the program.
- The Moving part is done using the 'goTo' method, seen below. There's also code in the Act method for a walking animation, very similar to the Fly's but with a much longer delay.



The Lizard's Devious Methods

```
public void goTo()
{
    Player player = (Player)getWorld().getObjects(Player.class).get(0);
    int PlayerX = player.getX();
    int PlayerY = player.getY();
    turnTowards(PlayerX, PlayerY);
    move(1);
}
```

```
public void checkPlayerCollision()
```

```
{
    if(isTouching(Player.class))
    {
        removeTouching(Player.class);
        updateImage();
        Greenfoot.stop();
    }
}
```

```
public void updateImage()
```

```
{
    setRotation(0);
    setImage(new GreenfootImage("GAME\nOVER", 96, colors[colorNum], new Color(0, 0, 0, 0)));
    new Color(0, 0, 0, 0);
}
```



MyWorld

(probably the most important class)



- MyWorld has code for everything that couldn't be done by any particular actor.
- It creates the initial 'playing field' in the prepare method, with the Player and Lizard in set locations, and five Flies at random ones.
- Its' act method calls upon another method amusingly called randomThingMethod. This checks how many actors are currently in the world, and if there's less than 5 Flies/Fireflies, it will have a chance to create one of them. The chance is based on another of Greenfoot's inbuilt methods, 'getRandomInt', followed by a few conditional statements.
- There is also the method increaseScore. The method has adds its integer parameter to the existing 'score' variable, then if score ≤ 10 , it displays text congratulating the player and stops the game.

randomThingMethod and increaseScore.

```
public void RandomThingMethod()
{
    //love this method, as it can completely replace the 'spawner' actor.
    //just made a few little tweaks to the numbers to let it spawn flies more often than fireflies, and to stop it from spawning too many.

    int objectCount = numberOfObjects() - 2; //the two being subtracted are the Player and the Lizard

    int z = Greenfoot.getRandomNumber(100);
    int x = Greenfoot.getRandomNumber(getWidth() - 20) + 10;
    int y = Greenfoot.getRandomNumber(getHeight() - 20) + 10;

    //flies have (roughly) a 10% chance of spawning, Fireflies are at 5%.
    //it won't spawn both at the same time, and it won't spawn either if there's 5 bugs already on screen.

    if(objectCount < maxObjects)
    {
        if(z >= 90)
        {
            addObject(new Fly(), x, y);
        }

        if(z <= 5)
        {
            addObject(new Firefly(), x, y);
        }
    }
}
```

```
public void increaseScore (int amount)
{
    score+=amount;
    showText("Score is: " + score, 70,20);

    if (score>=winCondition)
    {
        showText("CONGRATULATIONS, YOU'VE WON: " + score, 300, 300);
        Greenfoot.stop();
    }
}
```



Development: Yes, there were issues.

- Less code used by using the MyWorld Class – once we discovered we could do this.
- Player Score caused issues, resolved by Bug Fixing.
- Animations needed turning as they were facing the wrong way.



Testing

- Testing happened.
- This resolved some issues.
- Gave us new ideas once the game was tested in the earlier stages.
- Could have tested more, main learning takeaway from the project - more testing = more results.
- TESTING IN GITHUB



Evaluation

- Game Development went well, some issues which were resolved eventually.
- Collaborative working was useful even if the team was late in being put together.
- Really happy with the gameplay, would be nice if there were some more animations.
- Lessons learned – More screenshots of testing including the console logs. Keeping it simple really works. Add extras after the basics are done.
- FULL EVALUATION IN GITHUB.