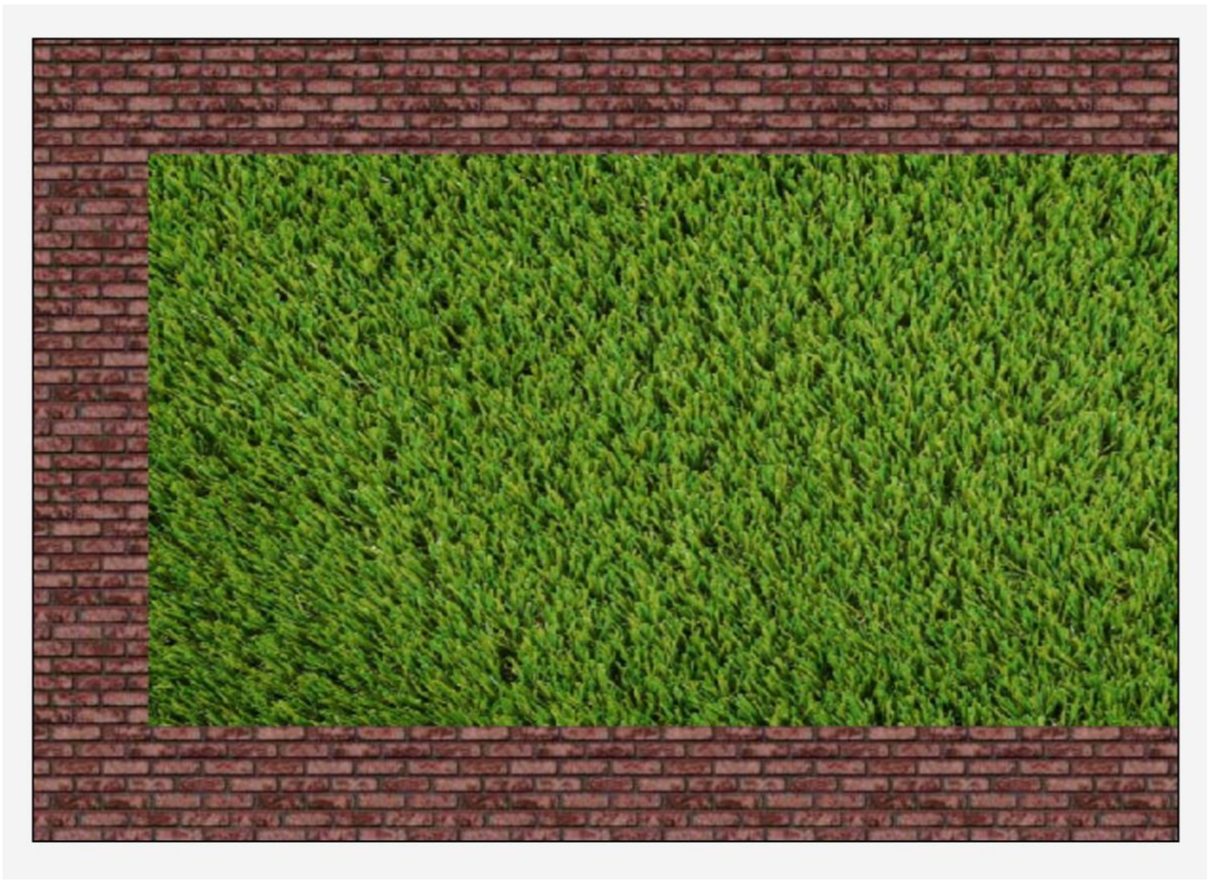
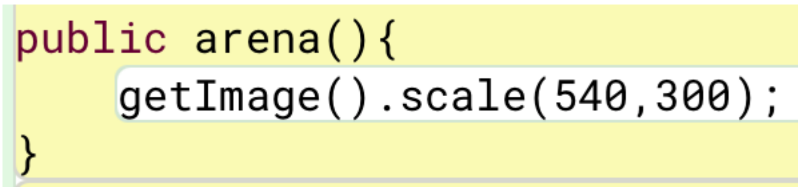
Development stage

Stage 1:

Following my dev plan, I began with my map, which I've decided (for now at least) should be a grassy land, inspired from "Plants vs. Zombies", which is where the enemies and towers will go.



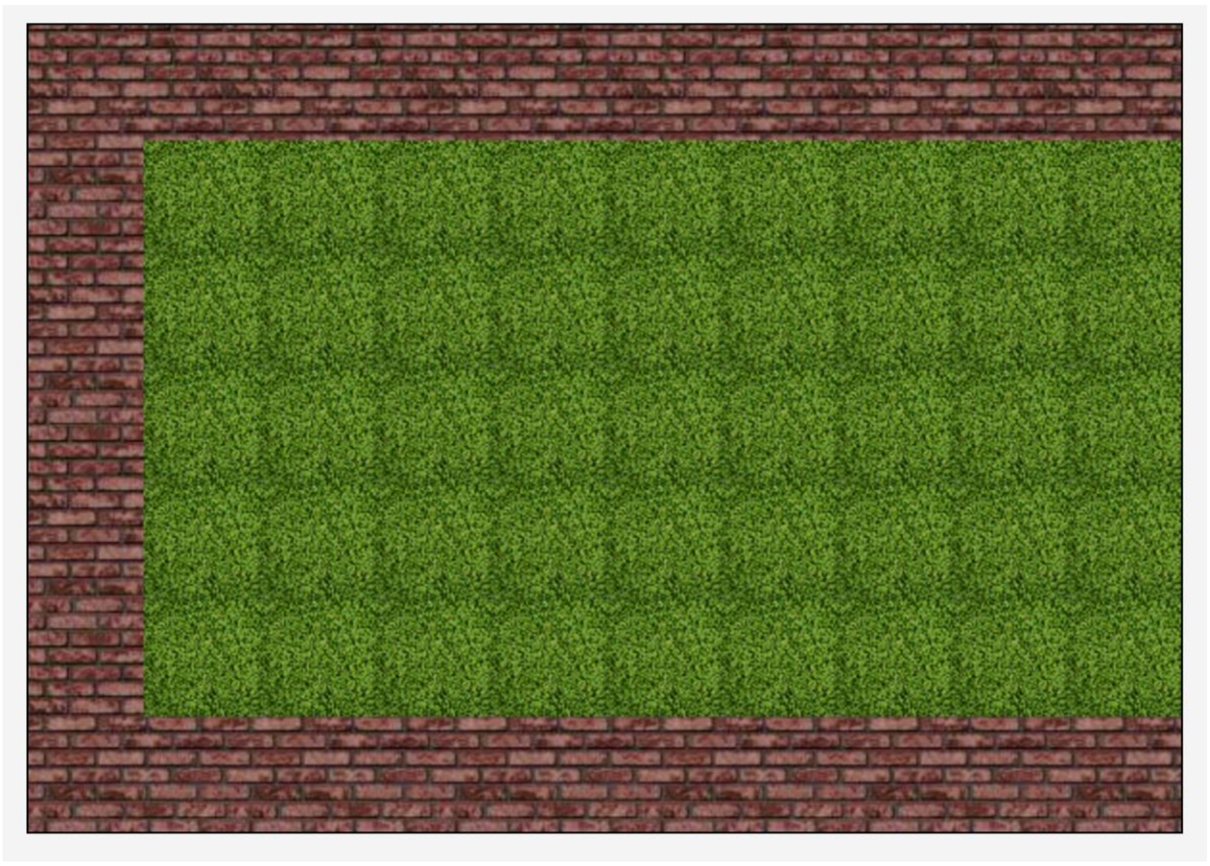
And to accommodate for the other UI elements I left space around the top, bottom, and left side of the grass, and set the background to a contrasting brick so that it would be obvious what's the gameplay and what's the UI. I did this by resizing the sprite and saving the world.



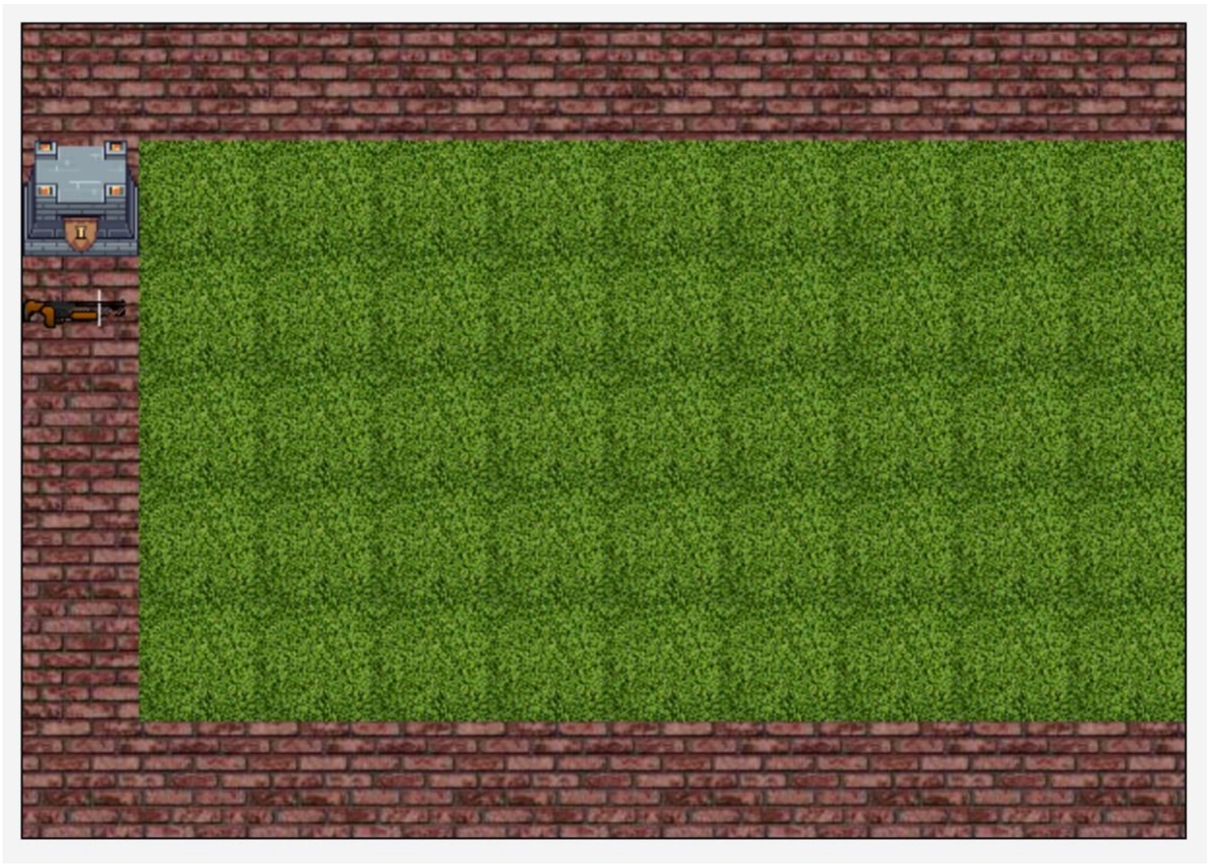
```
public arena(){
    getImage().scale(540,300);
}
```

I chose to make the world a very specific size and scale as this meant that instead of having to code in the different towers and enemies "snapping" to a grid, they did it automatically due to the size of the world. I did this because at this stage in the program I'm heavily leaning more towards function and getting as many features in the game and with 'good enough' functionality rather than making sure everything is perfect. I'd much rather have a somewhat playable game that I can tweak later on than one or two amazing features but the game is fundamentally unplayable.

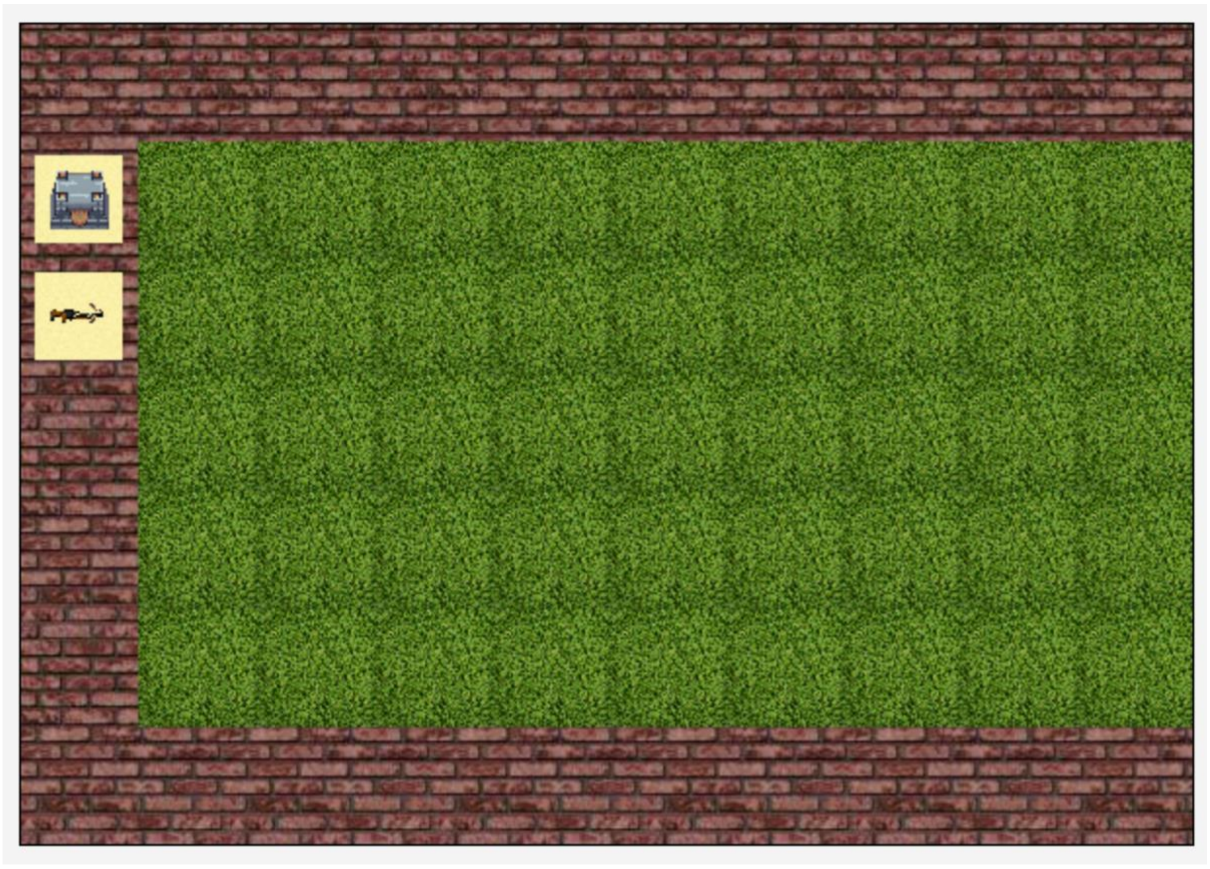
Having said this, I thought the current look of the grass was really ugly and just looked a bit lazy, because it kind of was, just having a jpg of grass in the middle of my game. But I also felt like I could find a better way to represent the grid, so I can use minimal text to inform the player of what's going on, SHOWING rather than TELLING. So, I made a new subclass so that I could go back to my original idea incase I didn't like this one, and made a grid that covered the same area, out of many smaller images of the same piece of grass.



Admittedly, this stage was quite small and relatively straight forward to do, but I decided to make it this way because I wanted to get a feel for how long things would take to do, and make sure that I wasn't stressing myself out by bombarding myself with tasks from the get-go. Having said this, I had a little bit of extra time in this stage so I decided to figure out what the icons for buying towers would look like on the left hand side. I originally began with just adding what the currency boosting tower and the projectile tower would look like on the left hand side.



However, I felt like this would look like a mistake in the game’s code as they look as if they are meant to be on the grassy area but have bugged out of bounds. So, I decided to make them look more like icons for the towers rather than the towers themselves, and I think this has a much better look.

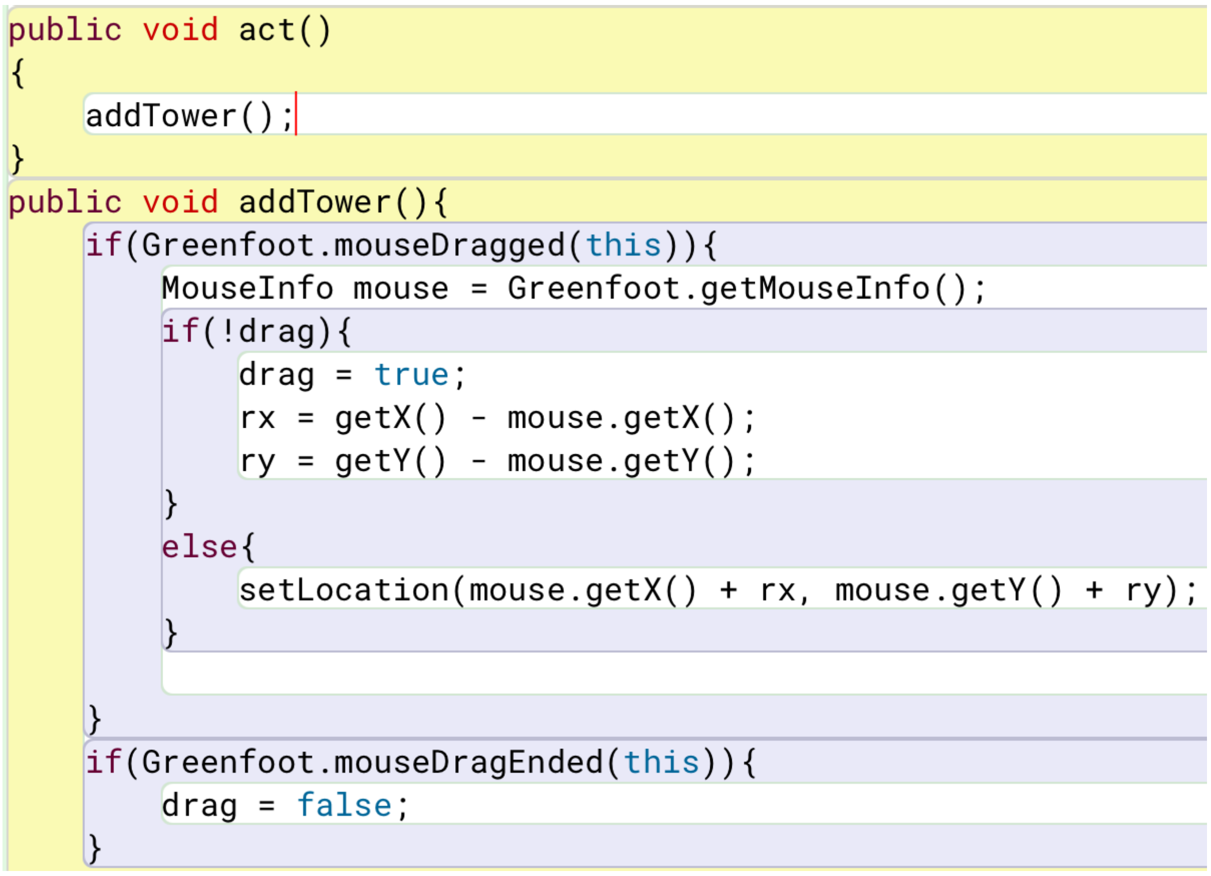


I put a shrunken down image of the tower on top of a tile of contrasting color which looks much better. I think this change is beneficial because it now feels much more intentional and looks more like what I want, that being an icon you click and drag onto the gameplay area to place down the tower.

Stage 2:

I began by developing the ability to click and drag towers in the gameplay area, as the first step of stage 2 is to develop the projectile tower, so I thought this “infrastructure” that’ll be used for all future towers, would be a good starting point. I didn’t really know where to start with this, so I found and followed a YouTube tutorial on how to make a simple click and drag function, which worked perfectly. Source:

https://www.youtube.com/watch?v=7pBkjbHqPJw

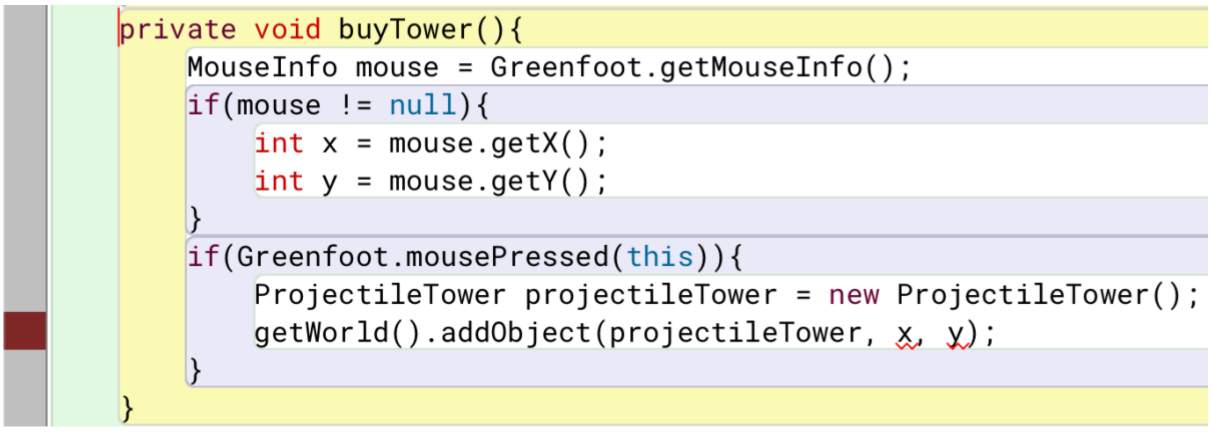


```
public void act()
{
    addTower();
}
public void addTower(){
    if(Greenfoot.mouseDragged(this)){
        MouseInfo mouse = Greenfoot.getMouseInfo();
        if(!drag){
            drag = true;
            rx = getX() - mouse.getX();
            ry = getY() - mouse.getY();
        }
        else{
            setLocation(mouse.getX() + rx, mouse.getY() + ry);
        }

    }
    if(Greenfoot.mouseDragEnded(this)){
        drag = false;
    }
```

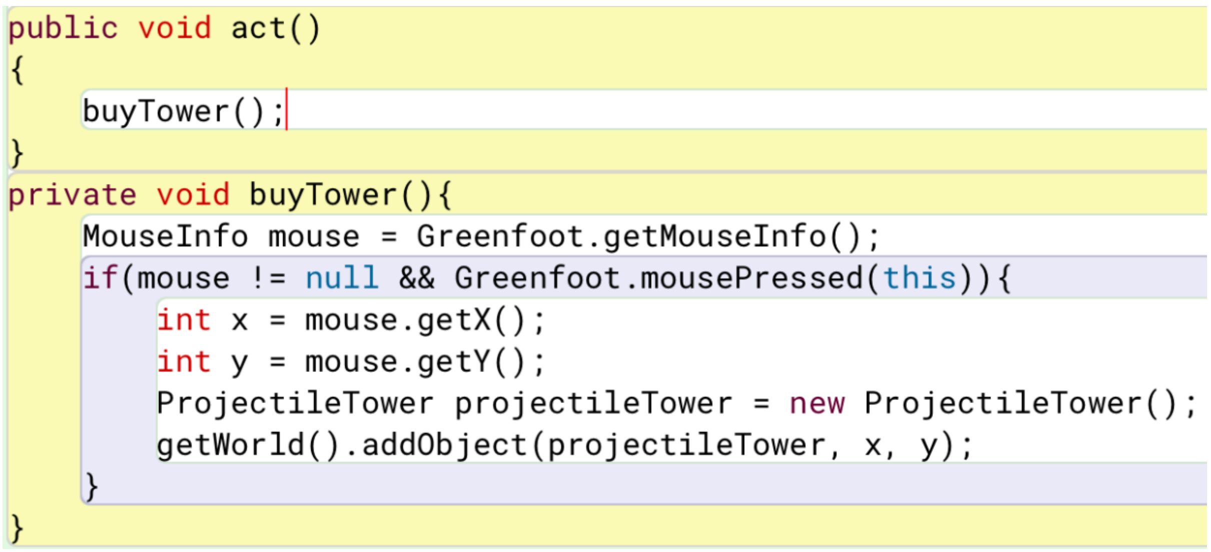
But next I need to make it so that when you click the one on the left and drag it, it brings a full size tower AND so that you can only place it down on grass, and once you've placed it, you can't move it again.

Next I tried to make it so when you click on the "Buy Projectile Tower" you can drag a tower onto the gameplay area to start defending against enemies, but I can't get it to work.



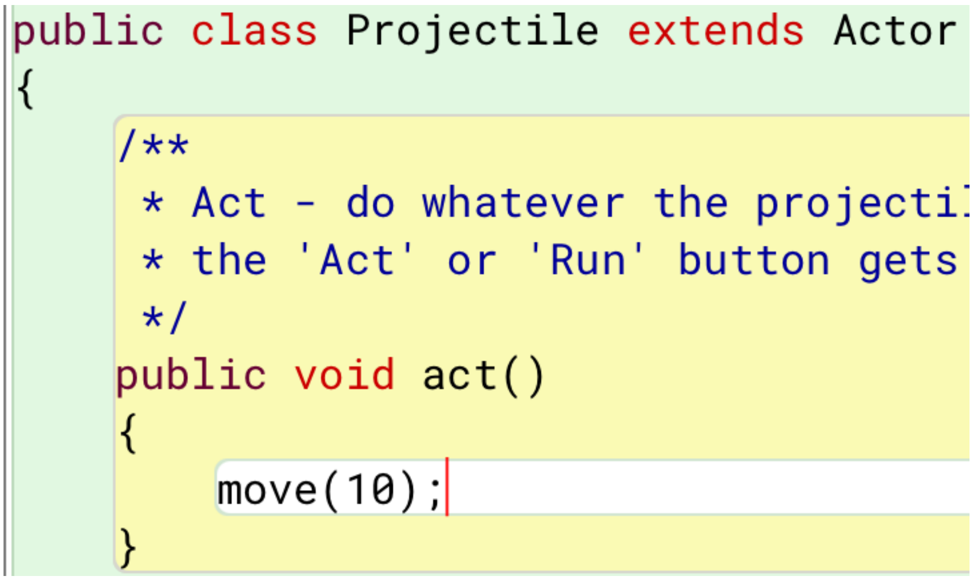
```
private void buyTower(){
    MouseInfo mouse = Greenfoot.getMouseInfo();
    if(mouse != null){
        int x = mouse.getX();
        int y = mouse.getY();
    }
    if(Greenfoot.mousePressed(this)){
        ProjectileTower projectileTower = new ProjectileTower();
        getWorld().addObject(projectileTower, x, y);
    }
}
```

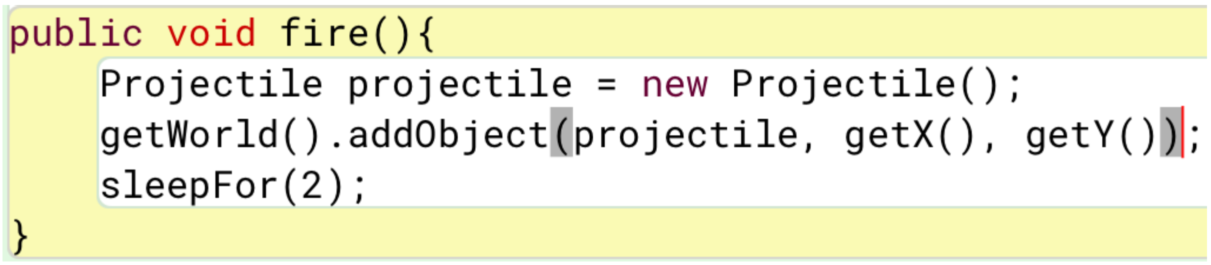
But then after reading the error I realized that it's a pretty easy fix and Greenfoot just didn't like me declaring "x" and "y" in a separate if statement, so I just combined them into one if statement and now the code works


```java
public void act()
{
    buyTower();
}
private void buyTower(){
    MouseInfo mouse = Greenfoot.getMouseInfo();
    if(mouse != null && Greenfoot.mousePressed(this)){
        int x = mouse.getX();
        int y = mouse.getY();
        ProjectileTower projectileTower = new ProjectileTower();
        getWorld().addObject(projectileTower, x, y);
    }
}
```

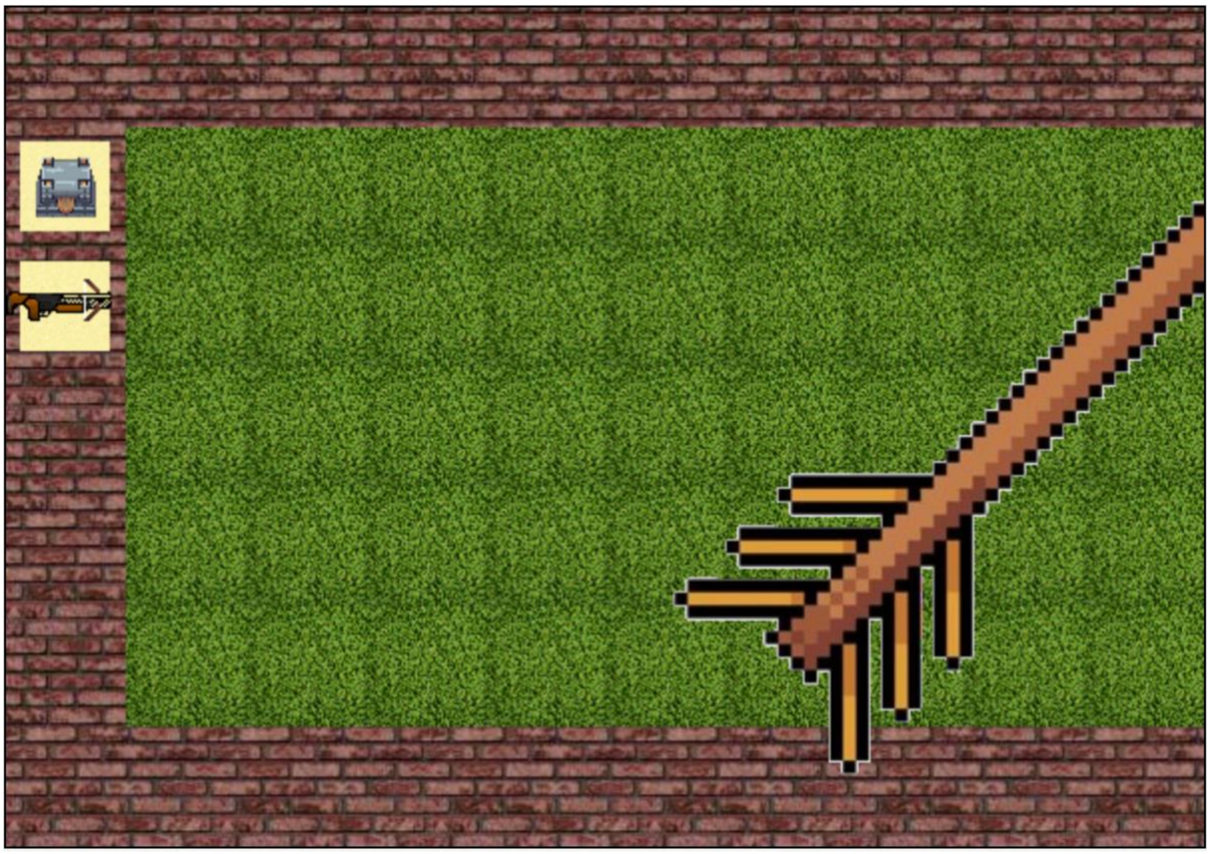
But now another problem arises in the form of you needing to click on the “buy projectile tower” you need to click again to drag the tower out into the gameplay area, but as this is more of a quality of life issue so I’ll fix this later down the line.

Next, I wanted to make the projectile tower fire the projectiles towards the right of the screen, so I created the projectile subclass, and wrote some simple code in the projectile tower class to instantiate a projectile every two seconds.


```java
public class Projectile extends Actor
{
    /**
     * Act - do whatever the projecti
     * the 'Act' or 'Run' button gets
     */
    public void act()
    {
        move(10);
    }
```

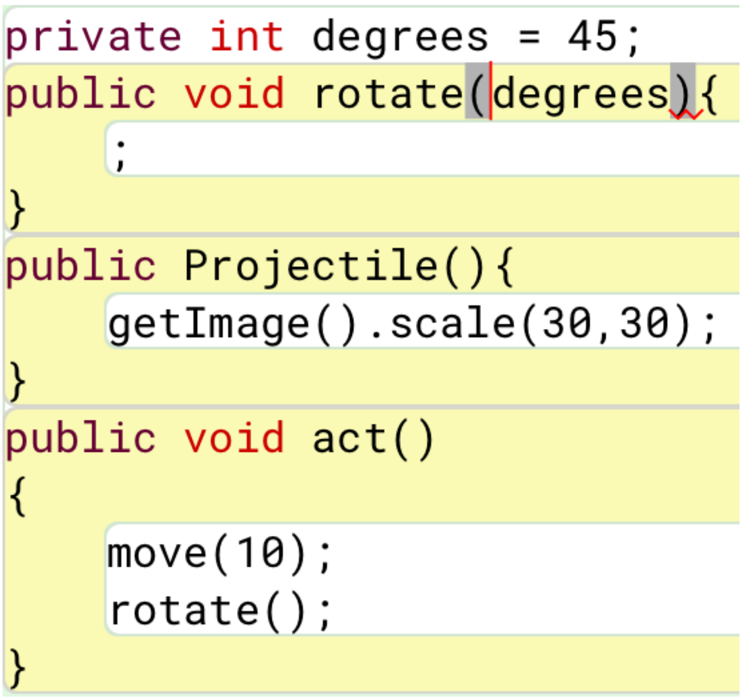

```java
public void fire(){
    Projectile projectile = new Projectile();
    getWorld().addObject(projectile, getX(), getY());
    sleepFor(2);
}
```

But after testing it I concluded that it does not work yet.

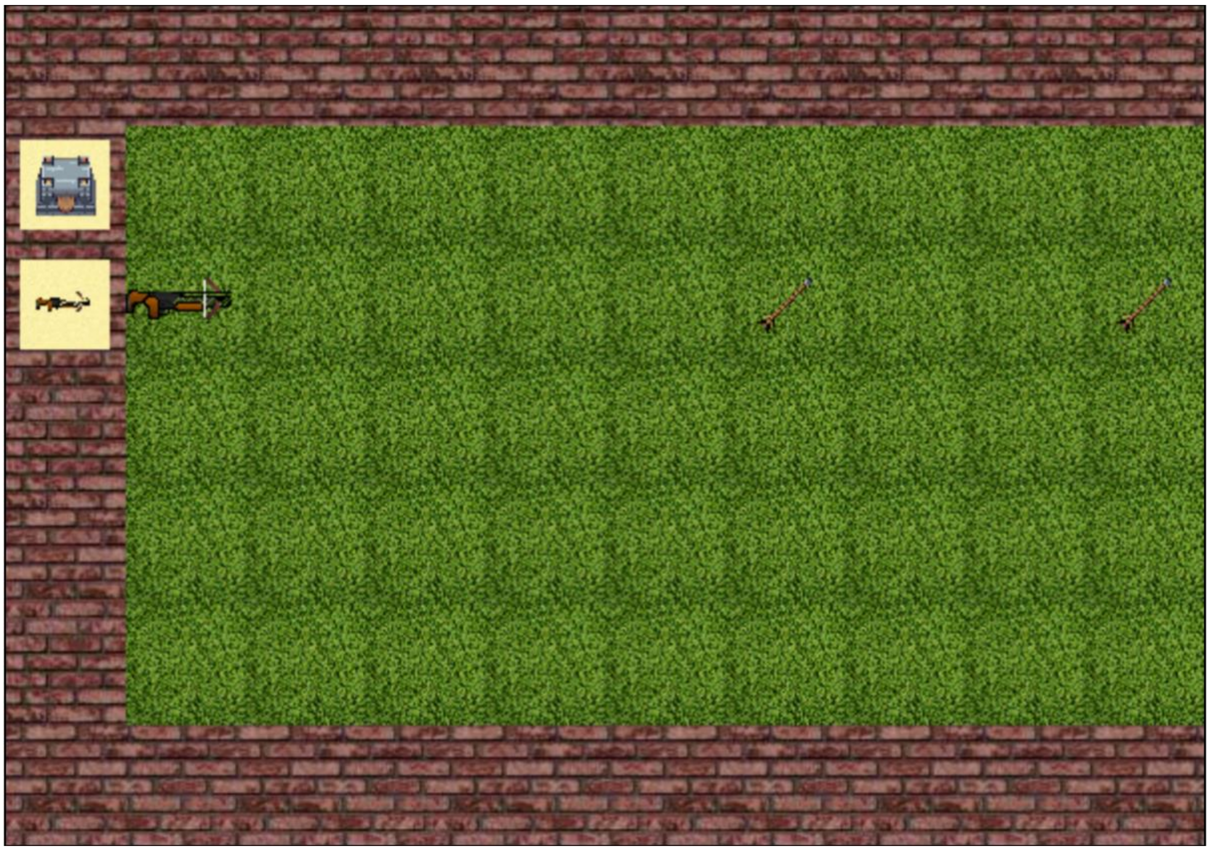


But I knew this was an easy fix to make, all I had forgotten to do was set the scale of the class. However, I also wanted to find a way to make the arrow actually point towards the right of the screen because as much as this part of the project isn't about vanity, I want it to feel somewhat "realistic".

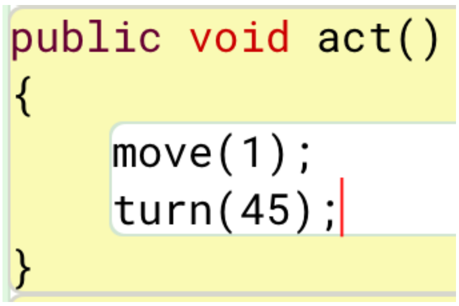
So, first I scaled it down which was easy as I've had to do it with all my other classes and looked on the Greenfoot Index API (https://www.greenfoot.org/files/javadoc/index-all.html) and saw that there was a rotate() function that I thought I could use in my project, with some modifications.



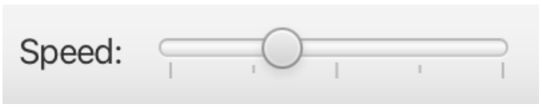
```
private int degrees = 45;
public void rotate(degrees){
    ;
}
public Projectile(){
    getImage().scale(30,30);
}
public void act()
{
    move(10);
    rotate();
}
```



This successfully scaled down the sprite, but did not change it's rotation / orientation at all. So, I kept looking for the solution to getting the arrow to point to the right, because I was now very dead set on having this completed before moving on, and found the turn() command in the Greenfoot API.

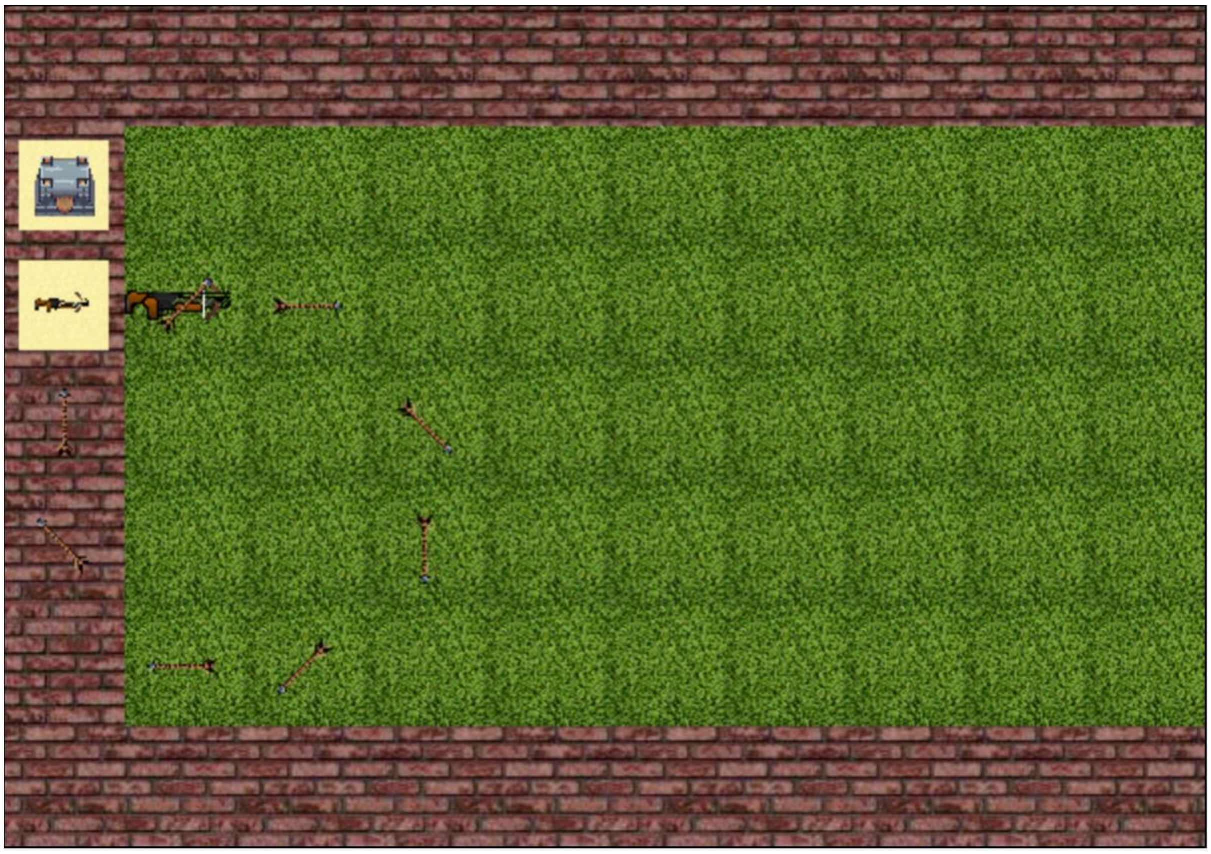


```
public void act()
{
    move(1);
    turn(45);
}
```



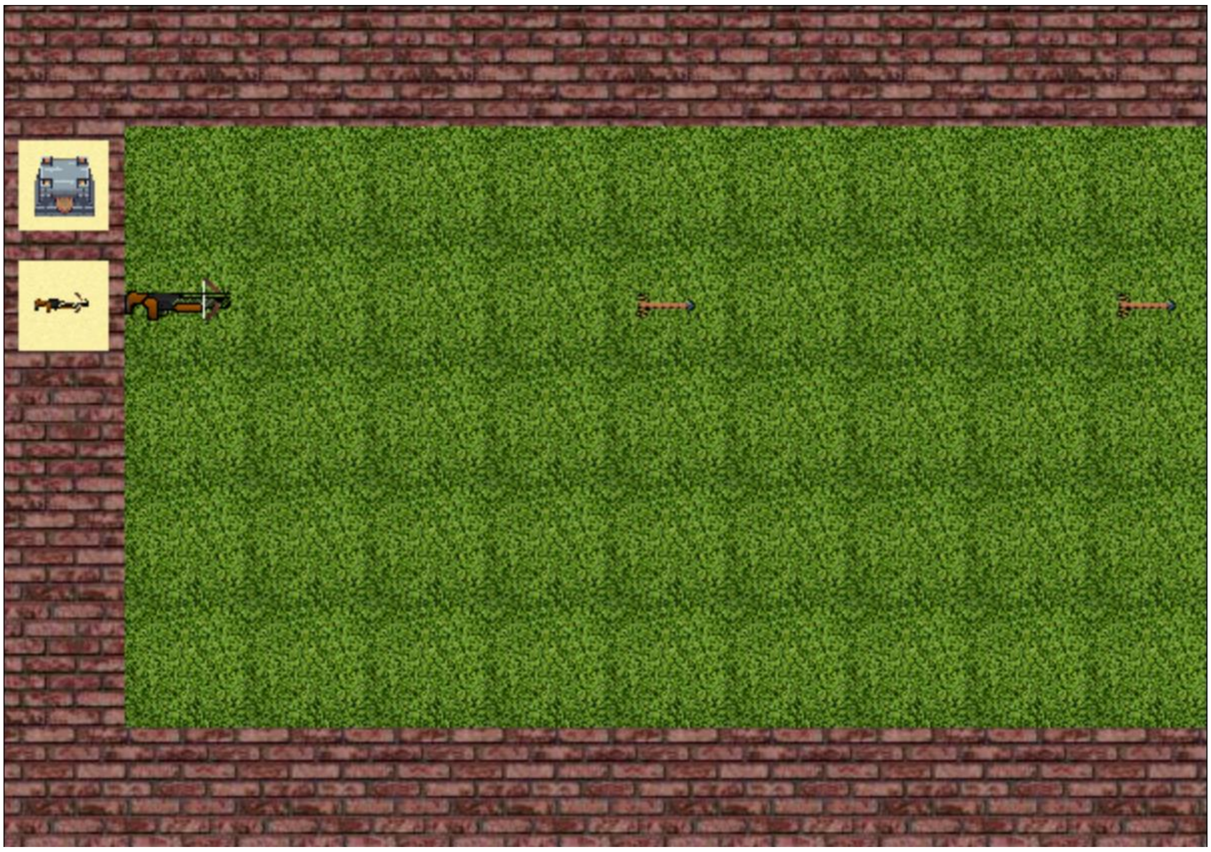
It was also at this point where I realized the arrows were flying out way too fast, which would unfairly favor the player when I integrated the enemies into the game as their fire rate would be way too high, clearing out the enemies far too easily. So I lowered the speed of the projectiles significantly because I wanted the game to feel balanced and fun because if the game was too easy, it wouldn't be fun to play and users would click off quite quickly.

But, sadly "turn" didn't quite mean what I thought it did:

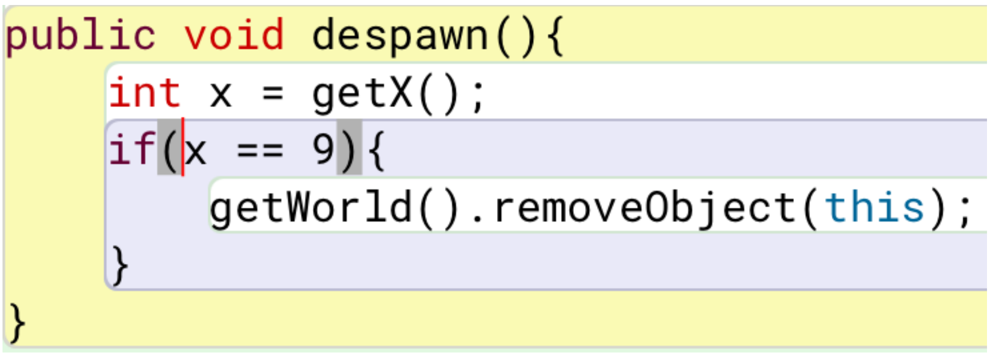


So, I think it would be easiest to just edit the image of the sprite so that it points to the right of the screen, which I think is a better choice because it'll result in tidier code when I'll inevitably come back to edit or debug it later, and because there's less code the program will run faster and smoother.

So now I have a tower that fires projectiles at a consistent rate and it actually points towards the direction it's travelling in, but now I've got the problem of them all just sitting on the right hand side of the screen, so I need to make them despawn when they make it to that side of the screen without hitting an enemy because this would not only look terrible from a gameplay point of view and would fill up RAM very quickly with more or less infinite projectile objects, so they need to go when they get that far.



This was as far as I was able to get for writing this command off my own steam because for the life of me I couldn't remember how to remove an object from the world if a specific condition was met.



```
public void despawn(){
    int x = getX();
    if(x == 9){
        getWorld().removeObject(this);
    }
}
```

But for whatever reason, this didn't work.