Appendix - Actor

```
package src;

import java.awt.*;

public abstract class Actor {

	/**
	 * Movement Vector Thing
	 */
	private Movement velocity = new Movement(0.0, 0.0);

	private Position position = new Position(0.0, 0.0);
	/**
	 * The rotation off of north of the object
	 */
	private Color color;

	public double radius;

	public int group;

	protected double speed;

	protected double rotation;

	public boolean edgeCollision;

	/*
	 * Boolean check for if needs to be removed
	 */
	private boolean needsRemoval;

	public boolean getRemoval() {
		return needsRemoval;
	}

	public Actor(Dictator d, Position position, Movement velocity, double radius) {

		group = d.rand.nextInt(15)+1;
		edgeCollision = false;
		color = Color.WHITE;
		this.velocity = velocity;
		this.position = position;
```

```
		this.radius = radius;
		this.needsRemoval = false;

}

/**
 * @param amount
 */
public void rotate(double amount) {
		this.rotation += amount;
		this.rotation %= Math.PI * 2;
}

public void setRotation(double angle) {
		rotation = angle;
}

public double getRotation() {
		return rotation;
}

public Position getPosition() {
		return position;
}

public Movement getVelocity() {
		return velocity;
}

public Movement getMovement() {
		return velocity;
}

public void setPosition(Position a) {
		position = a;
}

public void setMovement(Movement a) {
		velocity = a;
}

public boolean removeCheck() {
		return needsRemoval;
}

public void remove() {
```

```
		this.needsRemoval = true;
	}

	public double getRadius() {
		return radius;
	}

	public void update(Dictator d) {

		if (!edgeCollision) {
			if (position.getX() < 0.0f) {
				position.addX(d.SIZE_X);
			}
			if (position.getY() < 0.0f) {
				position.addY(d.SIZE_Y);
			}
			double posx = position.getX();
			position.setX(posx %= d.SIZE_X);
			double posy = position.getY();
			position.setY(posy %= d.SIZE_Y);
		} else {
			if (this.getClass() == Bullet.class) {
				if (position.getX() < 0.0f) {
					d.score--;
					remove();
				}
				if (position.getX() > d.SIZE_X) {
					d.score--;
					remove();
				}
				if (position.getY() < 0.0f) {
					d.score--;
					remove();
				}
				if (position.getY() > d.SIZE_Y) {
					d.score--;
					remove();
				}
			}
		}
	}

	public Color getColor() {
		return color;
	}
```

```
	public void setColor(Color c) {
		color = c;
	}

	public boolean colliding(Actor a) {

		double radius = a.getRadius() + getRadius();
		return (position.getDistanceToSquared(a.position) < radius * radius);
	}

	public abstract void collided(Actor a, Dictator dic);

	public abstract void draw(Graphics2D g, Dictator d);
}
```