

```
00001: package hevs.fragil.patapon.physics;
00002:
00003: import com.badlogic.gdx.math.Polygon;
00004: import com.badlogic.gdx.math.Vector2;
00005:
00006: import ch.hevs.gdx2d.components.bitmaps.BitmapImage;
00007: import ch.hevs.gdx2d.lib.GdxGraphics;
00008:
00009: public class Spear extends Projectile{
00010:     // for every arrow
00011:     private static BitmapImage img;
00012:     //TODO put this boolean into the projectile super class
00013:     private boolean flipped;
00014:     private static float[] spearVertices = { -1, 0, -1, 90, 0, 100, 1, 90, 1, 0 };
00015:
00016:     public Spear(Vector2 startPos, int startAngle, int distance, int collisionGroup, int damage) {
00017:         super(startPos,startAngle,collisionGroup,distance,damage,getSpearVertices(startAngle,(distance<0)),"arrow");
00018:         this.flipped = (distance<0);
00019:     }
00020:
00021:     public Vector2 getSpike() {
00022:         Vector2 temp = getBodyWorldCenter();
00023:         double angle = getBodyAngle() + startAngle;
00024:         temp.add((float) (Math.cos(angle) * 28), (float) (Math.sin(angle) * 28));
00025:         return temp;
00026:     }
00027:
00028:     private static Vector2[] getSpearVertices(int angle, boolean flipped) {
00029:         Polygon poly = new Polygon(spearVertices);
00030:         poly.setOrigin(0, 40);
00031:         if(flipped)
00032:             poly.rotate(270-angle);
00033:         else
```

```
00034:         poly.rotate(angle - 90);
00035:         return verticesToVector2(poly.getTransformedVertices());
00036:     }
00037:
00038:     @Override
00039:     public void draw(GdxGraphics g) {
00040:         float angleDegrees;
00041:         if(flipped)
00042:             angleDegrees = getBodyAngleDeg() - startAngle + 180 ;
00043:         else
00044:             angleDegrees = getBodyAngleDeg() + startAngle;
00045:         //         // better penetration depending of the impact angle
00046:         //         int distance = (int) (5 * Math.cos(angleRadians));
00047:         //         Vector2 offset = new Vector2 (
00048:         //             (float) Math.cos(angleRadians) * distance,
00049:         //             (float) Math.sin(angleRadians) * distance
00050:         //         );
00051:
00052:         Vector2 pos = getBodyWorldCenter();
00053:         //         pos = pos.add(offset);
00054:         g.drawAlphaPicture(pos.x, pos.y, angleDegrees, .4f, life, img);
00055:     }
00056:
00057:     public static void setImgPath(String url) {
00058:         img = new BitmapImage(url);
00059:     }
00060:
00061:     @Override
00062:     public void step(float dt) {
00063:         Vector2 v = getBodyLinearVelocity();
00064:         float angle = getBodyAngle();
00065:         double vNorm = Math.sqrt(v.x * v.x + v.y * v.y) * getBodyMass();
00066:
```

```
00067:         // process lift force relative to the angle and the velocity
00068:         float lift = (float) (-Math.cos(angle) * vNorm * 120 * dt);
00069:         if(v.x < 0)
00070:             lift = -lift;
00071:         // apply air damping
00072:         applyBodyTorque(lift, true);
00073:
00074:         // if this arrow is stuck, it start degrading itself
00075:         if (stuck)
00076:             this.life = Math.max(0, life - 0.005f);
00077:     }
00078: }
```