```
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: /**
00010:  *
00011:  * @author loicg
00012:  *
00013:  */
00014: public class Arrow extends Projectile{
00015:     // for every arrow
00016:     private static BitmapImage img;
00017:     private boolean flipped;
00018:     private static float[] arrowVertices = { -1, 0, -1, 40, 0, 50, 1, 40, 1, 0 };
00019:
00020:     public Arrow(Vector2 startPos, int startAngle, int distance, int collisionGroup, int damage) {
00021:         super(startPos,startAngle,collisionGroup,distance,damage,getArrowVertices(startAngle,(distance<0)),"arrow");
00022:         this.flipped = (distance<0);
00023:     }
00024:
00025:     public Vector2 getSpike() {
00026:         Vector2 temp = getBodyWorldCenter();
00027:         double angle = getBodyAngle() + startAngle;
00028:         temp.add((float) (Math.cos(angle) * 28), (float) (Math.sin(angle) * 28));
00029:         return temp;
00030:     }
00031:
00032:     private static Vector2[] getArrowVertices(int angle, boolean flipped) {
00033:         Polygon poly = new Polygon(arrowVertices);
```

```
00034:         poly.setOrigin(0, 40);
00035:         if(flipped)
00036:             poly.rotate(270-angle);
00037:         else
00038:             poly.rotate(angle - 90);
00039:         return verticesToVector2(poly.getTransformedVertices());
00040:     }
00041:
00042:     @Override
00043:     public void draw(GdxGraphics g) {
00044:         float angleDegrees;
00045:         if(flipped)
00046:             angleDegrees = getBodyAngleDeg() - startAngle + 180 ;
00047:         else
00048:             angleDegrees = getBodyAngleDeg() + startAngle;
00049: //        double angleRadians = getBodyAngle();
00050: //        // better penetration depending of the impact angle
00051: //        int distance = 3 + (int) (5 * Math.cos(angleRadians));
00052: //        Vector2 offset = new Vector2((float) Math.cos(angleRadians) * distance,
00053: //                (float) Math.sin(angleRadians) * distance);
00054:
00055:         Vector2 pos = getBodyWorldCenter();
00056: //        pos = pos.add(offset);
00057:         g.drawAlphaPicture(pos.x, pos.y, angleDegrees, .2f, life, img);
00058:     }
00059:
00060:     public static void setImgPath(String url) {
00061:         img = new BitmapImage(url);
00062:     }
00063:
00064:     @Override
00065:     public void step(float dt) {
00066:         Vector2 v = getBodyLinearVelocity();
```

```
00067:         float angle = getBodyAngle();
00068:         double vNorm = Math.sqrt(v.x * v.x + v.y * v.y) * getBodyMass();
00069:
00070:         // process lift force relative to the angle and the velocity
00071:         float lift = (float) (-Math.cos(angle) * vNorm * 12 * dt);
00072:         if(v.x < 0)
00073:             lift = -lift;
00074:         // apply air damping
00075:         applyBodyTorque(lift, true);
00076:
00077:         // if this arrow is stuck, it start degrading itself
00078:         if (stuck)
00079:             this.life = Math.max(0, life - 0.005f);
00080:     }
00081: }
```