

```
00001: package hevs.fragil.patapon.physics;
00002:
00003: import com.badlogic.gdx.Gdx;
00004: import com.badlogic.gdx.math.Vector2;
00005:
00006: import ch.hevs.gdx2d.components.physics.primitives.PhysicsPolygon;
00007:
00008: public class BodyPolygon extends PhysicsPolygon implements CollidedObject {
00009:     static Vector2 dimensions = new Vector2(3,80);
00010:     int collisionGroup;
00011:     private float life;
00012:     static int nArrows;
00013:     static Vector2 body[] = {
00014:         new Vector2(-30, 0),
00015:         new Vector2(-30, 60),
00016:         new Vector2(0, 80),
00017:         new Vector2(30, 60),
00018:         new Vector2(30, 0)
00019:     };
00020:
00021:     public BodyPolygon(Vector2 position, int collisionGroup, int life) {
00022:         //Ca c'est vraiment super !
00023:         super("arrow"+nArrows, position, body, 1000f, 0f, 1f, true);
00024:         this.life = life;
00025:         getBody().setBullet(true);
00026:         this.collisionGroup = collisionGroup;
00027:         setCollisionGroup(collisionGroup);
00028:         nArrows++;
00029:     }
00030:     public void moveToLinear(int position, double travelTime) {
00031:         travelTime *= 1000;
00032:         double distanceToTravel = position - (int)getBodyPosition().x;
00033:         double globalSpeed = distanceToTravel / travelTime;
```

```
00034:         int fps = Gdx.graphics.getFramesPerSecond();
00035:
00036:         // Check if this speed will cause overshoot in the next time step.
00037:         // If so, we need to scale the speed down to just enough to reach
00038:         // the target point.
00039:         double stepDistance = globalSpeed * (1.0/fps);
00040:         if ( Math.abs(stepDistance) > Math.abs(distanceToTravel) )
00041:             globalSpeed *= ( distanceToTravel / stepDistance );
00042:
00043:         double desiredAcceleration = Math.abs(globalSpeed) * distanceToTravel;
00044:         double changeInAcceleration = desiredAcceleration - this.getBodyLinearVelocity().x;
00045:
00046:         double force = this.getBodyMass() * fps * changeInAcceleration;
00047:         this.applyBodyForceToCenter((float)force, 0, true);
00048:     }
00049:     @Override
00050:     public int getCollisionGroup() {
00051:         return collisionGroup;
00052:     }
00053:     @Override
00054:     /** Return true when damage leads to death */
00055:     public boolean applyDamage(float damage) {
00056:         if(life > 0){
00057:             life -= damage;
00058:             if(life <= 0){
00059:                 kill();
00060:                 return true;
00061:             }
00062:             else return false;
00063:         }
00064:         else return false;
00065:
00066:     }
```

```
00067:     public float getLife(){
00068:         return life;
00069:     }
00070:     public boolean isDead(){
00071:         if(life <= 0) return true;
00072:         else return false;
00073:     }
00074:     public void kill() {
00075:         getBody().setFixedRotation(false);
00076:         applyBodyAngularImpulse(-1300, true);
00077:     }
00078: }
```