

```
00001: package hevs.fragil.patapon.units;
00002:
00003: import com.badlogic.gdx.Gdx;
00004: import com.badlogic.gdx.graphics.g2d.Animation.PlayMode;
00005: import com.badlogic.gdx.math.Vector2;
00006:
00007: import ch.hevs.gdx2d.lib.GdxGraphics;
00008: import hevs.fragil.patapon.drawables.SpriteSheet;
00009: import hevs.fragil.patapon.mechanics.CurrentLevel;
00010:
00011: public class UnitRender {
00012:     private Look look = Look.DEFAULT;
00013:
00014:     private Gesture gesture = Gesture.WALK;
00015:
00016:     private State state = State.WALK;
00017:
00018:     private float opacity = 1f;
00019:     protected int nAttacks;
00020:
00021:     private int bodyIndex;
00022:
00023:     private SpriteSheet body, eye, arms, legs;
00024:
00025:     protected float counter = -1;
00026:     protected float cooldownCounter;
00027:
00028:     private boolean gestureRunning = false;
00029:
00030:     private Stabilizer pos = new Stabilizer();
00031:
00032:     public boolean attack = false;
00033:
```

```
00034:  /**
00035:     * Constructor for a new UnitRender
00036:     * @param bodyIndex : the body sprite index
00037:     * @param preDelay : the delay for the attack animation
00038:     */
00039: public UnitRender (int bodyIndex){
00040:     this.bodyIndex = bodyIndex;
00041: }
00042:
00043:
00044: public Look getLook() {
00045:     return look;
00046: }
00047: public void setLook(Look expression) {
00048:     this.look = expression;
00049: }
00050: public Gesture getGesture() {
00051:     return gesture;
00052: }
00053: public void setGesture(Gesture gesture) {
00054:     this.gesture = gesture;
00055: }
00056:
00057: public void setState(State s) {
00058:     state = s;
00059: }
00060: public State getState() {
00061:     return state;
00062: }
00063:
00064: public Vector2 getPos(){
00065:     return pos.getStabilizedPos();
00066: }
```

```
00067:
00068:     public void draw(GdxGraphics g, float x, float y, float angle) {
00069:         if (state == State.DYING)
00070:             drawDead(g, pos.stabilized(x,y), angle);
00071:         else
00072:             drawAlive(g, pos.stabilized(x,y));
00073:     }
00074:     private void drawAlive(GdxGraphics g, Vector2 position) {
00075:         gestureSwitch();
00076:
00077:         float stateTime = CurrentLevel.getLevel().getStateTime();
00078:         int legsIndex = legs.drawAllFrames(stateTime, position);
00079:         body.drawWalkAnimation(legsIndex, bodyIndex, position.x, position.y-5);
00080:         eye.drawWalkAnimation(legsIndex, look.ordinal(), position.x, position.y-5);
00081:         arms.drawFrames(stateTime, gesture.ordinal() * 4, 4, position.x, position.y-5);
00082:     }
00083:
00084:     private void drawDead(GdxGraphics g, Vector2 position, float angle) {
00085:         gestureSwitch();
00086:         legs.drawRotatedFrame(0, angle, position.x, (float) (position.y-Math.cos(angle)*32));
00087:         body.drawRotatedFrame(bodyIndex, angle, position.x, (float) (position.y-Math.cos(angle)*32));
00088:         eye.drawRotatedFrame(Look.DYING.ordinal(), angle, position.x, (float) (position.y-Math.cos(angle)*32));
00089:         arms.drawRotatedFrame(0, angle, position.x, (float) (position.y-Math.cos(angle)*32));
00090:     }
00091:     private void gestureSwitch() {
00092:         float dt = Gdx.graphics.getDeltaTime();
00093:
00094:         if(counter >= 0){
00095:             counter += dt;
00096:         }
00097:         if(counter >= 4 * arms.getFrameDuration()){
00098:             gesture = Gesture.WALK;
00099:             counter = -1;
```

```
00100:     }
00101: }
00102: protected void launch(Gesture a) {
00103:     if(gesture != a){
00104:         setGesture(a);
00105:         if(counter == -1)
00106:             counter = 0;
00107:     }
00108: }
00109: public boolean die() {
00110:     opacity -= 0.005f;
00111:     if (opacity <= 0) {
00112:         return true;
00113:     }
00114:     return false;
00115: }
00116:
00117: /** This is only to load files in the PortableApplication onInit method */
00118: public void setLegsSprite(String url, int cols, int rows, boolean isEnnemi) {
00119:     legs = new SpriteSheet(url, cols, rows, 0.2f, isEnnemi, PlayMode.LOOP);
00120: }
00121:
00122: /** This is only to load files in the PortableApplication onInit method */
00123: public void setBodySprite(String url, int cols, int rows) {
00124:     body = new SpriteSheet(url, cols, rows, 1f, false, PlayMode.LOOP);
00125: }
00126:
00127: /** This is only to load files in the PortableApplication onInit method */
00128: public void setEyeSprite(String url, int cols, int rows) {
00129:     eye = new SpriteSheet(url, cols, rows, 0.2f, false, PlayMode.LOOP);
00130: }
00131:
00132: /** This is only to load files in the PortableApplication onInit method */
```

```
00133:     public void setArmsSprite(String url, int cols, int rows, boolean isEnnemi) {
00134:         arms = new SpriteSheet(url, cols, rows, 0.2f, isEnnemi, PlayMode.NORMAL);
00135:     }
00136:     public boolean gestureRunning() {
00137:         return gestureRunning;
00138:     }
00139: }
```