```
00001: package hevs.fragil.patapon.mechanics;
00002:
00003: import com.badlogic.gdx.Gdx;
00004: import com.badlogic.gdx.math.Interpolation;
00005:
00006: import ch.hevs.gdx2d.components.audio.SoundSample;
00007: import hevs.fragil.patapon.units.Company;
00008: import hevs.fragil.patapon.units.Section;
00009: import hevs.fragil.patapon.units.State;
00010: import hevs.fragil.patapon.units.Unit;
00011:
00012: /**
00013:  * This class manages the company movements and actions. It helps creating animations.
00014:  * This is called every FPS, and process the automatic movements of the units.
00015:  */
00016: public abstract class SequenceTimer{
00017:     private static float step = 0;
00018:     private static float deltaTime;
00019:     private static float feverScore;
00020:     private static float start, progression, end;
00021:     private static SoundSample lalala;
00022:     private static boolean playing = false;
00023:
00024:     public static void run(Company c, int fever) {
00025:         float dt = Gdx.graphics.getRawDeltaTime();
00026:         deltaTime = dt;
00027:         feverScore = fever;
00028:         switchAction(c.getAction(), c);
00029:     }
00030:     private static void switchAction(State a, Company c){
00031:         boolean finished = false;
00032:         if(a != null){
00033:             if(playing == false){
```

```
00034:                 //for instance, this song is way too annoying (so we don't play it)
00035:                 //lalala.play();
00036:                 playing = true;
00037:             }
00038:
00039:             // process moves
00040:             c.aiMove();
00041:
00042:             switch(a){
00043:                 case WALK :     finished = walk(c);
00044:                                 break;
00045:                 case ATTACK :     finished = attack(c);
00046:                                 break;
00047:                 case DEFEND :     finished = defend(c);
00048:                                 break;
00049:                 case MIRACLE :     finished = miracle(c);
00050:                                 break;
00051:                 case RETREAT :     finished = retreat(c);
00052:                                 break;
00053:                 case CHARGE :     finished = charge(c);
00054:                                 break;
00055:                 case IDLE :        finished = stop(c);
00056:                                 playing = false;
00057:                                 lalala.stop();
00058:                                 break;
00059:                 default :
00060:                                 break;
00061:             }
00062:             if(finished){
00063:                 playing = false;
00064:                 c.actionFinished();
00065:             }
00066:         }
```

```
00067:     }
00068:     private static boolean charge(Company c) {
00069:         // Disable automatic unit placement
00070:         c.regroupUnits();
00071:
00072:         //increase attack skills for this time
00073:         return wait(Param.CHARGE_TIME, c);
00074:     }
00075:     private static boolean miracle(Company c) {
00076:         // Disable automatic unit placement
00077:         c.regroupUnits();
00078:
00079:         //TODO new screen launch !
00080:         return true;
00081:     }
00082:     private static boolean defend(Company c) {
00083:         // Disable automatic unit placement
00084:         c.regroupUnits();
00085:
00086:         //TODO increase defend skills for this time
00087:         return wait(Param.DEFEND_TIME, c);
00088:     }
00089:     private static boolean shift(float totalTime, int distance, Company c){
00090:         if(progression == 0f){
00091:             start = c.getPosition();
00092:             end = start + distance;
00093:         }
00094:
00095:         progression += deltaTime/totalTime;
00096:         c.setPosition((int)Interpolation.fade.apply(start, end, progression));
00097:
00098:         if(progression >= 1f){
00099:             progression = 0f;
```

```
00100:             return true;
00101:         }
00102:         else return false;
00103:     }
00104:     private static boolean wait(double time, Company c){
00105:         progression += deltaTime;
00106:
00107:         if(progression >= time ){
00108:             progression = 0f;
00109:             return true ;
00110:         }
00111:
00112:         else return false ;
00113:
00114:     }
00115:     private static boolean walk(Company c){
00116:         float time = Param.WALK_TIME;
00117:
00118:         // Disable automatic unit placement
00119:         c.regroupUnits();
00120:
00121:         //add bonus time (faster move with fever)
00122:         time -= Param.WALK_TIME_BONUS/100.0f * feverScore;
00123:
00124:         return shift(time, Param.WALK_WIDTH, c);
00125:     }
00126:     private static boolean retreat(Company c){
00127:         // Disable automatic unit placement
00128:         c.regroupUnits();
00129:
00130:         float time = Param.RETREAT_TIME;
00131:         float bonus = (float) (Param.RETREAT_TIME_BONUS/100.0f * feverScore);
00132:         if (step == 0f) {
```

```
00133:             if(shift(time/4f - bonus, -Param.RETREAT_WIDTH, c))
00134:                 step++;
00135:         }
00136:
00137:         else if(step == 1f){
00138:             if(wait(time/2f + bonus, c))
00139:                 step++;
00140:         }
00141:
00142:         else if (shift(time/4f, Param.RETREAT_WIDTH, c)){
00143:             step = 0;
00144:             return true;
00145:         }
00146:
00147:         return false;
00148:     }
00149:     private static boolean attack(Company c){
00150:         progression += deltaTime;
00151:
00152:         // Enable automatic unit placement
00153:         c.freeUnits();
00154:
00155:         for (Section s : c.sections) {
00156:             for (Unit u : s.units) {
00157:                 u.attackRoutine();
00158:             }
00159:         }
00160:
00161:         //action ended
00162:         if(progression >= Param.ATTACK_TIME) {
00163:             for (Section s : c.sections) {
00164:                 for (Unit u : s.units) {
00165:                     u.resetGesture();
```

```
00166:                 }
00167:             }
00168:             progression = 0f;
00169:             return true;
00170:         }
00171:         return false;
00172:     }
00173:     private static boolean stop(Company c){
00174:         end = start;
00175:         progression = 0f;
00176:         step = 0;
00177:         deltaTime = 0f;
00178:         return true;
00179:     }
00180:     public static void loadFiles(){
00181:         lalala = new SoundSample("data/music/lalala.mp3");
00182:     }
00183: }
```