```
    1: //////////////////////////////////////////////////////////
    2: //
    3: // Copyright (C) 2014 Maximilian Wagenbach (aka. Foaly) (foaly.f@web.de)
    4: //
    5: // This software is provided 'as-is', without any express or implied warr
anty.
    6: // In no event will the authors be held liable for any damages arising
    7: // from the use of this software.
    8: //
    9: // Permission is granted to anyone to use this software for any purpose,
   10: // including commercial applications, and to alter it and redistribute it
 freely,
   11: // subject to the following restrictions:
   12: //
   13: // 1. The origin of this software must not be misrepresented;
   14: // you must not claim that you wrote the original software.
   15: // If you use this software in a product, an acknowledgment
   16: // in the product documentation would be appreciated but is not required.
   17: //
   18: // 2. Altered source versions must be plainly marked as such,
   19: // and must not be misrepresented as being the original software.
   20: //
   21: // 3. This notice may not be removed or altered from any source distribut
ion.
   22: //
   23: //////////////////////////////////////////////////////////
   24:
   25: #include "AnimatedSprite.hpp"
   26:
   27: AnimatedSprite::AnimatedSprite(sf::Time frameTime, bool paused, bool loop
ed) :
   28:     m_animation(NULL), m_frameTime(frameTime), m_currentFrame(0),
   29:     m_isPaused(paused), m_isLooped(looped), m_texture(NULL) {
   30: }
   31:
   32: void AnimatedSprite::setAnimation(const Animation& animation) {
   33:     m_animation = &animation;
   34:     m_texture = m_animation->getSpriteSheet();
   35:     m_currentFrame = 0;
   36:     setFrame(m_currentFrame);
   37: }
   38:
   39: void AnimatedSprite::setFrameTime(sf::Time time) {
   40:     m_frameTime = time;
   41: }
   42:
   43: void AnimatedSprite::play() {
   44:     m_isPaused = false;
   45: }
   46:
   47: void AnimatedSprite::play(const Animation& animation) {
   48:     if (getAnimation() != &animation)
   49:         setAnimation(animation);
   50:     play();
   51: }
   52:
   53: void AnimatedSprite::pause() {
   54:     m_isPaused = true;
   55: }
   56:
   57: void AnimatedSprite::stop() {
   58:     m_isPaused = true;
   59:     m_currentFrame = 0;
   60:     setFrame(m_currentFrame);
   61: }
```

```
   62:
   63: void AnimatedSprite::setLooped(bool looped) {
   64:     m_isLooped = looped;
   65: }
   66:
   67: void AnimatedSprite::setColor(const sf::Color& color) {
   68:     // Update the vertices' color
   69:     m_vertices[0].color = color;
   70:     m_vertices[1].color = color;
   71:     m_vertices[2].color = color;
   72:     m_vertices[3].color = color;
   73: }
   74:
   75: const Animation* AnimatedSprite::getAnimation() const {
   76:     return m_animation;
   77: }
   78:
   79: sf::FloatRect AnimatedSprite::getLocalBounds() const {
   80:     sf::IntRect rect = m_animation->getFrame(m_currentFrame);
   81:
   82:     float width = static_cast<float>(std::abs(rect.width));
   83:     float height = static_cast<float>(std::abs(rect.height));
   84:
   85:     return sf::FloatRect(0.f, 0.f, width, height);
   86: }
   87:
   88: sf::FloatRect AnimatedSprite::getGlobalBounds() const {
   89:     return getTransform().transformRect(getLocalBounds());
   90: }
   91:
   92: bool AnimatedSprite::isLooped() const {
   93:     return m_isLooped;
   94: }
   95:
   96: bool AnimatedSprite::isPlaying() const {
   97:     return !m_isPaused;
   98: }
   99:
  100: sf::Time AnimatedSprite::getFrameTime() const {
  101:     return m_frameTime;
  102: }
  103:
  104: void AnimatedSprite::setFrame(std::size_t newFrame, bool resetTime) {
  105:     if (m_animation) {
  106:         // calculate new vertex positions and texture coordiantes
  107:         sf::IntRect rect = m_animation->getFrame(newFrame);
  108:
  109:         m_vertices[0].position = sf::Vector2f(0.f, 0.f);
  110:         m_vertices[1].position = sf::Vector2f(0.f, static_cast<float>(rec
t.height));
  111:         m_vertices[2].position = sf::Vector2f(static_cast<float>(rect.wid
th),
  112:                                               static_cast<float>(rect.heigh
t));
  113:         m_vertices[3].position = sf::Vector2f(static_cast<float>(rect.wid
th), 0.f);
  114:
  115:         float left = static_cast<float>(rect.left) + 0.0001f;
  116:         float right = left + static_cast<float>(rect.width);
  117:         float top = static_cast<float>(rect.top);
  118:         float bottom = top + static_cast<float>(rect.height);
  119:
  120:         m_vertices[0].texCoords = sf::Vector2f(left, top);
  121:         m_vertices[1].texCoords = sf::Vector2f(left, bottom);
  122:         m_vertices[2].texCoords = sf::Vector2f(right, bottom);
```

```
  123:         m_vertices[3].texCoords = sf::Vector2f(right, top);
  124:     }
  125:
  126:     if (resetTime)
  127:         m_currentTime = sf::Time::Zero;
  128: }
  129:
  130: void AnimatedSprite::update(sf::Time deltaTime) {
  131:     // if not paused and we have a valid animation
  132:     if (!m_isPaused && m_animation) {
  133:         // add delta time
  134:         m_currentTime += deltaTime;
  135:
  136:         // if current time is bigger then the frame time advance one fram
e
  137:         if (m_currentTime >= m_frameTime) {
  138:             // reset time, but keep the remainder
  139:          m_currentTime = sf::microseconds(m_currentTime.asMicroseconds()
%
  140:                                                 m_frameTime.asMicroseconds())
;
  141:
  142:             // get next Frame index
  143:             if (m_currentFrame + 1 < m_animation->getSize()) {
  144:                 m_currentFrame++;
  145:             } else {
  146:                 // animation has ended
  147:                 if (!m_isLooped) {
  148:                     m_isPaused = true;
  149:                 } else {
  150:                     m_currentFrame = 0;
  151:                     // reset to start
  152:                 }
  153:             }
  154:
  155:             // set the current frame, not reseting the time
  156:             setFrame(m_currentFrame, false);
  157:         }
  158:     }
  159: }
  160:
  161: void AnimatedSprite::draw(sf::RenderTarget& target, sf::RenderStates stat
es) const {
  162:     if (m_animation && m_texture) {
  163:         states.transform *= getTransform();
  164:         states.texture = m_texture;
  165:         target.draw(m_vertices, 4, sf::Quads, states);
  166:     }
  167: }
```