```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using System;
using System.Collections.Generic;
using System.Text;

namespace GameDemo.Shared
{
    class AnimationPlayer
    {
        Animation animation;
        public Animation Animation
        {
            get
            {
                return animation;
            }
        }

        public int FrameIndex
        {
            get
            {
                return frameIndex;
            }

        }
        public Vector2 Origin
        {
            get { return new Vector2
              (this.Animation.FrameWidth/2,this.Animation.FrameHeight/2);  }

        }
        int frameIndex;
        /// <summary>

        /// The amount of time in seconds that the current frame has been
          shown for.

        /// </summary>
        float time;

        /// <summary>

        /// Begins or continues playback of an animation.

        /// </summary>
        public void PlayAnimation(Animation animation)
        {
            if (Animation == animation)
            {
                return;
            }
            this.animation = animation;
            this.frameIndex = 0;
            this.time = 0;
```

```
55
56          }
57          /// <summary>
58
59          /// Advances the time position and draws the current frame of the
              animation.
60
61          /// </summary>
62
63          public void Draw(GameTime gameTime, SpriteBatch spriteBatch, Vector2
              position, SpriteEffects spriteEffects,float scale)
64          {
65              if (Animation == null)
66
67                  throw new NotSupportedException("No animation is currently
                      playing.");
68
69              // Process passing time.
70
71              time += (float)gameTime.ElapsedGameTime.TotalMilliseconds;
72
73              while (time > Animation.MillisecondPerFrame)
74              {
75                  time -= Animation.MillisecondPerFrame;
76
77                  // Advance the frame index; looping or clamping as
                      appropriate.
78
79                  if (Animation.isLooping)
80                  {
81                      frameIndex = (frameIndex + 1) % Animation.totalFrames;
82                  }
83                  else
84                  {
85                      frameIndex = Math.Min(frameIndex + 1,
                          Animation.totalFrames - 1);
86
87                  }
88              }
89
90              // Calculate the source rectangle of the current frame.
91
92              Rectangle source = new Rectangle(FrameIndex *
                  Animation.FrameWidth, 0, Animation.FrameWidth,
                  Animation.FrameHeight);
93
94              // Draw the current frame.
95
96              spriteBatch.Draw(Animation.Texture, position, source,
                  Color.Aquamarine, 0.0f, Origin, scale, spriteEffects, 0.0f);
97
98
99          }
100
101     }
102 }
```