```
////////////////////////////////////////////////////////////
//
// Copyright (C) 2014 Maximilian Wagenbach (aka. Foaly) (foaly.f@web.de)
//
// This software is provided 'as-is', without any express or implied warranty.
// In no event will the authors be held liable for any damages arising from the use of this software.
//
// Permission is granted to anyone to use this software for any purpose,
// including commercial applications, and to alter it and redistribute it freely,
// subject to the following restrictions:
//
// 1. The origin of this software must not be misrepresented;
// you must not claim that you wrote the original software.
// If you use this software in a product, an acknowledgment
// in the product documentation would be appreciated but is not required.
//
// 2. Altered source versions must be plainly marked as such,
// and must not be misrepresented as being the original software.
//
// 3. This notice may not be removed or altered from any source distribution.
//
////////////////////////////////////////////////////////////

//little changes

#include "stdafx.h"


#include "Animation.hpp"

Animation::Animation() : m_texture(nullptr)
{
    m_active = -1;

}

Animation::~Animation()
{
    m_texture = nullptr;

    m_frames.clear();
    m_aniName.clear();
    m_animations.clear();
}

void Animation::setActiveAnimation(std::string ani)
{
    auto it = m_animations.find(ani);
    if (it != m_animations.end())
    {
        m_frames = it->second;

        auto itr = find(m_aniName.begin(), m_aniName.end(), ani);
        if (itr != m_aniName.end())
        {
            m_active = itr - m_aniName.begin();

        }

    }
}

void Animation::addFrame(std::vector<std::pair< sf::IntRect, sf::Time>> rect, const std::string animation)
{
    m_animations.insert(std::pair<std::string, std::vector<std::pair< sf::IntRect, sf::Time>>>(animation,
    rect));
    m_frames = rect;
    m_aniName.push_back(animation);
    m_active = m_animations.size() -1 ;
    //m_frames.push_back(rect);
}

void Animation::setSpriteSheet(const sf::Texture& texture)
```

```
{
    m_texture = &texture;
}

const sf::Texture* Animation::getSpriteSheet() const
{
    return m_texture;
}

std::size_t Animation::getSize() const
{
    return m_frames.size();
}

const std::pair< sf::IntRect, sf::Time >& Animation::getFrame(std::size_t n) const
{
    if (n < m_frames.size())
    {
        return m_frames[n];

    }
    else
    {
        std::pair< sf::IntRect, sf::Time > t;
        t.first = { 0, 0, 0, 0 };
        t.second = sf::seconds(0.f);
        return t ;
    }
}

const std::vector<std::string>* Animation::GetNames()
{
    return &m_aniName;
}


const std::string Animation::getActiveAnimation()
{
    if (m_active != -1)
    {
        return m_aniName[m_active];
    }
    else
    {
        return "";
    }
}
```