# Animation

Player.h

|  |
| --- |
| #pragma once  #include <SFML/Graphics.hpp>  #include "Animation.h"  class Player  {    public:  enum class PlayerAnimationState { walk, run, jump, idle };  PlayerAnimationState animationState = PlayerAnimationState::walk;  Animation animations[5];  AnimationSheet animSheet;  sf::Vector2f position;  sf::Sprite& spriteSheet;    Player (sf::Sprite& spheet) : spriteSheet(spheet)  {    }        void InitAnimationData()  {  animations[int(PlayerAnimationState::walk)] = { 8, 36,10,true };  animations[int(PlayerAnimationState::run)] = { 15,20,100,true };  animations[int(PlayerAnimationState::jump)] = { 4,0,40,false };  animations[int(PlayerAnimationState::idle)] = { 2,300,80,true };  animationState = PlayerAnimationState::walk;  animSheet.Init(spriteSheet.getGlobalBounds().width, spriteSheet.getGlobalBounds().height, 5, 9);  }  void startAnimaton(PlayerAnimationState \_animationState)  {  animationState = \_animationState;  animSheet.StartAnimation(animations[int(animationState)]);  }  void Draw(sf::RenderWindow& win)  {  animSheet.nextFrame();  spriteSheet.setTextureRect(animSheet.GetFrame());  spriteSheet.setPosition(100, 100);  win.draw(spriteSheet);  }  void Update(double dt)  {  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))  {  if (animationState != PlayerAnimationState::walk)  {  startAnimaton(PlayerAnimationState::walk);  }  }  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Space))  {  if (animationState !=PlayerAnimationState::jump)  {  startAnimaton(PlayerAnimationState::jump);  }  }  }  }; |

Animation.h

|  |
| --- |
| #pragma once  #include <iostream>  class Animation {  public:  int numframes;  int startOffset;  int speed;  bool loop;    };  class AnimationSheet  {  public:    int frameCounter;  sf::Vector2i frameSize{ };  int numRows = 0;  int numCols = 0;  Animation curAnimation;  int curFrameOffset = 0;    void Init(int \_pixelWidth, int \_pixelHeight, int \_numRows, int \_numCols)  {    numCols = \_numCols;  numRows = \_numRows;  frameSize.x = \_pixelWidth / numCols;  frameSize.y = \_pixelHeight / numRows;  curFrameOffset = 0;  }  sf::IntRect GetFrame()  {  int row = curFrameOffset / numCols;  int col = curFrameOffset % numCols;  sf::IntRect frameRectangle;  frameRectangle.left = col \* frameSize.x;  frameRectangle.top = row \* frameSize.y;  frameRectangle.width = frameSize.x;  frameRectangle.height = frameSize.y;  return frameRectangle;  }  void StartAnimation(Animation curAnimation)  {  this->curAnimation = curAnimation;  frameCounter = 0;  curFrameOffset = curAnimation.startOffset;  }  void nextFrame()  {  if (frameCounter > curAnimation.speed)  {    frameCounter = 0;  if (curFrameOffset >= curAnimation.startOffset + curAnimation.numframes - 1)  {  if (curAnimation.loop == true)  {  curFrameOffset = curAnimation.startOffset;  }  else {  curFrameOffset = curAnimation.startOffset + curAnimation.numframes - 1;  }  }  else {  std::cout << curFrameOffset << std::endl;  curFrameOffset++;  }  }  else {  frameCounter++;  }  }  }; |

Game.cpp

|  |
| --- |
| #include "Game.h"  #include <iostream>  #include <stdlib.h>  #include <time.h>  // Updates per milliseconds  static double const MS\_PER\_UPDATE = 10.0;  ////////////////////////////////////////////////////////////  Game::Game()  : m\_window(sf::VideoMode(ScreenSize::s\_width, ScreenSize::s\_height, 32), "SFML Playground", sf::Style::Default),  player (playerSpriteSheet)    {    if (!playerTextureSheet.loadFromFile("character\_robot\_sheet.png"))  {  // error...  }    playerSpriteSheet.setTexture(playerTextureSheet);      m\_window.setVerticalSyncEnabled(true);  starField.Init(m\_window);  player.InitAnimationData();    player.startAnimaton(Player::PlayerAnimationState::walk);    }  ////////////////////////////////////////////////////////////  void Game::run()  {  sf::Clock clock;  sf::Int32 lag = 0;    while (m\_window.isOpen())  {  sf::Time dt = clock.restart();  lag += dt.asMilliseconds();  processEvents();  while (lag > MS\_PER\_UPDATE)  {  update(lag);  lag -= MS\_PER\_UPDATE;  }  update(lag);    render();  }  }  ////////////////////////////////////////////////////////////  void Game::processEvents()  {  sf::Event event;  while (m\_window.pollEvent(event))  {  if (event.type == sf::Event::Closed)  {  m\_window.close();  }  //processGameEvents(event);  }  }  ////////////////////////////////////////////////////////////  void Game::processGameEvents(sf::Event& event)  {    if (sf::Event::KeyPressed == event.type)  {  switch (event.key.code)  {  case sf::Keyboard::Escape:  m\_window.close();  break;    default:  break;  }  }  }  ////////////////////////////////////////////////////////////  void Game::update(double dt)  {  // get keyboard input.  player.Update(dt);  starField.Update(m\_window);  }  ////////////////////////////////////////////////////////////  void Game::render()  {  m\_window.clear(sf::Color(0, 0, 0, 0));      starField.Draw(m\_window);  player.Draw(m\_window);  m\_window.display();  } |

Game.h

|  |
| --- |
| #pragma once  #pragma once  #include <SFML/Graphics.hpp>  #ifdef \_DEBUG  #pragma comment(lib,"sfml-graphics-d.lib")  #pragma comment(lib,"sfml-audio-d.lib")  #pragma comment(lib,"sfml-system-d.lib")  #pragma comment(lib,"sfml-window-d.lib")  #pragma comment(lib,"sfml-network-d.lib")  #else  #pragma comment(lib,"sfml-graphics.lib")  #pragma comment(lib,"sfml-audio.lib")  #pragma comment(lib,"sfml-system.lib")  #pragma comment(lib,"sfml-window.lib")  #pragma comment(lib,"sfml-network.lib")  #endif  #include "ScreenSize.h"  #include <stdlib.h>  #include <time.h>  #include "Starfield.h"  #include "Player.h"  class Game  {  public:  Game();  Starfield starField;    void run();  protected:      void update(double dt);    void render();    void processEvents();  /// <summary>  /// @brief Handles all user input.  /// </summary>  /// <param name="event">system event</param>  void processGameEvents(sf::Event&);    // main window  sf::RenderWindow m\_window;  Player player;  sf::Texture playerTextureSheet;  sf::Sprite playerSpriteSheet;  }; |