CMP105 Games Programming

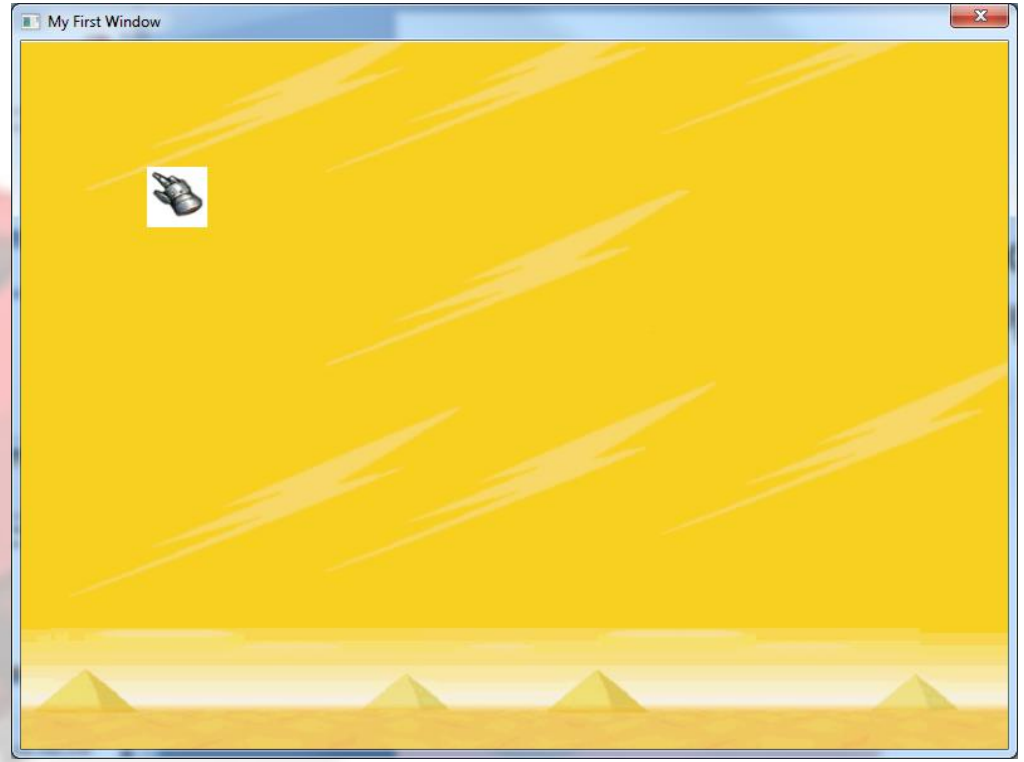
Animated Sprites

This week

- Transparency
- Sprite sheets
- Animated sprites

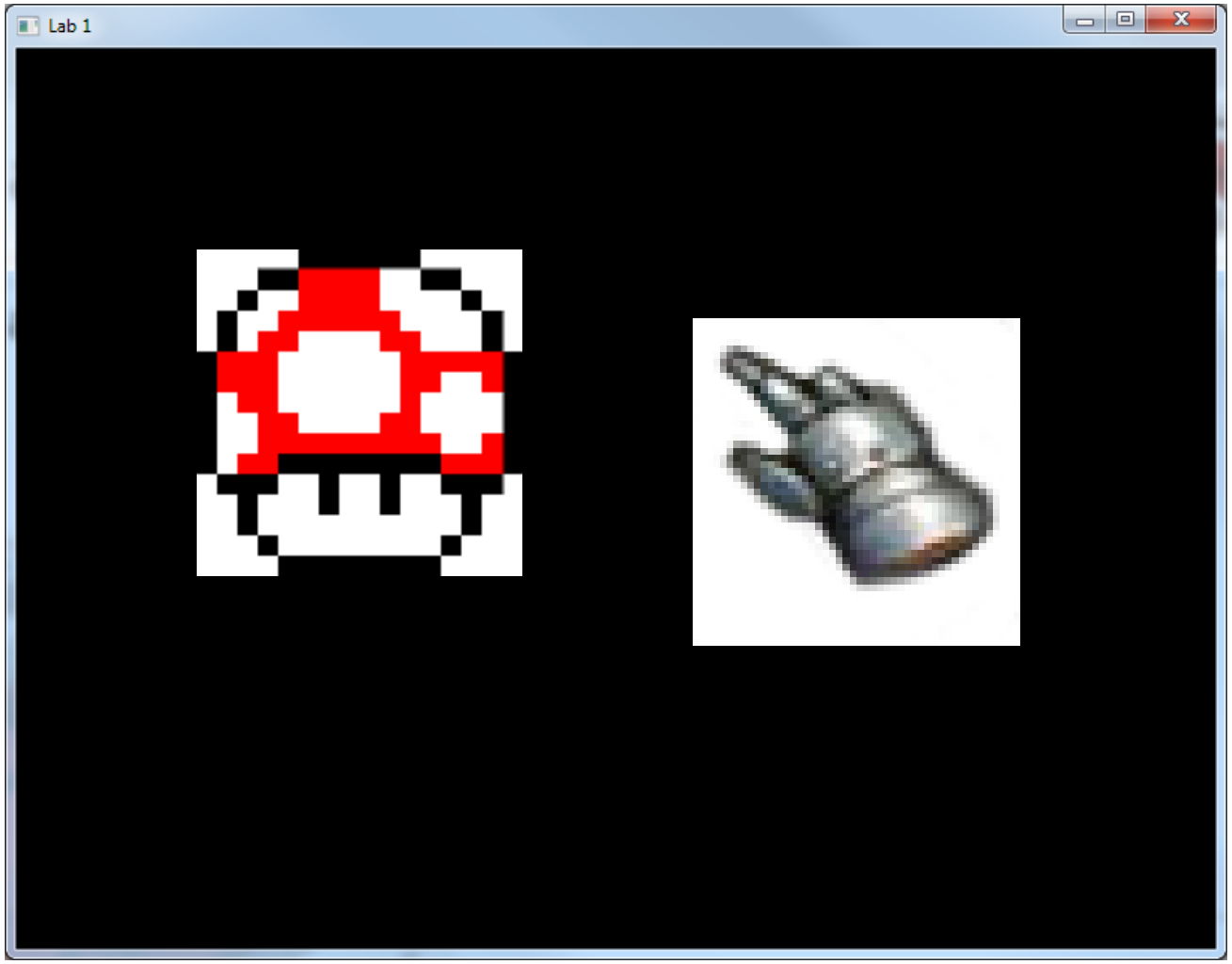
Transparency

- When making a complex scene we are going to have images drawn on top of images
  - This leads to some ... problems



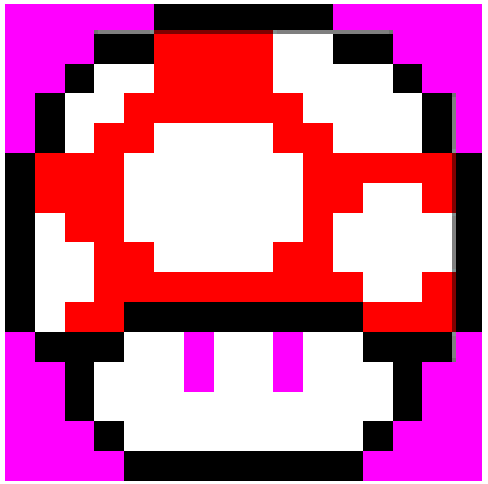
Transparency

- Three techniques
  - Define a transparent colour
  - Use a black and white mask
  - Use alpha channel



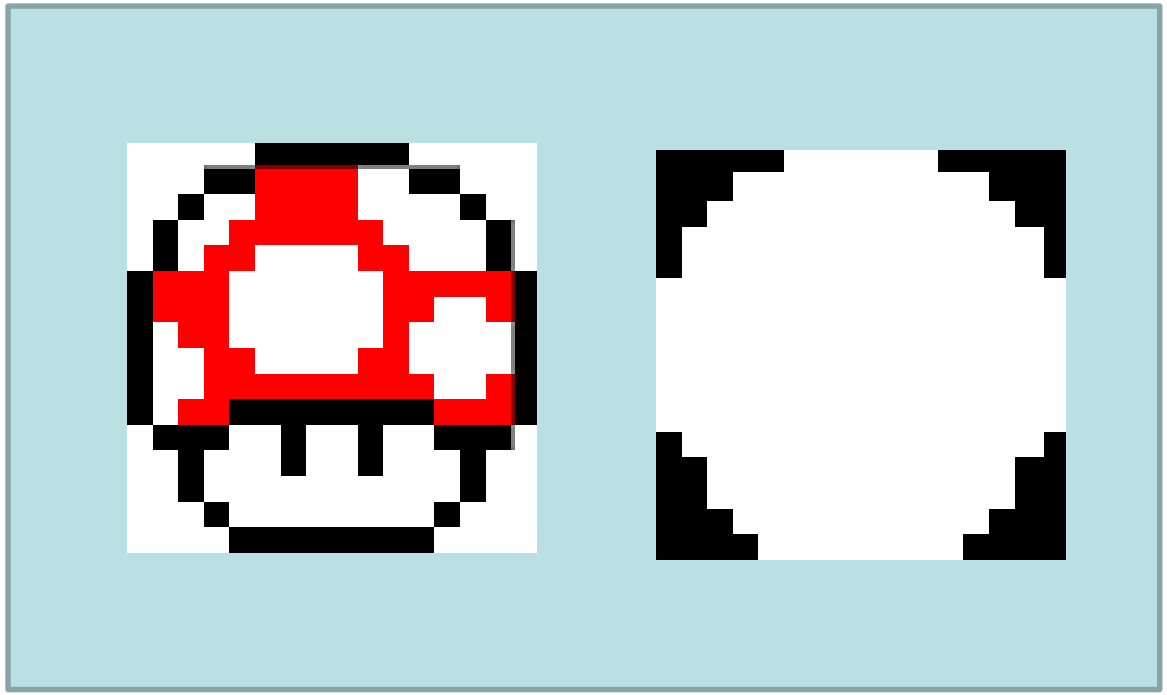
Transparency

- Define a transparent colour
  - This colour is not copied from the image to the back buffer making it transparent
  - Colour choice is important!
  - If all pixels of the colour will not be copied, could break our shape
  - Choose a colour not commonly used
  - Best practice is magenta (255, 0, 255)

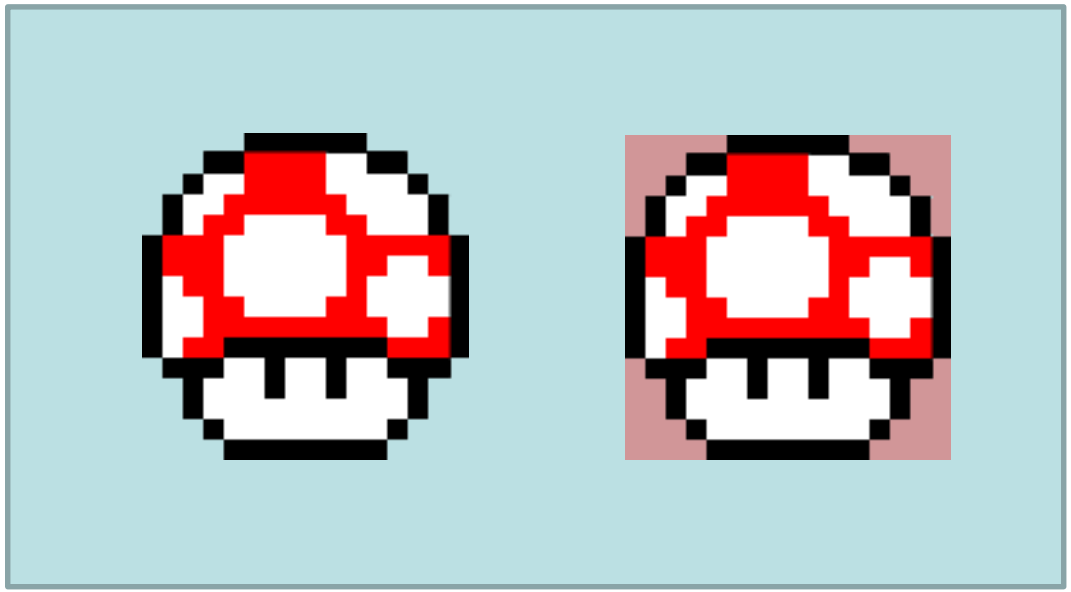


Transparency

- Mask
  - Second image that outlines what pixels should be used
  - Usually a black and white mask
  - Can be built in real-time using a transparency colour (previous)
  - Dis/advantages
    - Requires two images (waste of space)
    - Can use all colours (not likely an issue)



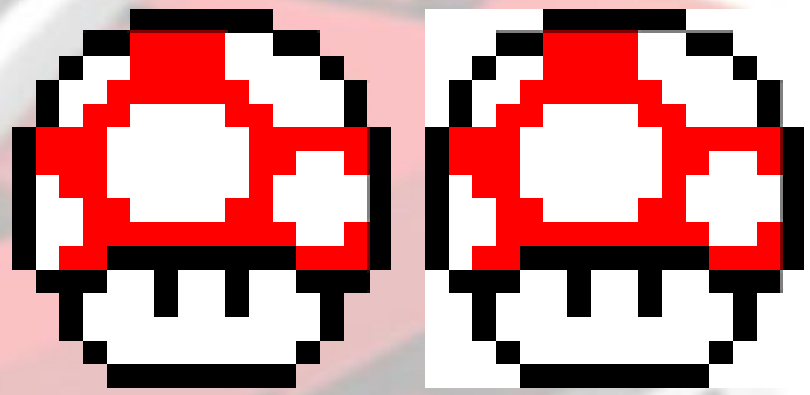
Transparency



- Alpha channel
  - Some images types support an alpha channel
  - Colour is defined as RGBA
    - Red, Green, Blue, Alpha
  - Supported by image formats
    - PNG and GIF
  - Not supported by
    - BMP or JPG
  - Big advantage, supported gradient transparency
    - Not just on or off transparency like previous techniques

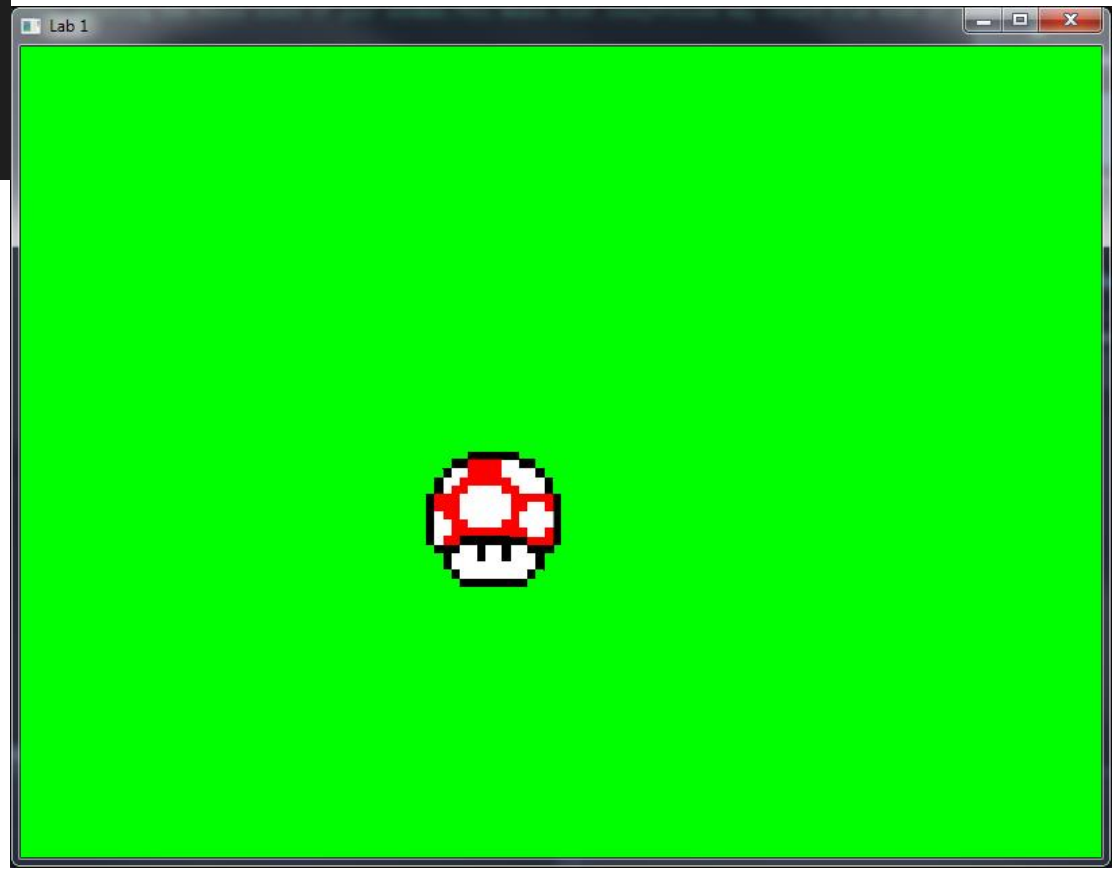
Code / Example

- Easy option
  - Load image as normal
  - Image must have transparent colour
  - Best option PNG
  - MushroomTrans.png



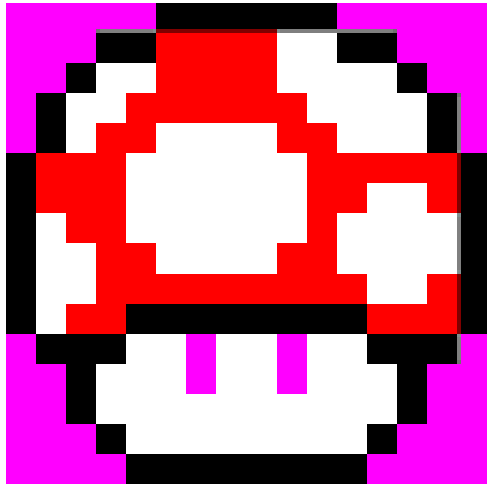
Code / Example

```
// Transparent sprite
m_mushroomTexture.loadFromFile("gfx/MushroomTrans.png");
m_mushroom.setTexture(&m_mushroomTexture);
m_mushroom.setSize(sf::Vector2f(100, 100));
m_mushroom.setPosition(300, 300);
```



Code / Example

- Colour mask
  - Can be done, not recommend
  - Sprite with colour
  - Mask colour
  - Make texture



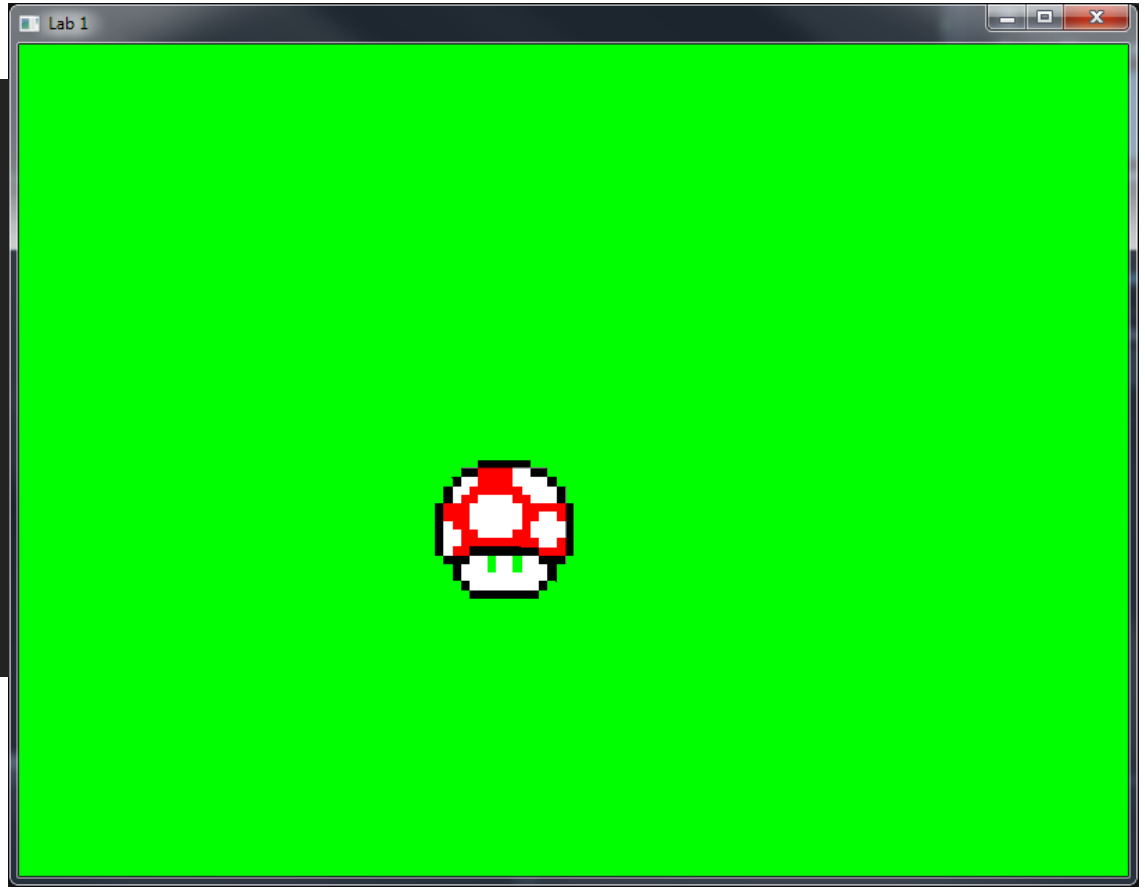
Code / Example

```
    sf::Image mask;
    mask.loadFromFile("gfx/MushroomMask.png");
    mask.createMaskFromColor(sf::Color::Magenta);

    m_mushroomTexture2.loadFromImage(mask);

    m_mushroom.setTexture(&m_mushroomTexture2);
    m_mushroom.setSize(sf::Vector2f(100, 100));
    m_mushroom.setPosition(300, 300);

}
```



Animated sprites

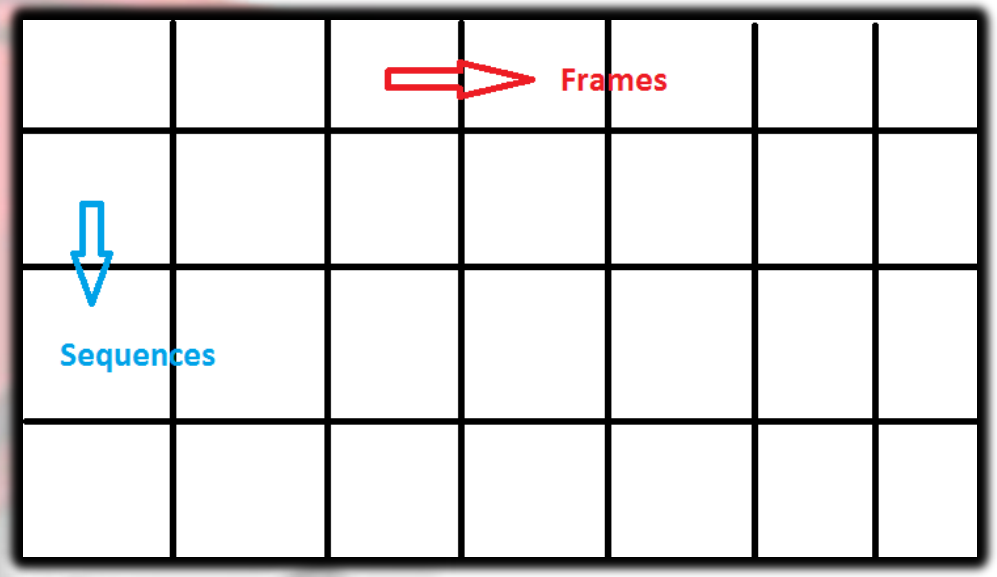
- Sprite animation is the electronic version of Cel Animation (traditional animation)
  - Where a transparent sheet had images painted and/or drawn on it
  - A sequence of cels was then used to create the illusion of motion, where you didn't have to re-draw the rest of the image
- Widely used in 2D games
  - Check out the Metal Slug series
- Providing characters with a collection of looping animation cycles
  - Idle
  - Walking
  - Running
  - etc

Animation

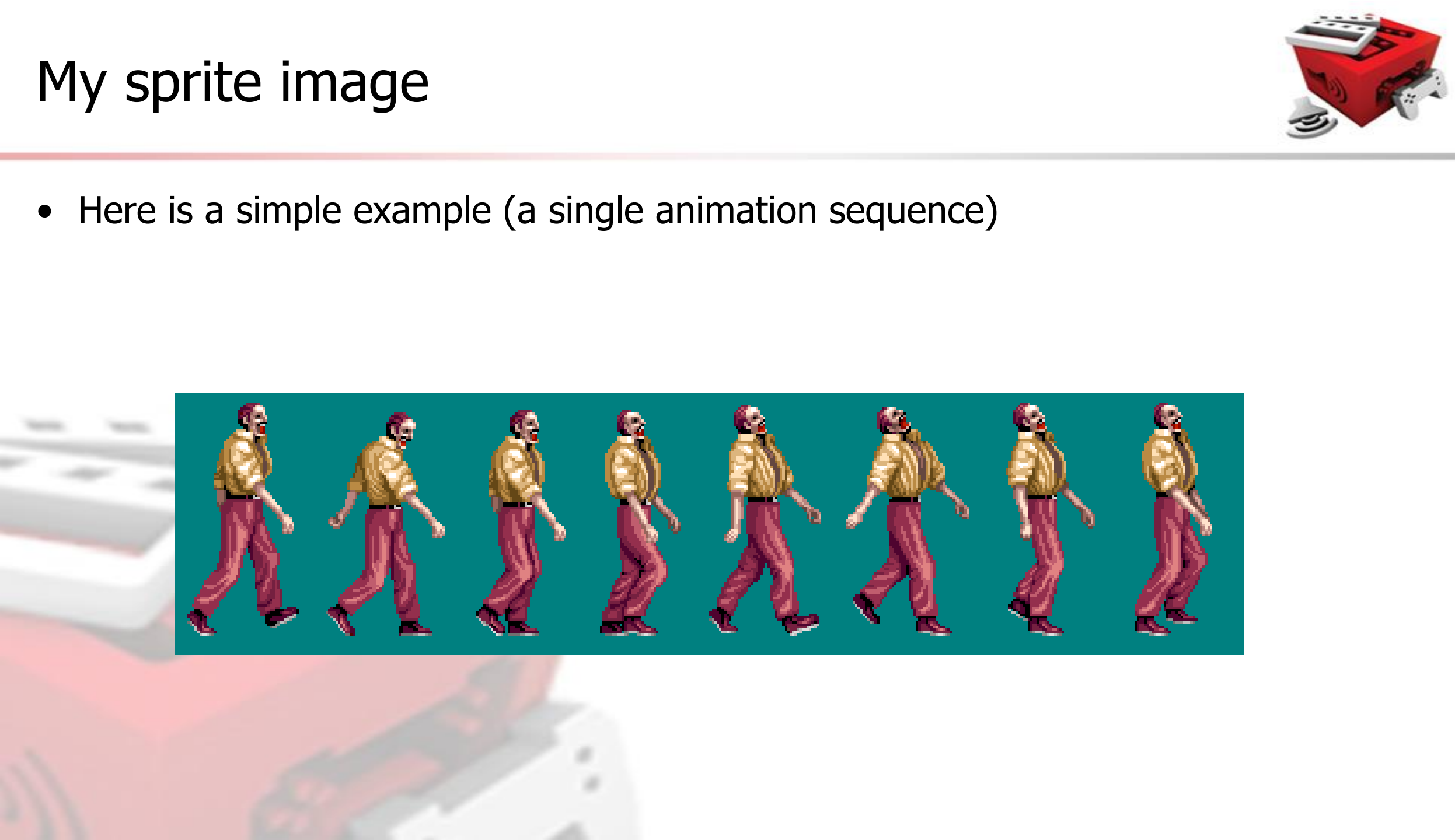
- Previously animation swapping was done by a frame count
  - Once X number of frames has past, next frame of animation
  - With poor or high FPS this would slow down / speed up the animation
- Solution
  - Accumulate delta time every frame
  - Once enough time has past, next frame

Sprite sheet

- A sprite sheet contains the sprite images used for animation
  - Both in the x and y direction
  - Commonly going from left to right defined one animation sequence
  - Where going down defined each separate animation
- This is not always the case
  - To make sprite sheets smaller, many short sequences can go on one line



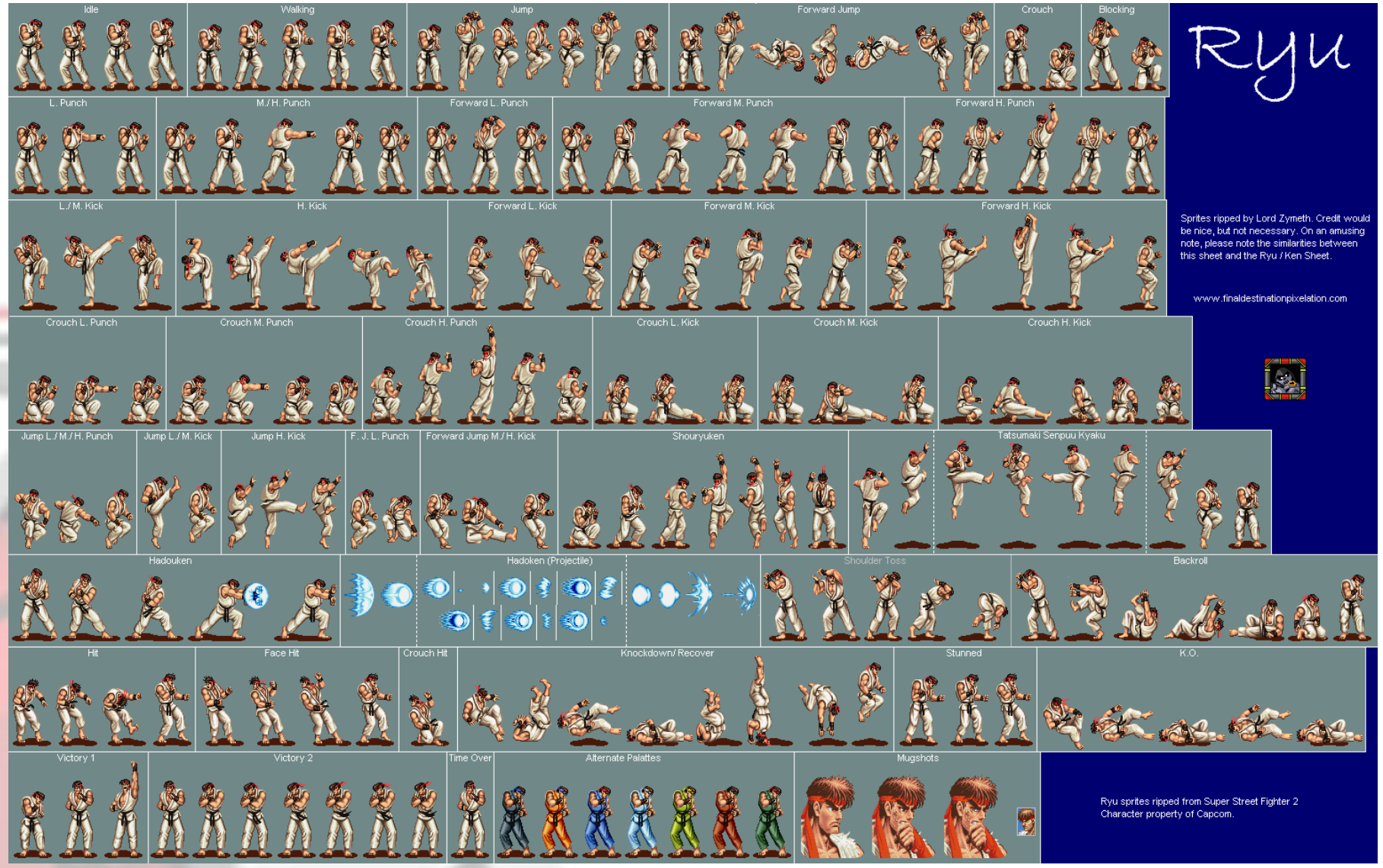
My sprite image



- Here is a simple example (a single animation sequence)

Example sprite sheet

Example sprite sheet



Sprite animation

- Render a sub-section of the image (frame)
- After set amount of time, render next frame
- If, past last frame, jump to start frame

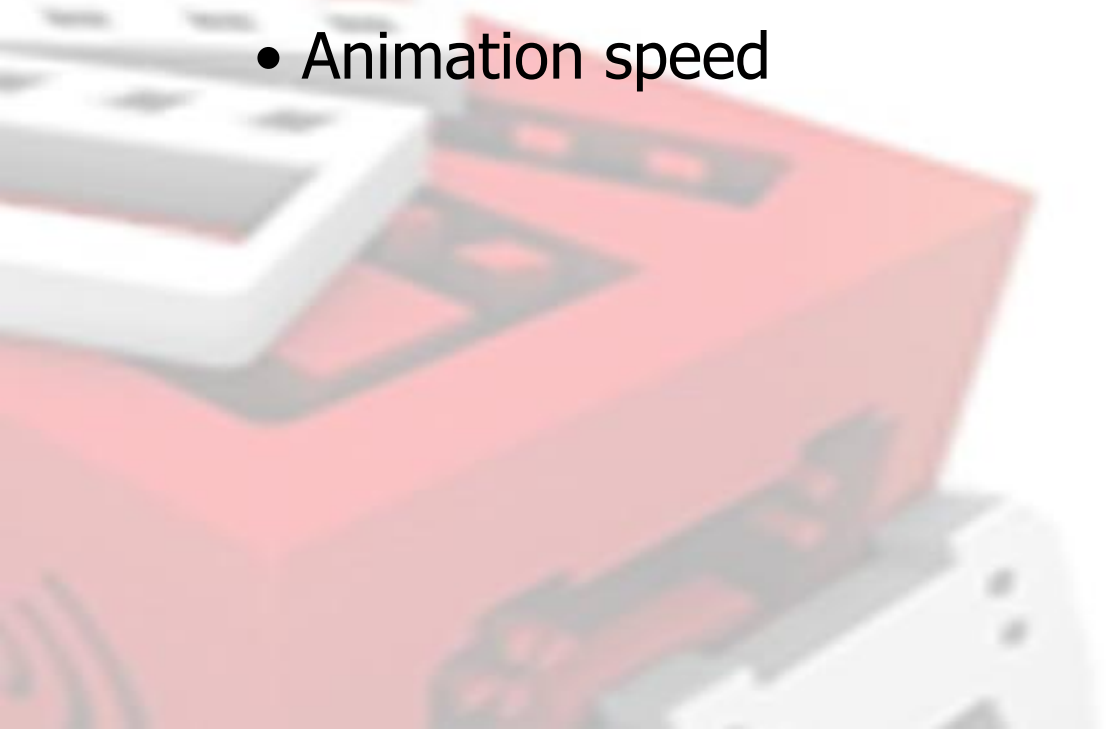
Example

- Simple, single, looping sprite animation
  - All frames are the same size
  - Best case scenario

- Need
  - Class to handle animated sprite
  - Sprite sheet

Animated sprite class

- Animated Sprite class
  - Inherits from sprite
  - Added variable/functions for animation
    - Including frame size
    - Animation speed



AnimatedSprite.h

```h
#pragma once
#include "Sprite.h"

class AnimatedSprite : public Sprite
{
public:
    AnimatedSprite(const sf::Vector2f &size = sf::Vector2f(0, 0));
    ~AnimatedSprite();

    void update(float dt);

    void setFrameSize(int width, int height);
    int getFrameWidth() { return frame.width; };
    int getFrameHeight() { return frame.height; };
    void setAnimationSpeed(float aspeed);

protected:
    // variable for controlling animation
    float frameTime;
    float elapsedTime;
    float animationSpeed;
    sf::IntRect frame;

};
```

AnimatedSprite.cpp

```cpp
#include "AnimatedSprite.h"

AnimatedSprite::AnimatedSprite(const sf::Vector2f &size) : Sprite(size)
{
    // Configure default values
    elapsedTime = 0.f;
    animationSpeed = 1.0f;
    frame = sf::IntRect(0, 0, 0, 0);
}

AnimatedSprite::~AnimatedSprite()
{}

void AnimatedSprite::update(float dt)
{
    // increment time
    elapsedTime += dt;

    // if enough time has passed move onto next frame
    if (elapsedTime >= animationSpeed)
    {
        frame.left += frame.width;
        // check if we have passed last frame of animation
        if (frame.left > getTexture()->getSize().x - frame.width)
        {
            frame.left = 0;
        }
        setTextureRect(frame);
        // reset timer
        elapsedTime = 0;

    }
}
```

AnimatedSprite.cpp

```
void AnimatedSprite::setAnimationSpeed(float speed)
{
    animationSpeed = speed;
}

void AnimatedSprite::setFrameSize(int width, int height)
{
    frame.width = width;
    frame.height = height;
    setTextureRect(frame);
}
```

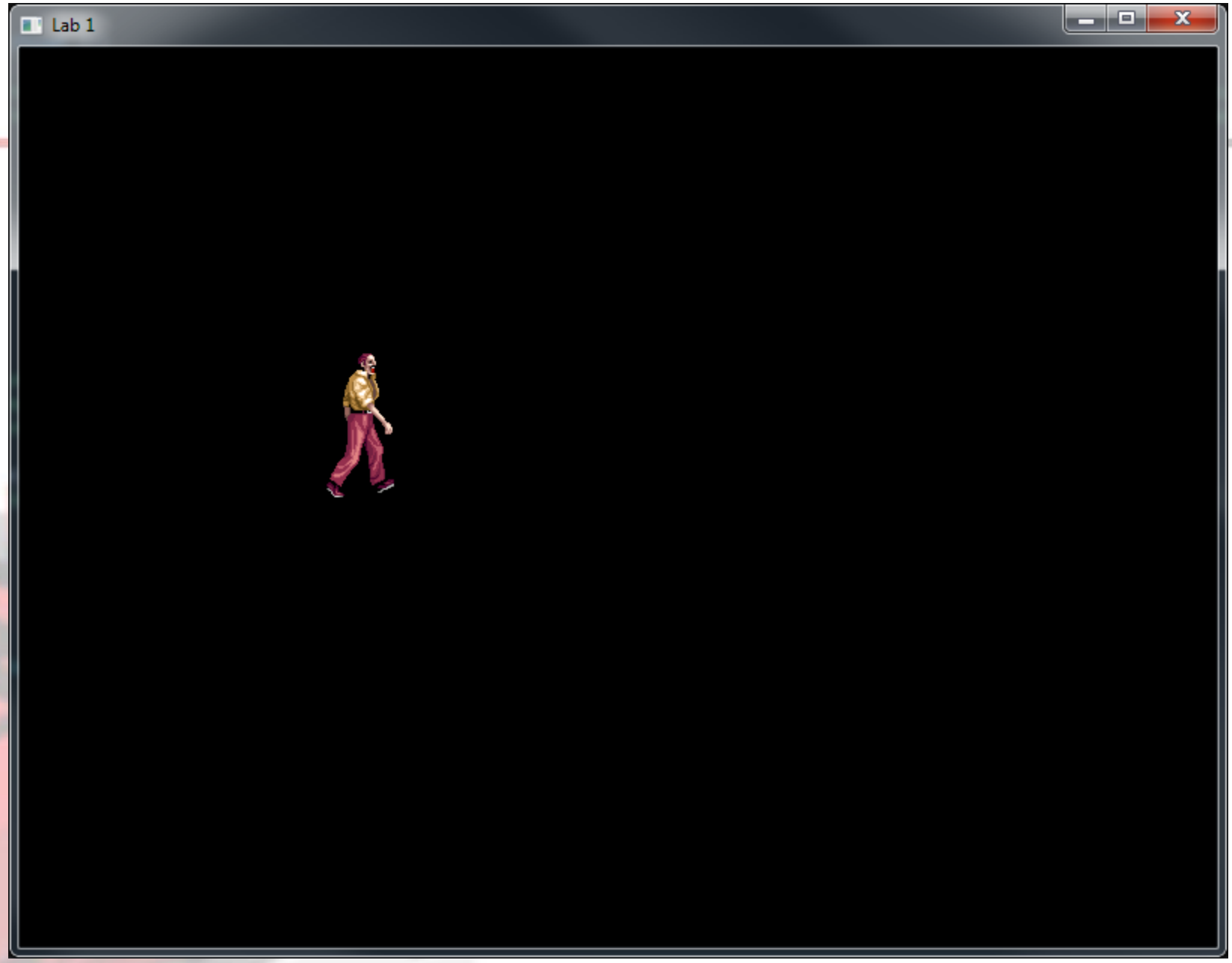
Sprite setup

- Zombie animated sprite
- Texture
  - Zombie sprite sheet

```
AnimatedSprite zombie;
sf::Texture zombTexture;
```

- Size match or ratio of frame size
  - Otherwise image will stretch/shrink/distort
- Set up frame size
- Set up animation speed
  - Amount of time to pass before moving onto next frame

```
zombTexture.loadFromFile("gfx/animZombie.png");
zombie.setSize(sf::Vector2f(55, 108));
zombie.setPosition(200, 200);
zombie.setTexture(&zombTexture);
zombie.setFrameSize(55, 108);
zombie.setAnimationSpeed(0.1);
```


Lab 1


Lab 1

Multiple animations

- A little tricky
- Not every animation will have the same
  - Speed
  - Number of frames
  - Size
- Create a class to store all the animation specific information
  - Avoids hard coding everything
  - Could expand on this to create a generic class for handling animation

Animation class

- What do we need to store?
  - Number of frames
  - Current frame of animation
  - Frame dimensions
  - Animation speed (how long between frames)

```
//Animation.h
#pragma once
#include "SFML\Graphics.hpp"

class Animation
{
public:
        Animation();
        ~Animation();

        void init(int left, int top, int width, int height, int frames, float speed);

        sf::IntRect getCurrentFrame();
        float getAnimationTime() { return animationSpeed; };
        void nextFrame();
        // could add functionality for play, pause, stop and loop

protected:
        sf::IntRect frame;
        int numberOfFrames;
        int currentFrame;
        float animationSpeed;
};
```

```cpp
// Animation.cpp
#include "Animation.h"

Animation::Animation()
{
        currentFrame = 0;
}
Animation::~Animation()
{}

void Animation::init(int left, int top, int width, int height, int numFrames, float speed)
{
        frame.left = left;
        frame.top = top;
        frame.width = width;
        frame.height = height;
        numberOfFrames = numFrames;
        animationSpeed = speed;
}

sf::IntRect Animation::getCurrentFrame()
{
        sf::IntRect temp;
        temp = sf::IntRect(frame.width * currentFrame, frame.top, frame.width, frame.height);
        return temp;
}
```

Animation.cpp

```
void Animation::nextFrame()
{
	currentFrame++;
	if (currentFrame >= numberOfFrames)
	{
		currentFrame = 0;
	}
}
```

New animation class

- New class, inheriting from AnimatedSprite
  - Each animation has a variable to describe it
- During the initialise the animations are configured

```
protected:
    Animation walk;
    Animation swim;
    Animation duck;
    Animation* currentAnimation;
```

```
Mario::Mario(const sf::Vector2f &size) : AnimatedSprite(size)
{
    walk.init(0, 0, 15, 21, 4, 0.3f);
    swim.init(0, 21, 16, 20, 3, 0.2f);
    duck.init(0, 41, 16, 20, 2, 1.f);
    currentAnimation = &walk;
    frame = currentAnimation->getCurrentFrame();
    setTextureRect(frame);
}
```

New animation class

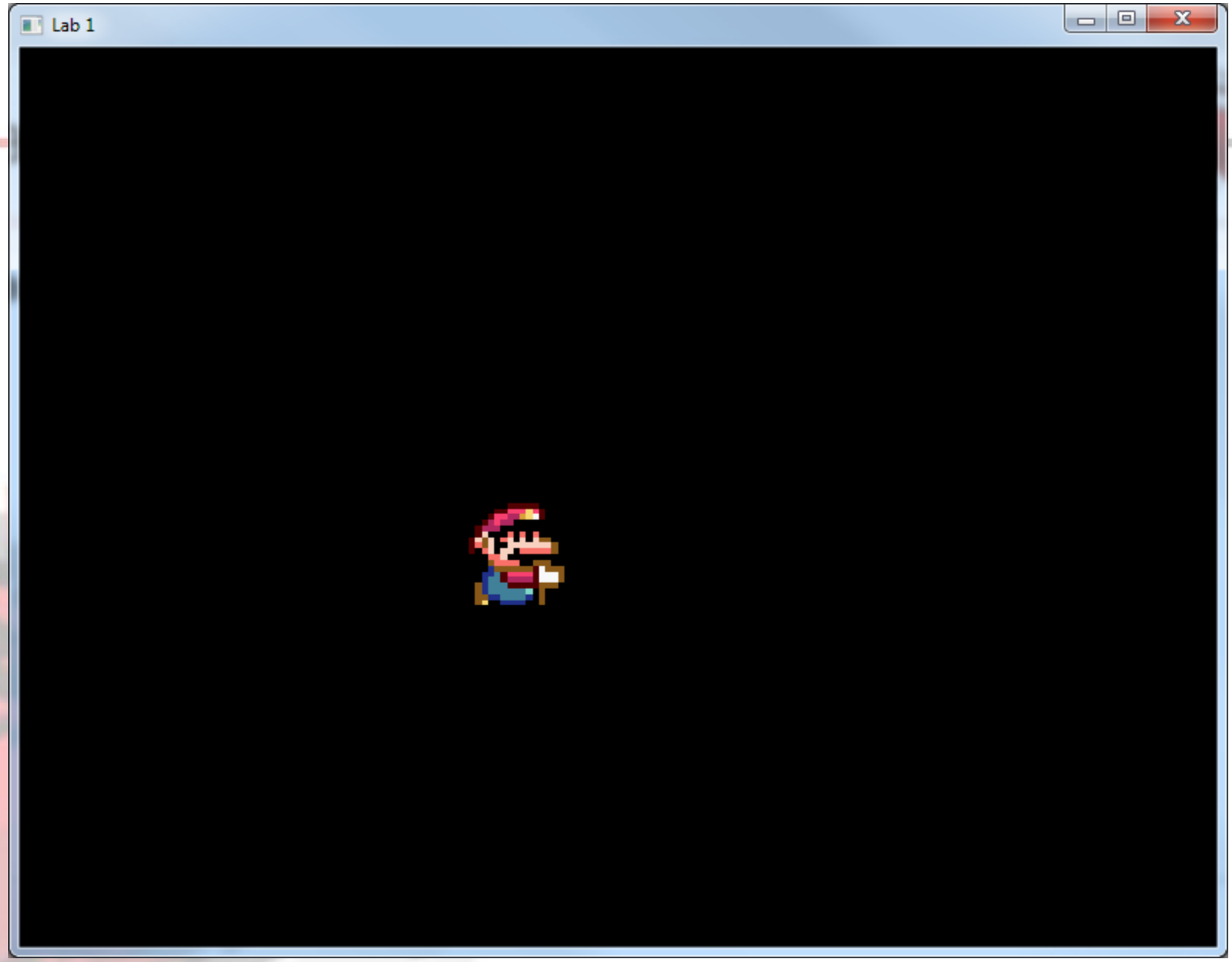
- Update() modified to handle new animation system
  - Not a major update
  - Uses current animation pointer
  - When we want to change the animation, we update the pointer

```
void Mario::update(float dt)
{
    elapsedTime += dt;
    if (elapsedTime >= currentAnimation->getAnimationTime())
    {
        // next frame
        currentAnimation->nextFrame();
        setTextureRect(currentAnimation->getCurrentFrame());
        elapsedTime = 0;
    }
```

Sprite setup

- Similar as before
  - Load the sprite sheet
  - Mario is a modified class (to handle multiple animations)

```
marioTexture.loadFromFile("gfx/MarioSheetT.png");
mario.setSize(sf::Vector2f(64, 80));
mario.setPosition(300, 300);
mario.setTexture(&marioTexture);
mario.setInput(input);
```

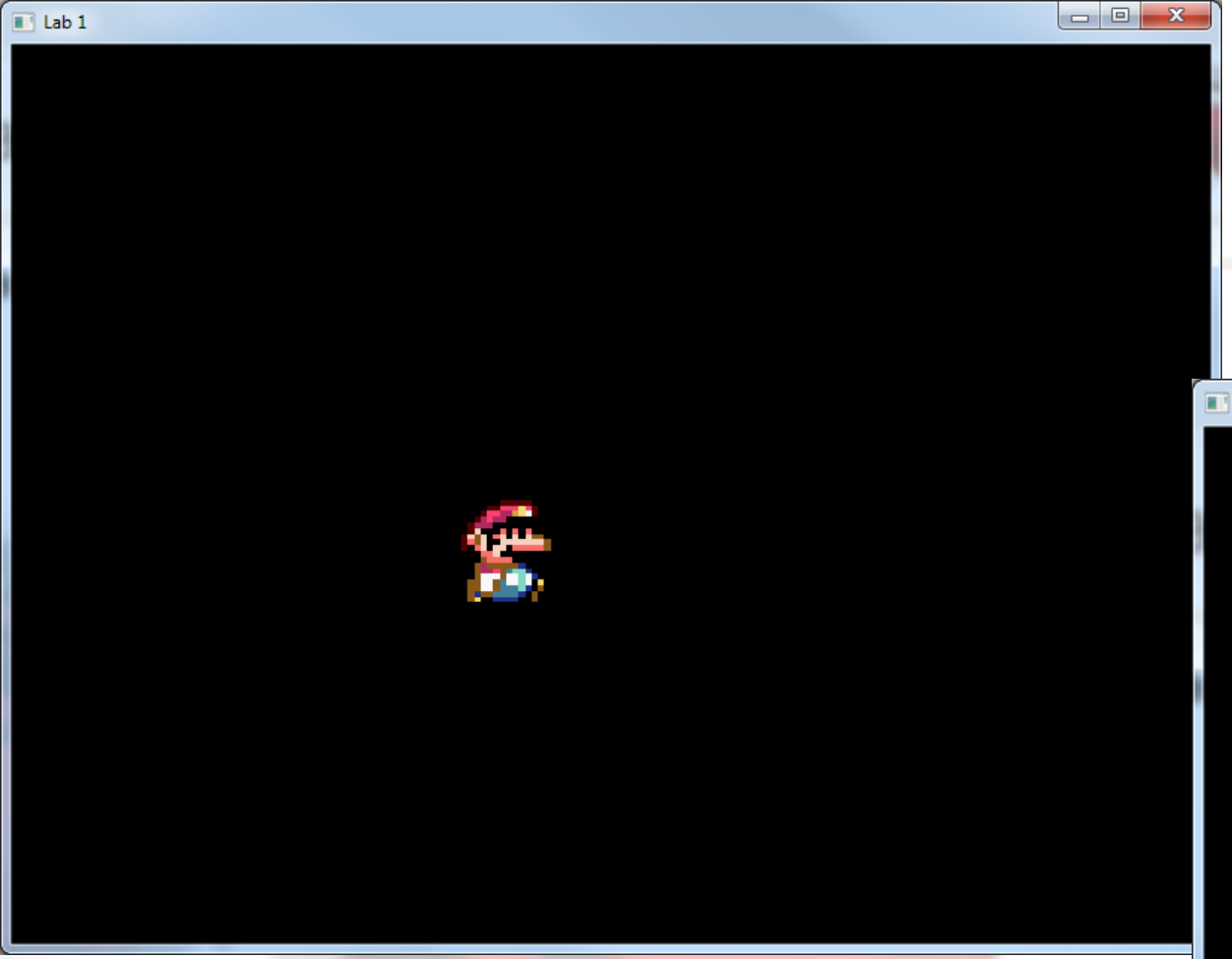

Lab 1

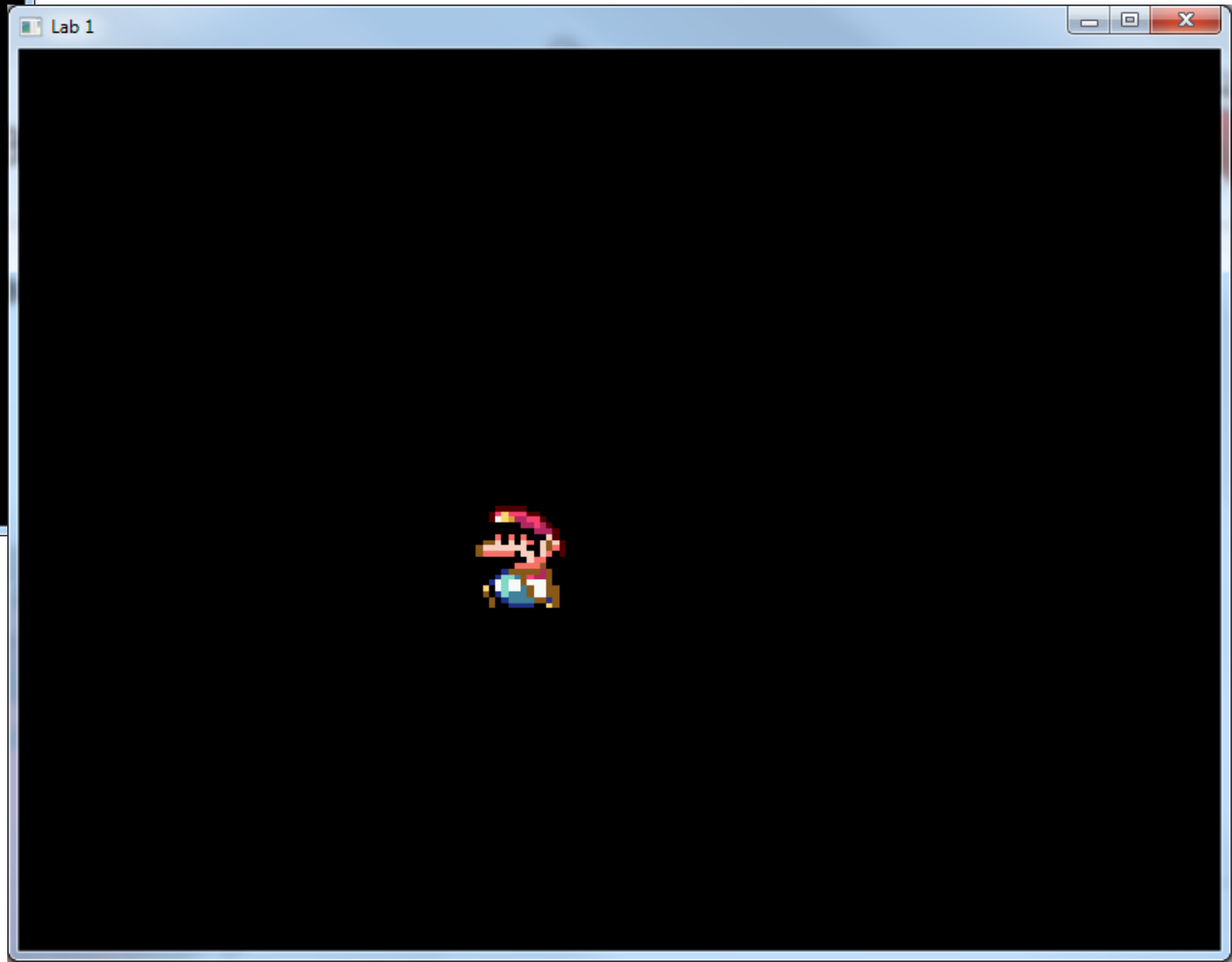
Live demo

- Sweet stolen Mario sprites!

Flipping aniamtion

- Did you notice all the characters faced to the right
  - How do they walk left?
  - Moon walk?
- In code we can flip the animation
  - No need to double sprite sheet storage
- The sf::IntRect that describes the frame to be render can be inverted
  - Instead of 0, 0, 55, 108
  - 55, 0, -55, 108


Lab 1


Lab 1

In the labs

- Building animated sprite class
- Making animated sprites

- Last week I linked a few sprite resources
  - Use them to get sprite sheets (may require some modification)
- Or create your own
  - Helpful tool: http://www.piskelapp.com/

Life of a programmer