

Новосибирский государственный технический университет
Кафедра вычислительной техники



**Пояснительная записка к
Расчётно-графической работе.**

Вариант 12

**«Использование графической библиотеки
“Graphics.h”»**

Группа: АВТ-716
Студент: Нестеров П.А.

Новосибирск 2017г.

1. Задание

«Шагающий человечек».

2. Основные идеи и методы решения

Нужно создать персонажа, который может передвигаться, при этом используется анимация передвижения. В моем случае используется Простая и быстрая Мультимедиа Библиотека (SFML).

2.1. Анимация передвижения

Для реализации анимации передвижения персонажа, используется изображение готового персонажа 256x384.

(картинка должна лежать в папке проекта)

```
image.loadFromFile("image.png"); //путь к картинке с персонажем
texture.loadFromImage(image); //преобразование в текстуру
sprite.setTexture(texture); //заключение текстуры в прямоугольник
sprite.setPosition(50,50); // радиус прямоугольника
```



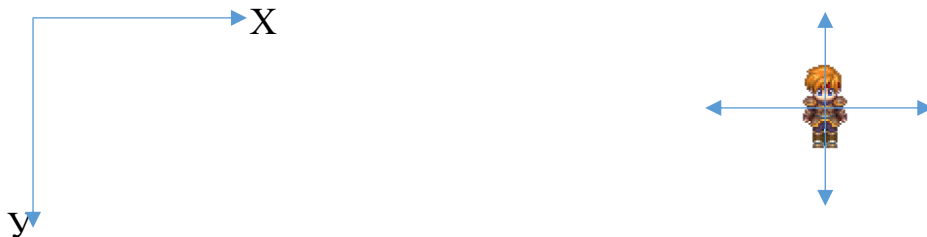
Чтобы не выводить на экран такую картинку, нужно вывести часть этой картинки. Картинка состоит из 4 кусков по вертикали и по горизонтали. Для этого нужно 256 разделить на 4 и 384 разделить на 4. В результате будет 64 и 96. Это разрешение для одной части персонажа.

```
sprite.setTextureRect(sf::IntRect(0,0,64,96)); //вырез из картинки
```



Таким методом используются все остальные части передвижения. Передвижение персонажа зависит от нажатия клавиши.

Координатная плоскость окна задана таким образом:



Для того чтобы персонаж шел в лево нужно от X отнять заданную величину (например: 0.1).

`float speed = 0.1; //скорость персонажа`

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) //нажатие на кнопку влево
{
    sprite.move(-speed,0);
}
```

Чтобы использовать анимацию идти в лево – используются координаты второй строки (по Y: от 64 до 128; по X: от 0 до 256)



```
sprite.setTextureRect(sf::IntRect(int(CurrentFrame)*64,96,64,96)); //вырез из картинки
```

Изменение анимации передвижения зависит от процессора ПК. Т.К процессоры разные программа может работать некорректно. Для этого передвижение нужно привязать к времени.

Создаем таймер:

```
sf::Clock clock; // добавление временного таймера
```

И добавляем в цикл строки для зап времени

```
//засекаем время:
```

```
float time = clock.getElapsedTime().asMicroseconds();
```

```
clock.restart();
```

```
time = time / 800;
```

Поэтому изменяем след строки:

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) //нажатие на кнопку влево
```

```
{
```

```
HeroDirection = 1;
```

```
CurrentFrame += 0.005*time; //счетчик
```

```
std::cout << CurrentFrame << std::endl; //вывод для консоли
```

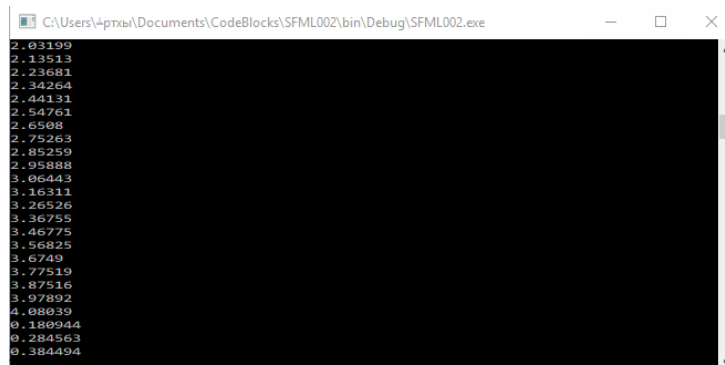
```
if(CurrentFrame>4) CurrentFrame -= 4; //для движения персонажа
```

```
sprite.setTextureRect(sf::IntRect(int(CurrentFrame)*64,96,64,96)); //вырез из картинки
```

```
sprite.move(-speed*time,0); //присоединение движ к времени
```

```
}
```

После это анимация будет зависеть от времени. В консоли будут отображаться следующее:



```
C:\Users\4ptx\Documents\CodeBlocks\SFML002\bin\Debug\SFML002.exe
2.03199
2.13513
2.23681
2.34264
2.44131
2.54761
2.6508
2.75263
2.85259
2.95888
3.06443
3.16311
3.26526
3.36755
3.46775
3.56825
3.6749
3.77519
3.87516
3.97892
4.08039
0.180944
0.284563
0.384494
```

Анимация изменяется только тогда, когда изменяется целое число.

2.2. Препятствия при передвижении

Чтобы персонажу не обитал в пустом пространстве, а мог с чем-нибудь взаимодействовать, то можно ввести след фигуры:

- Стена в виде прямоугольника;
- Круг, который можно толкать;
- Круг, который исчезает при взаим. с персонажем;
- Квадрат в виде поверхности (лед);
- Прямоугольники для телепортации.

Для прорисовки фигур следует написать следующие строки:

```
sf::CircleShape circle(50,4); //задаем квадрат
sf::RectangleShape rectangle(sf::Vector2f(150,70)); // твердый прямоугольник
sf::RectangleShape rectangleTele(sf::Vector2f(10,70)); //телепортирующая фигура
sf::RectangleShape rectangleTele2(sf::Vector2f(10,70)); //2-я фигура для телепорта
sf::CircleShape circle1(25,10); //исчезающий круг
sf::CircleShape circle2(20,30); //двигающийся круг
```

Цвет и размер задаются следующие строки:

```
circle.setFillColor(sf::Color::Yellow); //цвет квадрата
circle.setPosition(200,350); //радиус квадрата
circle.rotate(45); // поворот фигуры (если убрать, то будет ромб)
```

```
rectangle.setFillColor(sf::Color::Cyan); // цвет тверд. прямоугольника
rectangle.setPosition(200,200); //радиус прямоугольника
```

```
circle1.setFillColor(sf::Color::Yellow); //цвет исч. круга
circle1.setPosition(300,400); //радиус круга
```

```
circle2.setFillColor(sf::Color::Red); //цвет двиг. круга
circle2.setPosition(500,200); //радиус круга
```

```
rectangleTele.setFillColor(sf::Color::Green); //телепортирующая фигура
rectangleTele.setPosition(100,100); // радиус
```

```
rectangleTele2.setFillColor(sf::Color::Green); // 2-я фигура для телепорта
rectangleTele2.setPosition(200,100); // радиус
```

Чтобы использовать фигуры, нужно изъять их значение в `getGlobalBounds`:

```
sf::FloatRect spriteBounds = sprite.getGlobalBounds(); // изымаем данные персонажа
sf::FloatRect rectangleBounds = rectangle.getGlobalBounds(); // изымаем данные прямоугольника
sf::FloatRect circle1Bounds = circle1.getGlobalBounds(); // изымаем данные исч. круга
sf::FloatRect circle2Bounds = circle2.getGlobalBounds(); // изымаем данные двиг. круга
sf::FloatRect circleBounds = circle.getGlobalBounds(); //изымаем данные квадрата
sf::FloatRect Teleportation = rectangleTele.getGlobalBounds(); //изымаем данные телепорта
```

После этого можно создавать события взаимодействия персонажа и фигур.

2.2.1. Стена в виде прямоугольника

Когда прямоугольник персонажа попадает в пространство прямоугольника, то прямоугольник персонажа выталкивается в противоположную сторону.

//-----взаимодействие с прямоугольником

```
if (spriteBounds.intersects(rectangleBounds))
{
    // std:: cout << "collision"<< std::endl; //(эта строка для проверки)
    if (HeroDirection==0)
        sprite.move(speed*time,0); //отталкивание персонажа от прямоугольника вправо
    if (HeroDirection==2)
        sprite.move(-speed*time,0); //отталкивание персонажа от прямоугольника влево
    if (HeroDirection==3)
        sprite.move(0,speed*time); //отталкивание персонажа от прямоугольника вниз
    if (HeroDirection==4)
        sprite.move(0,-speed*time); //отталкивание персонажа от прямоугольника вверх
}
```

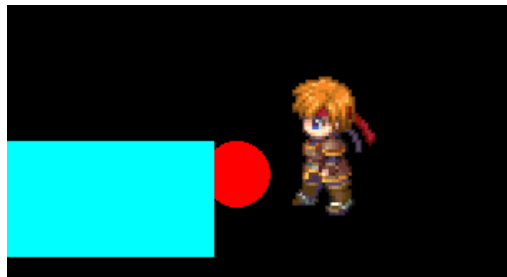
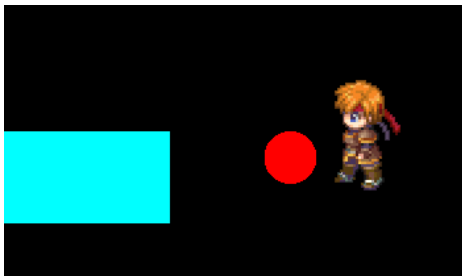
2.2.2. Круг, который можно толкать

Аналогично прямоугольной стене, только наоборот – фигура отталкивается от персонажа.

//-----взаимодействие с двиг. кругом

```
if (spriteBounds.intersects(circle2Bounds))
{
    // std:: cout << "collision"<< std::endl; //(эта строка для проверки)
    if (HeroDirection==0)
        circle2.move(-speed*time,0); // отталкивание фигуры от персонажа влево
    if (HeroDirection==2)
        circle2.move(speed*time,0); // отталкивание фигуры от персонажа вправо
    if (HeroDirection==3)
        circle2.move(0,-speed*time); // отталкивание фигуры от персонажа вверх
    if (HeroDirection==4)
```

```
circle2.move(0,speed*time); // отталкивание фигуры от персонажа вниз
}
```

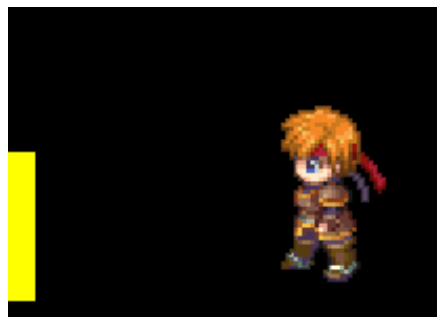
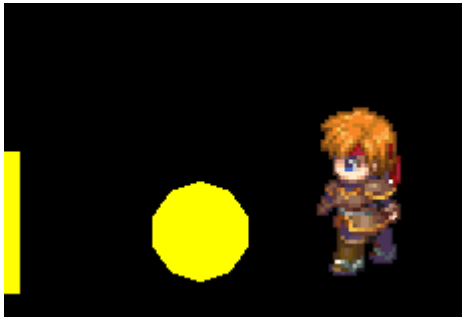


2.2.3. Круг, который исчезает при взаим. с персонажем

Фигура исчезновение при приближении персонажа.

//-----взаимодействие с исч.кругом

```
if (spriteBounds.intersects(circle1Bounds))
circle1.setFillColor(sf::Color::Black); // исчезновение при приближении персонажа
```



2.2.4. Квадрат в виде поверхности (лед)

При взаим. с областью квадрата персонажа перемещает без нажатия на клавишу.

//-----взаимодействие с квадратом

```
if (spriteBounds.intersects(circleBounds))
{
// std:: cout << "collision"<< std::endl; //(эта строка для проверки)
if (HeroDirection==0)
sprite.move(-speed*time,0); //при нажатии влево отталкивание персонажа влево
if (HeroDirection==2)
sprite.move(speed*time,0); //при нажатии вправо отталкивание персонажа вправо
if (HeroDirection==3)
sprite.move(0,-speed*time); //при нажатии вверх отталкивание персонажа вверх
if (HeroDirection==4)
sprite.move(0,speed*time); //при нажатии вниз отталкивание персонажа вниз
}
```

2.2.5. Прямоугольники для телепортации

При взаим. с областью прямоугольника персонажа перемещает на заданную координату.

//-----событие телепорт персонажа

```
if (spriteBounds.intersects(Teleportation))
```

```
    sprite.setPosition(sf::Vector2f(200,100)); //при приближении персонажа - телепортирует его в заданную точку
```

Второй прямоугольник нужен только для декорации!

3. Функциональное описание программы

Программа содержит следующие данные:

```
#include <SFML/Graphics.hpp> //исп библиотеки sfml графики
```

```
#include <cstdlib> //исп библиотеки для исп std
```

```
#include <windows.h> // исп библиотеки для работы в windows
```

```
#include <iostream> //исп стандартной библиотеки
```

```
using namespace sf; // ( это для подстраховки:] )
```

```
sf::RenderWindow window; //задаем окно
```

```
sf::CircleShape circle(50,4); //задаем квадрат
```

```
sf::RectangleShape rectangle(sf::Vector2f(150,70)); // твердый прямоугольник
```

```
sf::RectangleShape rectangleTele(sf::Vector2f(10,70)); //телепортирующая фигура
```

```
sf::RectangleShape rectangleTele2(sf::Vector2f(10,70)); //2-я фигура для телепорта
```

```
sf::CircleShape circle1(25,10); //исчезающий круг
```

```
sf::CircleShape circle2(20,30); //двигающийся круг
```

```
sf::Image image; //имя картинки
```

```
sf::Texture texture; //текстурка
```

```
sf::Sprite sprite;
```

```
float speed = 0.1; //скорость персонажа
```

```
float CurrentFrame = 0;
```

```
int HeroDirection; //персонаж
```

```
int main()
```

```
{
```

```
//создание экрана
```

```
    window.create(sf::VideoMode(640,480),"Walking man");
```

```
    window.setFramerateLimit(60); //кол-во кадр/сек
```



```
//=====
circle.setFill(sf::Color::Yellow); //цвет квадрата
circle.setPosition(200,350); //радиус квадрата
circle.rotate(45); // поворот фигуры (если убрать, то будет ромб)

rectangle.setFill(sf::Color::Cyan); // цвет тверд. прямоугольника
rectangle.setPosition(200,200); //радиус прямоугольника

circle1.setFill(sf::Color::Yellow); //цвет исч. круга
circle1.setPosition(300,400); //радиус круга

circle2.setFill(sf::Color::Red); //цвет двиг. круга
circle2.setPosition(500,200); //радиус круга

//=====
image.loadFromFile("image.png"); //путь к картинке с персонажем
texture.loadFromImage(image); //преобразование в текстуру
sprite.setTexture(texture);
sprite.setPosition(50,50); // радиус прямоугольника
sprite.setTextureRect(sf::IntRect(0,0,64,96)); //вырез из картинку

//=====
rectangleTele.setFill(sf::Color::Green); //телепортирующая фигура
rectangleTele.setPosition(100,100); // радиус

rectangleTele2.setFill(sf::Color::Green); // 2-я фигура для телепорта
rectangleTele2.setPosition(200,100); // радиус

//=====
sf::Clock clock; // добавление временного таймера

while (window.isOpen()) // если открыто окно
{
//то засекаем время:
float time = clock.getElapsedTime().asMicroseconds();
clock.restart();
time = time / 800;

//=====
```

```

//-----событие для окна:
sf::Event event;//проверка открытости окна
while (window.pollEvent(event))
{
//если нажать на кнопку закрыть:
    if (event.type == sf::Event::Closed)
        window.close();
}

//=====
//----события взаимодействия персонажа с объектами
sf::FloatRect spriteBounds = sprite.getGlobalBounds(); // изымаем данные персонажа
sf::FloatRect rectangleBounds = rectangle.getGlobalBounds(); // изымаем данные прямоугольника
sf::FloatRect circle1Bounds = circle1.getGlobalBounds(); // изымаем данные исч.круга
sf::FloatRect circle2Bounds = circle2.getGlobalBounds(); // изымаем данные двиг. круга
sf::FloatRect circleBounds = circle.getGlobalBounds(); //изымаем данные квадрата
sf::FloatRect Teleportation = rectangleTele.getGlobalBounds(); //изымаем данные телепорта

//=====
//-----взаимодействие с прямоугольником
if (spriteBounds.intersects(rectangleBounds))
{
// std:: cout << "collision"<< std::endl; //(эта строка для проверки)
    if (HeroDirection==0)
        sprite.move(speed*time,0); //отталкивание персонажа от прямоугольника вправо
    if (HeroDirection==2)
        sprite.move(-speed*time,0); //отталкивание персонажа от прямоугольника влево
    if (HeroDirection==3)
        sprite.move(0,speed*time); //отталкивание персонажа от прямоугольника вниз
    if (HeroDirection==4)
        sprite.move(0,-speed*time); //отталкивание персонажа от прямоугольника вверх
}

//=====
//-----взаимодействие с исч.кругом
if (spriteBounds.intersects(circle1Bounds))
    circle1.setFill(sf::Color::Black); // исчезновение при приближении персонажа

//=====
//-----взаимодействие с двиг. кругом

```

```

if (spriteBounds.intersects(circle2Bounds))
{
// std:: cout << "collision"<< std::endl; //(эта строка для проверки)
if (HeroDirection==0)
circle2.move(-speed*time,0); // отталкивание фигуры от персонажа влево
if (HeroDirection==2)
circle2.move(speed*time,0); // отталкивание фигуры от персонажа вправо
if (HeroDirection==3)
circle2.move(0,-speed*time); // отталкивание фигуры от персонажа вверх
if (HeroDirection==4)
circle2.move(0,speed*time); // отталкивание фигуры от персонажа вниз
}

//=====
//-----взаимодействие с квадратом
if (spriteBounds.intersects(circleBounds))
{
// std:: cout << "collision"<< std::endl; //(эта строка для проверки)
if (HeroDirection==0)
sprite.move(-speed*time,0); //при нажатии влево отталкивание персонажа влево
if (HeroDirection==2)
sprite.move(speed*time,0); //при нажатии вправо отталкивание персонажа вправо
if (HeroDirection==3)
sprite.move(0,-speed*time); //при нажатии вверх отталкивание персонажа вверх
if (HeroDirection==4)
sprite.move(0,speed*time); //при нажатии вниз отталкивание персонажа вниз
}

//=====
//-----событие телепорт персонажа
if (spriteBounds.intersects(Teleportation))
sprite.setPosition(sf::Vector2f(200,100)); //при приближении персонажа - телепортирует его в
заданную точку

//=====
//-----событие кнопки 1
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) //нажатие на кнопку влево
{
HeroDirection = 0;
CurrentFrame += 0.005*time; //счетчик

```

```

std::cout << CurrentFrame << std::endl; //вывод для консоли
if(CurrentFrame>4) CurrentFrame -= 4; //для движения персонажа
sprite.setTextureRect(sf::IntRect(int(CurrentFrame)*64,96,64,96)); //вырез из картини
sprite.move(-speed*time,0); //просоединение движ к времени
}
//-----событие кнопки 2
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) //нажатие на кнопку вправо
{
    HeroDirection = 2;
    CurrentFrame += 0.005*time; //счетчик
    std::cout << CurrentFrame << std::endl; //вывод для консоли
    if(CurrentFrame>4) CurrentFrame -= 4; //для движения персонажа
    sprite.setTextureRect(sf::IntRect(int(CurrentFrame)*64,192,64,96)); //вырез из картини
    sprite.move(speed*time,0); //просоединение движ к времени
}
//-----событие кнопки 3
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) //нажатие на кнопку вверх
{
    HeroDirection = 3;
    CurrentFrame += 0.005*time; //счетчик
    std::cout << CurrentFrame << std::endl; //вывод для консоли
    if(CurrentFrame>4) CurrentFrame -= 4; //для движения персонажа
    sprite.setTextureRect(sf::IntRect(int(CurrentFrame)*64,288,64,96)); //вырез из картини
    sprite.move(0,-speed*time); //просоединение движ к времени
}
//-----событие кнопки 4
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) //нажатие на кнопку вниз
{
    HeroDirection = 4;
    CurrentFrame += 0.005*time; //счетчик
    std::cout << CurrentFrame << std::endl; //вывод для консоли
    if(CurrentFrame>4) CurrentFrame -= 4; //для движения персонажа
    sprite.setTextureRect(sf::IntRect(int(CurrentFrame)*64,0,64,96)); //вырез из картини
    sprite.move(0,speed*time); //просоединение движ к времени
}

//=====
//----рисование фигур:
window.clear(sf::Color::Black); //заливка фона
window.draw(circle); //заливка квадрата

```

```
window.draw(circle1); //заливка круга
window.draw(circle2); //заливка круга
window.draw(rectangle); //заливка прямоугольника
window.draw(rectangleTele); //заливка телепорт фигуры
window.draw(rectangleTele2); //заливка 2-й теле. фигуры
window.draw(sprite); //вывод картинки
window.display(); //дисплей
}
return 0;
}
```

//200!

4. Что происходит при запуске?

