Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

**Высшая школа программной инженерии**

**ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ**

по дисциплине «Объектно-ориентированное программирование»

Выполнил

студент гр. 3530904/90030 <*подпись*> Минервин Д.А.

Преподаватель

<*подпись*> Круглов С.К.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021г.

Санкт-Петербург

2021

1. **Постановка задачи**

Цель данной работы познакомится и научиться работать с библиотекой **SFML.** Библиотека **SFML** является фреймворком С++, также доступен на Java, Ruby, Python, Go и т.д.

Это хорошая 2D библиотека, с помощью которой можно создавать как 2D игры, так и **GUI (graphical user interface).**

Мною была создана игра 2D платформер, в которой игрок должен бегая по локации выживая как можно дольше собирая сердечки и уклоняясь от врагов и препятствий.

1. **Обзор аналогов**

Самым ярким представителем игры данного жанра является игра на мобильные телефоны **Hungry Shark Evolution.** В игре вы отыгрываете роль акулу которая плавая по локации поедает морских жителей и людей. В качестве противников выступают вооружённые люди, ядовитые рыбы, более крупные акулы, мины и т.д. Ваша акула также теряет жизнь если долго ничего ни ест. Так же пока вы поедаете рыб вам может попасться золотая рыбка за которую начисляют золотые монеты.

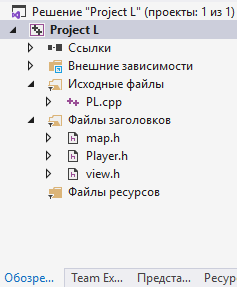
В игре представлено много разных видов акул от самой маленькой до самой большой. За монеты собранные во время плаванья вы будете открывать новые виды акул и улучшать их.

Также **Hungry Dragon** разработчики Ubisoft. Эта игра как и перва на мобильные устройства. Вы играете за дракона летая по локация кушая все подряд. Если вы не будете кушать то игра закончится.

**Hungry Dragon** была выпущена в 2018 году.

1. **Структура программы**

Программа состоит из 4 файлов: 3 заглавных и 1 исходного.



В исходном файле **PL.cpp** расположена функция **int** **main(),** меню игры, рестарт игры и тело игры.

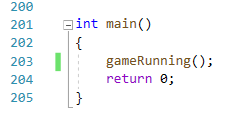
В заглавном файле **map.h** находится карта игры и рандомно добавляем на карту сердечки.

В **Player.h** расположены классы игрока и врагов и их взаимодействие с картой и друг с другом, управление героем, конец игры.

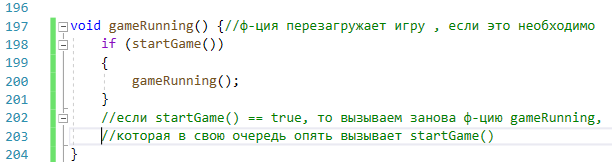
В **view.h** реализован захват камеры на герое и границы карты.

**PL.cpp**

В функции **main** реализован вызов процесса игры:



Функция **gameRanning** отвечает за рестарт игры:

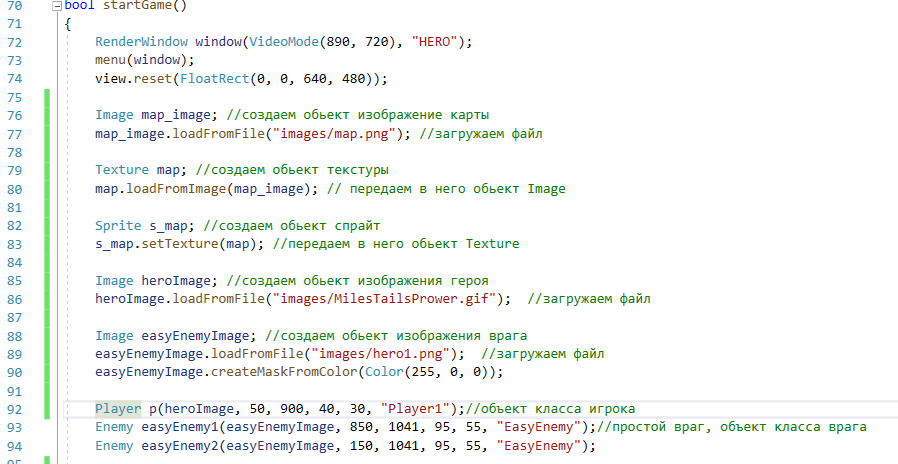


Функция **startGame** имеет тип **bool.** Начинается тело игры с создания окна с размерами 890 на 720 и названием окна **"HERO"**:

**RenderWindow window(VideoMode(890, 720), "HERO");**

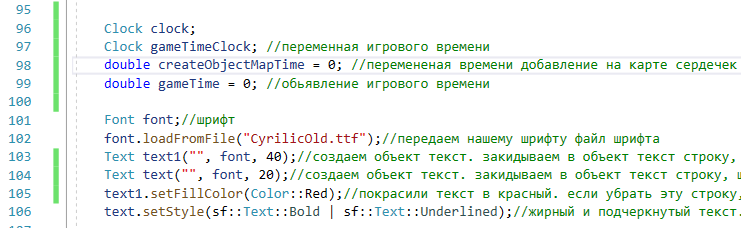
После это реализуем вызов функции **menu(),** которая вызывает меню игры.

Реализуем переменные которые будут отвечать за создание графики игры:



Мы загружаем изображение частей карты, игрока и врага, а также реализуем объект класса игрока **р** и объект класса врага **easyEnemy2** и **easyEnemy2**. В **р** передается изображение игрока, начальные координаты, размер изображение и имя игрока. Аналогично и для врага.

Реализуем переменные времени игры и переменные текста и шрифта текста в окне игры:



На этом реализация переменных заканчивается.

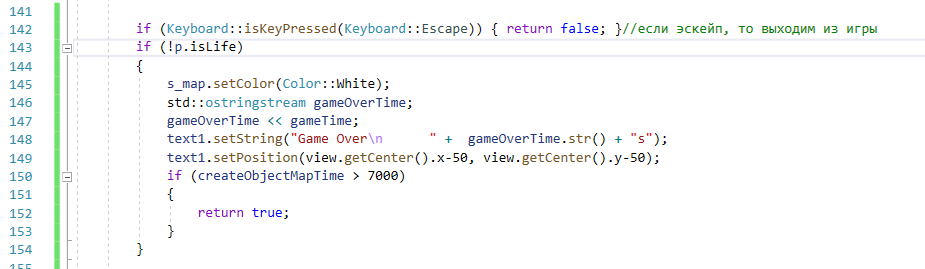
Далее реализуем цикл который подразумевает “Пока окно открыто – выполняй то, что внутри цикла”. Для закрытия окна реализуется событие **event,** которое должно принять значение **Closed.**

Создаем переменную времени **time** в миллисекундах, она отвечает за скорость игры. Реализуем случайное добавление на карту сердечек, а также остановку времени игры когда жизнь игрока 0.

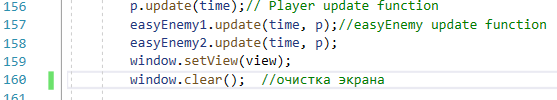


Реализуем выход из игры используя кнопку **ESC.**

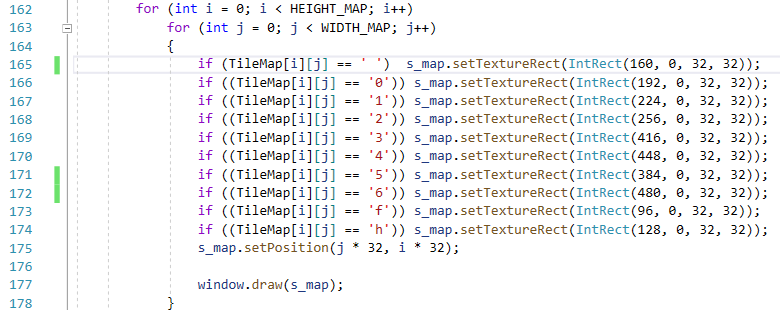
После того как игра окончена выводим на экран надпись «Конец игры» и время игры. Через 7с игра закрывается и выводится окно меню.



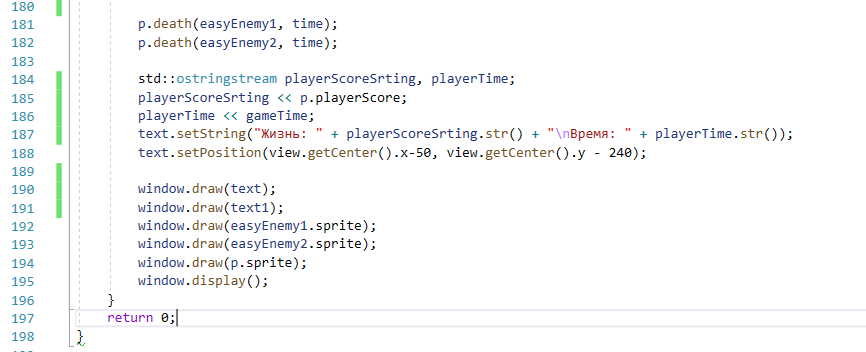
Вызываем функции **update** «оживления» класса игрок и врага. **update** отвечает за передвижение игрока и врагов. Очищаем экран.



Загружаем текстуры карты:



Вызываем функцию взаимодействия игра и врага. Выводим строки количество жизни и время игры. Используя **window.draw** прорисовываем объекты, а **window.display(); -** показываем все объекты.



Функция **menu** реализовывает меню игры. Создаем переменные графического меню и загружаем файлы. Задаем координаты объектов меню. Задаем логическую переменную меню и переменную номера кнопки меню.



Реализуем цикл открытия меню. Очищаем экран. Реализуем изменение цвета кнопки на синий при нажатии.

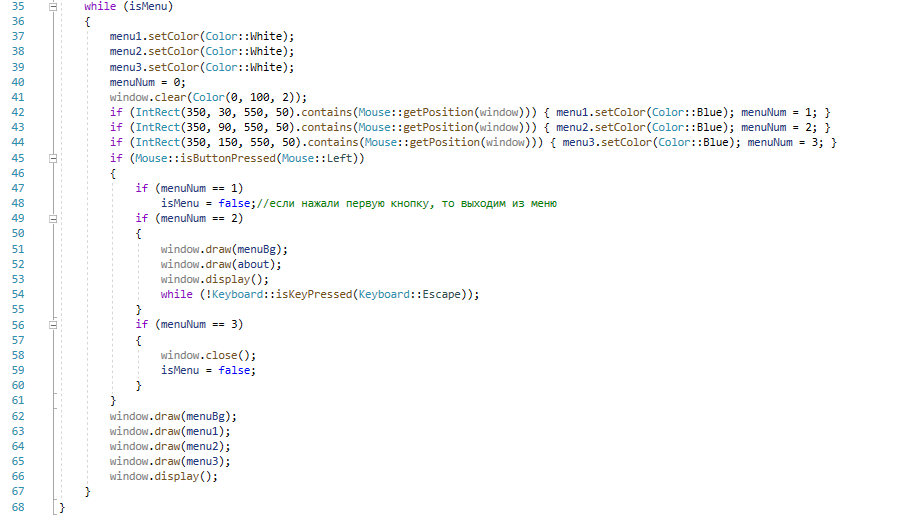
Реализуем условия:

-Если нажата первая кнопка то запускаем игру;

-Если вторая кнопка то выводить правила игры;

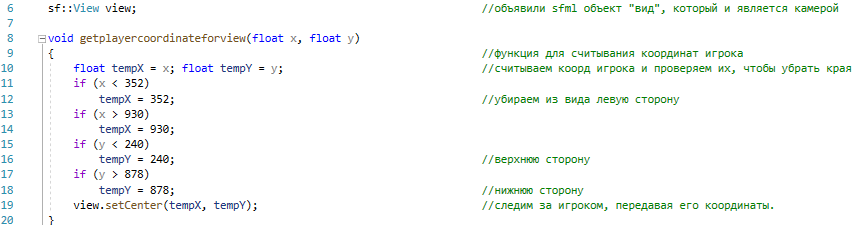
-Если третья – выходит из меню и закрывает приложение.

Прорисовываем меню и показываем все обьекты.



**view.h**

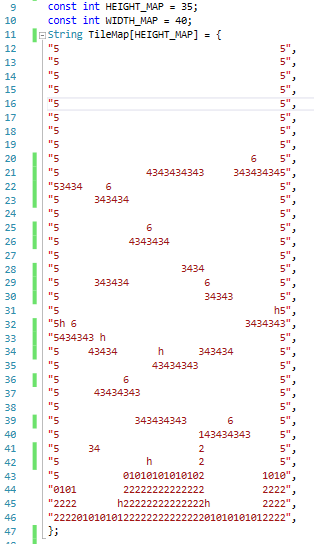
В этом заглавном файле реализуем функцию захвата камеры на игроке и границы карты.



**map.h**

В этом заглавном файле реализуем карту и случайное появление на карте сердечек.

Реализуем переменные высоты и ширины карты, и строковую переменную двумерно массива которая и будет нашей картой. В массиве мы рисуем нашу карту: препятствия, ловушки, землю по которой будет передвигаться игрок.



Для случайного заполнения карты сердечками реализуем функцию, в которой задаем условия появления на карте.

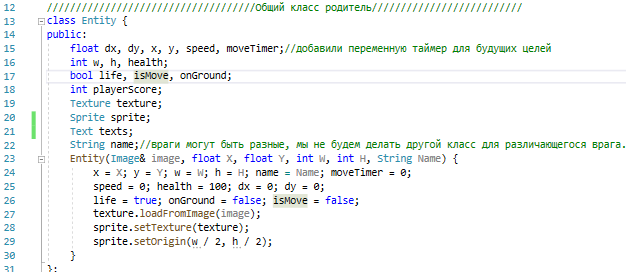


**Player.h**

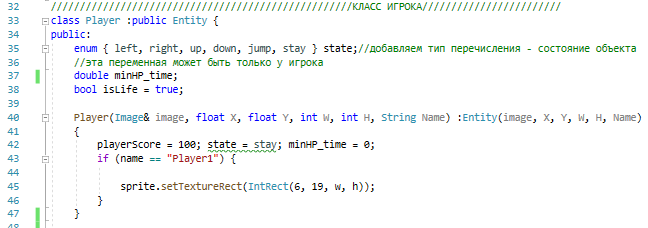
Здесь реализуем классы игрок и врага. Их передвижение, взаимодействия с картой и друг с другом.

Реализуем общий класс родитель. Задаем переменные координат, высоты и ширины, количество жизни, строковые переменные имени, логические переменные жизни, притяжения, движения. Задаем переменные текстур, спрайта, текста.

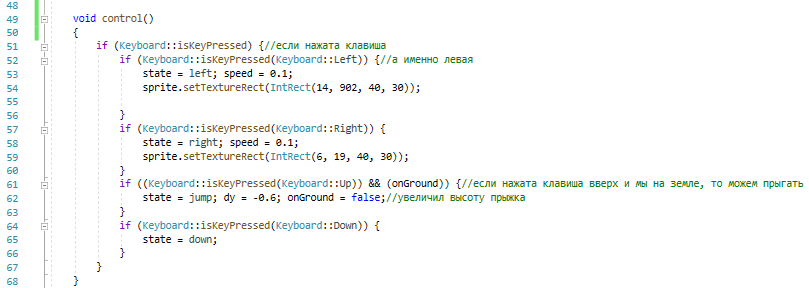
Реализуем конструктор родительского класса.



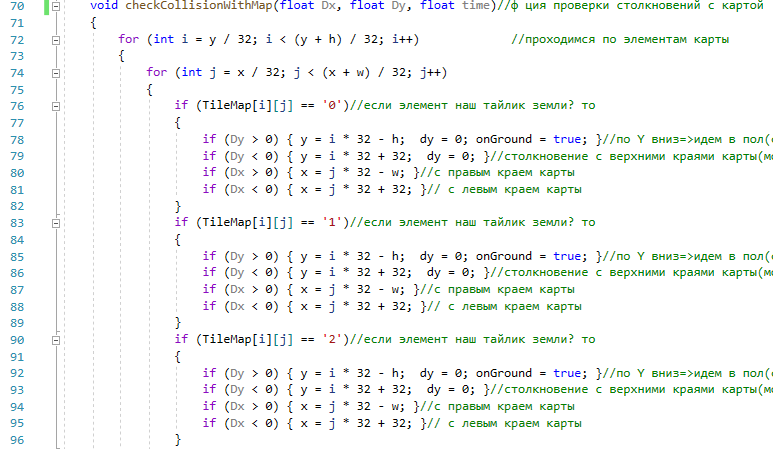
Далее реализуем класс игрок. Задаем дополнительные переменные только для класса игрок, и конструктор класса, в котором передаем изображение игрока, переменные родительского класса и дополнительные переменные.

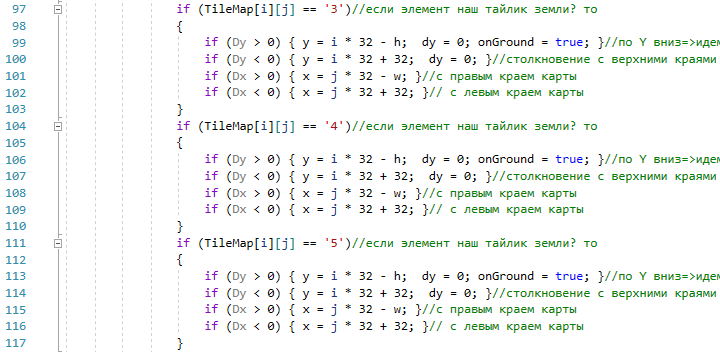


В функции контроля игрока реализуем передвижения игрока с помощью клавиш клавиатуры.



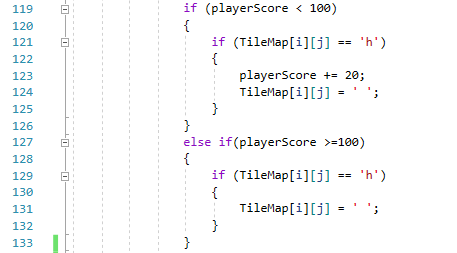
Следующий метод реализует взаимодействие с разными элементами карты.



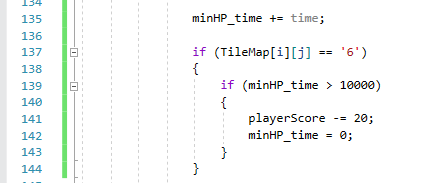


После взаимодействия игра с элементами карты по которой он передвигается реализуем сбора сердечек и взаимодействия с ловушками.

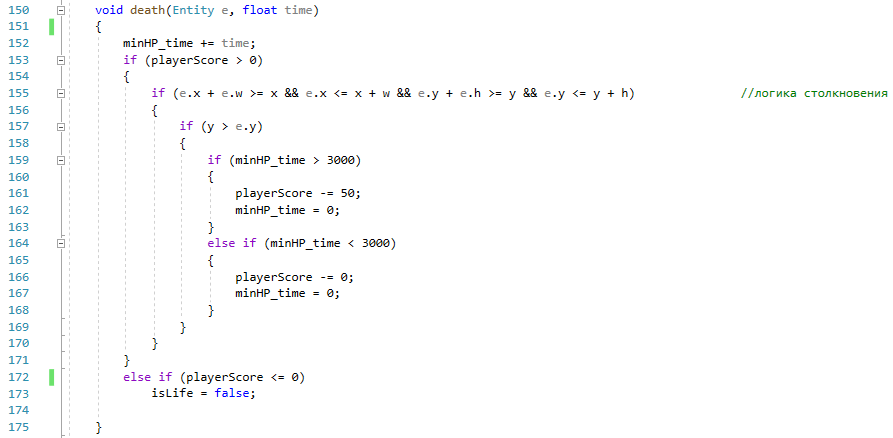
Реализация сбора сердечек:



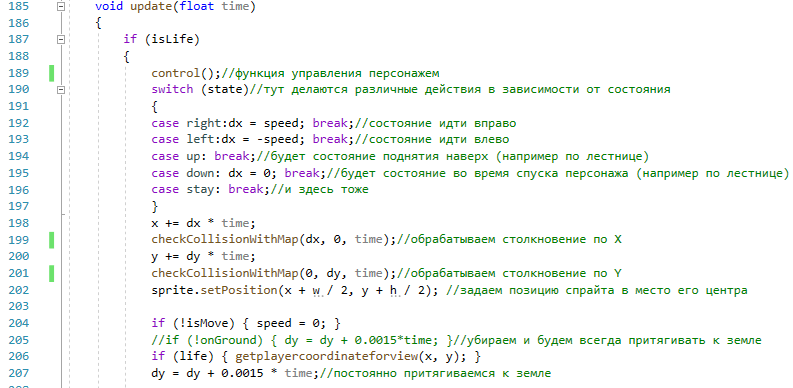
Взаимодействие с ловушками:



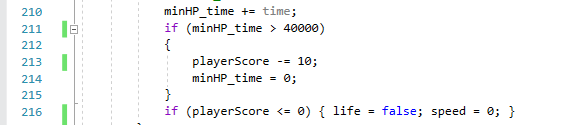
Теперь реализуем метод взаимодействие с врагами. Если мы попадаем к врагу в лапы то теряем 50 очков жизни.



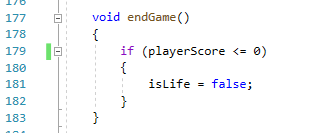
Метод **update** является основным методом. В нем реализуется управление персонажем, вызов метода **control,** обработка столкновение с картой.



Также в этом методе реализуем условие при котором через определенное время у героя будет отниматься жизнь:

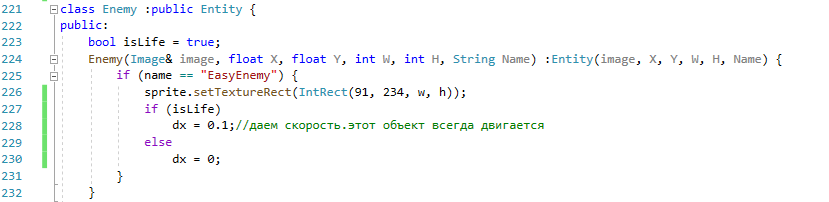


И последний метод в этом классе это «конец игры». Он реализуется так:

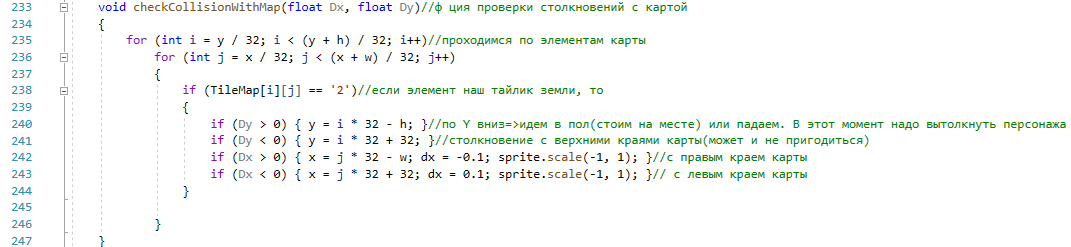


На этом с классом игрок все.

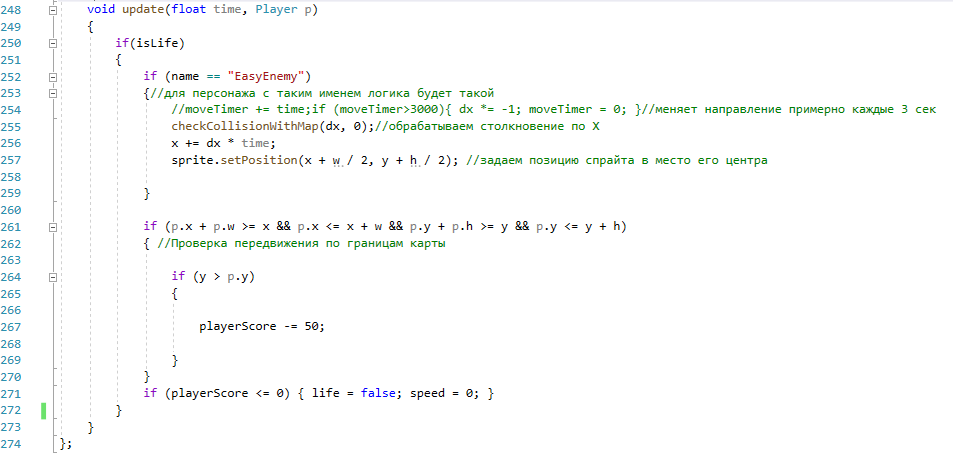
Реализуем класс врага. В нем как и в классе игрок реализуем конструктор, в котором передаем изображение врага, переменные родительского класса. Реализуем движение врага.



Реализуем взаимодействие врага с элементами карты. Так как он все время двигается, то нужна сделать так чтоб он не заходил за текстуры карты и передвигался в разных направлениях, а не только в одном.



И последний метод но основной это **update.** Здесь реализуем обработку столкновение с картой, задаем позицию спрайта.



На этом реализация кода игры заканчивается.

1. **Описание игры**

Основное правило игры – это проживи как можно дольше. Игра с одной стороны простая, а с другой сложная (самое дольше что я смог продержаться это 51с).

При запуске игры появляется меню. На выбор предоставляется 3 кнопки: новая игра, о програме, и выход.



«Новая игра» запускает игру. «О программе» выводит окно с описанием игры, что бы вернуться в основное меню надо нажать ESC. «Выходи» закрывает меню и выходит из игры.



В игре героем является лиса, за которую и играет игрок. Врагом – лев. Игрок должен передвигаясь по карте собирать сердечки, за которые восстанавливается жизнь лисы **(+20 очков жизни)**. Каждые 8с будет отниматься жизнь **(-10 очков жизни)**. Если игрок натыкается на льва то жизнь уменьшается на половину **(то есть -50 очков).** Также по карте встречаются ловушки, которые, если в них попасть, то же отнимают жизнь **( -20 очков жизни).**

Игра заканчивается если жизнь лисы становится ноль. По окончанию игры появляется надпись «Game Over» и сколько времени прожил ваш персонаж.



Ловушки в игре:



Сердечко:



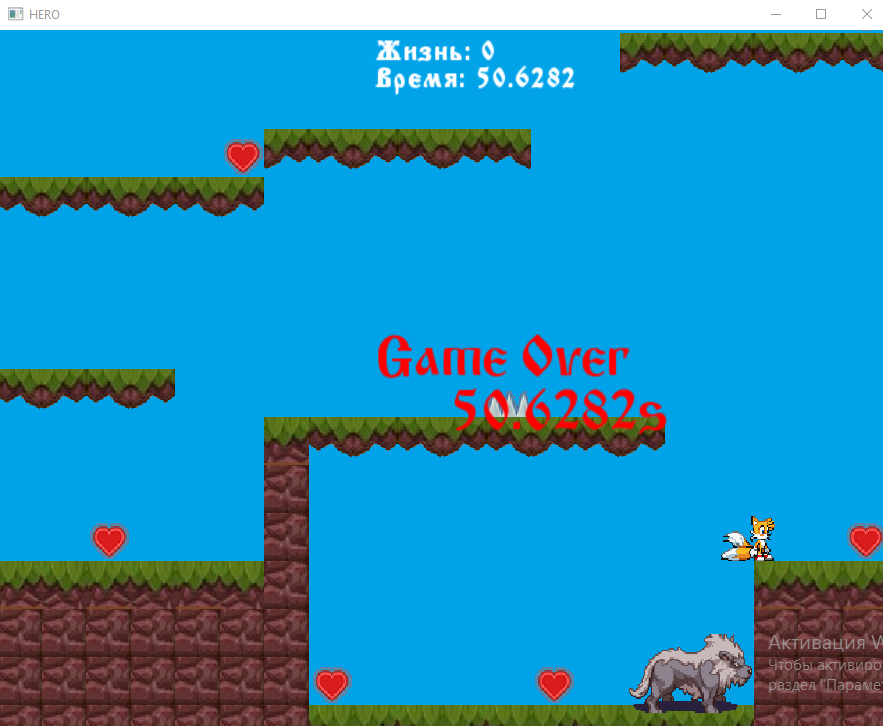
Враг – лев:



Количество жизней и время игры:



Конец игры :



1. **Литература**

<https://kychka-pc.ru/category/sfml> - SFML(разработка 2D игр), язык С++