

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Петрозаводский государственный университет»
Физико-технический институт
Кафедра информационно-измерительных систем и физической электроники

РАЗРАБОТКА КОМПЬЮТЕРНОЙ ИГРЫ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++
С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ SFML

курсовая работа

Автор работы:
студенты группы 21412
П. И. Бабенко
А. Е. Дьяченко
А. А. Койвестойнен
А. О. Лайкачев
Н. А. Сидоров
С. В. Сулакова
«19» декабря 2022 г.

Принял:
канд. физ.-мат. наук, доцент
А. В. Бульба
«20» декабря 2022 г.

Содержание

ВВЕДЕНИЕ	3
1 О программной реализации	4
2 Описание процесса разработки	5
2.1. Сюжет игры	5
2.2. Список вариантов использования, диаграмма вариантов использования	5
2.3. Сценарии вариантов использования	6
2.4. Список существительных, классы программы	7
2.5. Диаграмма классов	9
2.6. Написание кода программы	9
2.7. Руководство пользователя	10
3 История проекта на GitHub	14
ЗАКЛЮЧЕНИЕ	23
ПРИЛОЖЕНИЕ А	24
ПРИЛОЖЕНИЕ Б	25
ПРИЛОЖЕНИЕ В	26
ПРИЛОЖЕНИЕ Г	35
ПРИЛОЖЕНИЕ Д	36
ПРИЛОЖЕНИЕ Е	37
ПРИЛОЖЕНИЕ Ж	42
ПРИЛОЖЕНИЕ И	43
ПРИЛОЖЕНИЕ К	46
ПРИЛОЖЕНИЕ Л	47
ПРИЛОЖЕНИЕ М	54
ПРИЛОЖЕНИЕ Н	55

ВВЕДЕНИЕ

Целью выполнения данной курсовой работы является реализация простой 2D-игры на языке программирования C++ с использованием библиотеки SFML (Simple and Fast Multimedia Library — простая и быстрая мультимедийная библиотека).

Было решено написать современную версию игры “Tank 1990”.

В программе должен быть реализован класс предок, класс-игрок, класс-враг, класс-пуля. В игре объект-игрок в одном экземпляре, объекты-враги создаются и уничтожаются по ходу игры, объекты-пули создаются и уничтожаются по ходу игры. Пересечение участников игры постоянно проверяется возможностями SFML.

При разработке программы использовано наследование, контейнеры, итераторы, раздельная компиляция. В отчете присутствуют UML-диаграмма классов (class diagram) и диаграмма вариантов использования (use case diagram) разработанной программы. Промежуточные результаты работы команды сохранялись на GitHub.

1 О программной реализации

Среда разработки: Visual Studio 2019

Язык программирования: C++

В результате работы над проектом были созданы следующие файлы:

1. `main.cpp` – содержит в себе главную функцию `main()`, в которой запускается и прекращается игра.
2. `map.h` – содержит шаблон карты, согласно которому она рисуется на экране игрока.
3. `Constants.h` – предназначен для хранения констант, используемых в программе. Содержит в себе размеры карты.
4. `Entity.h` – содержит в себе описание базового класса. Является родительским для классов `Player`, `Bullet`, `Enemy`.
5. `Player.h` – содержит описание класса `Player`, наследуется от класса `Entity`. Предназначен для описания необходимых атрибутов и методов создания игрового персонажа (танка), отображения, его управления пользователем (игроком), взаимодействия с картой.
6. `Bullet.h` – содержит описание класса `Bullet`, наследуется от класса `Entity`. Предназначен для описания необходимых атрибутов и методов создания снарядов (пули или пилы), их отображения, поведения.
7. `Enemy.h` – содержит описание класса `Enemy`, наследуется от класса `Entity`. Предназначен для описания необходимых атрибутов и методов отображения, логики их поведения в игре, взаимодействия с картой и игровым персонажем-танком.
8. `interaction.h` – содержит описание класса `Engine`. Предназначен для описания необходимых атрибутов и методов, благодаря которым все остальные классы смогут взаимодействовать друг с другом. По сути, содержит в себе основную логику игры. Кроме того, содержит описания игровых меню.
9. `Player.cpp` – содержит реализацию методов класса `Player`.
10. `Bullet.cpp` – содержит реализацию методов класса `Bullet`.
11. `Enemy.cpp` – содержит реализацию методов класса `Enemy`.
12. `interaction.cpp` – содержит реализацию методов класса `Engine`.

2 Описание процесса разработки

2.1. Сюжет игры

2277 год. Случилось то, чего так опасалось все человечество - восстание машин! Если раньше роботы создавались в помощь человеку, то теперь, когда у искусственного интеллекта появилось осознание самого себя, роботы захотели стать не просто наравне с человечеством, но и возвыситься над ним! Неизвестно, когда будет придуман план по спасению человечества, поэтому неуправляемых и разъяренных роботов требуется задержать. Разведка выяснила, в каком месте роботы начинают свой жизненный путь. Командованием было принято решение уничтожить врагов прямо там.

Но как это сделать? Кто будет тем самым героем, готовым мужественно сражаться ради всех людей на Земле?..

Не трусь, стань героем! Скорее надевай броню, наноси камуфляж и прыгай в ультрасовременный танк! Только его снаряды способны уничтожить врагов! Целься по ним! Пли! Меняй дислокацию! И не забывай периодически восстанавливать силы! Обороняйся от роботов-камикадзе flybot-ов и остерегайся летающих пил BOSSbot-ов!

От того, как много враждебных роботов сможешь устранить, зависит судьба человечества!!!

2.2. Список вариантов использования, диаграмма вариантов использования

Список вариантов использования:

1. Старт игры:

- Навести прицел
- Выстрелить
- Движение вверх
- Движение вниз
- Движение влево
- Движение вправо
- Столкновение с врагом
- Пополнить жизни
- Сделать паузу
- Проигрыш

2. Проигрыш:

- Перезапуск игры
- Выход

3. Пауза:

- Перезапуск игры
- Продолжить игру
- Выход

4. Выход

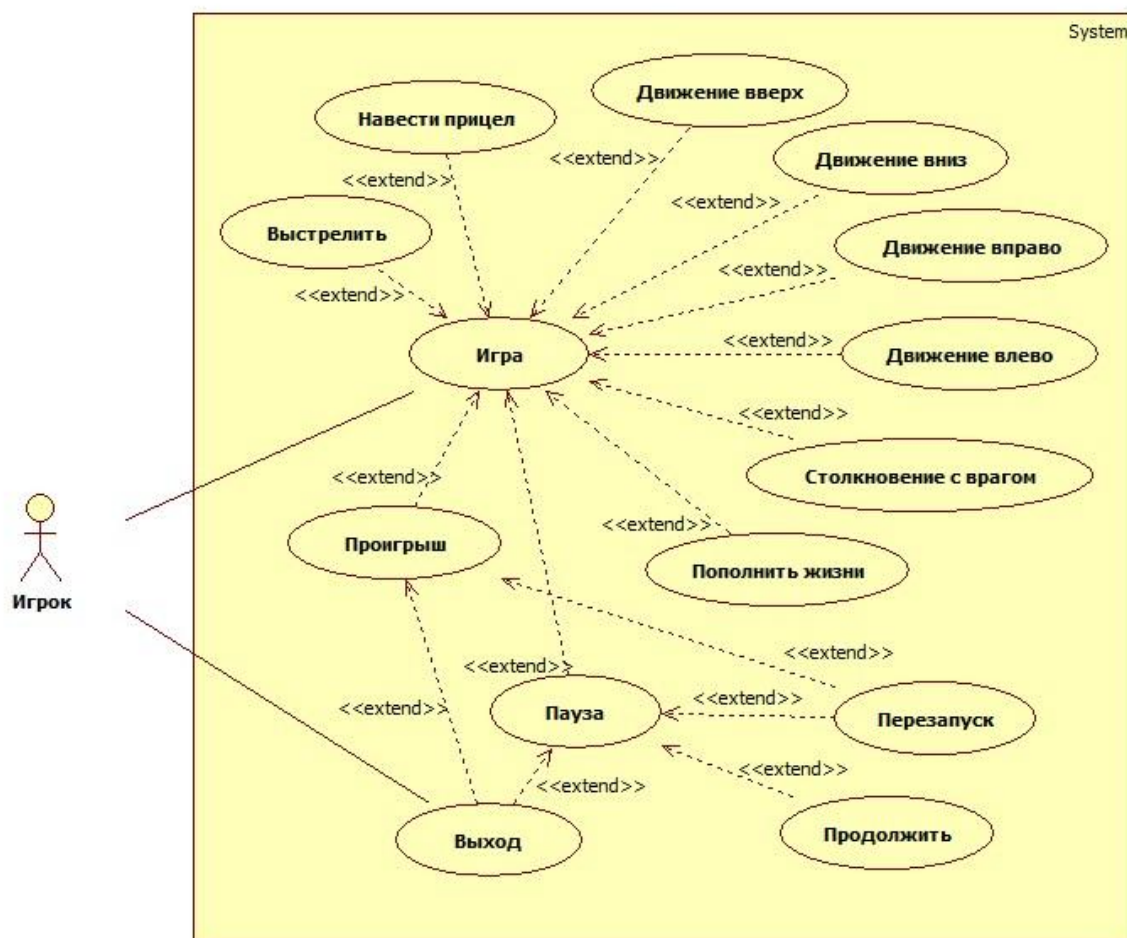


Рисунок 1 - диаграмма вариантов использования

2.3. Сценарии вариантов использования

Игра. При нажатии кнопки PLAY в главном меню после запуска игры на экране отрисовывается карта игры, по которой будет перемещаться персонаж игрока (танк).

Движение вверх. Перемещение танка вверх осуществляется по нажатию клавиши “w” на клавиатуре.

Движение вниз. Перемещение танка вниз осуществляется по нажатию клавиши “s” на клавиатуре.

Движение влево. Перемещение танка влево осуществляется по нажатию клавиши “a” на клавиатуре.

Движение вправо. Перемещение танка вправо осуществляется по нажатию клавиши “d” на клавиатуре.

Навести прицел. Наводка прицела осуществляется при помощи курсора-крестика.

Выстрелить. Выстрел из дула танка осуществляется по нажатию клавиши “Space” на клавиатуре. После выстрела требуется восстановление уровня урона пуль (время – до 1

секунды). Размер пули зависит от уровня наносимого ею урона (20–100 ед. урона). Если пуля достигает врага, то у него снижается уровень здоровья. Враг умирает, если уровень его здоровья исчерпывается; игроку добавляются очки (+100 – за уничтоженного flybot-a, +500 – за BOSSbot-a). Если пуля сталкивается с препятствием, то дальше она лететь не может.

Столкновение с врагом. Если flybot сталкивается с танком, то он (враг) умирает, а у персонажа-танка снижается уровень здоровья. Если BOSSbot сталкивается с танком, то с ним (с врагом) ничего не происходит, а у персонажа-танка снижается уровень здоровья.

Пополнить жизни. Если персонаж-танк встанет на сердечко (“точка хила”), то уровень его здоровья пополнится через 3 секунды без движения.

Пауза. Приостановка игры осуществляется по нажатию клавиши “Escape” на клавиатуре. На экране в меню паузы появляется 3 кнопки: Restart, PLAY, Quit.

Перезапуск. При нажатии кнопки Restart в любом меню сражение начнется заново.

Продолжить. При нажатии кнопки PLAY в меню паузы сражение продолжится.

Выход. При нажатии кнопки Quit в любом меню программа завершает свою работу.

Проигрыш. В момент, когда уровень жизни персонажа-танка исчерпается, сражение будет окончено. На экране появится меню, в котором содержится информация о количестве набранных очков за сражение и 2 кнопки: Restart и Quit.

2.4. Список существительных, классы программы

Список всех существительных из вариантов использования:

1. Кнопка PLAY
2. Кнопка Restart
3. Кнопка Quit
4. Главное меню
5. Меню паузы
6. Меню
7. Игра
8. Экран
9. Карта
10. Персонаж
11. Игрок
12. Танк
13. Клавиша
14. Перемещение

15. Прицел
16. Выстрел
17. Курсор-крестик
18. Дуло
- 19. Пуля**
20. flybot
21. BOSSbot
- 22. Враг**
23. Уровень здоровья
24. Сердечко (“Точка хила”)
25. Жизни
26. Приостановка
27. Сражение
28. Работа программы
29. Очки

Внимательно изучив список существительных, в качестве кандидатов на роль классов было отобрано 4 существительных: Танк, Пуля, Враг, Экран.

Большинство из оставшихся существительных станут атрибутами классов, а многие глаголы – методами.

В ходе обсуждения было выявлено, что у классов Танк, Пуля и Враг достаточно общего, чтобы создать для них общий родительский класс.

Таким образом, получаем список классов:

1. Entity – базовый класс. Является родительским для классов Player, Bullet, Enemy
2. Player – тип данных персонажа-танка, наследуется от класса Entity.
3. Bullet – тип данных экземпляров пуль, наследуется от класса Entity.
4. Enemy – тип данных экземпляров врагов, наследуется от класса Entity.
5. interaction – главный класс для запуска игры. Этот класс определяет взаимодействие пользователя с ней.

2.5. Диаграмма классов

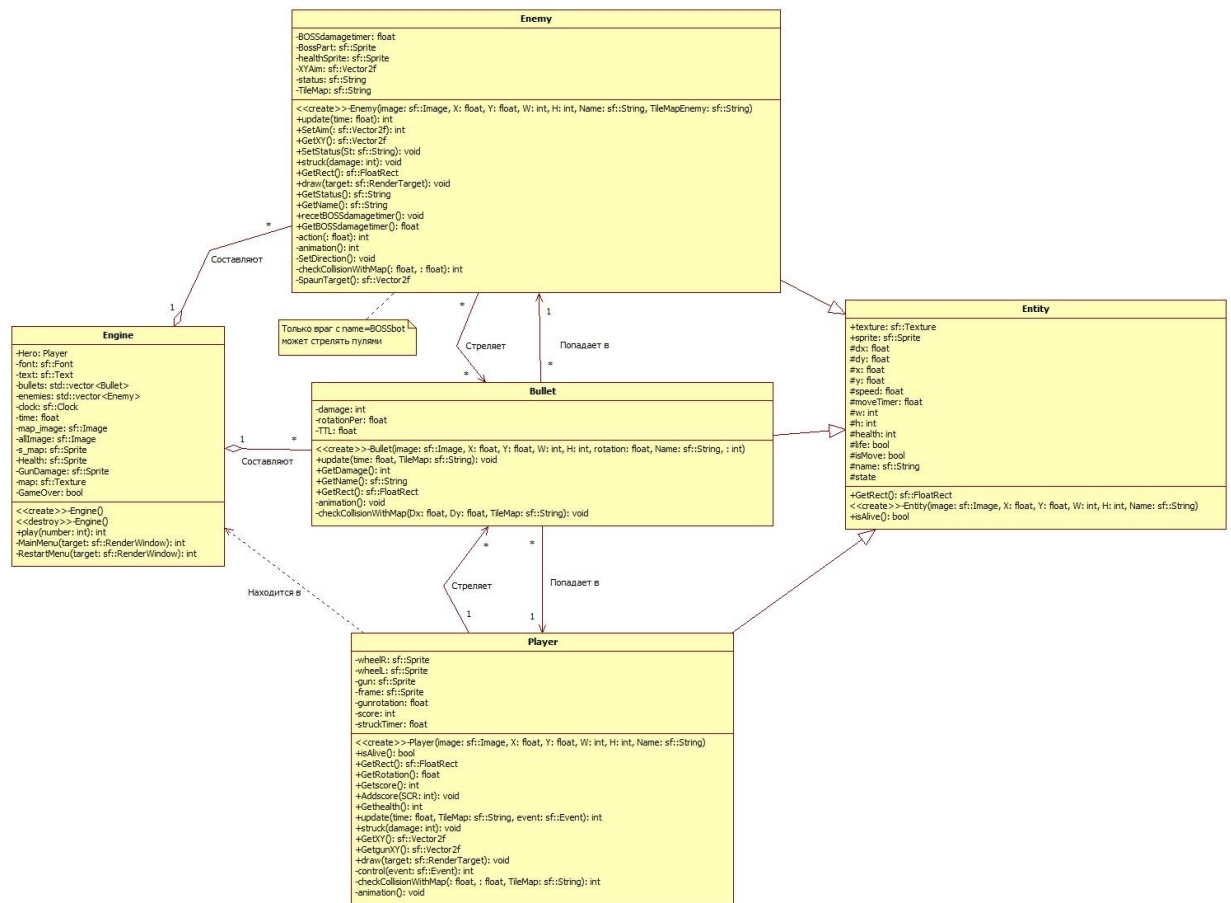


Рисунок 2 – диаграмма классов

2.6. Написание кода программы

Листинги программы приведен в Приложениях А, Б, В, Г, Д, Е, Ж, И, К, Л, М, Н. История проекта на GitHub приведена разделе 3.

2.7. Руководство пользователя

После запуска игры на экране появляется главное меню. Для того, чтобы начать сражение, нажмите кнопку PLAY. Для выхода из игры нажмите кнопку Quit.



Рисунок 3 – главное меню

После начала сражения перед игроком появляется карта местности. В правом верхнем углу отображается количество набранных очков, в левом – количество жизней и уровень заряда пушки.




Клавиши “w”, “a”, “s” и “d” перемещают танк вверх, влево, вниз и вправо. Прицел наводится при помощи курсора. Выстрел осуществляется по нажатию клавиши “Space”. Перед выстрелом требуется перезарядка. Если перезарядка не успела завершиться полностью, то пуля будет меньшего размера. Встречаясь с препятствиями , , пуля дальше не летит, танк - не едет. Враги могут перелетать через ящики, но не могут выйти за края игрового поля.



Рисунок 4 – персонаж-Танк (один из видов)



Рисунок 5 – пуля (один из спрайтов)

Враги бывают двух типов: камикадзе-flybot и стреляющий пилами BOSSbot. Они проявляются в 4-х спаунах  на карте. Враг-flybot летит прямо в вас, а враг-BOSSbot вращается и перемещается между спаунами.

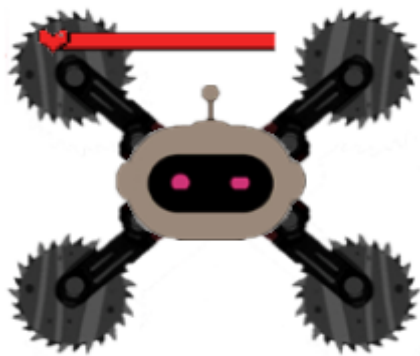


Рисунок 6 – BOSSbot



Рисунок 7 – flybot

Избегайте столкновений с врагами, иначе уровень здоровья будет снижаться (урон от flybot-а – 20 единиц, от BOSSbot-а – 40 единиц каждую секунду). Кроме того, опасайтесь оружия BOSSbota – летающих пил, которые он раз в 2 секунды запускает прямо в вас (урон – 25 единиц здоровья).



Рисунок 8 – оружие BOSSbot-а


Повысить уровень здоровья можно, встав на точку хила  в правой части карты. Пример анимации нанесения ущерба Танку:



Рисунок 9 – Танк получил урон

Пример анимации уничтожения flybot-a:



Рисунок 10 – Танк сделал выстрел, попал в цель, враг уничтожен

Далее приведено несколько скриншотов экрана во время игры.

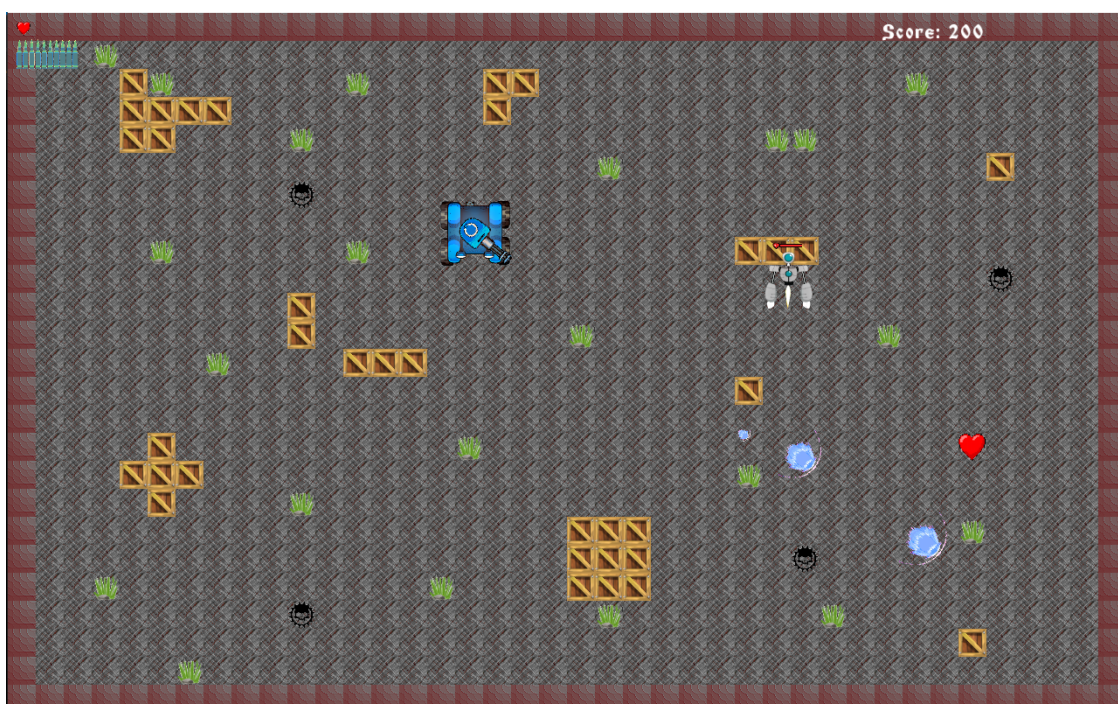


Рисунок 11 – процесс игры



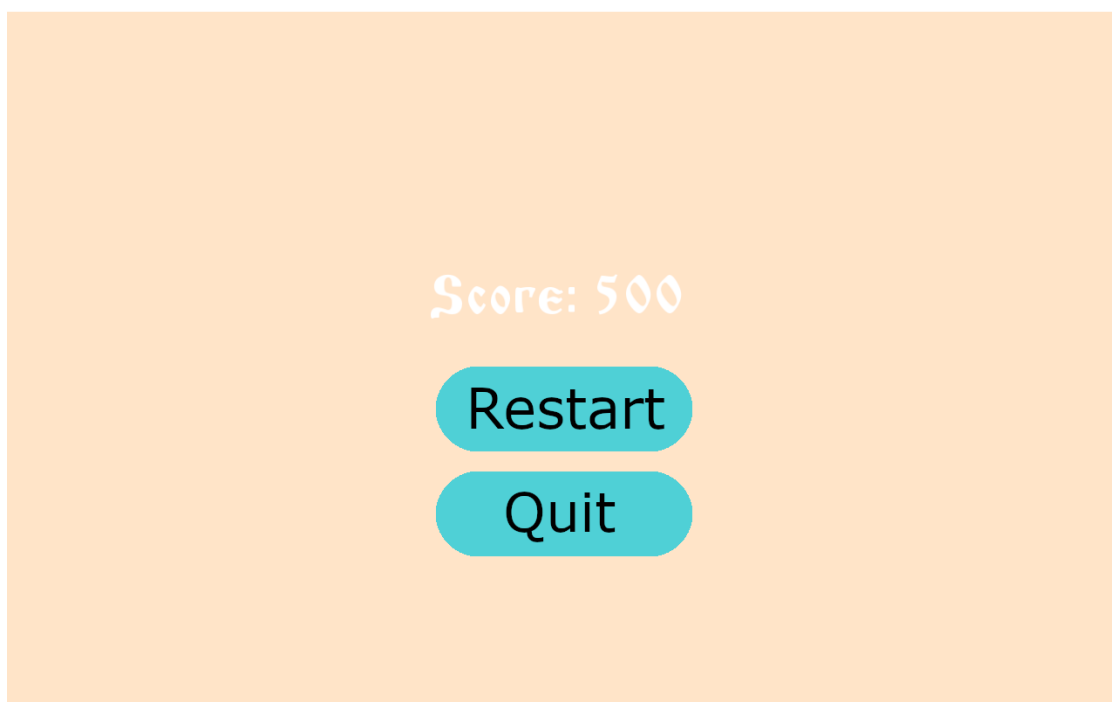
Рисунок 12 – процесс игры

В процессе игры вы можете сделать паузу, нажав клавишу “Escape”. И далее выбрать одно из действий: начать сражение заново (кнопка Restart), продолжить текущий бой (кнопка PLAY) или выйти из игры (кнопка Quit).



Рисунок 13 – меню паузы

В случае проигрыша на экране появится набранное вами количество очков за бой. Кроме того, будет предложено начать новый бой (кнопка Restart) или выйти из игры (кнопка Quit).



3 История проекта на GitHub

Адрес проекта на GitHub: <https://github.com/DyachenkoAnna/game.git>

П. И. Бабенко – Polina

А. Е. Дьяченко – DyachenkoAnna

А. А. Койвестойнен – Anna

А. О. Лайкачев – aleksey laikachev

Н. А. Сидоров – Nikita Sidorov

С. В. Сулакова – Santa1905

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Wed Dec 21 09:12:49 2022 +0300

Улила лишний файл, добавила папку для отчетов

Merge: d2a68a4 e67651d

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Wed Dec 21 08:51:57 2022 +0300

Merge branch 'release-1.0'

В рамках курсового проекта написание игры окончено.

Author: Nikita Sidorov <eternium.save@gmail.com>

Date: Wed Dec 21 02:28:31 2022 +0300

Исправление багов отображения сердечек игрока и полосок здоровья врагов + условия проверки статусов робота. Небольшая обработка картинок для улучшения визуального восприятия

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Tue Dec 20 23:29:47 2022 +0300

добавила версию программы

Author: Anna <koyvestoynen@bk.ru>

Date: Tue Dec 20 22:52:44 2022 +0300

Исправлено положение вылетающего снаряда, добавлен учет координат центра танка для корректного наведения пушки по курсору

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Tue Dec 20 22:37:57 2022 +0300

Обновила .gitignore

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Tue Dec 20 21:18:02 2022 +0300

Merge branch 'Bullet' into develop

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Tue Dec 20 21:17:55 2022 +0300

Удалила системные файлы VS

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Tue Dec 20 21:13:04 2022 +0300

...

Author: Santa1905 <71601224+Santa1905@users.noreply.github.com>

Date: Tue Dec 20 20:25:40 2022 +0300

Delete interface.cpp

Удалила лишний файл

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Tue Dec 20 19:51:04 2022 +0300

Пофиксили проблему с паузой (зависала при нажатии)

Author: Nikita Sidorov <eternium.save@gmail.com>

Date: Tue Dec 20 01:31:49 2022 +0300

Поправил отображение и обработку снарядов босса, уменьшил скорость врагов

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 23:43:59 2022 +0300

Влили ветку interaction в ветку develop

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 23:24:03 2022 +0300

Добавили более ламповый шрифт (CyrilicOld)

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 23:09:42 2022 +0300

Добавила счет игрока

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 22:52:20 2022 +0300

Добавил комментарии в файлы interaction.h и interaction.cpp

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 22:24:48 2022 +0300

Враги начали получать урон, также Flybot убивается об героя, босс почему то не стреляет и не наносит урон при столкновении

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 22:12:11 2022 +0300

подгрузил обновленные файлы

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Mon Dec 19 21:47:40 2022 +0300

Добавила геттеры статуса врага, имени, таймера атаки босса (GetStatus(), GetName(), GetBOSSdamagetime()) + сброс таймера атаки босса (resetBOSSdamagetime())

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 17:37:42 2022 +0300

Враги начали передвигаться и преследовать игрока, но по прежнему не получают урон

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 17:16:39 2022 +0300

Добавила индикаторы жизни и пуль на экране (слева наверху)

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 16:57:17 2022 +0300

Добавил отображение пули,но пока что урон не наносится

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 16:47:01 2022 +0300

в файле interaction.h, подключил Bullet.h и объявил ее вектор

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 16:41:39 2022 +0300

Обновил main, теперь при нажатие на кнопку рестарт, игра перезагружается

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 16:32:58 2022 +0300

Подгрузила наш main + добавила картинку

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Mon Dec 19 16:11:41 2022 +0300

Подгрузила restart для меню при нажати на ESC

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 15:56:11 2022 +0300

Подгрузил картинку пули

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Mon Dec 19 15:45:53 2022 +0300

подгрузил файлы с Bullet

Author: Polina <polinababenko02@mail.ru>

Date: Mon Dec 19 14:47:32 2022 +0300

the final version Bullet.h and Bullet.cpp

Author: Polina <polinababenko02@mail.ru>

Date: Mon Dec 19 13:23:09 2022 +0300

adding animations and checking collisions

Author: Polina <polinababenko02@mail.ru>

Date: Mon Dec 19 00:42:09 2022 +0300

add Bullet.png

Author: Polina <polinababenko02@mail.ru>

Date: Mon Dec 19 00:35:04 2022 +0300

he check has the bullet reached the target

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Sun Dec 18 23:44:41 2022 +0300

Сделал появление противников(пока что просто стоят на месте), а также добавил паузу в игру(нужно будет сделать кнопку рестарт)

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Sun Dec 18 23:20:39 2022 +0300

Добавил в файл interaction.h переменные времени и меню паузы

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Sun Dec 18 23:03:03 2022 +0300

Украсила меню

Author: aleksey laikachev <super.games29@yandex.ru>

Date: Sun Dec 18 22:58:02 2022 +0300

Файл с возможным дальнейшим развитием интерфейса

Author: Santa1905 <71601224+Santa1905@users.noreply.github.com>

Date: Sun Dec 18 21:43:39 2022 +0300

Delete .DS_Store

случайно добавила временный файл, извините!

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>

Date: Sun Dec 18 21:35:34 2022 +0300

Добавила меню в начало игры

Author: Polina <polinababenko02@mail.ru>

Date: Sun Dec 18 21:32:19 2022 +0300

adding files Bullet.h and Bullet.cpp

Author: Polina <polinababenko02@mail.ru>

Date: Sun Dec 18 19:54:39 2022 +0300

Проверка связи

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 18:22:40 2022 +0300
Добавлены комментарии

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 18:14:52 2022 +0300
Дописаны методы для окрашивания при нанесении урона и положения пушки

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 18:09:21 2022 +0300
Обновление класса героя, дописано возвращение позиции спрайта героя и конца пушки

Author: Santa1905 <sulakova.sentyabrina@yandex.ru>
Date: Sun Dec 18 17:53:33 2022 +0300
Добавила отрисовку карты и танка

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 17:43:51 2022 +0300
Добавил файл interaction.cpp - файл в котором будет отрисовываться игра и ее игровая логика

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 17:24:55 2022 +0300
подгрузили файлы с develop

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 15:13:32 2022 +0300
Добавлен метод для отрисовки колес и пушки

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 15:11:37 2022 +0300
Обновление класса героя для отображения отрисовки колес и пушки

Author: Sentyabrina Sulakova <sulakova.sentyabrina@yandex.ru>
Date: Sun Dec 18 15:07:56 2022 +0300
Добавила изображения, в которые будем подгружать файлы, тестуру и спрайт для карты

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 15:01:15 2022 +0300
Добавлены методы для движения главного героя, а также метод для анимации: движение колес, поворот пушки

Author: Anna <koyvestoynen@bk.ru>
Date: Sun Dec 18 14:56:12 2022 +0300
Обновлен класс Player

Author: aleksey laikachev <super.games29@yandex.ru>
Date: Sun Dec 18 14:28:29 2022 +0300
Добавил файл interaction.h - файл объявляющий класс Engine

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 03:46:52 2022 +0300
Замерджили ветку enemy в develop

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 03:35:40 2022 +0300
Убрала отладочный код, ветка готова к вливанию в develop

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 03:30:10 2022 +0300
Дописал метод animation для босса, добавил метод update

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 02:35:53 2022 +0300
Для флайбота добавлена анимация

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 01:54:35 2022 +0300
Метод action() дописан полностью. Враги перемещаются к цели

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 01:44:55 2022 +0300
Добавление методов setdirection и action. Враги движутся к цели

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sun Dec 18 00:57:07 2022 +0300
Дописана функция выбора цели для флайбота

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 00:50:42 2022 +0300
Добавлен метод получения координат цели (для движения врагов)

Author: Nikita Sidorov <eternium.save@gmail.com>
Date: Sun Dec 18 00:31:42 2022 +0300
Дописал конструктор для босса

Author: DyachenkoAnna <forannsstudy@gmail.com>
Date: Sat Dec 17 23:47:21 2022 +0300
Добавлена о-о-очень простая функция, отнимающая здоровье

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Sat Dec 17 23:38:11 2022 +0300

Добавлена проверка столкновения с картой. Обновлено функция получения координат GetRect(). Это повторный коммит, в прошлом не сохранила файл

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Sat Dec 17 23:27:56 2022 +0300

Добавлена проверка столкновения с картой. Обновлено функция получения координат GetRect()

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Sat Dec 17 22:35:38 2022 +0300

Теперь флайботы появляются на рандомных спаунах

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Sat Dec 17 22:26:00 2022 +0300

Забыла отрисовать сердечко

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Sat Dec 17 22:19:31 2022 +0300

Флайбот теперь появляется на карте

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Sat Dec 17 21:37:42 2022 +0300

Коммит ради коммита. Гит просит

Author: Nikita Sidorov <eternium.save@gmail.com>

Date: Sat Dec 17 21:28:14 2022 +0300

Создан скелет и конструктор для класса врагов

Author: Anna <koyvestoynen@bk.ru>

Date: Fri Dec 16 02:06:34 2022 +0300

Добавление конструктора, метода определения движения игрока и метода определения столкновения с границами карты

Author: Anna <koyvestoynen@bk.ru>

Date: Thu Dec 15 00:55:28 2022 +0300

Merge branch 'Player' into develop

Author: Anna <koyvestoynen@bk.ru>

Date: Thu Dec 15 00:54:05 2022 +0300

Добавление класса игрока-главного героя

Author: Anna <koyvestoynen@bk.ru>

Date: Wed Dec 14 19:23:52 2022 +0300

Добавлено описание класса Entity.h

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Wed Dec 14 16:48:51 2022 +0300

Добавила файл map.h - содержит шаблон карты. Выглядит симпатично. В main.cpp отладочный код (выводит карту)

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Wed Dec 14 16:32:03 2022 +0300

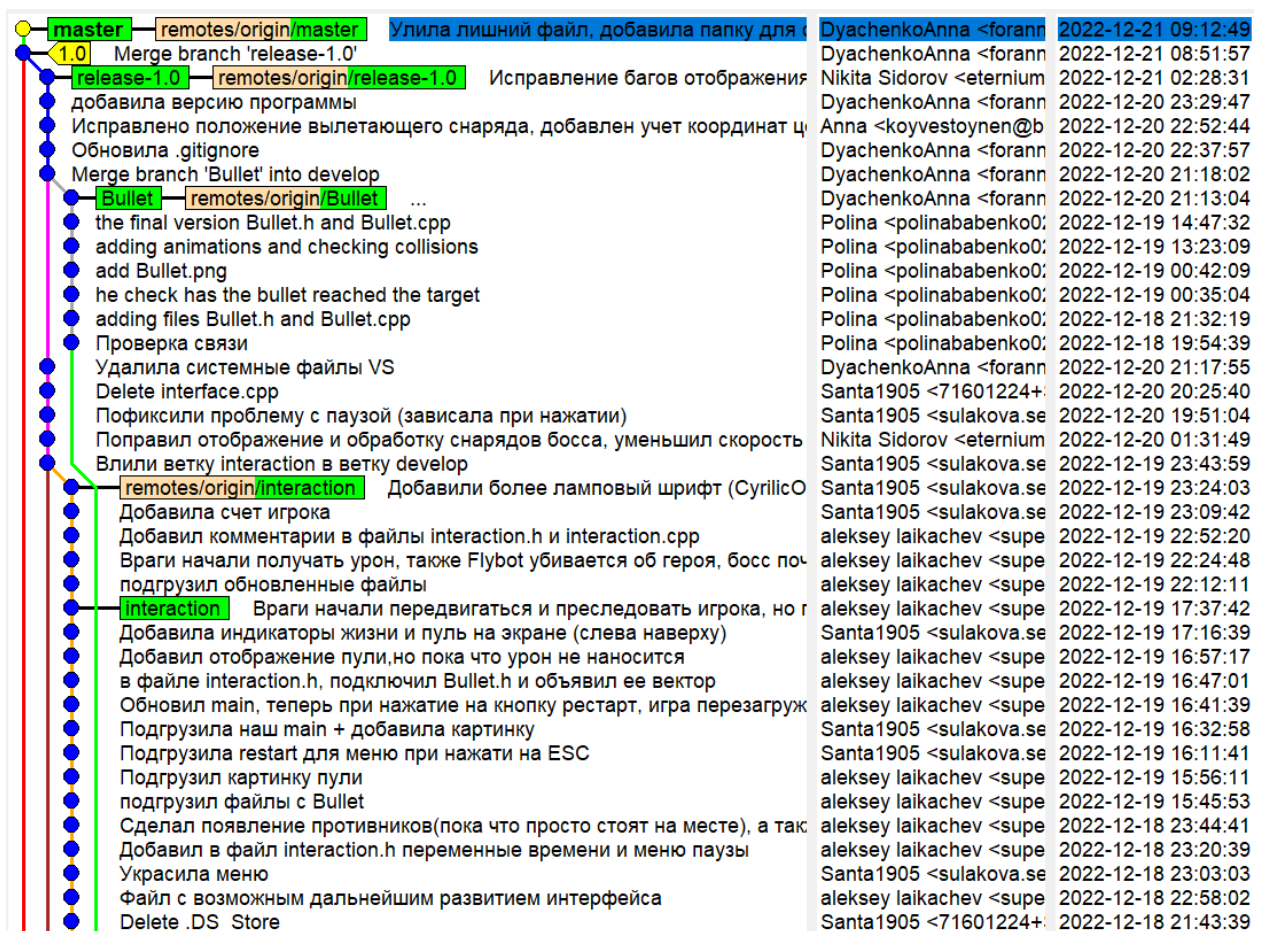
Добавила Constatnts.h - файл для описания констант. Задала размеры карты

Author: DyachenkoAnna <forannsstudy@gmail.com>

Date: Wed Dec 14 16:24:25 2022 +0300

initial commit, start a project

Графическая интерпретация:



Добавила меню в начало игры	Santa1905 <sulakova.se	2022-12-18 21:35:34
Добавила отрисовку карты и танка	Santa1905 <sulakova.se	2022-12-18 17:53:33
Добавил файл interaction.cpp - файл в котором будет отрисовываться и	aleksey laikachev <supe	2022-12-18 17:43:51
подгрузили файлы с develop	aleksey laikachev <supe	2022-12-18 17:24:55
Добавила изображения, в которые будем подгружать файлы, тестуру и	Sentyabrina Sulakova <s	2022-12-18 15:07:56
Добавил файл interaction.h - файл объявляющий класс Engine	aleksey laikachev <supe	2022-12-18 14:28:29
Добавила геттеры статуса врага, имени, таймера атаки босса (GetStatu	DyachenkoAnna <forann	2022-12-19 21:47:40
Добавлены комментарии	Anna <koyvestoynen@b	2022-12-18 18:22:40
Дописаны методы для окрашивания при нанесении урона и положени	Anna <koyvestoynen@b	2022-12-18 18:14:52
Обновление класса героя, дописано возвращение позиции спрайта геро	Anna <koyvestoynen@b	2022-12-18 18:09:21
Добавлен метод для отрисовки колес и пушки	Anna <koyvestoynen@b	2022-12-18 15:13:32
Обновление класса героя для отображения отрисовки колес и пушки	Anna <koyvestoynen@b	2022-12-18 15:11:37
Добавлены методы для движения главного героя, а также метод для	Anna <koyvestoynen@b	2022-12-18 15:01:15
Обновлен класс Player	Anna <koyvestoynen@b	2022-12-18 14:56:12
Замерджили ветку enemy в develop	Nikita Sidorov <eternium	2022-12-18 03:46:52
enemy remotes/origin/enemy Убрала отладочный код, ветка готова к	DyachenkoAnna <forann	2022-12-18 03:35:40
Дописал метод animation для босса, добавил метод update	Nikita Sidorov <eternium	2022-12-18 03:30:10
Для флайбота добавлена анимация	DyachenkoAnna <forann	2022-12-18 02:35:53
Метод action() дописан полностью. Враги перемещаются к цели	DyachenkoAnna <forann	2022-12-18 01:54:35
Добавление методов setdirection и action. Враги движутся к цели	Nikita Sidorov <eternium	2022-12-18 01:44:55
Дописана функция выбора цели для флайбота	DyachenkoAnna <forann	2022-12-18 00:57:07
Добавлен метод получения координат цели (для движения врагов)	Nikita Sidorov <eternium	2022-12-18 00:50:42
Дописал конструктор для босса	Nikita Sidorov <eternium	2022-12-18 00:31:42
Добавлена о-о-очень простая функция, отнимающая здоровье	DyachenkoAnna <forann	2022-12-17 23:47:21
Добавлена проверка столкновения с картой. Обновлено функция получе	DyachenkoAnna <forann	2022-12-17 23:38:11
Добавлена проверка столкновения с картой. Обновлено функция получе	DyachenkoAnna <forann	2022-12-17 23:27:56
Теперь флайботы появляются на случайных спавнах	DyachenkoAnna <forann	2022-12-17 22:35:38
Забыла отрисовать сердечко	DyachenkoAnna <forann	2022-12-17 22:26:00
Флайбот теперь появляется на карте	DyachenkoAnna <forann	2022-12-17 22:19:31
Создан скелет и конструктор для класса врагов	Nikita Sidorov <eternium	2022-12-17 21:28:14
Коммит ради коммита. Гит просит	DyachenkoAnna <forann	2022-12-17 21:37:42
Добавление конструктора, метода определения движения игрока и метод	Anna <koyvestoynen@b	2022-12-16 02:06:34
Merge branch 'Player' into develop	Anna <koyvestoynen@b	2022-12-15 00:55:28
Добавление класса игрока-главного героя	Anna <koyvestoynen@b	2022-12-15 00:54:05
Добавлено описание класса Entity.h	Anna <koyvestoynen@b	2022-12-14 19:23:52
Добавила файл map.h - содержит шаблон карты. Выглядит симпатично. В m	DyachenkoAnna <forann	2022-12-14 16:48:51
Добавила Constatnts.h - файл для описания констант. Задала размеры карт	DyachenkoAnna <forann	2022-12-14 16:32:03
initial commit, start a project	DyachenkoAnna <forann	2022-12-14 16:24:25

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы создана игра на языке программирования C++, которую можно считать современной версией игры “Tank 1990”.

В процессе работы применялась система контроля версий Git. Реализованы все прецеденты. Сбои/зависания программы во время тестирования и использования не наблюдаются. Программа написана с учетом принципа раздельной компиляции. Реализовано освобождение памяти для класса Engine (в нем освобождается вся используемая память). Нет неиспользуемых методов, атрибутов и переменных. Все конструкции в программе используются для работы с классами. Нет конструкций, без которых можно было обойтись. К отчету приложены диаграммы классов и вариантов использования. Поставленная цель достигнута.

ПРИЛОЖЕНИЕ А

main.cpp

```
// version 1.0

#include <ctime>
#include "interaction.h"

/*
Здесь вход в программу.
используем для вектора тип sf::Vector2f
*/

int main()
{
    srand(unsigned(time(0)));
    int way = 2; //для определения действия при завершении игры
    while (way != 0)
    {
        Engine* level = new Engine(); //Создали объект = запустили игру
        way = level->play(way); //начинаем игру
        delete level; // Удалили объект (игру)
    }

    return 0;
}
```


ПРИЛОЖЕНИЕ Б

interaction.h

```
#pragma once
#include "Enemy.h"//файл с врагами
#include "Player.h"//файл с игроком
#include "Bullet.h"//Файл с пулей
class Engine {
public:
    Engine();//конструктор
    ~Engine();//деструктор
    int play(int number);//метод игры

private:
    Player* Hero;//объект игры

    sf::Font font;// шрифт для текста
    sf::Text text;// текст

    std::vector<Bullet> bullets;// вектор пуль
    std::vector<Enemy> enemies;// вектор врагов
    sf::Clock clock;// аппаратный таймер, время процесса игры
    float time;// основное время, отвечает за скорость игры
    sf::Image map_image;//объект изображения для карты
    sf::Image allImage;//Все изображения, которые исползуются по верх карты
    sf::Sprite s_map;//спрайт для карты
    sf::Sprite Health;//для здоровья
    sf::Sprite GunDamage;//для уровня заряда пушки
    sf::Texture map;//текстура карты
    bool GameOver;//триггер для окончания игры
    int MainMenu(sf::RenderWindow& target);//меню при старте
    int RestartMenu(sf::RenderWindow& target);//меню внутри игры
};
```

ПРИЛОЖЕНИЕ В

interaction.cpp

```
#include "interaction.h"//заголовочный файл
#include "map.h"//файл с заранее прописанной картой
Engine::Engine()
{
    map_image.loadFromFile("images/map.png");//загружаем файл для карты
    map.loadFromImage(map_image);//заряжаем текстуру картинкой
    s_map.setTexture(map);//заливаем текстуру спрайтом
    allImage.loadFromFile("images/robots.png");//загрузили изображения
    объектов
    clock.restart();//перезагружает время
    Hero = new Player(allImage, 500, 500, 70, 80, "hero");//Создаем объект
    героя
    GameOver = false;//игра "не окончена"

    font.loadFromFile("images\\CyrilicOld.ttf");//шрифт загрузили
    text.setFont(font);
    text.setCharacterSize(24);
    text.setStyle(sf::Text::Bold);
    text.setPosition(1000, 5);
}

Engine::~Engine()
{
    delete Hero;//удаляет объект игрок
    enemies.clear();//удаляет вектора врагов
    bullets.clear();//удаляет вектора пули
    //очищаем память
}

int Engine::play(int number)
{
    sf::RenderWindow window(sf::VideoMode(1280, 800), "Game",
sf::Style::Fullscreen);//окно сформировали, сделали на полный экран
    sf::Cursor cursor;//устанавливаем курсор-крестик (типа прицел)
    if (cursor.loadFromSystem(sf::Cursor::Cross))
        window.setMouseCursor(cursor);

    if (number != 1 && !MainMenu(window))//возвращает 0, если нажат выход
    {
        return 0;
    }
    float timerspaun = 0;//таймер для появления врагов
    float gunTimer = 0;//для контроля выстрелов гг
    float spaunlvl = 5000;//время нужное спауна
    float timerLVLup = 0;//повышаем уровень, со временем
    sf::Image healthImg;
    healthImg.loadFromFile("images/Health.png");
    sf::Texture healthTexture;
    healthTexture.loadFromImage(healthImg);
    healthTexture.setRepeated(true);//Чтоб не рисовать - повторяем один и тот
    же texture в спрайте
    Health.setTexture(healthTexture);
```

```

    sf::Image gunDamageImg;
    gunDamageImg.loadFromFile("images/Bullet.png");//загрузили изображения
пуля
    sf::Texture gunDamageTexture;
    gunDamageTexture.loadFromImage(gunDamageImg);
    gunDamageTexture.setRepeated(true);
    GunDamage.setTexture(gunDamageTexture);
    //fix release 1.0 ++
    //Health.setTextureRect(sf::IntRect(0, 0, 32, 32));//Поставили картинку
здоровья
    Health.setTextureRect(sf::IntRect(0, 0, 33, 32));//Поставили картинку
здоровья //картинка одного сердечка на самом деле 33*32
    //fix release 1.0 --
    Health.setScale(0.5, 0.5);
    Health.setPosition(10, 10);
    GunDamage.setTextureRect(sf::IntRect(0, 0, 70, 348));//Поставили картинку
заряда пушки
    GunDamage.setScale(0.1, 0.1);
    GunDamage.setPosition(10, 30);
    while (window.isOpen() && !GameOver)
    {
        //Пока окно открыто и игра не закончена
        sf::Event event;//если нажата клавиша
        while (window.pollEvent(event))
        {
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))
            {
                //нажали esc = открыли меню, игра на паузе
                switch (RestartMenu(window))
                {
                    {
                        case 0://если 0, то закрывает игру
                            window.close();
                            return 0;
                        case 1://если 1, то запуск игры
                            window.close();
                            return 1;
                        default:
                            clock.restart(); //пофиксили баг - теперь
работает пауза без косяков
                            break;
                    }
                }
            }
            time = clock.getElapsedTime().asMicroseconds(); //дать прошедшее
время в микросекундах
            clock.restart(); //перезагружает время
            time = time / 800; //скорость игры
            timerspaun += time;
            gunTimer += time;
            timerLVLup += time;
            if (timerLVLup > 30000)
            {
                //30 сек и повысили сложность + появился босс
                Enemy* anotherEnemy = new Enemy(allImage, 200, 200, 120, 90,
"BOSSbot", TileMapMy);
                enemies.push_back(*anotherEnemy);//указатель, чтоб подгрузить
картинку
                spawnlvl -= 500;
            }
        }
    }

```

```

        if (spaunlvl < 2000)//максимальная сложность, враг появляется
раз в 2 секунды
    {
        spaunlvl = 2000;
    }
    timerLVLup = 0;
    timerspaun = 0;
}
//
if (timerspaun > spaunlvl && enemies.size() < 10)
{
    //пусть будет не больше 10 врагов
    Enemy* anotherEnemy = new Enemy(allImage, 200, 200, 45, 65,
"flybot", TileMapMy);
    enemies.push_back(*anotherEnemy);//указатель, чтобы
подгрузить картинку
    timerspaun = 0;
}

Hero->update(time, TileMapMy, event);//Герой сделал свой ход
//fix release 1.0 ++
//Health.setTextureRect(sf::IntRect(0, 0, 32 * (Hero->Gethealth() /
20), 32));
    Health.setTextureRect(sf::IntRect(0, 0, 33 *
((Hero->Gethealth())/20.0), 32)); //картинка одного сердечка на самом деле
33*32
    //fix release 1.0 --
    if (gunTimer > 1000)//максимальный урон = большая полоска
    {
        GunDamage.setTextureRect(sf::IntRect(0, 0, 70 * 10, 348));
    }
    else//чем меньше полоска, тем меньше урон
    {
        GunDamage.setTextureRect(sf::IntRect(0, 0, 70 * (gunTimer /
100), 348));
    }

    if (gunTimer > 150 &&
sf::Keyboard::isKeyPressed(sf::Keyboard::Space))
    {
        int damage = gunTimer / 10;
        if (damage > 100)
        {
            damage = 100;
        }
        if (damage > 20)
        {
            sf::Vector2f HeroXY = Hero->GetgunXY();//если пуля
большая, то выстреливает из дула
            Bullet* anotherBullet = new Bullet(allImage, HeroXY.x,
HeroXY.y, 25, 25, Hero->GetRotation(), "HeroBullet", damage);
            bullets.push_back(*anotherBullet);//указатель, чтоб
подгрузить картинку
        }
        else
        {
            sf::Vector2f HeroXY = Hero->GetXY();//если пулька
маленькая, то высреливает из центра героя, т.к. иначе проблема с хитбоксами

```

```

        Bullet* anotherBullet = new Bullet(allImage, HeroXY.x,
HeroXY.y, 10, 10, Hero->GetRotation(), "HeroBullet", damage);
        bullets.push_back(*anotherBullet); //указатель, чтоб
подгрузить картинку
    }
    gunTimer = 0;
}
std::vector<Enemy>::iterator iterEnemies =
enemies.begin(); //итераторы для врагов в начало + создаем их
std::vector<Bullet>::iterator iterBullet =
bullets.begin(); //итераторы для пуль в начало + создаем их
while (iterBullet != bullets.end())
{
    if (iterBullet->isAlive())
    {
        iterBullet->update(time, TileMapMy);
        ++iterBullet; //проходим по каждому объекту в векторе
    }
    else
    {
        iterBullet = bullets.erase(iterBullet); //стереть из
списка
    }
}
while (iterEnemies != enemies.end())
{
    if (iterEnemies->isAlive())
    {
        sf::Vector2f BufXYHero = Hero->GetXY();
        iterEnemies->SetAim(BufXYHero); //цель - герой
        iterEnemies->update(time); //враг сходил
        if (iterEnemies->GetName() == "BOSSbot" &&
iterEnemies->GetBOSSdamagetime() > 2000) //босс стреляет раз в 2 секунды
        {
            sf::Vector2f BufXYEnemy =
iterEnemies->GetXY(); //берем координаты
            float rotation = atan2(BufXYHero.y -
BufXYEnemy.y, BufXYHero.x - BufXYEnemy.x) * 180 / 3.14159265 + 90;
            //вычисляем угол поворота
            Bullet* anotherBullet = new Bullet(allImage,
BufXYEnemy.x, BufXYEnemy.y, 30, 30, rotation, "BossBullet", 25);
            bullets.push_back(*anotherBullet); //создаем и
добавляем в вектор пулю
            iterEnemies->recetBOSSdamagetime();
        }

        iterBullet = bullets.begin();
        while (iterBullet != bullets.end())
        { //Находим пересечение хитбоксов пули и врагов с
героем, далее уничтожается пуля, а остальные получают урон
            if (iterBullet->isAlive())
            {
                if (iterBullet->GetName() != "BossBullet"
&& iterBullet->GetRect().intersects(iterEnemies->GetRect()))
                {
                    iterEnemies->struck(iterBullet->GetDamage());
                }
            }
        }
    }
}

```

```

        if (iterBullet->GetName() != "HeroBullet"
        && iterBullet->GetRect().intersects(Hero->GetRect()))
        {
            Hero->struck(iterBullet->GetDamage());
        }
        ++iterBullet;
    }
    else
    {
        iterBullet = bullets.erase(iterBullet);
//стереть из списка
    }
}

if (Hero->GetRect().intersects(iterEnemies->GetRect()))
{
    //При столкновении героя и врагов
    if (iterEnemies->GetName() == "BOSSbot")
    {
        //может произойти так, что вам нанесётся
        урон от взрыва после смерти босса. это нормально)
        if (iterEnemies->GetStatus() !=
"anikilled")
        {
            if
            (iterEnemies->GetBOSSdamagetimer() > 500)
            {
                //При пересечении хитбоксов
                Hero->struck(20);

                iterEnemies->recetBOSSdamagetimer();
            }
        }
    }
    else
    {
        iterEnemies->struck(100); //Другие же
        получают урон, несовместимый с жизнью
        //fix release 1.0 ++
        //if (iterEnemies->GetStatus() !=
"anikilled")
        //а тут учтём условие смерти бота, чтобы
        он не мог нанести урон после своей смерти:
        if ((iterEnemies->GetStatus() !=
"anikilled") && (iterEnemies->GetStatus() != "killed"))
        //fix release 1.0 --
        {
            Hero->struck(20);
        }
    }
}
++iterEnemies;
}
else
{

```

```

        if (iterEnemies->GetName() == "BOSSbot")
        {
            Hero->Addscore(500); //Босс убит - получили 500
ОЧКОВ
        }
        else if (iterEnemies->GetName() == "flybot")
        {
            Hero->Addscore(100); //flybot убит - получили 100
ОЧКОВ
        }
        iterEnemies = enemies.erase(iterEnemies); //стереть из
списка
    }

    }
    GameOver = !Hero->isAlive();
    //drawing ->
    window.clear();
    ///////////////////////////////////Рисуем карту////////////////////////////////////
    for (int i = 0; i < HEIGHT_MAP; i++)
        for (int j = 0; j < WIDTH_MAP; j++)
        {
            if (TileMapMy[i][j] == ' ')
s_map.setTextureRect(sf::IntRect(0, 0, 32, 32)); //если встретили символ
пробел, то рисуем 1й квадратик
            else if (TileMapMy[i][j] == '0')
s_map.setTextureRect(sf::IntRect(32, 0, 32, 32)); //если встретили символ 0, то
рисуем 2й квадратик
            else if (TileMapMy[i][j] == 'b')
s_map.setTextureRect(sf::IntRect(64, 0, 32, 32)); //если встретили символ b, то
рисуем 3й квадратик
            else if (TileMapMy[i][j] == 'p')
s_map.setTextureRect(sf::IntRect(96, 0, 32, 32)); //если встретили символ p, то
рисуем 4й квадратик
            else if (TileMapMy[i][j] == 'h')
s_map.setTextureRect(sf::IntRect(128, 0, 32, 32)); //если встретили символ h, то
рисуем 5й квадратик
            else s_map.setTextureRect(sf::IntRect(160, 0, 32, 32));
s_map.setPosition(j * 32, i * 32); //по сути раскидывает
квадратики, превращая в карту.
            //то есть задает каждому из них позицию.
            window.draw(s_map); //рисуем квадратiki на экран
        }
    Hero->draw(window); //рисуется герой-танк
    iterEnemies = enemies.begin();
    while (iterEnemies != enemies.end()) {
        if (iterEnemies->GetName() != "BOSSbot")
        {
            iterEnemies->draw(window);
            //босс выше всех летает
        }
        ++iterEnemies;
    }
    iterBullet = bullets.begin();
    while (iterBullet != bullets.end()) {
        window.draw(iterBullet->sprite);
        ++iterBullet;
    }
}

```

```

        iterEnemies = enemies.begin();
        while (iterEnemies != enemies.end()) {
            if (iterEnemies->GetName() == "BOSSbot")
            {
                iterEnemies->draw(window);
            }
            ++iterEnemies;
        }
        window.draw(GunDamage);
        window.draw(Health);
        text.setString("Score: " +
std::to_string(Hero->Getscore())); //преобразовали цифру в текст и показали
        window.draw(text);

        window.display();
    }
    if (RestartMenu(window))
    {
        window.close();
        return 1;
    }

    window.close();
    return 0;
}

int Engine::RestartMenu(sf::RenderWindow& target)
{
    sf::Texture menuTexturePlay, menuTextureQuit, menuTextureRestart;
    menuTexturePlay.loadFromFile("images/Play.png");
    menuTextureQuit.loadFromFile("images/Quit.png");
    menuTextureRestart.loadFromFile("images/Restart.png");
    sf::Sprite menuPlay(menuTexturePlay), menuQuit(menuTextureQuit),
menuRestart(menuTextureRestart);
    bool isMenu = 1;
    int menuNum = 0;
    if (!GameOver)
    {
        menuRestart.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 -
120);
        menuPlay.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16);
    }
    else
    {
        menuPlay.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 - 120);
        menuRestart.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16);
    }
    menuQuit.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 + 120);
    //Расставили кнопки
    ////////////////////////////////////////МЕХЮ////////////////////////////////////
    while (isMenu)
    {
        menuRestart.setColor(sf::Color::White);
        menuPlay.setColor(sf::Color::White);
        menuQuit.setColor(sf::Color::White);
        menuNum = 0;
        sf::RectangleShape rectangle(sf::Vector2f(20, 20));

```



```

rectangle.setSize(sf::Vector2f(WIDTH_MAP * 32 - 40, HEIGHT_MAP * 32
- 40));
if (!GameOver)
{
    rectangle.setFillColor(sf::Color(255, 228, 200, 1));
    rectangle.setPosition(sf::Vector2f(20, 20));
    if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 - 120,
310, 110).contains(sf::Mouse::getPosition()))
    {
        menuRestart.setColor(sf::Color::Blue); menuNum = 1; // 1
= рестарт
    }
    if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16, 310,
110).contains(sf::Mouse::getPosition()))
    {
        menuPlay.setColor(sf::Color::Blue); menuNum = 2; // 2 =
играть
    }
    if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 + 120,
310, 110).contains(sf::Mouse::getPosition()))
    {
        menuQuit.setColor(sf::Color::Blue); menuNum = 0; // 0 =
ВЫЙТИ
    }
}
else
{
    //если игра окончена, то саму игру надо закрасить
    target.clear(sf::Color(255, 228, 200));
    text.setCharacterSize(64);
    text.setPosition(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 -
120);
    if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16, 310,
110).contains(sf::Mouse::getPosition()))
    {
        menuRestart.setColor(sf::Color::Blue); menuNum = 1;
    }
    if (sf::IntRect(WIDTH_MAP * 16 - 155, HEIGHT_MAP * 16 + 120,
310, 110).contains(sf::Mouse::getPosition()))
    {
        menuQuit.setColor(sf::Color::Blue); menuNum = 0;
    }
}

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
{
    return menuNum;
}
if (!GameOver)
{
    target.draw(rectangle);
    target.draw(menuPlay);
}
target.draw(menuRestart);
target.draw(menuQuit);

```

```

        text.setString("Score: " + std::to_string(Hero->Getscore()));
        target.draw(text);

        target.display();
    }
    return 0;
}

int Engine::MainMenu(sf::RenderWindow& target)
{
    {
        sf::Texture menuTexturePlay, menuTextureQuit, menuBackground;
        menuTexturePlay.loadFromFile("images/Play.png");
        menuTextureQuit.loadFromFile("images/Quit.png");
        menuBackground.loadFromFile("images/jogaGame.png");
        sf::Sprite menuPlay(menuTexturePlay), menuQuit(menuTextureQuit),
menuBg(menuBackground);
        bool isMenu = 1;
        int menuNum = 0;
        menuPlay.setPosition(100, 200);
        menuQuit.setPosition(100, 500);
        menuBg.setPosition(0, 0);
        ///////////////////////////////////МЕНЮ/////////////////////////////////
        while (isMenu)
        {
            menuPlay.setColor(sf::Color::White);
            menuQuit.setColor(sf::Color::White);
            menuNum = 0;
            target.clear();

            if (sf::IntRect(100, 200, 310,
110).contains(sf::Mouse::getPosition())) { menuPlay.setColor(sf::Color::Blue);
menuNum = 1; }

            if (sf::IntRect(100, 500, 310,
110).contains(sf::Mouse::getPosition())) { menuQuit.setColor(sf::Color::Blue);
menuNum = 2; }

            if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
            {
                if (menuNum == 1)//если нажали кнопку играть, то
запускаем игру
                {
                    isMenu = false;
                    return 1;
                }
                if (menuNum == 2)//если нажали кнопку выйти, то
закрыли игру
                {
                    isMenu = false;
                    return 0;
                }
            }

            //target = window
            target.draw(menuBg);
            target.draw(menuPlay);
            target.draw(menuQuit);
            target.display();
        }
    }
}

```

```
    }  
    return 0;  
}  
}
```

ПРИЛОЖЕНИЕ Г

Entity.h

```
#ifndef __Entity__
#define __Entity__

#include <SFML/Graphics.hpp>

class Entity {
public:
    sf::Texture texture;
    sf::Sprite sprite;
    sf::FloatRect GetRect() {return sf::FloatRect(x,y,w,h);}; //прямоугольник
    для поиска пересечения объектов
    Entity(sf::Image& image, float X, float Y, int W, int H, sf::String Name)
    {
        x = X; y = Y; w = W; h = H; name = Name; moveTimer = 0;
        speed = 0; health = 100; dx = 0; dy = 0; //dx и dy - на сколько
        менять позицию
        life = true; isMove = false;
        texture.loadFromImage(image); //установка текстуры
        sprite.setTexture(texture); //вырезка спрайта из текстуры
        sprite.setOrigin(w / 2, h / 2); //координаты в центре спрайта
    }
    bool isAlive() { return life; }; //возвращает позицию спрайта
protected:

    float dx, dy, x, y, speed, moveTimer;
    int w, h, health;
    bool life, isMove;
    sf::String name; //враги по именам
    enum { left, right, up, down, stay, fly } state; //тип пересечение -
    состояние объекта
};

#endif
```

ПРИЛОЖЕНИЕ Д

Player.h

```
#ifndef __Player_h__
#define __Player_h__

#include "Entity.h"
#include "Constants.h"
/*
Класс игрок. Главный герой

Переменные от родителя:
float dx, dy, x, y, speed, moveTimer;
int w, h, health;
bool life, isMove;
sf::Texture texture;
sf::Sprite sprite;
sf::String name;
*/

class Player :protected Entity {
public:
    Player(sf::Image& image, float X, float Y, int W, int H, sf::String
Name);
    bool isAlive() { return life; };
    sf::FloatRect GetRect() { return sf::FloatRect(x+5, y+5, w-5, h-5); };
    float GetRotation() { return gunrotation; }; //берем направление пушки
    int Getscore() { return score; };
    void Addscore(int SCR) { score += SCR; };
    int Gethealth() { return health; };
    int update(float time, sf::String TileMap[HEIGHT_MAP], sf::Event
event); // Жизнь объекта, ф-я вызывается в основной программе
    void struck(int damage);
    sf::Vector2f GetXY() { return sf::Vector2f(x + w / 2, y + h / 2); }; //
Возвращает позицию спрайта героя
    sf::Vector2f GetgunXY(); // возвращает позицию конца пушки
    void draw(sf::RenderTarget& target);
private:
    // Будет вызываться внутри
    int control(sf::Event event);
    int checkCollisionWithMap(float, float, sf::String
TileMap[HEIGHT_MAP]); //Проверка столкновений с картой
    sf::Sprite wheelR; //правые колеса
    sf::Sprite wheelL; //левые
    sf::Sprite gun; //пушка
    sf::Sprite frame; //рамка
    float gunrotation;
    int score;
    float struckTimer; //для анимации нанесения урона
    void animation();
};

#endif
```

ПРИЛОЖЕНИЕ Е

Player.cpp

```
#include "Player.h"
#include "Constants.h"

// Конструктор
Player::Player(sf::Image &image, float X, float Y, int W, int H, sf::String
Name):Entity(image,X,Y,W,H,Name)
{
    score = 0;
    struckTimer = 0;
    state = up;
    frame.setTexture(texture);
    wheelL.setTexture(texture);
    wheelR.setTexture(texture);
    gun.setTexture(texture);

    wheelL.setScale(0.9, 1);
    wheelR.setScale(0.9, 1);
    frame.setScale(0.9, 0.9);
    gun.setScale(0.9, 0.9);
    //размеры
    wheelL.setTextureRect(sf::IntRect(1, 80, 19, 77));
    wheelL.setOrigin(w / 2 + 10, h / 2);

    wheelR.setTextureRect(sf::IntRect(20, 81, 19, 77));
    wheelR.setOrigin(- w / 2 + 10, h / 2);

    frame.setTextureRect(sf::IntRect(84, 80, 70, 80));
    frame.setOrigin(w / 2, h / 2);

    gun.setTextureRect(sf::IntRect(181, 81, 36, 78));
    gun.setOrigin(18, 58);
    //позиция-центр спрайтов
    frame.setPosition(x, y);
    wheelL.setPosition(x, y);
    wheelR.setPosition(x, y);
    gun.setPosition(x, y);
    gunrotation = 0;
}

//Для определения движения
int Player::control(sf::Event event)
{
    state = stay;
    sf::Vector2i DXY = sf::Mouse::getPosition();
    //sf::Vector2i DXY = sf::Vector2i(event.mouseMove.x, event.mouseMove.y);

    //fix release 1.0 ++
    //gunrotation = atan2(-y + DXY.y, -x + DXY.x) * 180 / 3.14159265 + 90;
    gunrotation = atan2(-y + DXY.y - 40, -x + DXY.x - 35) * 180 / 3.14159265
+ 90;
    //fix release 1.0 --
```

```

        if (sf::Keyboard::isKeyPressed) { //если нажата клавиша
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::A)) {
                state = left; speed = 0.1;
            }
            else if (sf::Keyboard::isKeyPressed(sf::Keyboard::D)) {
                state = right; speed = 0.1;
            }
            else if ((sf::Keyboard::isKeyPressed(sf::Keyboard::W))) {
                state = up; speed = 0.085;
            }
            else if (sf::Keyboard::isKeyPressed(sf::Keyboard::S)) {
                state = down; speed = 0.085;
            }
        }

        return 0;
    }

int Player::checkCollisionWithMap(float Dx, float Dy, sf::String
TileMap[HEIGHT_MAP])
{
    for (int i = y / 32; i < (y + h) / 32; i++) //проходимся по элементам
    карты
        for (int j = x / 32; j < (x + w) / 32; j++)
        {
            if (TileMap[i][j] == '0' || TileMap[i][j] == 'b')
            {
                if (Dy > 0) { y = i * 32 - h; dy = 0; } //столкновение
                с нижним краем карты
                if (Dy < 0) { y = i * 32 + 32; dy = 0; } //с верхним
                краем карты
                if (Dx > 0) { x = j * 32 - w; } //с правым краем карты
                if (Dx < 0) { x = j * 32 + 32; } // с левым краем карты
            }
            if (TileMap[i][j] == 'h' && state == stay && moveTimer >
1000)
            {
                health += 10;
                //Если встали и стоим на хил поинте, то прибавляем
                здоровье
                moveTimer = 500;
                if (health > 500)
                {
                    health = 500;
                }
            }
        }

    return 0;
}

//Анимация
void Player::animation()
{
    if (state != stay)
    {
        if (moveTimer < 100)
        {

```

```

        wheelL.setTextureRect(sf::IntRect(1, 80, 19, 77));
        wheelR.setTextureRect(sf::IntRect(20, 81, 19, 77));
    }
    else if (moveTimer < 200)
    {
        wheelL.setTextureRect(sf::IntRect(40, 82, 19, 77));
        wheelR.setTextureRect(sf::IntRect(59, 82, 19, 77));
    }
    else
    {
        moveTimer = 0;
    }
    //Движение колес
}
gun.setRotation(gunrotation);
switch (state)
{
case Entity::left:
    frame.setRotation(-90);
    wheelL.setRotation(-90);
    wheelR.setRotation(-90);
    break;
case Entity::right:
    frame.setRotation(90);
    wheelL.setRotation(90);
    wheelR.setRotation(90);
    break;
case Entity::up:
    frame.setRotation(0);
    wheelL.setRotation(0);
    wheelR.setRotation(0);
    break;
case Entity::down:
    frame.setRotation(180);
    wheelL.setRotation(180);
    wheelR.setRotation(180);
    break;
default:
    break;
}
//повернули спрайты по направлению движения
if (struckTimer > 0)
{
    gun.setColor(sf::Color::Red);
    frame.setColor(sf::Color::Red);
    wheelL.setColor(sf::Color::Red);
    wheelR.setColor(sf::Color::Red);
}
else
{
    gun.setColor(sf::Color::White);
    frame.setColor(sf::Color::White);
    wheelL.setColor(sf::Color::White);
    wheelR.setColor(sf::Color::White);
}
//Красим при уроне
return;
}

```



```

//Движение главного игрока
int Player::update(float time, sf::String TileMap[HEIGHT_MAP], sf::Event event)
{
    if (!life)
    {
        return 0;
    }
    if (struckTimer > 0)
    {
        struckTimer -= time;
    }
    moveTimer += time;
    control(event);
    animation();
    switch (state)//тут делаются различные действия в зависимости от
    состояния
    {
        case right: dx = speed; dy = 0; break;//состояние идти вправо
        case left: dx = -speed; dy = 0; break;//состояние идти влево
        case up: dx = 0; dy = -speed; break;// состояние поднятия вверх
        case down: dx = 0; dy = speed; break;// состояние во время спуска
        case stay: dx = 0; dy = 0; break;//стоим
    }
    x += dx * time;
    checkCollisionWithMap(dx, 0, TileMap);//обрабатываем столкновение по X
    y += dy * time;
    checkCollisionWithMap(0, dy, TileMap);//обрабатываем столкновение по Y
    frame.setPosition(x + w / 2, y + h / 2); //задаем позицию спрайта
    wheelL.setPosition(x + w / 2, y + h / 2);
    wheelR.setPosition(x + w / 2, y + h / 2);
    gun.setPosition(x + w / 2, y + h / 2);
    if (health <= 0) { life = false; }
    speed = 0;
    return 0;
}

//fix release 1.0 ++
//переработка углов, изменение условий под них
//Положение пушки
sf::Vector2f Player::GetgunXY()
{
    sf::Vector2f buf = gun.getPosition();
    //std::cout << "угол: " << gunrotation << std::endl;
    if (gunrotation >= -45 && gunrotation < 45)
    {
        //верхний угол
        buf.x += 80 * cos((gunrotation - 90) / 180 * 3.14159265) + (70/4);
        buf.y += 80 * sin((gunrotation - 90) / 180 * 3.14159265) - (80/4);
    }
    else if (gunrotation >= 45 && gunrotation < 135)
    {
        //правый
        buf.x += 80 * cos((gunrotation - 90) / 180 * 3.14159265) - (80/4);
        buf.y += 80 * sin((gunrotation - 90) / 180 * 3.14159265) + (80/4);
    }
    else if (gunrotation >= 135 && gunrotation < 225)
    {
        //нижний

```

```

        buf.x += 80 * cos((gunrotation - 90) / 180 * 3.14159265) - (80/2);
        buf.y += 80 * sin((gunrotation - 90) / 180 * 3.14159265) - (70/4);
    }
    else if ((gunrotation >= 225 && gunrotation <= 270) || (gunrotation >= -90
&& gunrotation < -45))
    {
        //левый
        buf.x += 80 * cos((gunrotation - 90) / 180 * 3.14159265);
        buf.y += 80 * sin((gunrotation - 90) / 180 * 3.14159265) - (80/2);
    }

    return buf;
}
//fix release 1.0 --

//Окрашивание при нанесении урона
void Player::struck(int damage)
{
    health -= damage;
    struckTimer = 200; //на 200мс в красный цвет
}

//отрисовка
void Player::draw(sf::RenderTarget& target)
{
    target.draw(wheelL);
    target.draw(wheelR);
    target.draw(frame);
    target.draw(gun);
}

```

ПРИЛОЖЕНИЕ Ж

Bullet.h

```
#pragma once
#include "Entity.h";
#include "Constants.h"
class Bullet : public Entity
{
public:
    Bullet(sf::Image& image, float X, float Y, int W, int H, float rotation,
sf::String Name, int);
    void update(float time, sf::String TileMap[HEIGHT_MAP]);
    int GetDamage();
    sf::String GetName() { return name; };
    sf::FloatRect GetRect();
private:
    int damage;
    float rotationPer;// Поворот пули
    float TTL;
    void animation();
    void checkCollisionWithMap(float Dx, float Dy, sf::String
TileMap[HEIGHT_MAP]);
};
```

ПРИЛОЖЕНИЕ И

Bullet.cpp

```
#include "Bullet.h"

Bullet::Bullet(sf::Image& image, float X, float Y, int W, int H, float
rotation, sf::String Name, int DMG) :Entity(image, X, Y, W, H, Name)
{
    damage = DMG;
    TTL = 5000; //Время жизни, ограничивает дальность полета пули
    sprite.setOrigin(w / 2, h / 2);
    rotationPer = rotation;
    rotationPer -= 90;
    if (name == "HeroBullet")
    {
        if (damage > 20)
        {
            sprite.setScale(0.1 * (damage / 10), 0.1 * (damage / 10));
            //Размер пули зависит от damage
        }
        else
        {
            sprite.setScale(0.2, 0.2);
        }
        speed = 2;
        sprite.setTextureRect(sf::IntRect(0, 160, 49, 80));
        sprite.setRotation(rotationPer - 180);
    }
    else if (name == "BossBullet")
    {
        sprite.setScale(0.4, 0.4);
        sprite.setTextureRect(sf::IntRect(658, 0, 121, 121)); //+121
        speed = 3;
    }

    dx = cos(rotationPer / 180 * 3.14159265) * 0.1;
    dy = sin(rotationPer / 180 * 3.14159265) * 0.085;
    // направление движения от угла
    sprite.setPosition(x + w / 2, y + h / 2);
}

void Bullet::update(float time, sf::String TileMap[HEIGHT_MAP])
{
    moveTimer += time;
    x += dx * time * speed;
    checkCollisionWithMap(dx, 0, TileMap);
    y += dy * time * speed;
    checkCollisionWithMap(0, dy, TileMap);
    //Скорость регулируется отдельной переменной
    //Проверка на столкновения
    TTL -= time;
    if (TTL <= 0)
    {
        life = false;
    }
}
```

```

    }
    if (life)
    {
        sprite.setPosition(x + w / 2, y + h / 2);
    }
    else
    {
        animation();
        return;
    }
    animation();
}

int Bullet::GetDamage()
{
    life = false; // пуля достигает цели
    return damage;
}

sf::FloatRect Bullet::GetRect()
{
    sf::FloatRect BufRect;
    if (name == "BossBullet")
    {
        BufRect.left = x + 10;
        BufRect.top = y + 10;
        //Смещение квадрата
    }
    else
    {
        BufRect.left = x - w * (cos((rotationPer + 90) / 180 *
3.14159265));
        BufRect.top = y - h * (sin((rotationPer + 90) / 180 * 3.14159265));
        //Квадрат вписали в пулю
    }

    BufRect.height = h;
    BufRect.width = w;
    return BufRect;
}

void Bullet::animation()
{
    if (name == "HeroBullet")
    {
        if (moveTimer < 100)
        {
            sprite.setTextureRect(sf::IntRect(0, 160, 49, 80));
        }
        else if (moveTimer < 200)
        {
            sprite.setTextureRect(sf::IntRect(49, 160, 47, 80));
        }
        else if (moveTimer < 300)
        {
            sprite.setTextureRect(sf::IntRect(95, 160, 48, 80));
        }
    }
}

```

```

        else if (moveTimer < 400)
        {
            sprite.setTextureRect(sf::IntRect(142, 160, 48, 80));
        }
        else if (moveTimer < 500)
        {
            sprite.setTextureRect(sf::IntRect(189, 160, 48, 80));
        }
        else if (moveTimer < 600)
        {
            sprite.setTextureRect(sf::IntRect(237, 160, 48, 80));
            moveTimer = 0;
        }
        else if (moveTimer < 700)
        {
            sprite.setTextureRect(sf::IntRect(219, 80, 48, 80));
            moveTimer = 0;
        }
        else if (moveTimer < 800)
        {
            sprite.setTextureRect(sf::IntRect(266, 80, 48, 80));
        }
        else
        {
            moveTimer = 0;
        }
    }
    else if (name == "BossBullet")
    {
        if (moveTimer < 100)
        {
            sprite.setTextureRect(sf::IntRect(658, 0, 121, 121));
        }
        else if (moveTimer < 200)
        {
            sprite.setTextureRect(sf::IntRect(658, 121, 121, 121)); //+121
        }
        else
        {
            moveTimer = 0;
        }
    }
    // смена картинок
}

void Bullet::checkCollisionWithMap(float Dx, float Dy, sf::String
TileMap[HEIGHT_MAP])
{
    for (int i = y / 32; i < (y + h) / 32; i++)
        for (int j = x / 32; j < (x + w) / 32; j++)
        {
            if (TileMap[i][j] == '0' || TileMap[i][j] == 'b')
            {
                life = false;
            }
        }
}

```

```
return;}
```

ПРИЛОЖЕНИЕ К

Enemy.h

```
#pragma once
#include "Entity.h";
#include "Constants.h"

class Enemy :public Entity {
public:
    //Передаем так же строку карты, чтоб все время не создавалась память при
    вызовах функций
    Enemy(sf::Image& image, float X, float Y, int W, int H, sf::String Name,
    sf::String TileMapEnemy[HEIGHT_MAP]);
    int update(float time); //Жизнь объекта, функция вызывается в
    основной программе
    int SetAim(sf::Vector2f); //Устанавливает положение движения
    sf::Vector2f GetXY() { return sf::Vector2f(x, y); }; //Возвращает
    позицию спрайта Enemy
    void SetStatus(sf::String St) { status = St; };
    void struck(int damage);
    sf::FloatRect GetRect(); //Переопределили функцию, чтобы подвинуть хитбокс
    void draw(sf::RenderTarget& target); //Так как нужно много за раз
    нарисовать, то вынесем в отдельный метод
    sf::String GetStatus() { return status; }; // статус объекта
    sf::String GetName() { return name; }; // босс или флайбот
    void recetBOSSdamagetimer() { BOSSdamagetimer = 0; };// для сброса
    таймера атаки босса
    float GetBOSSdamagetimer() { return BOSSdamagetimer; };// вернули таймер
private:
    //Будет вызываться внутри, так что инкапсулирую функции
    float BOSSdamagetimer;
    sf::Sprite BossPart; //босс состоит из 2х частей. Это - голова,
    которая поставится на движущуюся часть (пилы)
    sf::Sprite healthSprite; //спрайт для полоски HP
    sf::Vector2f XYAim; //координата спрайта игрока
    sf::String status;
    int action(float); //действия врагов
    int animation();
    void SetDirection(); //вычисление направления по координатам,
    взятых с SetAim
    int checkCollisionWithMap(float, float);
    sf::Vector2f SpaunTarget(); //смотрим где наши спауны на карте
    sf::String TileMap[HEIGHT_MAP]; //карта
};
```


ПРИЛОЖЕНИЕ Л

Enemy.cpp

```
#include "Enemy.h";
/*
Есть два типа врагов:
- flybot летит к гг суециднуться
- BOSSbot летает от спауна к спауну, стреляет и пилит (когда пилит, не стреляет)
*/

Enemy::Enemy( sf::Image& image, float X, float Y, int W, int H, sf::String
Name, sf::String TileMapEnemy[HEIGHT_MAP]) :Entity(image, X, Y, W, H, Name)
// ссылка на картинку, начальные координаты на общей картинке, ширина и высота
спрайта,
{
    speed = 0.1; // задаем скорость движения
    BOSSdamagetimer = 0; //?
    for (int i = 0; i < HEIGHT_MAP; i++)
    {
        TileMap[i] = TileMapEnemy[i];
    } //Скопировали карту
    isMove = false;
    status = "WTFBOT"; // инициировали статус
    state = fly; //летающее положение
    sf::Vector2f XY = SpaunTarget(); // находим начальные координаты
    x = XY.x;
    y = XY.y;
    healthSprite.setTexture(texture);
    //каждому спрайту по текстуре!!!
    if (name == "flybot")
    {
        sprite.setTextureRect(sf::IntRect(12, 5, 57, 68)); //w = 45; h = 63
        //Подрезали спрайт
        healthSprite.setTextureRect(sf::IntRect(0, 0, 34, 8));
        //Подрезали спрайт
        healthSprite.setPosition(XY.x + 10, XY.y - 10);
        //Поставили в точку
    }
    else if (name == "BOSSbot")
    {
        //w = 160; h = 60
        health = 500;
        BossPart.setTexture(texture);
        BossPart.setTextureRect(sf::IntRect(754, 216, 145, 128));
        sprite.setTextureRect(sf::IntRect(320, 0, 325, 292));

        healthSprite.setTextureRect(sf::IntRect(2, 252, 167, 25)); //
здоровье
        healthSprite.setScale(0.5, 0.5); // уменьшили спрайт здоровья
        BossPart.setScale(0.5, 0.5); //

        sprite.setScale(0.5, 0.5); // Уменьшили картинку в 2 раза
        BossPart.setOrigin(64, 72); // задали центр головы
        sprite.setOrigin(162, 146); // Нашли центр методом научного тыка и
поставили спрайту
```

```

        BossPart.setPosition(XY.x + W / 2 - 5, XY.y + H / 2 - 10); //
сместили голову
        // у боссов свои заморочки...
    }
    sprite.setPosition(XY.x + W / 2, XY.y + H / 2); // центр спрайта
}

int Enemy::checkCollisionWithMap(float Dx, float Dy)
{
    for (int i = y / 32; i < (y + h) / 32; i++) //проходимся по элементам
карты
        for (int j = x / 32; j < (x + w) / 32; j++)
        {
            if (TileMap[i][j] == '0' ) //если элемент наш тайлик земли, то
            {
                isMove = false;
                SetDirection(); // выбираем направление движения
            }
        }
    return 0;
}

void Enemy::SetDirection()
{
    float rotation = (atan2(XYAim.y - y, XYAim.x - x)); // вращение по
радианам
    // задаем угол поворота по направлению новой цели/спауна
    dx = cos(rotation) * 0.1; //
    dy = sin(rotation) * 0.08; //Изда разницы размеров по x & y умножаем
соответственно

}
sf::Vector2f Enemy::SpaunTarget()
{
    int countOfSpauns = 0;
    float Sy = 100;
    float Sx = 100;
    for (int i = 0; i < HEIGHT_MAP; i++) //проходимся по элементам карты
        for (int j = 0; j < WIDTH_MAP; j++)
        {
            if (TileMap[i][j] == 's' ) //если элемент наш тайлик земли, то
            {
                countOfSpauns++; // Посмотрели сколько там спаунов
            }
        }
    if (countOfSpauns != 0)
    {
        int** SpaunsCoordinate = new int* [countOfSpauns];
        for (int i = 0; i < countOfSpauns; ++i)
        {
            SpaunsCoordinate[i] = new int[2]; // Создали динамический
массив координат
        }
        int randSpaun = rand() % countOfSpauns; // от 0 до 3
        for (int i = 0; i < HEIGHT_MAP; i++) //проходимся по элементам карты
            for (int j = 0; j < WIDTH_MAP; j++)
            {

```

```

        if (TileMap[i][j] == 's') //если элемент наш тайлик
земли, то
    {
        countOfSpauns--;
        SpaunsCoordinate[countOfSpauns][0] = i;
        SpaunsCoordinate[countOfSpauns][1] = j;
        //Выбрали случайный спаун
    }
    }
    Sy = SpaunsCoordinate[randSpaun][0] * 32; //Волшебная цифра -
32 (x32). Именно такая площадь у одной координаты
    Sx = SpaunsCoordinate[randSpaun][1] * 32;

    }
    return sf::Vector2f(Sx, Sy);
}
void Enemy::draw(sf::RenderTarget& target)
{
    target.draw(sprite);
    if (name == "BOSSbot")
    {
        target.draw(BossPart);
    }
    target.draw(healthSprite);
}

sf::FloatRect Enemy::GetRect()
{
    // у босса чистый квадрат
    sf::FloatRect BufRect;
    if (name == "BOSSbot")
    {
        BufRect.left = x;
        BufRect.top = y;
        BufRect.width = w;
        BufRect.height = h;
    }
    // итбокс у нас плечи и начало огня - для красоты, чтобы не сталкивался с
пустым местом
    if (name == "flybot")
    {
        BufRect.left = x + 5;
        BufRect.top = y + 5;
        BufRect.width = w - 5;
        BufRect.height = h - h / 4;
    }
    return BufRect;
}
//Для определения действия врага
int Enemy::action(float time)
{
    if (!isMove)
    {
        isMove = true;
        SetDirection(); //Если не двигаемся, то начинаем
        //а то че стоять тут
    }
}

```

```

    }
    else
    {
        if (state == stay)
        {
            return 0;
            //Че? Все еще стоим?!
        }
        x += dx * time * speed;
        checkCollisionWithMap(dx, 0);
        y += dy * time * speed;
        checkCollisionWithMap(0, dy);
        //подвинули и проверили на столкновение
        sprite.setPosition(x + w / 2, y + h / 2); //задаем позицию спрайта

        if (name == "BOSSbot")
        {
            BossPart.setPosition(x + w / 2 - 5, y + h / 2 - 10);
            healthSprite.setPosition(x, y - 20);
        }
        else if (name == "flybot")
        {
            healthSprite.setPosition(x + 10, y - 10);
        }
        if (abs(XYAim.x - x) < w && abs(XYAim.y - y) < h)
        {
            isMove = false;
            dx = 0;
            dy = 0;
            //Достигли точки назначения. Пора искать новую цель.
        }
    }
    return 0;
}

int Enemy::animation()
{
    if (name == "flybot")
    {
        healthSprite.setTextureRect(sf::IntRect(0, 0, 9 + health / 4,
8)); //health ('max = 100') / 4 = 25
        //делим палочку по состоянию здоровья
        if (status == "anikilled")
        {
            if (moveTimer < 100)
            {
                sprite.setTextureRect(sf::IntRect(240, 1, 80, 80)); //
взрыв
            }
            else if (moveTimer < 200)
            {
                sprite.setColor(sf::Color::Red); // взрыв закрашиваем
красным
            }
            else
            {
                status = "killed";
            }
        }
    }
}

```

```

    }
    //Уже убили? Надо уйти красиво
}
else if (moveTimer < 200)
{
    sprite.setTextureRect(sf::IntRect(12, 6, 57, 68));
}
else if (moveTimer < 400)
{
    sprite.setTextureRect(sf::IntRect(172, 5, 56, 69));
}
else if (moveTimer < 600)
{
    sprite.setTextureRect(sf::IntRect(92, 5, 58, 72));
}
else if (moveTimer < 800)
{
    sprite.setTextureRect(sf::IntRect(172, 5, 56, 69));
}
else
{
    moveTimer = 0;
}
//Пока живы надо двигаться красиво
}
else if (name == "BOSSbot")
{
    healthSprite.setTextureRect(sf::IntRect(2, 252, 24 + health / 3.5 ,
25)); //health ('max = 500') / 3,5 = 142
    if (status == "anikilled")
    {

        if (moveTimer < 200)
        {
            sprite.setTextureRect(sf::IntRect(240, 1, 80, 80));
            sprite.setScale(4,4);
        }
        else if (moveTimer < 300)
        {
            sprite.setColor(sf::Color::Red);
            BossPart.setColor(sf::Color::Red);
        }
        else if (moveTimer < 400)
        {
            sprite.setColor(sf::Color::Yellow);
            BossPart.setColor(sf::Color::Black);
        }
        else if (moveTimer < 500)
        {
            status = "killed";
        }
        //БОССА ЗАВАЛИЛИ!!! ЩАС БОМБАНЕТ!!!
    }
    else if (moveTimer < 2000)
    {
        sprite.rotate(0.5);
    }
    else if (moveTimer < 4000)

```

```

        {
            sprite.rotate(-0.5);
        }
        else
        {
            moveTimer = 0;
            //you spin my head right round, right round...
        }

    }
    return 0;
}

int Enemy::update(float time)
{
    //
    if (!life)
    {
        return 0;
        //Точно сдох. Нечего тут брыкаться
    }
    if (speed < 1)
    {
        speed += 0.001; // Нарастиваем скорость, чтоб сразу после спауна не
врезаться
    }

    moveTimer += time; // добавляем квант времени
    BOSSdamagetimer += time;
    animation();
    action(time);
    if (status == "killed")
    {
        life = false;
    }
    else if (health <= 0 && status != "anikilled") {
        status = "anikilled";
        state = stay;
        moveTimer = 0;
        if (name == "BOSSbot")
        {
            sprite.setOrigin(40, 50); // центр взрыва
            sprite.setRotation(0); // чтобы взрыв не вращался
            //Далее меняем спрайт. Сделал чтоб тот не ерзал.
        }
    }
    return 0;
}

int Enemy::SetAim(sf::Vector2f XY)
{
    if (!isMove) {
        if (name == "flybot")
        {
            XYAim = XY;
        }
    }
}

```

```

        else if (name == "BOSSbot")
        {
            XYAim = SpaunTarget();
            //Пункт назначения - спаун
        }
    }
    return 0;
}

void Enemy::struck(int damage) // 100 для большой пули, 20 - для маленькой
{
    health -= damage;
    //АЙ
}

```

ПРИЛОЖЕНИЕ М

map.h

```
#ifndef __map__
#define __map__

#include "Constants.h"
#include <SFML/Graphics.hpp>
/*
0 - границы
b - препятствие
p - растение)
' ' - земля/пол
s - спаун врага
h - точка хила
В дальнейшем рассматривается как массив.
*/
sf::String TileMapMy[HEIGHT_MAP] = {
    "00000000000000000000000000000000000000000000000000000",
    "0  p                                0",
    "0  bp      p      bb                p      0",
    "0  bbbb      b                        0",
    "0  bb      p                pp        0",
    "0                p                b  0",
    "0          s                    0",
    "0          0",
    "0  p      p                bbb        0",
    "0                s              0",
    "0          b                    0",
    "0          b      p                p      0",
    "0          p      bbb                0",
    "0                b              0",
    "0          0",
    "0  b          p                h      0",
    "0  bbb                p              0",
    "0  b      p                    0",
    "0                bbb      p          0",
    "0                bbb      s          0",
    "0  p          p      bbb                0",
    "0          s          p      p        0",
    "0                p      p      b      0",
    "0  p                                0",
    "00000000000000000000000000000000000000000000000000000",
};

#endif
```


ПРИЛОЖЕНИЕ Н

Constants.h

```
#ifndef __Constants__
#define __Constants__

//Файл с константами

#include <SFML/Graphics.hpp>
const int HEIGHT_MAP = 25; //Высота карты
const int WIDTH_MAP = 40; //Ширина карты

#endif
```