**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Физико-технический институт**

**Командное задание в рамках курсового проекта «Разработка игры «Space invaders»**

**Индивидуальный отчет**

**Выполнил:**

Студент

Физико-технического

Института группы 21318

Сергиенко Даниил

**Преподаватель:**

Бульба Артем Владимирович

**Петрозаводск 2023**

**Цель работы:** Разработать аналог игры «space invaiders». Вести разработку с помощью системы контроля версий Git через веб-сервис GitHub. Закрепить полученные знания, использования языков С++ с использованием библиотеки

SFML и UML.

**Задачи:**

1. Познакомится с принципами работы в команде.
2. Научится использовать при работе систему контроля версии Git.
3. Научится строить UML-диаграммы.
4. Научится использовать SFML библиотеки в программной среде С++.
5. Написание отчёта о проделанной работе.

Кратко о программной реализации:

Программная реализация выполнена на языке c++ с использованием библиотеки SFML, среда разработки - IDE Qt Creator (Qt 5.4.2)

Единицы компиляции:

Globals.h

создан для хранения констант и более удобного управления их значениями в целях отладки

Entity.h

Entity.cpp

Содержат абстрактный класс, описывающий общее поведение объектов в рамках игры

Bullet.h

Bullet.cpp

Создание пули, метод взаимодействия с ними, а также их изменение со временем

Enemy.h

Enemy.cpp

Описание класса враг и его проведение как отдельного объекта

Player.h

Player.cpp

Описание класса игрока. В нем содержатся метод управления самолетом, список пуль, выпущенных игроком, и проверку на попадание по самолету

Enemies.h

Enemies.cpp

Описание класса враги, то есть список врагов и их поведение как одно целое. Отвечает за передвижение списка врагов по карте, обработке их смерти при попадании пули игрока по ним

Game.h

Game.cpp

Описание класса игры, собирающий все выше перечисленные классы вместе и обеспечивающий взаимодействие между ними

Список существительных:

* Игра

Основной класс программы, в нем хранятся экземпляры классов игрока и врагов, а также обрабатывается основная логика игры, например проигрыш игрока

* Сущность

Абстрактный класс, представляющий собой объект на экране

* Игрок

Класс, представляющий игрока. Он отвечает за передвижение, выстрелы самолета и отслеживание попадания по пришельцам. А также хранением списка пуль, выпущенных игроком, и управлением им

* Пришелец

Класс, описывающий поведение отдельного пришельца, то есть его передвижение, отрисовка и выстрелы

* Группа пришельцев

Класс, отвечающий за совместное поведение пришельцев, движение по траектории (если крайний пришелец достигает конца экрана все пришельцы меняют направление и сдвигаются вниз), определение противника который в данный момент должен выстрелить, а также отслеживание попадания по игроку.

* Пуля

Класс, описывающий поведение пули. В зависимости от того кто создает пулю она может идти как вверх(в случае создания игроком), так и вниз(в случае создания врагом)

Заголовочные файлы

Enemy.h

#ifndef ENEMY

#define ENEMY

#include <random>

#include "bullet.h"

#include "entity.h"

class Enemy: public Entity{

enum direction {left=-1,right=1} state;

public:

Enemy(Image &image, float X, float Y, int W, int H, RenderWindow\* i\_window);

void change\_frame();

bool move(int dir);

void *update*(float time);

void move\_down();

Bullet\* shoot(Image &image);

IntRect get\_hitbox();

bool check\_collisions(IntRect player\_bullet);

};

#endif // ENEMY

Исполнительные файлы

Enemy.cpp

#include"enemy.h"

Enemy::Enemy(Image &image, float X, float Y, int W, int H, RenderWindow\* i\_window) :Entity(image, X,

Y, W, H, i\_window){

sprite.setTextureRect(IntRect(CurrentFrame\*w,0,w,h));

state = right;

sprite.setPosition(x, y);

window->draw(sprite);

set\_speed(STEP);

}

void Enemy::change\_frame(){

if(CurrentFrame==0){

set\_current\_frame(1);

}else{

set\_current\_frame(0);

}

}

bool Enemy::move(int dir){

change\_frame();

bool return\_value=0;

if (state==right){

return\_value = (x + BASE\_SIZE + STEP >= SCREEN\_WIDTH);

}else{

return\_value= (x - STEP <=0);

}

x+=speed\*state\*dir;

return return\_value;

}

void Enemy::*update*(float time){

sprite.setTextureRect(IntRect(CurrentFrame\*w,0,w,h));

sprite.setPosition(x, y);

window->draw(sprite);

}

void Enemy::move\_down(){

if (state==left){

state = right;

}else{

state = left;

}

y+=BASE\_SIZE;

}

Bullet\* Enemy::shoot(Image &image){

Bullet\* tmp\_bullet = new Bullet(image,x,y,BASE\_SIZE,BASE\_SIZE,window,ENEMY\_BULLET\_SPEED);

return tmp\_bullet;

}

IntRect Enemy::get\_hitbox()

{

return IntRect(x + 0.25f \* BASE\_SIZE, y + 0.25f \* BASE\_SIZE, 0.5f \* BASE\_SIZE, 0.5f \* BASE\_SIZE);

}

bool Enemy::check\_collisions(IntRect player\_bullet){

if (get\_hitbox().intersects(player\_bullet)){

return true;

}else{

return false;

}

}

История моих коммитов:

add Enemy class header

добавлен готовый заголовочный файл класса Enemy

add Enemy class realization

добавлены реализации к методам заголовочного файла

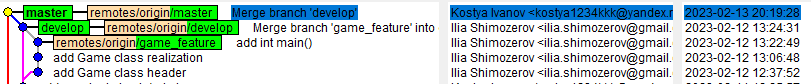
Enemy class is done

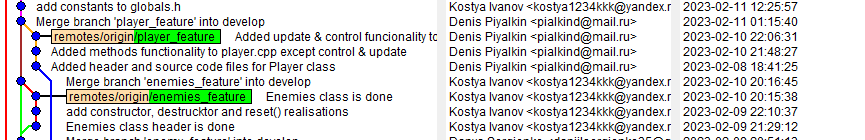
Добавлены методы shoot(), get\_hitbox(), check\_collisions() а также их реализации

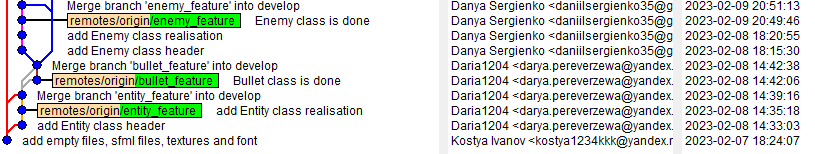
Merge branch 'enemy\_feature' into develop

Вливание ветки enemy\_feature в develop

История проекта Github:







**Вывод:** В ходе выполнения командного задания в рамках курсового проекта использовалась среда разработки Qt язык C++ с использованием библиотеки SFML. Применялась система контроля версий git. Реализованы все прецеденты. В ходе тестирования игры зависания или сбои не были обнаружены. Для классов Player и Enemies, хранящих в себе списки пуль, была реализована очистка памяти в их деструкторах. Неиспользуемые переменные в ходе разработки старались удалять. В алгоритмы методов старались не добавлять не нужных циклов, массивов и др. К отчету приложены диаграмма вариантов использования, диаграммы деятельности, диаграммы последовательности и диаграмма классов. Все они, за исключением диаграммы вариантов использования, находятся в пошаговом описании процесса разработки. Мы считаем что наша команда достигла поставленных целей.