**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

Пиялкин Денис

**Преподаватель:**

Бульба Артем Владимирович

**Петрозаводск 2023**

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

Player.h

Player.cpp

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

Описание сюжета игры

Игрок управляет самолетом, передвигая его горизонтально, в нижней части экрана, а также отстреливая инопланетян, надвигающихся сверху экрана. Целью игры является уничтожение всех инопланетян на экране, которые двигаются горизонтально, а также вертикально, по направлению к низу экрана. Игрок имеет бесконечное количество патронов. Попадая в инопланетянина, игрок уничтожает его, за что получает очки. При уничтожении инопланетян увеличивается скорость движения оставшихся. При уничтожении всех инопланетян появляется новая волна. Количество новых волн инопланетян неограниченно, что делает игру бесконечной.

Инопланетяне пытаются уничтожить самолет, стреляя по нему. При попадании в самолет он уничтожается, то есть игра заканчивается. При достижении хотя бы одним из противников нижней части экрана происходит инопланетный захват, и игра также заканчивается.

Заголовочные файлы

Player.h:

#ifndef PLAYER

#define PLAYER

#include <SFML/Graphics.hpp>

#include "globals.h"

#include "bullet.h"

#include <list>

#include "entity.h"

using namespace sf;

using namespace std;

class Player:public Entity{

private:

unsigned int score;

enum { left, right, stay} state;

list<Bullet\*> bullet\_list;

list<Bullet\*>\* enemy\_bullets;

list<Bullet\*>::iterator iter;

Image bullet\_img;

Clock reload\_timer;

public:

Player(Image &image, float X, float Y, int W, int H, RenderWindow\* i\_window, list<Bullet\*>\* enemy\_bul);

~Player();

void control();

void reset(bool win);

void die();

void *update*(float time);

list<Bullet\*>\* get\_player\_bullet\_list();

IntRect get\_hitbox();

bool is\_alive();

string get\_score();

void set\_score(unsigned int sc);

};

#endif // PLAYER

Исполнительные файлы

Player.cpp:

#include "player.h"

Player::Player(Image &image, float X, float Y, int W, int H, RenderWindow\* i\_window, list<Bullet\*>\* enemy\_bul) :Entity(image, X,

Y, W, H, i\_window),enemy\_bullets(enemy\_bul){

score=0;

bullet\_list.clear();

state = stay;

sprite.setTextureRect(IntRect(0, 0, w, h));

bullet\_img.loadFromFile("Resources/Images/PlayerBullet.png");

}

Player::~Player(){

while(!bullet\_list.empty()){

iter= bullet\_list.begin();

bullet\_list.erase(iter);

}

}

void Player::control(){

if (Keyboard::isKeyPressed(Keyboard::Left)) {

if(!(x+BASE\_SIZE/2<0)){

state = left;

speed = 0.1;}

}

if (Keyboard::isKeyPressed(Keyboard::Right) ) {

if(!(x+BASE\_SIZE/2>SCREEN\_WIDTH)){

state = right;

speed = 0.1;

}

}

if (Keyboard::isKeyPressed(Keyboard::Up)) {

if (reload\_timer.getElapsedTime().asMilliseconds() > RELOAD\_TIME){

Bullet\* tmp\_bullet= new Bullet(bullet\_img,x,y-BASE\_SIZE,BASE\_SIZE,BASE\_SIZE,window,-0.2);

bullet\_list.push\_back(tmp\_bullet);

reload\_timer.restart();

}

}

}

void Player::reset(bool win){

x=0;

y=SCREEN\_HEIGHT-BASE\_SIZE-1;

if (!win){

score=0;

life=true;

}

while(!bullet\_list.empty()){

iter= bullet\_list.begin();

bullet\_list.erase(iter);

}

}

void Player::die(){

life=false;

}

void Player::*update*(float time)

{

if (life) {

control();

switch (state)

{

case right:{

dx = speed;

break;

}

case left:{

dx = -speed;

break;

}

case stay:{

dy = speed;

dx = speed;

break;

}

}

x += dx\*time;

speed = 0;

sprite.setPosition(x, y);

window->draw(sprite);

iter = bullet\_list.begin();

while (iter != bullet\_list.end()){

(\*iter)->update(time);

if((\*iter)->get\_y()<-BASE\_SIZE || (\*iter)->get\_life()==false){

if ((\*iter)->get\_life()==false)

score+=ENEMY\_VALUE;

bullet\_list.erase(iter);

}

\*iter++;

}

for (iter=enemy\_bullets->begin();iter!=enemy\_bullets->end();iter++){

if (get\_hitbox().intersects((\*iter)->get\_hitbox())){

die();

}

}

}

}

list<Bullet\*>\* Player::get\_player\_bullet\_list(){

return &bullet\_list;

}

IntRect Player::get\_hitbox()

{

return IntRect(x + 0.125f \* BASE\_SIZE, y + 0.125f \* BASE\_SIZE, 0.75f \* BASE\_SIZE, 0.75f \* BASE\_SIZE);

}

bool Player::is\_alive(){

return life;

}

string Player::get\_score(){

string str = to\_string(score);

return str;

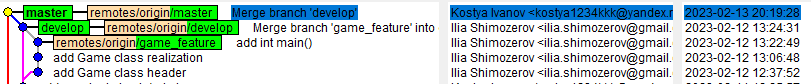
}

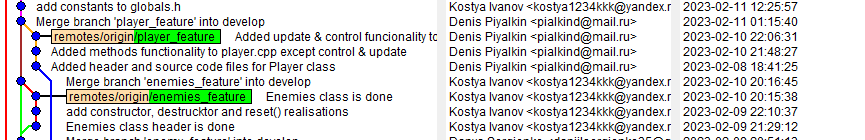
void Player::set\_score(unsigned int sc){

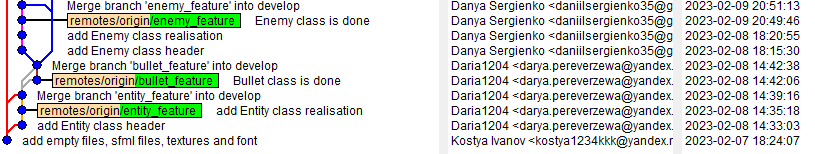
score=sc;

}

История проекта Github:







История моих коммитов:

Added header and source code files for Player class

Добавлен готовый заголовочный файл класса Player и заглушки под методы

Added methods functionality to player.cpp except control & update

Добавлены реализация всем методам кроме control() и update()

Added update & control funcionality to player.cpp

Добавлена реализация методам control() и update()

Merge branch 'player\_feature' into develop

Слияние ветки player\_feature в develop

**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 класса Player, хранящего в себе списки пуль, была реализована очистка памяти в их деструкторах. Я считаю, что мной и моей командой были достигнуты все поставленные цели.