**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 invaiders»**

**Индивидуальный отчет**

**Выполнили:**

Студенты

Физико-технического

Института группы 21318

Иванов Константин

**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

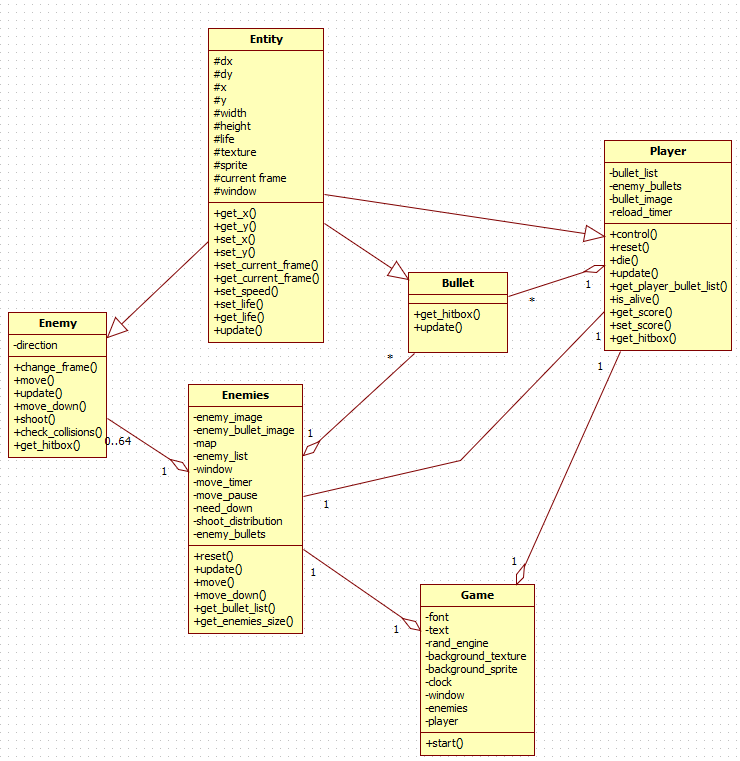


Рис1. Диаграмма классов

Заголовочные файлы

Enemies.h

#ifndef ENEMIES

#define ENEMIES

#include <SFML/Graphics.hpp>

#include <list>

#include <random>

#include "enemy.h"

#include "globals.h"

using namespace std;

class Enemies{

Image enemy\_img;

Image bullet\_img;

String map;

list<Enemy\*> enemy\_list;

list<Enemy\*>::iterator iter;

RenderWindow\* window;

Clock move\_timer;

float move\_pause;

bool need\_down, below;

uniform\_int\_distribution<unsigned short> shoot\_distribution;

list<Bullet\*> enemy\_bullets;

list<Bullet\*>::iterator bullet\_iter;

public:

~Enemies();

Enemies(RenderWindow\* i\_window);

void reset();

void update(float time,std::mt19937& engine, list<Bullet\*>\* player\_bullet);

void move(std::mt19937& engine);

void move\_down();

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

int get\_enemies\_size();

bool get\_below();

};

#endif // ENEMIES

Исполнительные файлы

Enemies.cpp

#include "enemies.h"

Enemies::~Enemies(){

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

iter= enemy\_list.begin();

enemy\_list.erase(iter);

}

while(!enemy\_bullets.empty()){

bullet\_iter=enemy\_bullets.begin();

enemy\_bullets.erase(bullet\_iter);

}

}

Enemies::Enemies(RenderWindow\* i\_window){

need\_down=false;

move\_timer.restart();

window=i\_window;

enemy\_img.loadFromFile(ENEMY\_IMG);

bullet\_img.loadFromFile(ENEMY\_BULLET);

map = "0000000000000000\n0000000000000000\n0000000000000000\n0000000000000000";

reset();

}

void Enemies::reset(){

below=false;

unsigned char enemy\_x = 0;

unsigned char enemy\_y = 0;

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

iter= enemy\_list.begin();

enemy\_list.erase(iter);

}

while(!enemy\_bullets.empty()){

bullet\_iter=enemy\_bullets.begin();

enemy\_bullets.erase(bullet\_iter);

}

shoot\_distribution = uniform\_int\_distribution<unsigned short>(0,ENEMY\_SHOOT\_RATE);

for(char map\_part : map){

enemy\_x++;

switch(map\_part){

case '\n':

{

enemy\_x=0;

enemy\_y++;

break;

}

case '0':

{

Enemy\* tmp\_enemy = new Enemy(enemy\_img,BASE\_SIZE\*(1+enemy\_x),BASE\_SIZE\*(1+enemy\_y),BASE\_SIZE,BASE\_SIZE,window);

if ((enemy\_x+enemy\_y)%2){

tmp\_enemy->change\_frame();

}

enemy\_list.push\_back(tmp\_enemy);

break;

}

}

}

}

void Enemies::update(float time,std::mt19937& engine, list<Bullet\*>\* player\_bullet){

move(engine);

if (!enemy\_bullets.empty()){

bullet\_iter=enemy\_bullets.begin();

while(bullet\_iter != enemy\_bullets.end()){

(\*bullet\_iter)->update(time);

if ((\*bullet\_iter)->get\_y()>SCREEN\_HEIGHT){

enemy\_bullets.erase(bullet\_iter);

}

bullet\_iter++;

}

}

if(!enemy\_list.empty()){

iter = enemy\_list.begin();

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

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

if ((\*iter)->get\_y()>=SCREEN\_HEIGHT-3\*BASE\_SIZE/2){

below=true;

}

for(bullet\_iter=player\_bullet->begin();bullet\_iter!=player\_bullet->end();bullet\_iter++){

if ((\*iter)->check\_collisions((\*bullet\_iter)->get\_hitbox())){

enemy\_list.erase(iter);

(\*bullet\_iter)->set\_life(false);

break;

}

}

iter++;

}

}

}

void Enemies::move(std::mt19937& engine){

if (move\_timer.getElapsedTime().asMilliseconds()>MOVE\_TIME-ENEMY\_SPEED\_INCREASE\*(64-enemy\_list.size())){

if(!need\_down){

if(!enemy\_list.empty()){

iter = enemy\_list.begin();

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

if ((\*iter)->move(1)){

need\_down=true;

}

if (shoot\_distribution(engine)==0){enemy\_bullets.push\_back((\*iter)->shoot(bullet\_img));}

iter++;

}

}

move\_timer.restart();

}else{

move\_down();

need\_down=false;

}

}

}

void Enemies::move\_down(){

if(!enemy\_list.empty()){

iter = enemy\_list.begin();

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

(\*iter)->move\_down();

(\*iter)->move(-1);

iter++;

}

}

}

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

return &enemy\_bullets;

}

int Enemies::get\_enemies\_size(){

return enemy\_list.size();

}

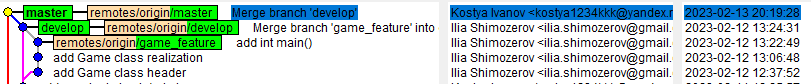
bool Enemies::get\_below()

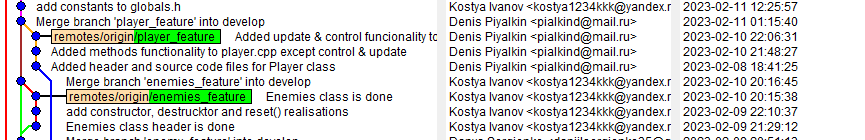
{

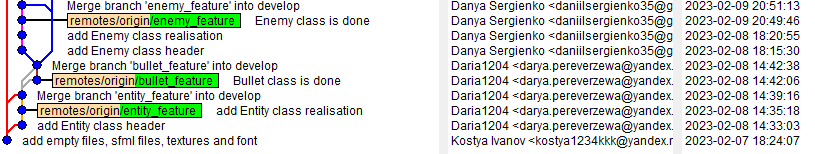
return below;

}

История проекта Github:







(Иванов Константин)

Add empty files, sfml files, textures and font

В проект добавлены пустые файлы под классы, текстуры, шрифт, и файлы необходимые для sfml

Enemies class header is done

добавлен готовый заголовочный файл класса Enemies

add constructor, destructor and reset() realisations

добавлены реализации конструктора, деструктора и метода reset()

Enemies class is done

Добавлены остальные реализации методов класса Enemies

Merge branch 'enemies\_feature' into develop

Вливание ветки enemies\_feature в develop

add constants to globals.h

добавлен заголовочный файл с константами

Merge branch 'develop'

Вливание ветки develop в master

**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.