Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

Учреждение высшего образования

«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Физико-технический институт

Дисциплина «Технология программирования»

Отчет по командному проекту:

**«Написание 2D-игры на языке С++ с использованием библиотеки SFML»**

Выполнили:

Студенты группы 21312 ФТИ

**Молчанов Павел Андреевич**

**Орсаг Павел Владимирович**

**Архипов Виталий Игоревич**

Проверил:

[к. ф.-м. н.](https://sokrasheniya.academic.ru/25072/%D0%BA._%D1%84.-%D0%BC._%D0%BD.)

**Бульба Артем Владимирович**

# Цель работы

Цель данной работы заключается в создании командного проекта – 2D игры.

# Задачи работы

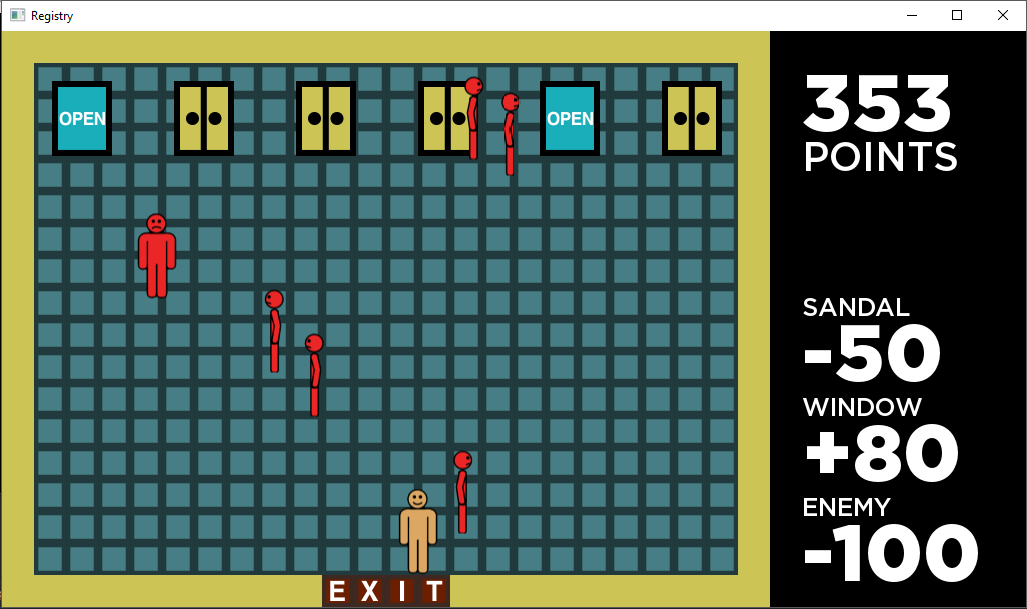
1. Ознакомиться с принципами работы над проектом в команде.
2. Научиться работать с системой контроля версий Git: научиться создавать локальные репозитории и ветки, сохранять изменения (делать коммиты), отправлять измененные файлы проекта в глобальный репозиторий, скачивать изменения из глобального репозиторий, устранять конфликты слияний.
3. Научиться строить UML-диаграммы проекта с использованием CASE-системы WhiteStarUML, включая диаграммы классов, use case диаграммы и диаграммы деятельности.
4. Написать 2D-игру на языке программирования С++ с использованием библиотеки SFML.
5. Составить отчеты по проделанной работе.

# Требования к программе

1. Программа не должна вызывать системных сбоев или зависать.
2. Программа должна освобождать всю выделенную память.
3. Должны отсутствовать конструкции, приводящие к неопределённому поведению.
4. Переменные должны быть, по возможности, локальны. Не должно быть неиспользуемых переменных.
5. Алгоритмы не должны быть избыточны (нет циклов, массивов без которых можно обойтись).
6. Свои классы необходимо реализовывать в отдельных файлах.
7. Необходимо отделять интерфейс от реализации.
8. Необходимо использовать STL-контейнеры и итераторы.

# Описание сюжета игры

В игре «Регистратура» игроку необходимо продержаться как можно дольше среди психически неуравновешенных людей, набрав при этом как можно больше номерков к врачам из окошек регистратуры. С течением времени количество очков, набранных игроком будет уменьшаться. Основным препятствием к этой цели станут пациенты больницы, при контакте с которыми игрок также будет терять очки. Пациенты также будут запускать в игрока снаряды, которые тоже забирают очки при попадании в игрока. При достижении игроком нуля очков игра заканчивается.



# Программная реализация

Все члены команды использовали среду разработки приложений Visual Studio 2017, с подключенной к ней библиотекой SFML версии 2.5.1. Исходный тест программы написан на языке программирования С++.

Так как при создании проекта использовался объектный подход, необходимо было написать программу так, чтобы она удовлетворяла главным принципам ООП: принципам инкапсуляции и наследования. Кроме того, программа должна удовлетворять принципам раздельной компиляции.

**Подключаемые заголовочные файлы**

1. "map.h" - интерфейс класса карты. (map.cpp)
2. "Player.h" - интерфейсная часть класса игрока. (player.cpp)
3. "Enemy.h" - интерфейсная часть класса врага. (Enemy.cpp)
4. "Place.h" - интерфейсная часть классов окон регистратуры. (Place.cpp)
5. "Bullet.h" - интерфейсная часть класса пуль. (Bullet.cpp)

# Функционал главного файла исходного кода

Главным файлом исходного кода в данном проекте является файл **Registry.cpp**

1. Сначала в Registry.cpp подключаются все необходимые для работы заголовочные файлы:

#include <sstream>

#include <iostream>

#include <list>

#include "Player.h"

#include "Enemy.h"

#include "Place.h"

#include "map.h"

#include "Bullet.h"

1. После создается окно, и загружаются текстуры игрока, врагов, пуль и окон. Вызывается конструктор класса игрока.

VideoMode desktop = VideoMode::getDesktopMode();

RenderWindow window(VideoMode(1024, 576, desktop.bitsPerPixel), "Registry");

Image map\_image;

map\_image.loadFromFile("images/map.png");

Texture map;

map.loadFromImage(map\_image);

Sprite s\_map;

s\_map.setTexture(map);

Image heroImage;

heroImage.loadFromFile("images/hero.png");

Image BulletImage;

BulletImage.loadFromFile("images/sandal.png");

Image EnemyImage;

EnemyImage.loadFromFile("images/enemy.png");

Image WindowImage;

WindowImage.loadFromFile("images/window.png");

Player p(heroImage, 492, 453, 40, 86, "Player");

1. Создаются три списка - для врагов, пуль, окон.

list<Enemy\*> Enemies;

list<Enemy\*>::iterator ite;

list<Bullet\*> Bullets;

list<Bullet\*>::iterator itr;

list <Place\*> windows;

list <Place\*>::iterator it;

1. Заполняются списки врагов и окон. Создаются объекты соответствующих классов.

const int W\_COUNT = 6;

int wCount = 0;

const int E\_COUNT = 6;

int eCount = 0;

for (int i = 0; i < E\_COUNT; i++)

{

float xr = 150 + rand() % 500;

float yr = 150 + rand() % 350;

Enemies.push\_back(new Enemy(EnemyImage, xr, yr, 40, 86, "Psycho"));

eCount += 1;

}

for (int i = 0;i < W\_COUNT;i++)

{

float xr = 50 + 172 \* i;

float yr = 50;

windows.push\_back(new Place(WindowImage, xr, yr, 60, 75));

wCount++;

}

1. Активация методов обновления всех объектов

p.update(time, p.x, p.y);

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

(\*ite)->update(time, p.x, p.y);

}

for (it = windows.begin(); it != windows.end(); it++)

{

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

}

for (itr = Bullets.begin(); itr != Bullets.end(); itr++)

{

(\*itr)->update(time, p.x, p.y);

}

1. Реализация случайной смены направления движения врагов.

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

float switchdir;

switchdir = rand() % 50;

if (switchdir > 48)

{

(\*ite)->direction = rand() % 3;

}

}

1. Генерация выстрелов врага.

if (p.PlayerScore > 0)

{

int testTime = test.getElapsedTime().asSeconds();

if (createBullet < 300) {

createBullet += testTime;

}

else

{

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

int switchdirbul = rand() % 10;

if (switchdirbul > 3)

Bullets.push\_back(new Bullet(BulletImage, (\*ite)->x, (\*ite)->y, 16, 16, "Bullet",rand()%(7)));

}

createBullet = 0;

test.restart();

}

}

1. Проверка пересечения объектов друг с другом.

for (it = windows.begin(); it != windows.end(); it++)

if (p.getRect().intersects((\*it)->getRect()) && (\*it)->isOpen)

{

p.PlayerScore += 80; (\*it)->CurrentFrame = 1; }

if (p.life == true) {

for (itr = Bullets.begin(); itr != Bullets.end(); itr++) {

if ((\*itr)->getRect().intersects(p.getRect()) && ((\*itr)->name == "Bullet"))

{

p.PlayerScore -= 50;

(\*itr)->life = false;

}

}

}

if (p.life == true)

{

int ToW = enemyclock.getElapsedTime().asMilliseconds();

if (ToW < 300) {

}

else

{

for (ite = Enemies.begin(); ite != Enemies.end(); ite++) {

if ((p.getRect().intersects((\*ite)->getRect())) && ((\*ite)->name == "Psycho") && p.invinc == false)

{

p.PlayerScore -= 100; //забираем очки

p.invinc = true; //даем временную неуязвимость

}

}

enemyclock.restart();

}

p.invinc = false;

}

1. Очистка контейнера пуль.

for (itr = Bullets.begin(); itr != Bullets.end(); ) {

Bullet \*t = \*itr;

t->update(time,p.x,p.y);

if (t->life == false)

{

itr = Bullets.erase(itr);

delete t;

}

else

itr++;

}

1. Отрисовка карты и объектов классов.

for (int i = 0; i < Height\_Map; i++)

for (int j = 0; j < Width\_Map; j++)

{

if ((TileMap[i][j] == '0')) s\_map.setTextureRect(IntRect(0, 0, 32, 32 if (TileMap[i][j] == ' ') s\_map.setTextureRect(IntRect(32, 0, 32, 32)); if ((TileMap[i][j] == 'X')) s\_map.setTextureRect(IntRect(96, 0, 32, 32)); s\_map.setPosition(j \* 32, i \* 32);

window.draw(s\_map);

}

window.draw(p.sprite);

for (it = windows.begin();it != windows.end();it++)

{

window.draw((\*it)->sprite);

}

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

window.draw((\*ite)->sprite);

}

for (itr = Bullets.begin(); itr != Bullets.end(); itr++)

{

if ((\*itr)->life)

window.draw((\*itr)->sprite);

}

1. Снятие очков со временем.

if (p.PlayerScore > 0)

p.PlayerScore -= 0.02;

if (p.PlayerScore < 0)

p.PlayerScore = 0;

1. Умертвление всех объектов после конца игры.

if (p.PlayerScore <= 0)

{

p.life = false;

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

(\*ite)->life = false;

for (itr = Bullets.begin(); itr != Bullets.end(); itr++)

(\*itr)->life = false;

for (it = windows.begin();it != windows.end();it++)

{

(\*it)->CurrentFrame = 1;

}

}

# Заключение

В данной работе была разработана 2D-игра с использованием языка программирования С++ и мультимедийной библиотеки SFML. Для удобства командной разработки активно использовалась система контроля версий GIT.

При построении проекта, были использованы принципы раздельной компиляции – каждый класс находится в отдельном компиляционном листе. Кроме того, реализация удовлетворяет принципам ООП: наследование, инкапсуляция и полиморфизм.

Были использованы STL-контейнеры.

Заключительная версия проекта соответствует всем заявленным требованиям – разработаны все прецеденты, описанные в Use Case диаграмме.

# Приложение 1. Листинг файла Registry.cpp

#include <sstream>

#include <iostream>

#include <list>

#include "Player.h"

#include "Enemy.h"

#include "Place.h"

#include "map.h"

#include "bullet.h"

int main()

{

VideoMode desktop = VideoMode::getDesktopMode();

RenderWindow window(VideoMode(1024, 576, desktop.bitsPerPixel), "Registry");

Image map\_image;

map\_image.loadFromFile("images/map.png");

Texture map;

map.loadFromImage(map\_image);

Sprite s\_map;

s\_map.setTexture(map);

Image heroImage;

heroImage.loadFromFile("images/hero.png");

Image BulletImage;

BulletImage.loadFromFile("images/sandal.png");

Image EnemyImage;

EnemyImage.loadFromFile("images/enemy.png");

Image WindowImage;

WindowImage.loadFromFile("images/window.png");

Player p(heroImage, 364, 453, 40, 86, "Player");

list<Enemy\*> Enemies; //список врагов

list<Enemy\*>::iterator ite;

list<Bullet\*> Bullets; //список пуль

list<Bullet\*>::iterator itr;

list <Place\*> windows;

list <Place\*>::iterator it;

const int W\_COUNT = 6; //максимальное количество окон в игре

int wCount = 0; //текущее количество окон в игре

const int E\_COUNT = 6; //максимальное количество врагов в игре

int eCount = 0;

for (int i = 0; i < E\_COUNT; i++)

{

float xr = 150 + rand() % 500; //случайная координата врага на поле игры по оси “x”

float yr = 150 + rand() % 350; //случайная координата врага на поле игры по оси “y”

Enemies.push\_back(new Enemy(EnemyImage, xr, yr, 40, 86, "Psycho")); //создаем врагов и помещаем в список

eCount += 1; //увеличили счётчик врагов

}

for (int i = 0;i < W\_COUNT;i++)

{

float xr = 50 + 122 \* i;

float yr = 50; //задаём позицию окон

windows.push\_back(new Place(WindowImage, xr, yr, 60, 75)); //вносим их в список

wCount++;

}

Clock clock, test,enemyclock;

Clock gameTimeClock;

float gameTime = 0;

float createBullet = 0;

// Основной (бесконечный) цикл

while (window.isOpen())

{

int time = clock.getElapsedTime().asMicroseconds();

if (p.life) gameTime = gameTimeClock.getElapsedTime().asSeconds();

clock.restart();

time = time / 800;

Event event;

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();//Закрываем окно, если событие “Closed”

}

///////////////////////////////////Оживление объектов/////////////////////////////////////

p.update(time, p.x, p.y); //Игрока

for (ite = Enemies.begin(); ite != Enemies.end(); ite++) //Врагов

{

(\*ite)->update(time, p.x, p.y); //запускаем метод update()

}

for (it = windows.begin(); it != windows.end(); it++) //Окон

{

(\*it)->update(time);//применяем метод update(time) класса Place для объектов из списка

}

for (itr = Bullets.begin(); itr != Bullets.end(); itr++) //Пуль

{

(\*itr)->update(time, p.x, p.y); //запускаем метод update() для пуль

}

/////////////////Смена направления///////////

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

float switchdir;

switchdir = rand() % 50;

if (switchdir > 48)

{

(\*ite)->direction = rand() % 3;

}

}

/////////////Генерация выстрелов/////////////////////////////////////////

if (p.PlayerScore > 0)

{

int testTime = test.getElapsedTime().asSeconds();

if (createBullet < 300) {

createBullet += testTime;

}

else

{

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

int switchdirbul = rand() % 10;

if (switchdirbul > 3)

Bullets.push\_back(new Bullet(BulletImage, (\*ite)->x, (\*ite)->y, 16, 16, "Bullet",rand()%(7))); //добавляем в список Bullets пулю

}

createBullet = 0;

test.restart();

}

}

///////////////////////////////////// Пересечения/////////////////////////////////////////////////////////

////////////////////////С окнами////////////////////////////

for (it = windows.begin(); it != windows.end(); it++) //С окнами//

if (p.getRect().intersects((\*it)->getRect()) && (\*it)->isOpen)

{ //проверка пересечения игрока с окном

p.PlayerScore += 80; //начисление очков, если окно открыто

(\*it)->CurrentFrame = 1; //закрытие окна после получения очков

}

/////////////////////////Игрока со снарядами //////////////////////////////////

if (p.life == true) { //если игрок жив

for (itr = Bullets.begin(); itr != Bullets.end(); itr++) { //бежим по списку врагов

if ((\*itr)->getRect().intersects(p.getRect()) && ((\*itr)->name == "Bullet"))

{

p.PlayerScore -= 50; //снятие очков

(\*itr)->life = false;

}

}

}

///////////////Врага и игрока///////////////

if (p.life == true) //если игрок жив

{

int ToW = enemyclock.getElapsedTime().asMilliseconds();

if (ToW < 300) {

}

else

{

for (ite = Enemies.begin(); ite != Enemies.end(); ite++) //бежим по списку врагов

{

if ((p.getRect().intersects((\*ite)->getRect())) && ((\*ite)->name == "Psycho") && p.invinc == false)

{

p.PlayerScore -= 100; //забираем очки

p.invinc = true; //даем временную неуязвимость

}

}

enemyclock.restart();

}

p.invinc = false;

}

////////////////Очистка контейнера с пулями/////////////

for (itr = Bullets.begin(); itr != Bullets.end(); ) //проходимся от начала до конца

{ //если этот объект мертв, то удаляем его

Bullet \*t = \*itr;

t->update(time,p.x,p.y);

if (t->life == false)

{

itr = Bullets.erase(itr);

delete t;

}

else

itr++; //и идем итератором к след объекту.

}

window.clear();

Font font1, font2;

font1.loadFromFile("Fonts/GothaProBla.otf");

font2.loadFromFile("Fonts/GothaProMed.otf");

Text bigtext("", font1, 80),//создаем объект текст. Помещаем в объект текст-строку, шрифт, размер

smalltext("", font2, 40),//шрифта(в пикселях)

rule1("", font2, 25), rule2("", font2, 25), rule3("", font2, 25), //big text

ruleOne("", font1, 80), ruleTwo("", font1, 80), ruleThree("", font1, 80); //small text

//////////////////////////////////Карта//////////////////////////////////////////

for (int i = 0; i < Height\_Map; i++)

for (int j = 0; j < Width\_Map; j++)

{

if ((TileMap[i][j] == '0')) s\_map.setTextureRect(IntRect(0, 0, 32, 32));//если встретили 0, то рисуем 1й квадратик

if (TileMap[i][j] == ' ') s\_map.setTextureRect(IntRect(32, 0, 32, 32)); //если встретили пробел, то рисуем 2й квадратик

if ((TileMap[i][j] == '-')) s\_map.setTextureRect(IntRect(64, 0, 32, 32));//если встретили -, то рисуем черноту

if ((TileMap[i][j] == 'E')) s\_map.setTextureRect(IntRect(96, 0, 32, 32));// рисуем EXIT

if ((TileMap[i][j] == 'X')) s\_map.setTextureRect(IntRect(128, 0, 32, 32));//

if ((TileMap[i][j] == 'I')) s\_map.setTextureRect(IntRect(160, 0, 32, 32));//

if ((TileMap[i][j] == 'T')) s\_map.setTextureRect(IntRect(192, 0, 32, 32));//

s\_map.setPosition(j \* 32, i \* 32); //раскладываем квадратики в карту.

window.draw(s\_map);

}

///////////////////////////////Отрисовка объектов///////////////////////////////////////////////

window.draw(p.sprite); //игрок

for (it = windows.begin();it != windows.end();it++) //окна

{

window.draw((\*it)->sprite);

}

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

{

window.draw((\*ite)->sprite); //рисуем enemies объекты

}

///////////////////////////////draw bullets//////////////////////////////////

for (itr = Bullets.begin(); itr != Bullets.end(); itr++)

{

if ((\*itr)->life) //если пули живы

window.draw((\*itr)->sprite); //рисуем объекты

}

/////////Снятие очков со временем/////

if (p.PlayerScore > 0)

p.PlayerScore -= 0.02; //со временем очки сгорают, если счёт дойдет до нуля, игра окончена

if (p.PlayerScore < 0)

p.PlayerScore = 0;

////// Конец игры/////////

ostringstream playerScoreString, timeString; // объявили переменную

if (p.PlayerScore <= 0)

{

p.life = false;

for (ite = Enemies.begin(); ite != Enemies.end(); ite++)

(\*ite)->life = false;

for (itr = Bullets.begin(); itr != Bullets.end(); itr++)

(\*itr)->life = false;

for (it = windows.begin();it != windows.end();it++)

{

(\*it)->CurrentFrame = 1;

}

}

playerScoreString << int (p.PlayerScore); //занесли в нее число очков, то есть формируем строку

bigtext.setString(playerScoreString.str()); //задаем строку тексту и вызываем

smalltext.setString("POINTS"); //сформированную выше строку методом .str()

bigtext.setPosition(800, 20);//задаем позицию текста

smalltext.setPosition(800, 100);

window.draw(bigtext);//рисуем этот текст

window.draw(smalltext);

rule1.setString("SANDAL"); rule2.setString("WINDOW"), rule3.setString("ENEMY");

ruleOne.setString("-50"); ruleTwo.setString("+80"); ruleThree.setString("-100");

ruleOne.setPosition(800, 270); window.draw(ruleOne);

ruleTwo.setPosition(800, 370); window.draw(ruleTwo);

ruleThree.setPosition(800, 470); window.draw(ruleThree);

rule1.setPosition(800, 260); window.draw(rule1);

rule2.setPosition(800, 360); window.draw(rule2);

rule3.setPosition(800, 460); window.draw(rule3);

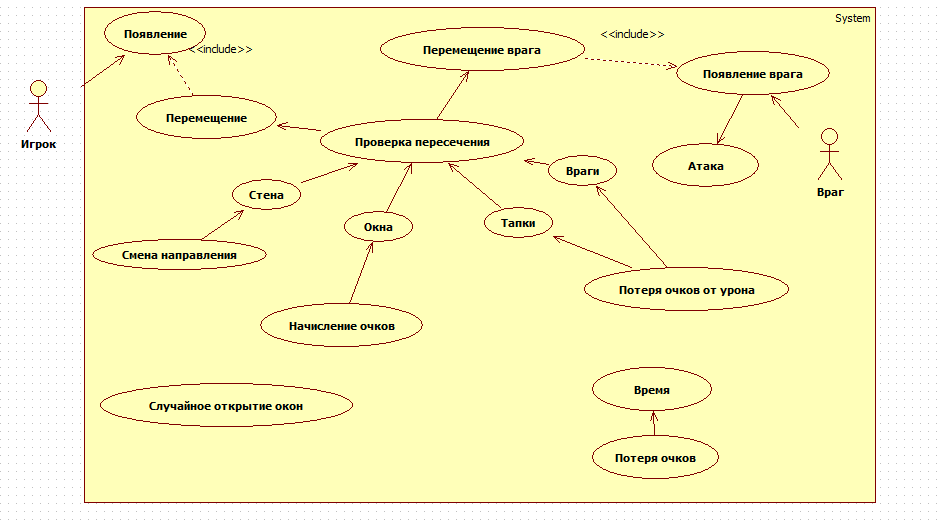
window.display();

}

return 0;

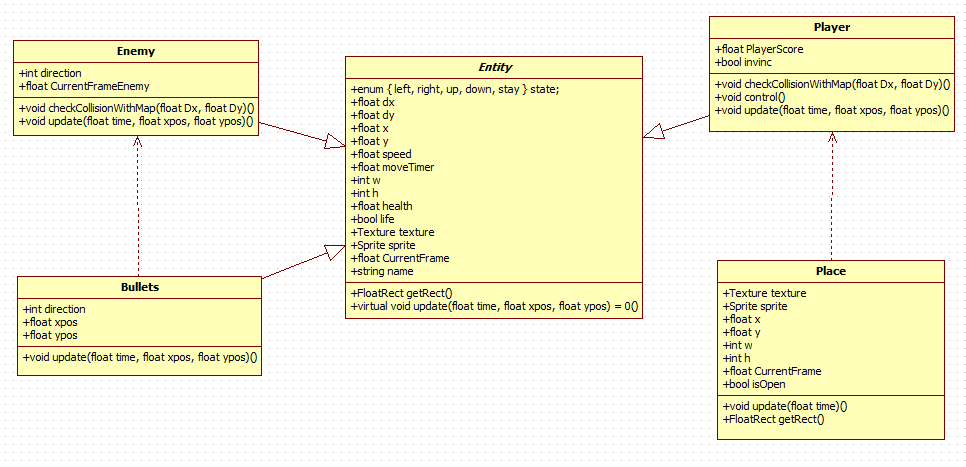
}

**Приложение 2. Диаграмма вариантов использования**

****

**Приложение 3. Диаграмма классов**

**Приложение 3. Диаграмма классов**

****