Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Петрозаводский государственный университет»

Кафедра информационно-измерительных систем и физической электроники

АКВАРИУМ

курсовой проект по дисциплине: «Технология программирования»

 Авторы работы:

 студенты группы 21312

\_\_\_\_\_\_\_\_\_\_\_\_ С. А. Семенов

\_\_\_\_\_\_\_\_\_\_\_\_ В. Р. Бояркин

\_\_\_\_\_\_\_\_\_\_\_\_ А. Д. Барановский

\_\_\_\_\_\_\_\_\_\_\_\_ С. И. Смирнов

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.

 Научный руководитель:

 канд. физ.-мат. наук, доцент:

\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Бульба

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.

Петрозаводск 2021

СОДЕРЖАНИЕ

Введение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1 О программной реализации . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4   
2 Описание процесса разработки . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1 Краткое словесное описание сюжета . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5

2.2 Список вариантов использования . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .6

2.3 Отдельное описание вариантов использования . . . . . . . . . . . . . . . . . . . . . . . .7

2.4 Список существительных .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.5 Диаграмма классов . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.6 Коды программы .h . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.7 Коды программы .cpp . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.8 Руководство пользователя . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .22   
3 История проекта на GitHub . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

Заключение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

Введение

Целью данной работы является укрепление навыков и обобщение полученных ранее знаний путём разработки игры на определённую тематику.

1 О программной реализации

Среда разработки: Microsoft Visual Studio 2019, но с набором инструментов платформы Visual Studio 2017 (v141)

Язык программирования: C++

Созданные единицы компиляции:

1. Main.cpp   
   (Используется для запуска игры и самого взаимодействия с ней)
2. Animation.h   
   (Хранит методы для анимации)
3. Animation.cpp   
   (Реализация методов анимации)
4. Menu.h   
   (Хранит метод вывода меню)
5. Menu.cpp   
   (Реализация метода вывода меню)
6. Entity.h   
   (Хранит родительский класс “Сущность”)
7. Entity.cpp   
   (Реализация методов класса “Сущность”)
8. Enemy.h   
   (Хранит дочерний класс “Враг”, описывает поведение врага.)
9. Enemy.cpp   
   (Реализация методов класса “Враг”)
10. Player.h   
    (Хранит дочерний класс “Игрок”, описывает поведение игрока.)
11. Player.cpp   
    (Реализация методов класса “Враг”)
12. Bubble.h   
    (Хранит дочерний класс “Пузырик”, описывает поведение пузырика.)
13. Bubble.cpp   
    (Реализация методов класса “Враг”)

2 Описание процесса разработки

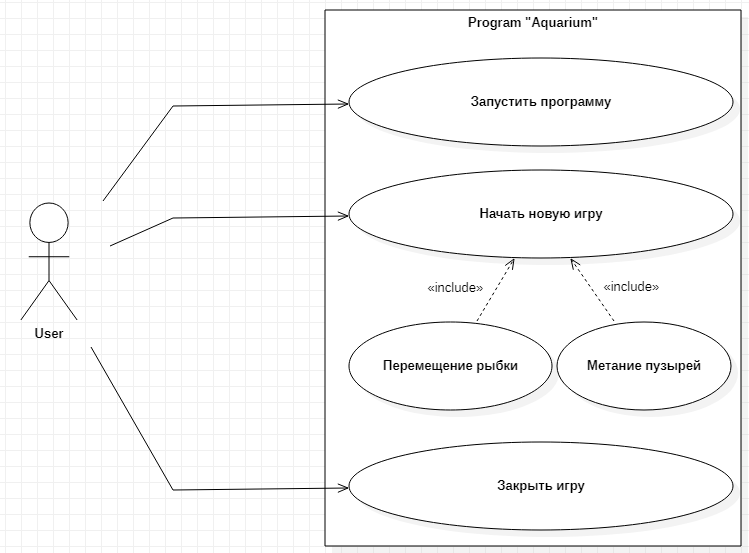
2.1 Краткое словесное описание сюжета (Полстраницы)

В большом морском аквариуме, в дали от других рыб, в уединении живут две рыбы — муж и жена, Марлин и Корал. Корал наметала икры, Марлин наметал молок, супруги готовятся к появлению потомства, строят планы на будущее, как вдруг все обрывает их новый сосед - барракуда, внезапно появившаяся на рифе. Она пожирает всю икру, Корал погибает. Марлин остается с одной икринкой, из которой вскоре появился сын — Немо, которого он растит один. Океанариум и существующие в нём опасности очень страшат Марлина, и он, как может, ограждает сына от них.

Однажды любопытный Немо, чтобы доказать, что он уже взрослый, плывет в другие районы аквариума и попадает в неприятности. Марлин отправляется на поиски сына.

2.2 Список вариантов использования

1. Запустить программу
2. Начать новую игру
   1. Перемещение рыбки
   2. Метание пузырей
3. Закрыть игру



2.3 Отдельное описание вариантов использования

1. Запустить программу – открыть главное меню
2. Начать новую игру – отобразить главного героя и врагов
   1. Перемещение рыбки – изменение положения рыбки в пространстве
   2. Метание пузырей – запустить снаряд
3. Закрыть игру

2.4 Список существительных

Сущность

Рыба – игрок

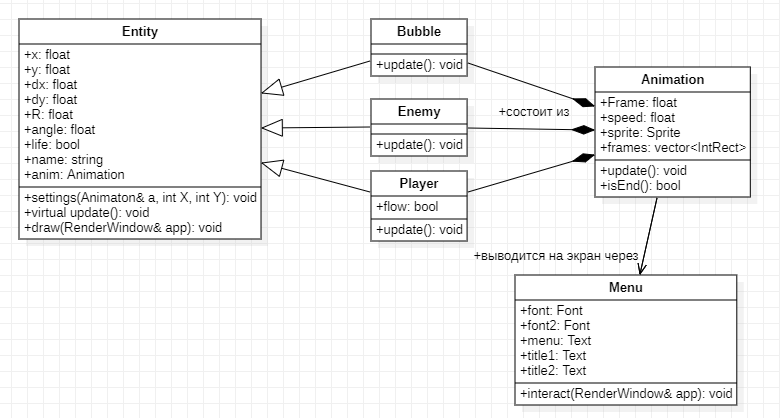
Рыба – враг

Пузырик

Меню

Анимация

2.5 Диаграмма классов



2.6 Коды программы .h

**Animation.h**

#pragma once

#include <SFML/Graphics.hpp>

#include <time.h>

#include <list>

#include "Menu.h"

class Animation

{

public:

float Frame, speed;

Sprite sprite;

vector<IntRect> frames;

Animation(){}

Animation(Texture& t, int x, int y, int w, int h, int count, float Speed);

void update();

bool isEnd();

};

**Bubble.h**

#pragma once

#include "Entity.h"

const float deg = 0.017453;

class Bubble : public Entity

{

public:

Bubble();

void update();

};

**Enemy.h**

#pragma once

#include "Entity.h"

class Enemy : public Entity

{

public:

Enemy();

void update();

};

**Entity.h**

#pragma once

#include "Animation.h"

class Entity

{

public:

float x, y, dx, dy, R, angle;

bool life;

string name;

Animation anim;

Entity();

void settings(Animation& a, int X, int Y, float Angle = 0, int radius = 1);

virtual void update() {};

void draw(RenderWindow& app);

virtual ~Entity() {};

};

**Menu.h**

#pragma once

#include <SFML/Graphics.hpp>

#include <time.h>

#include <list>

#include <iostream>

using namespace sf;

using namespace std;

const int W = 1200;

const int H = 700;

class Menu

{

private:

Font font, font2;

Text menu, title1, title2;

public:

void interact(RenderWindow& app);

Menu();

~Menu();

};

**Player.h**

#pragma once

#include "Entity.h"

#include "Bubble.h"

class Player : public Entity

{

public:

bool flow;

Player();

void update();

};

2.7 Коды программы .cpp

**Animation.cpp**

#include "Animation.h"

Animation::Animation(Texture& t, int x, int y, int w, int h, int count, float Speed)

{

Frame = 0;

speed = Speed;

for (int i = 0; i < count; i++)

frames.push\_back(IntRect(x + i \* w, y, w, h));

sprite.setTexture(t);

sprite.setOrigin(w / 2, h / 2);

sprite.setTextureRect(frames[0]);

}

void Animation::update()

{

Frame += speed;

int n = frames.size();

if (Frame >= n) Frame -= n;

if (n > 0) sprite.setTextureRect(frames[int(Frame)]);

}

bool Animation::isEnd()

{

return Frame + speed >= frames.size();

}

**Bubble.cpp**

#include "Bubble.h"

Bubble::Bubble()

{

name = "Bubble";

}

void Bubble::update()

{

dx = cos(angle \* deg) \* 4;

dy = sin(angle \* deg) \* 4;

x += dx;

y += dy;

if (x > W || x<0 || y>H || y < 0) life = 0;

}

**Enemy.cpp**

#include "Enemy.h"

Enemy::Enemy()

{

dx = rand() % 4;

if (dx == 0)

{

dx -= 1;

}

else

{

dx \*= -1;

}

name = "Enemy";

}

void Enemy::update()

{

x += dx;

if (x > W) x = 0; if (x < 0) x = W;

if (y > H) y = 0; if (y < 0) y = H;

}

**Entity.cpp**

#include "Entity.h"

Entity::Entity()

{

life = 1;

}

void Entity::settings(Animation& a, int X, int Y, float Angle, int radius)

{

anim = a;

x = X; y = Y;

angle = Angle;

R = radius;

}

void Entity::draw(RenderWindow& app)

{

anim.sprite.setPosition(x, y);

anim.sprite.setRotation(angle + 90);

app.draw(anim.sprite);

CircleShape circle(R);

circle.setFillColor(Color(255, 0, 0, 170));

circle.setPosition(x, y);

circle.setOrigin(R, R);

}

**Player.cpp**

#include "Player.h"

Player::Player()

{

name = "Player";

}

void Player::update()

{

if (flow)

{

dx += cos(angle \* deg) \* 0.3;

dy += sin(angle \* deg) \* 0.3;

}

else

{

dx \*= 0.99;

dy \*= 0.99;

}

int maxSpeed = 2;

float speed = sqrt(dx \* dx + dy \* dy);

if (speed > maxSpeed)

{

dx \*= maxSpeed / speed;

dy \*= maxSpeed / speed;

}

x += dx;

y += dy;

}

**Menu.cpp**

#include <SFML/Graphics.hpp>

#include <time.h>

#include <list>

#include "Menu.h"

#include "Animation.h"

#include "Entity.h"

#include "Enemy.h"

#include "Player.h"

Menu::Menu()

{

}

Menu::~Menu()

{

}

void Menu::interact(RenderWindow& app)

{

srand(time(0));

app.setFramerateLimit(60);

Texture menuTexture1,menuTexture2, menuBackground;

menuBackground.setSmooth(true);

menuTexture1.loadFromFile("images/button.png");

menuTexture2.loadFromFile("images/button.png");

menuBackground.loadFromFile("images/background.jpg");

Sprite menu1(menuTexture1), menu2(menuTexture2), menuBg(menuBackground);

int menuNum = 0;

//Icon of app

Image icon;

if (!icon.loadFromFile("images/icon2.png"))

{

cout << "No icons here";

}

app.setIcon(512, 512, icon.getPixelsPtr());

//Menu title

if (!font.loadFromFile("Fonts/CricketInlineShaddow.ttf"))

{

cout << "No fonts here";

}

menu.setFont(font);

menu.setFillColor(Color::Blue);

menu.setString("Menu");

menu.setCharacterSize(90);

menu.setPosition(60, 30);

//End

//Title1 of menu

if (!font2.loadFromFile("Fonts/Gecko\_Personal\_Use\_Only.ttf"))

{

cout << "No fonts here";

}

title1.setFont(font2);

title1.setFillColor(Color::Black);

title1.setString("New game");

title1.setCharacterSize(30);

title1.setPosition(90, 210);

//End

//Title2 of menu

if (!font2.loadFromFile("Fonts/Gecko\_Personal\_Use\_Only.ttf"))

{

cout << "No fonts here";

}

title2.setFont(font2);

title2.setFillColor(Color::Black);

title2.setString("Exit");

title2.setCharacterSize(30);

title2.setPosition(130, 310);

//End

//Задаём положения заголовков

menu1.setPosition(60, 200);

menu2.setPosition(60, 300);

menuBg.setPosition(0, 0);

menu1.setColor(Color::White);

menu2.setColor(Color::White);

menuNum = 0;

app.clear();

//Проверка наведения курсора на кнопку

if (IntRect(60, 200, 200, 80).contains(Mouse::getPosition(app)))

{

menu1.setColor(Color::Blue);

menuNum = 1;

}

//Проверка наведения курсора на кнопку

if (IntRect(60, 300, 200, 80).contains(Mouse::getPosition(app)))

{

menu2.setColor(Color::Blue);

menuNum = 2;

}

//Проверка нажатия на кнопку

if (Mouse::isButtonPressed(Mouse::Left))

{

if (menuNum == 2)

{

app.close();

}

}

app.draw(menuBg);

app.draw(menu1);

app.draw(menu2);

app.draw(menu);

app.draw(title1);

app.draw(title2);

app.display();

}

**Main.cpp**

#include <SFML/Graphics.hpp>

#include <time.h>

#include <list>

#include "Menu.h"

#include "Animation.h"

#include "Entity.h"

#include "Enemy.h"

#include "Player.h"

#include <iostream>

#include <sstream>

using namespace sf;

using namespace std;

//Функция проверки столкновения объектов

bool isCollide(Entity\* a, Entity\* b)

{

return (b->x - a->x) \* (b->x - a->x) +

(b->y - a->y) \* (b->y - a->y) <

(a->R + b->R) \* (a->R + b->R);

}

int main()

{

setlocale(0, "rus");

RenderWindow app(VideoMode(W, H), "Aquarium", Style::Close | Style::Titlebar);

srand(time(0));

app.setFramerateLimit(60);

Menu menu;

Text score;

Font font;

int schet = 0;

//Score

if (!font.loadFromFile("Fonts/Gecko\_Personal\_Use\_Only.ttf"))

{

cout << "No fonts here";

}

score.setFont(font);

score.setFillColor(Color::Black);

score.setCharacterSize(40);

score.setPosition(60, 20);

//End

Texture t1, t2, t3, t4, t5, t7, t8, t9;

t1.loadFromFile("images/hero.png");

t2.loadFromFile("images/background.jpg");

t3.loadFromFile("images/explosions/bubble\_boom.png");

t4.loadFromFile("images/angry111.png");

t5.loadFromFile("images/bubble.png");

t7.loadFromFile("images/explosions/bubble\_boom.png");

t8.loadFromFile("images/angry2.png");

t9.loadFromFile("images/angry3.png");

t1.setSmooth(true);

t2.setSmooth(true);

Sprite background(t2);

Animation sExplosion(t3, 0, 0, 256, 256, 24, 0.1);

Animation sExplosion\_fish(t7, 0, 0, 180, 180, 24, 0.1);

Animation sAngry1(t4, 0, 0, 75, 75, 1, 0);

Animation sAngry2(t8, 0, 0, 75, 75, 1, 0);

Animation sAngry3(t9, 0, 0, 75, 75, 1, 0);

Animation sBubble(t5, 0, 0, 20, 20, 1, 0);

Animation sPlayer(t1, 0, 0, 90, 90, 1, 0);

Animation sPlayer\_go(t1, 0, 0, 90, 90, 1, 0);

//Генерируем врагов и игрока

list<Entity\*> entities;

for (int i = 0; i < 3; i++)

{

Enemy\* A = new Enemy();

A->settings(sAngry1, rand() % W, rand() % H, 0, 30);

entities.push\_back(A);

}

for (int i = 0; i < 3; i++)

{

Enemy\* A = new Enemy();

A->settings(sAngry2, rand() % W, rand() % H, 0, 30);

entities.push\_back(A);

}

for (int i = 0; i < 3; i++)

{

Enemy\* A = new Enemy();

A->settings(sAngry3, rand() % W, rand() % H, 0, 30);

entities.push\_back(A);

}

Player\* P = new Player();

P->settings(sPlayer, 200, 350, 0, 45);

entities.push\_back(P);

//Главный цикл

while (app.isOpen())

{ //Проверка положения курсора и нажатия на кнопку

if ((IntRect(60, 200, 200, 80).contains(Mouse::getPosition(app))) && (Mouse::isButtonPressed(Mouse::Left)))

{

while (app.isOpen())

{

Event event;

while (app.pollEvent(event))

{ //Проверка событий

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

app.close();

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

if (event.key.code == Keyboard::Escape)

{

app.close();

}

//Отработка клавиши стрельбы

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

if (event.key.code == Keyboard::Space)

{

Bubble\* B = new Bubble();

B->settings(sBubble, P->x + 20, P->y, P->angle, 10);

entities.push\_back(B);

}

}

//Отработка клавиши поворота вправо

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

{

if (P->angle == 45)

{

P->angle += 0;

}

else

{

P->angle += 3;

}

}

//Отработка клавиши поворота влево

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

{

if (P->angle == -45)

{

P->angle += 0;

}

else

{

P->angle -= 3;

}

}

//Отработка клавиши движения

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

{

P->flow = true;

}

else

{

P->flow = false;

P->dx += -0.03;

}

for (auto a : entities)

for (auto b : entities)

{ //Проверка столкновения врага и пузырька

if (a->name == "Enemy" && b->name == "Bubble")

if (isCollide(a, b))

{

a->life = false;

b->life = false;

Entity\* e = new Entity();

e->settings(sExplosion, a->x, a->y);

e->name = "boom";

entities.push\_back(e);

++schet;

}

//Проверка столкновения игрока и врага

if (a->name == "Player" && b->name == "Enemy")

if (isCollide(a, b))

{

b->life = false;

Entity\* e = new Entity();

e->settings(sExplosion\_fish, a->x, a->y);

e->name = "boom";

entities.push\_back(e);

P->settings(sPlayer, W / 2, H / 2, 0, 20);

P->dx = 0; P->dy = 0;

schet = 0;

}

//Проверка достижения правой стороны

if (a->name == "Player" && a->x > W - 150)

{

P->dx -= 0.1;

}

//Проверка достижения левой стороны

if (a->name == "Player" && a->x < 0)

{

Entity\* e = new Entity();

e->settings(sExplosion\_fish, a->x, a->y);

e->name = "boom";

entities.push\_back(e);

P->settings(sPlayer, W / 2, H / 2, 0, 20);

P->dx = 0; P->dy = 0;

schet = 0;

}

//Проверка прохождения верхней и нижней границ

if (a->name == "Player" && a->y > H) a->y = 0;

if (a->name == "Player" && a->y < 0) a->y = H;

}

//Условия анимации движения

if (P->flow) P->anim = sPlayer\_go;

else P->anim = sPlayer;

//Завершения анимации

for (auto e : entities)

if (e->name == "boom")

if (e->anim.isEnd()) e->life = 0;

//Генерация новых врагов

if (rand() % 300 == 0)

{

Enemy\* A = new Enemy();

A->settings(sAngry1, 0, rand() % H, 0, 30);

entities.push\_back(A);

}

if (rand() % 300 == 0)

{

Enemy\* A = new Enemy();

A->settings(sAngry2, 0, rand() % H, 0, 30);

entities.push\_back(A);

}

if (rand() % 300 == 0)

{

Enemy\* A = new Enemy();

A->settings(sAngry3, 0, rand() % H, 0, 30);

entities.push\_back(A);

}

// Очистка памяти итератором

for (auto i = entities.begin(); i != entities.end();)

{

Entity\* e = \*i;

e->update();

e->anim.update();

if (e->life == false) { i = entities.erase(i); delete e; }

else i++;

}

app.clear();

//Отрисовока объектов

app.draw(background);

for (auto i : entities) i->draw(app);

ostringstream Score;

Score << schet;

score.setString("Score:" + Score.str());

app.draw(score);

app.display();

}

}

else //Меню

{

menu.interact(app);

Event event;

while (app.pollEvent(event))

{

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

app.close();

}

}

}

return 0;

}

2.8 Руководство пользователя

При запуске программы показывается меню. В нем указаны все варианты использования программы пользователем



Для того чтобы запустить нужную функцию необходимо навести курсор мыши на соответствующий пункт и кликнуть “ЛКМ”.

При нажатии на пункт “New game” запускается игровое поле.

1. Появляется игровой фон (представлен в меню)
2. В левом верхнем углу формы появляется игровой счёт очков.   
   За каждое уничтожение рыбки прибавляется 1 очко. В случае если игрок умирает, счёт обнуляется.



Рисунок 1 Счёт

1. Появляется главный герой

Управление героем осуществляется стрелочками и клавишей “Space”.

* Клавиши “Left”, “Right” – отвечают за изменение угла поворота рыбы в диапазоне [-45;45] градусов.
* Клавиша “Up” – отвечает за ускорение рыбки.
* Клавиша “Space” – отвечает за произведение выстрела пузырьком.

Во время игрового процесса ГГ может стрелять пузырьками для поражения рыб-врагов. В случае попадания пузырик схлопывается, а рыба “лопается”. Помимо этого, нашему герою нельзя стоять на месте, так как его сносит поток и на него движется полчища злобно настроенных рыб-врагов. При соприкосновении с ними, обе рыбы “лопаются”, а в случае если рыбка не успеет отплыть от левого края игрового поля – она будет снесена потоком и “лопнет”. Достигнув правой границы у рыбки не хватает сил её преодолеть и, она отбрасывается назад.



Рисунок 3 Стрельба пузырьком

Рисунок 2 Главный герой



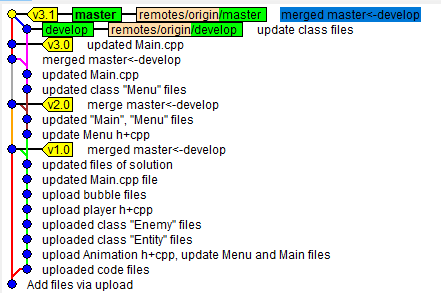
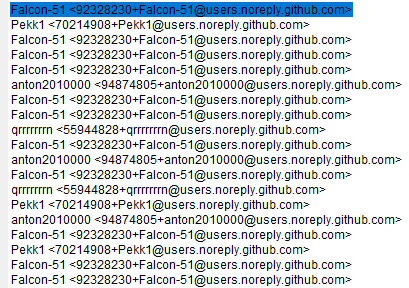
Рисунок 4 Схлопование при столкновении или попадании пузырька в врага

Рисунок 5 Рыбы-враги

3 История проекта на GitHub

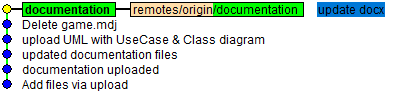
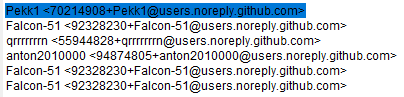
Ветка master

<https://github.com/Falcon-51/Aquarium>



* Add files via upload - gitignore uploaded (Семёнов)
* Uploaded code files - Загрузил главный файл. Создал класс Menu. Реализовал стартовый интерфейс и работу с ним (Семёнов)
* Upload Animation h+cpp; update Menu and Main files - Прописал класс Animation. Подключил к меню и мэйну (Бояркин)
* Uploaded class "Entity" files - Создал класс "Entity". Отредактировал главный файл. (Семёнов)
* Uploaded class "Enemy" files - Добавил дочерний класс "Enemy" от родительского класса "Entity" (Барановский)
* Upload player h+cpp - добавил дочерний класс Player от родительского Entity (Бояркин)
* Upload bubble files - cpp+h с классом bubble (потомок Entity) (Смирнов)
* Updated Main.cpp file - Доработал главный файл до рабочей версии продукта (Семёнов)
* Updated files of solution - Привёл к нормальному виду все файлы - подготовил к слиянию. (Барановский)
* Merged master<-develop - Рабочая версия (Семёнов)
* Update Menu h+cpp - Обновил класс меню. Отредактировал код. (Смирнов)
* Updated "Main", "Menu" files - Реализовал рабочее меню, Мелкие правки (Семёнов)
* Merge master<-develop - Объединил ветки master и develop (Семёнов)
* Updated class "Menu" files - Убрал кнопку "About". Поправил код. (Барановский)
* Updated Main.cpp - Добавил счётчик очков при уничтожении врага-рыбки. Подготовил файлы к слиянию (Cемёнов)
* Merged master<-develop - Слияние веток master и develop (Семёнов)
* Updated Main.cpp - Мелкие правки (Cемёнов)
* Update class files - Отредактировал код, cделал обработку исключений (Бояркин)
* Merged master<-develop - Слияние ветки master и develop (Семёнов)
* Updated files - Мелкие правки, реализовал обнуление счёта при смерти, изменил текстурку рыбки-врага№1 (Семёнов)

Ветка develop



* Add files via upload - gitignore uploaded (Семёнов)
* Documentation uploaded - DOCX: Добавлен шаблон отчёта (Семёнов)
* Updated documentation files - Отредактировал отчёт, добавил UML файл (Барановский)
* Upload UML with UseCase & Class diagram - Сделал диаграммы классов и вариантов использования (Смирнов)
* Delete game.mdj - Удалил лишний файл (Семёнов)
* Update docx - Добавил описание сюжета (Бояркин)
* Updated documentation files - Отредактировал отчёт (Семёнов)
* Updated documentation files - Отредактировал отчёт (Семёнов)

Заключение

В ходе выполнения курсового проекта вся работа велась в VS 2019, но с набором инструментов платформы VS 2017 (v141) на ЯП С++. Для удобной работы всех членов команды использовалась СКВ GitHub. На момент завершения работы все прецеденты были реализованы. Программа не имеет сбоев и зависаний. В разработке программы использовался принцип раздельной компиляции и очистка памяти. Программа не имеет неиспользуемых переменных, а её алгоритмы не избыточны ненужными циклами, массивами и т.д. В отчёте представлены диаграммы вариантов использования и диаграммы классов. По окончанию работы можно сделать вывод, что преследуемая цель достигнута, несмотря на трудности.