ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА

по дисциплине «Технологии разработки программного обеспечения»

на тему «KeyboardNinja»

Выполнил:
ст. гр. ИП-216
Русецкий А. С.

Проверил:
ст. преподаватель Токмашева Е. И.

Новосибирск, 23.05.2023

Содержание

Введение и постановка задачи

Клавиатурный тренажёр – программа, предназначенная для обучения или улучшения навыка печати, а также для запоминания расположения букв на клавишах клавиатуры.

Клавиатурный тренажёр представляет собой игру, предназначенную для оценки правильного выбора нужной клавиши на клавиатуре. Во время игры, в игровой области падают буквы. Ваша цель: успеть нажать нужную клавишу с буквой на клавиатуре, пока буква не достигла конца игровой области.

Передо мной и моей командной стояла цель: реализовать данную идею.

Задачи, которые необходимо было выполнить:

1. Придумать идею реализации данного проекта.
2. Написать рабочую программу.
3. Придумать дизайн программы.
4. Исправить баги и ошибки.
5. Покрытие тестами, включая unit тестирование.
6. Разбить проект на модули
7. Полностью подготовить проект к релизу.
8. Итоговая презентация.

Техническое задание

Цель проекта: Разработать интерактивную игру KeyboardNinja.

Стадии разработки и задачи на этих стадиях:

1. Разработать дизайн интерфейса игры

- Создать игровое меню
- Определить цветовую схему и общий стиль игры

2. Реализовать игровую логику

- Сделать запуск игры по нажатию кнопки SPACE
- Реализовать падение букв в игровом окне
- Добавить таймер для игры
- Реализовать проверку на правильность нажатой клавиши на клавиатуре

3. Реализовать несколько уровней сложности

- Создать меню для выбора уровня сложности
- Изменить время падения буквы на игровом окне в зависимости от уровня сложности

4. Подсчёт и оценка результата

- Добавить счётчик правильно и неправильно нажатых клавиш
- Вывести результаты в отдельное окно после окончания игры

5. Тестирование и отладка

- Протестировать функциональность на всех уровнях сложности
- Отладить игру для устранения ошибок и багов

6. Релиз и поддержка

- Выпустить игру и предоставить ее преподавателю

Описание выполненного проекта

Суть игры:

Игроку необходимо выбрать уровень сложности и нажать SPACE для старта.

После начала игры в игровом окне будут падать буквы с определённой скоростью, в зависимости от выбранного уровня сложности. Игроку необходимо правильно нажать нужную клавишу с буквой в соответствии с той, что показана на экране. По окончании таймера игроку будут представлены результаты игры (кол-во букв, кол-во правильных и неправильных нажатий).

Вид программы

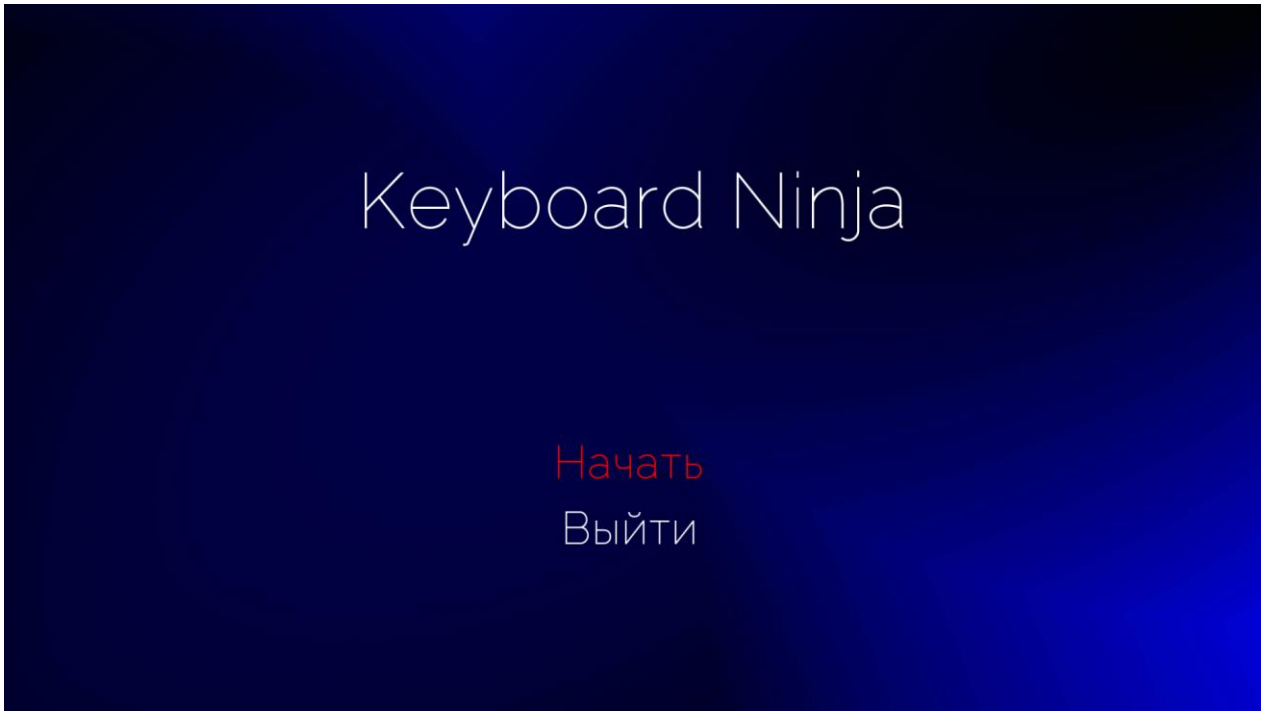

Рис. 1. Главное меню

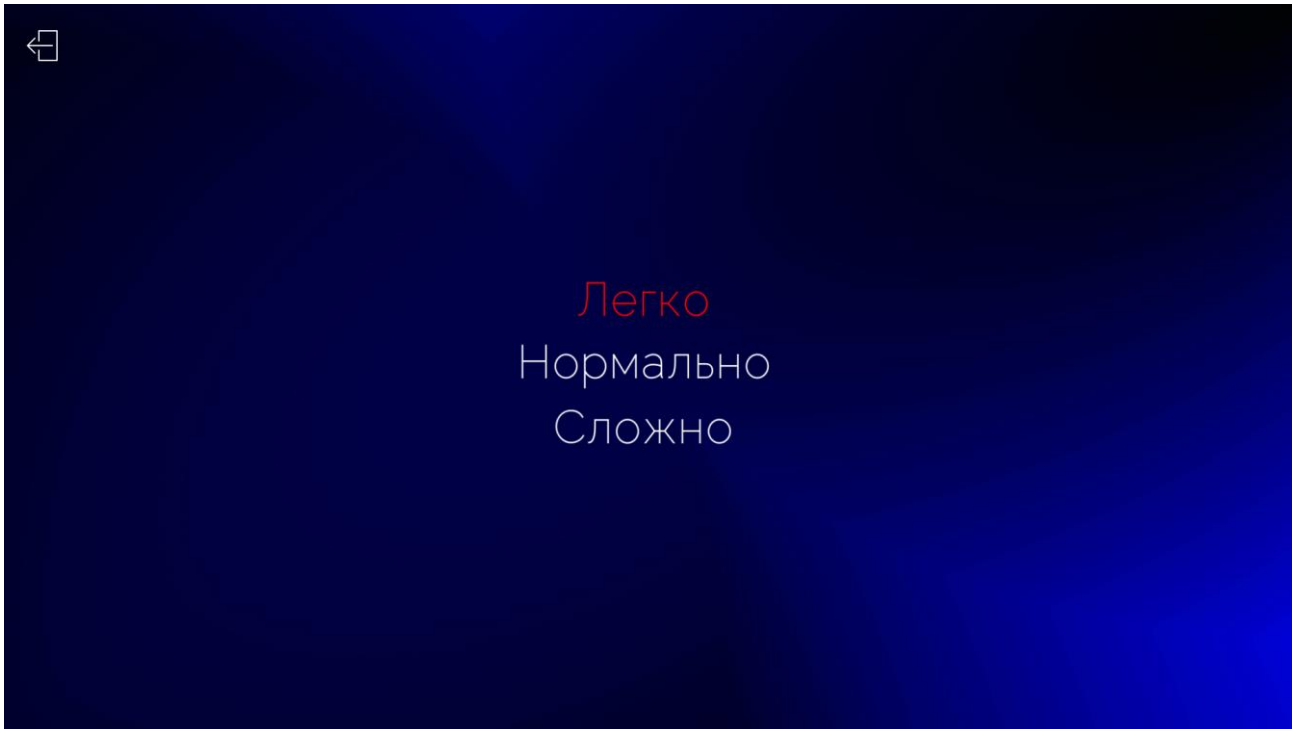

Рис. 2. Меню выбора уровня сложности

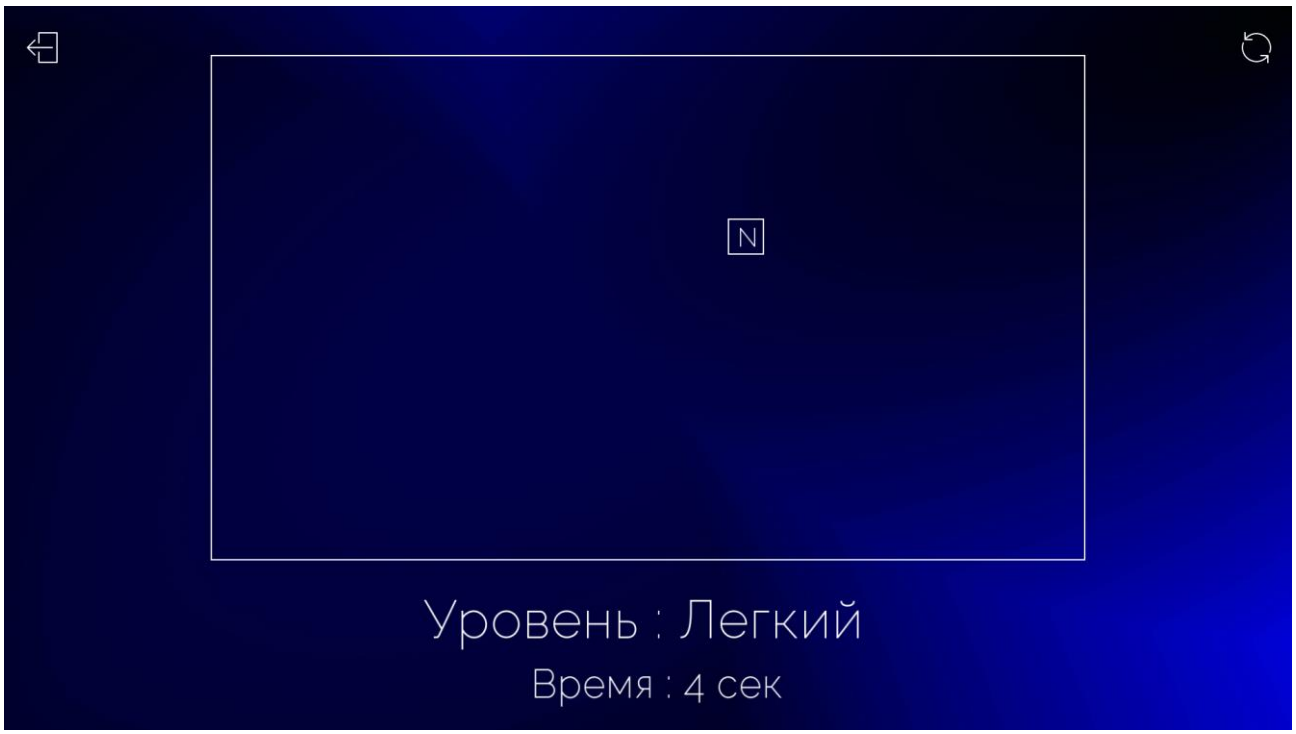

Рис. 3. Игровое меню и процесс

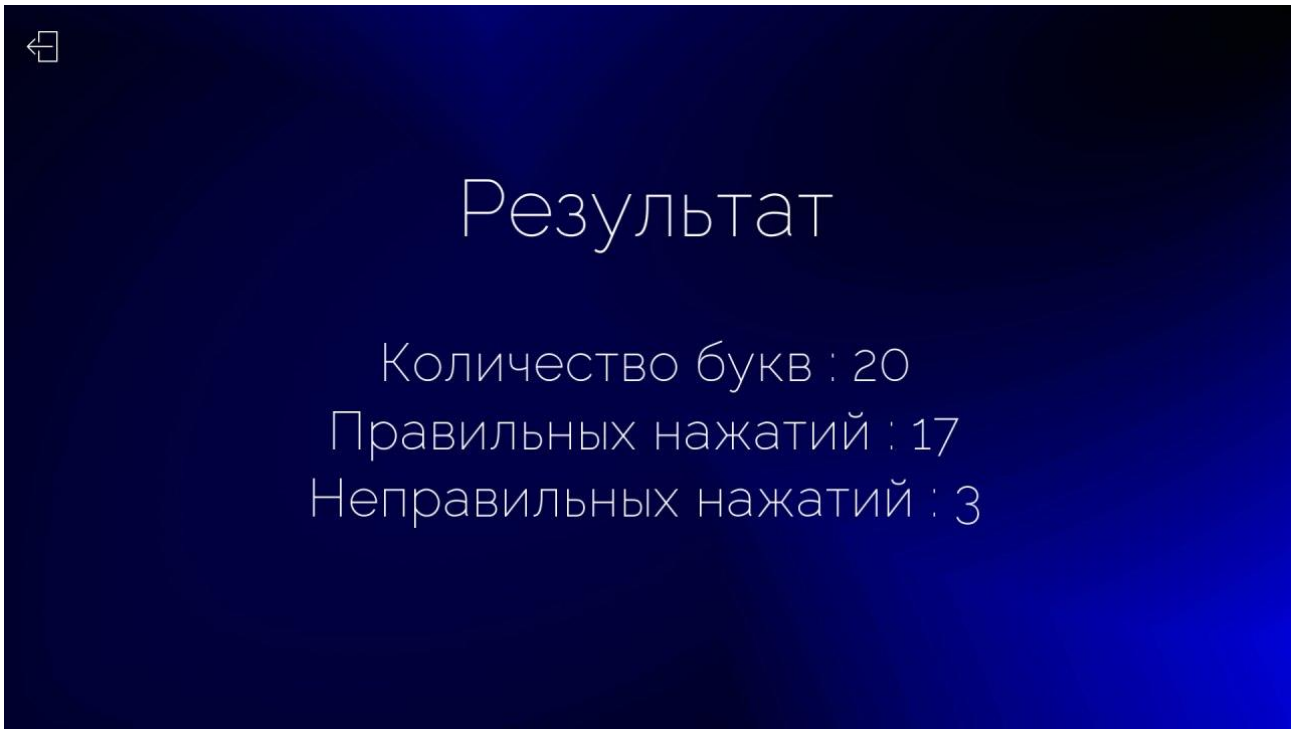

Рис. 4. Результаты

Личный вклад в проект

- Составление ТЗ и презентации продукта
- Частичная разработка интерфейса
- Разработка логики и механики игры
- Участие в разработке игрового процесса
- Помощь в написании MakeFile
- Разбиение проекта на модули

Текст программы

//file main.c

```
#include <iostream>
#include <SFML/Graphics.hpp>

#include <difficult.h>
#include <game.h>
#include <menu.h>
#include <globals.h>
#include <mode.h>
#include <window.h>
#include <fileLoad.h>

using namespace sf;

int main() {

    //Окно
    RenderWindow window;
    window.create(VideoMode::getDesktopMode(), "KeyBoardNinja", Style::Fullscreen);
    window.setFramerateLimit(60);


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Фон меню
    RectangleShape background(Vector2f(1920, 1080));
    Texture screen;
    loadTextureFromFile(screen, "Resource/Pictures/Background/3.jpg");
    background.setTexture(&screen);


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Шрифт
    Font font;
    loadFontFromFile(font, "Resource/Font/Raleway/static/Raleway-Thin.ttf");


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Заголовок
    Text title;
    initText(title, font, 120, L"Keyboard Ninja", 960, 300, Color::White);


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Кнопки

    Text buttonMenu[2];
```

```
initText(buttonMenu[0], font, 70, L"Начать", 960, 700, Color::Red);
initText(buttonMenu[1], font, 70, L"Выйти", 960, 800, Color::White);

Text buttonDifficult[3];
initText(buttonDifficult[0], font, 70, L"Легко", 960, 440, Color::Red);
initText(buttonDifficult[1], font, 70, L"Нормально", 960, 540, Color::White);
initText(buttonDifficult[2], font, 70, L"Сложно", 960, 630, Color::White);

RectangleShape logOutButton;    //Кнопка назад
Texture logOutTexture;
loadTextureFromFile(logOutTexture, "Resource/Pictures/Button/logout.png");
initButton(logOutButton, 25, 30, logOutTexture);

RectangleShape refreshButton;   //Кнопка перезапуска
Texture refreshTexture;
loadTextureFromFile(refreshTexture, "Resource/Pictures/Button/refresh.png");
initButton(refreshButton, 1835, 30, refreshTexture);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

//Текст
Text typeGamelvl[3];
initText(typeGamelvl[0], font, 80, L"Уровень : Легкий", 960, 920, Color::White);
initText(typeGamelvl[1], font, 80, L"Уровень : Нормальный", 960, 920, Color::White);
initText(typeGamelvl[2], font, 80, L"Уровень : Сложный", 960, 920, Color::White);

Text noticeMessage;
String noticeMessage_str = L"Отсчет времени начнется после нажатия SPACE";
initText(noticeMessage, font, 50, noticeMessage_str, 960, 1020, Color::White);

Text timeMessage;
timeMessage.setFont(font);
timeMessage.setStyle(Text::Bold);
timeMessage.setCharacterSize(60);
timeMessage.setFillColor(Color::White);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

//Рамки
RectangleShape board;

initFrame(board, 1300, 750, 960, 450);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

//Таймер
```

```
    Clock clock;
    int timer;


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Кубик

    RectangleShape cube;
    float xSize = 50, ySize = 50;
    float xPos = 935, yPos = 100;
    initFrame(cube, xSize, ySize, xPos, yPos);


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Буквы

    std::string letters[26];

    int z = 0;
    for (int i = 'A'; i <= 'Z'; i++) {
        letters[z] = i;
        z++;
    }

    Text letter;
    initText(letter, font, 40, "A", 935 + 10, 100 + 10, Color::White);


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    //Результат

    Text titleResult;
    initText(titleResult, font, 120, L"Результат", 960, 300, Color::White);
    Text textSumLetters;
    Text textCorrectTypes;
    Text textIncorrectTypes;


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////

    while (window.isOpen()) {
        Event ev;
        while (window.pollEvent(ev)) {
            modeMenu(window, ev, buttonMenu);
            modeDifficult(window, ev, buttonDifficult, logOutButton);
            modeGame(window, ev, logOutButton, refreshButton, clock, cube, letter);
            modeResult(window, ev, logOutButton);
        }

        window.clear();
```

```
        window.draw(background);

        switch (checkMode) {
            case 0 :
                windowMenu(window, title, buttonMenu[0], buttonMenu[1]);
                break;
            case 1 :
                windowDifficult(window, logOutButton, buttonDifficult[0], buttonDifficult[1], buttonDifficult[2]);
                break;
            case 2 :
                windowGame(window, board, typeGamelvl, noticeMessage, timeMessage, logOutButton,
refreshButton, clock, timer, cube, letter, letters);
                break;
            case 3:
                windowResult(window, font, titleResult, textSumLetters, textCorrectTypes, textIncorrectTypes,
logOutButton);
                break;
        }

        window.display();
    }

    return 0;
}
```

//file difficult.cpp

```
#include "difficult.h"

void MoveUpDifficult(Text button[], int& number) {
    if (number - 1 >= -1) {
        button[number].setFillColor(Color::White);
        number--;
        if (number == -1)
            number = 2;
        button[number].setFillColor(Color::Red);
    }
}

void MoveDownDifficult(Text button[], int& number) {
    if (number + 1 >= 1) {
        button[number].setFillColor(Color::White);
        number++;
        if (number == 3)
            number = 0;
        button[number].setFillColor(Color::Red);
    }
}

void buttonEasyCondition(Text buttonDifficult[]) {
    if (Mouse::getPosition().x >= 960 - 5 - buttonDifficult[0].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y >= 440 - buttonDifficult[0].getGlobalBounds().height / 2 &&
        Mouse::getPosition().x <= 960 + buttonDifficult[0].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y <= 440 + buttonDifficult[0].getGlobalBounds().height / 2) {
        numberDifficult = 0;
        buttonDifficult[0].setFillColor(Color::Red);
        buttonDifficult[1].setFillColor(Color::White);
```

```
        buttonDifficult[2].setFillColor(Color::White);
        if (Mouse::isButtonPressed(Mouse::Left)) {
            checkMode = 2;
        }
    }
}

void buttonNormalCondition(Text buttonDifficult[]) {
    if (Mouse::getPosition().x >= 960 - 5 - buttonDifficult[1].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y >= 540 - buttonDifficult[1].getGlobalBounds().height / 2 &&
        Mouse::getPosition().x <= 960 + buttonDifficult[1].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y <= 540 + buttonDifficult[1].getGlobalBounds().height / 2) {
        numberDifficult = 1;
        buttonDifficult[1].setFillColor(Color::Red);
        buttonDifficult[0].setFillColor(Color::White);
        buttonDifficult[2].setFillColor(Color::White);
        if (Mouse::isButtonPressed(Mouse::Left)) {
            checkMode = 2;
        }
    }
}

void buttonHardCondition(Text buttonDifficult[]) {
    if (Mouse::getPosition().x >= 960 - 5 - buttonDifficult[2].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y >= 630 - buttonDifficult[2].getGlobalBounds().height / 2 &&
        Mouse::getPosition().x <= 960 + buttonDifficult[2].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y <= 630 + buttonDifficult[2].getGlobalBounds().height / 2) {
        numberDifficult = 2;
        buttonDifficult[2].setFillColor(Color::Red);
        buttonDifficult[1].setFillColor(Color::White);
        buttonDifficult[0].setFillColor(Color::White);
        if (Mouse::isButtonPressed(Mouse::Left)) {
            checkMode = 2;
        }
    }
}
```

//file fileLoad.cpp

```
#include "fileLoad.h"

bool loadTextureFromFile(Texture& texture, std::string path) {
    if (!texture.loadFromFile(path)) return false;
    return true;
}

bool loadFontFromFile(Font& font, std::string path) {
    if (!font.loadFromFile(path)) return false;
    return true;
}
```

//file game.cpp

```
#include "game.h"

void initFrame(RectangleShape& board, float xSize, float ySize, float xPos, float yPos) {
```

```
    board.setSize(Vector2f(xSize, ySize));      //Размер рамки
    board.setFillColor(Color(0, 0, 0, 0));      //Внутренняя часть рамки прозрачная
    board.setOutlineThickness(2);               //Толщина рамки
    board.setOutlineColor(Color::White);        //Цвет рамки
    xPos = xPos - board.getSize().x / 2;
    yPos = yPos - board.getSize().y / 2;
    board.setPosition(xPos, yPos);              //Позиция рамки
}

void buttonBack(RectangleShape logOutButton, RenderWindow& window) {
    if (Mouse::getPosition().x >= 25 &&
        Mouse::getPosition().y >= 30  &&
        Mouse::getPosition().x <= 25 + logOutButton.getSize().x &&
        Mouse::getPosition().y <= 30 + logOutButton.getSize().y) {
        if (Mouse::isButtonPressed(Mouse::Left)) {
            checkMode = 0;
        }
    }
}

void buttonRefresh(RectangleShape refreshButton, RenderWindow& window, RectangleShape& cube, Text&
letter) {
    if (Mouse::getPosition().x >= 1835 &&
        Mouse::getPosition().y >= 30  &&
        Mouse::getPosition().x <= 1835 + refreshButton.getSize().x &&
        Mouse::getPosition().y <= 30 + refreshButton.getSize().y) {
        if (Mouse::isButtonPressed(Mouse::Left)) {
            sumLetters = 0;
            correctTypes = 0;
            incorrectTypes = 0;
            flagStart = 0;
            cube.setPosition(945, 110);
            letter.setPosition(945 + 12, 110);
        }
    }
}

void startTimer(Text &timeMessage, Clock& clock, RectangleShape& cube, Text& letter) {
    int timer = clock.getElapsedTime().asSeconds();
    String timerStr = L"Время : " + std::to_string(25 - timer) + L" сек";
    float xPos = 980 - 10 - timeMessage.getGlobalBounds().width / 2;    //Выравнивание по X
    float yPos = 1020 - 20 - timeMessage.getGlobalBounds().height / 2;   //Выравнивание по Y
    timeMessage.setString(timerStr);
    timeMessage.setPosition(xPos, yPos);   //Позиция текста

    if (timer >= 25)  {
        clock.restart();
        cube.setPosition(945, 110);
        letter.setPosition(945 + 12, 110);
        checkMode = 3;
        flagStart = 0;
    }
}

void gameKey(RenderWindow& window, RectangleShape& cube, Text& letter, std::string *letters) {
    srand(time(NULL));
```

```
    float speedFall = 0;
    switch (numberDifficult) {
        case 0:
            speedFall = 4;
            break;
        case 1:
            speedFall = 8;
            break;
        case 2:
            speedFall = 12;
            break;
    }
    float xPos = rand() % 1220 + 320;
    if (flagCorrect == 0) {
        if (cube.getPosition().y <= 768) {
            cube.move(0, speedFall);
            letter.move(0, speedFall);
        } else if (cube.getPosition().y >= 768) {
            cube.setPosition(xPos, 75);
            numberLetter = rand() % (25 + 1);
            letter.setString(letters[numberLetter]);
            letter.setPosition(xPos + 12, 75);
            incorrectTypes++;
            sumLetters++;
            flagCorrect = 0;
        }
    } else if (flagCorrect != 0) {
            cube.setPosition(xPos, 75);
            numberLetter = rand() % (25 + 1);
            letter.setString(letters[numberLetter]);
            letter.setPosition(xPos + 12, 75);
            flagCorrect = 0;
        }
    window.draw(letter);
    window.draw(cube);
}

void checkCorrect(Event& ev) {
    if (numberLetter == ev.key.code) {
        correctTypes++;
        flagCorrect = 1;
        sumLetters++;
    }
    else {
        sumLetters++;
        incorrectTypes++;
    }
}
```

//file globals.cpp

```
#include "globals.h"

int checkMode = 0;          //0 - Menu; 1 - Difficult; 2 - Game; 4 - Result
int numberButton = 0;       //0 - Start; 1 - Exit
int numberDifficult = 0;    //0 - Easy; 1 - Normal; 2 - Hard
int flagStart = 0;          //0 - !Start; 1 - Start
int flagCorrect = 0;
```

```
int numberLetter;
int sumLetters = 0;
int correctTypes = 0;
int incorrectTypes = 0;
```

//file menu.cpp

```
#include "menu.h"

void initText(Text& text, Font& font, int size, String str, float xPos, float yPos,  Color textColor) {
    text.setFont(font);             //Шрифт
    text.setStyle(Text::Bold);      //Толщина шрифта
    text.setCharacterSize(size);    //Размер шрифта
    text.setString(str);            //Текст

    xPos = xPos - 10 - text.getGlobalBounds().width / 2;    //Выравнивание по X
    yPos = yPos - 20 - text.getGlobalBounds().height / 2;   //Выравнивание по Y

    text.setPosition(xPos, yPos);   //Позиция текста
    text.setFillColor(textColor);   //Цвет текста

}

void initButton(RectangleShape& button, float xPos, float yPos, Texture& t) {
    button.setSize(Vector2f(64, 64));
    t.setSmooth(true);
    button.setTexture(&t);
    button.setPosition(xPos, yPos);
}

void MoveUp(Text button[], int& number) {
    if (number - 1 >= -1) {
        button[number].setFillColor(Color::White);
        number--;
        if (number == -1)
            number = 1;
        button[number].setFillColor(Color::Red);
    }
}

void MoveDown(Text button[], int& number) {
    if (number + 1 >= 1) {
        button[number].setFillColor(Color::White);
        number++;
        if (number == 2)
            number = 0;
        button[number].setFillColor(Color::Red);
    }
}

void buttonStartCondition(Text buttonMenu[]) {
    if (Mouse::getPosition().x >= 960 - 5 - buttonMenu[0].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y >= 700 - buttonMenu[0].getGlobalBounds().height / 2 &&
        Mouse::getPosition().x <= 960 + buttonMenu[0].getGlobalBounds().width / 2 &&
```

```
        Mouse::getPosition().y <= 700 + buttonMenu[0].getGlobalBounds().height / 2) {
        numberButton = 0;
        buttonMenu[0].setFillColor(Color::Red);
        buttonMenu[1].setFillColor(Color::White);
        if (Mouse::isButtonPressed(Mouse::Left)) {
            checkMode = 1;
        }
    }
}

void buttonExitCondition(Text buttonMenu[], RenderWindow& window) {
    if (Mouse::getPosition().x >= 960 - 5 - buttonMenu[1].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y >= 800 - buttonMenu[1].getGlobalBounds().height / 2 &&
        Mouse::getPosition().x <= 960 + buttonMenu[1].getGlobalBounds().width / 2 &&
        Mouse::getPosition().y <= 800 + buttonMenu[1].getGlobalBounds().height / 2) {
        numberButton = 1;
        buttonMenu[1].setFillColor(Color::Red);
        buttonMenu[0].setFillColor(Color::White);
        if (Mouse::isButtonPressed(Mouse::Left)) {
            window.close();
        }
    }
}
```

//file mode.cpp

```
#include "mode.h"

void modeMenu (RenderWindow& window, Event &ev, Text buttonMenu[]) {
    if (checkMode == 0) {
        buttonStartCondition(buttonMenu);
        buttonExitCondition(buttonMenu, window);
        switch (ev.type) {
            case Event::KeyPressed :
                switch (ev.key.code) {
                    case Keyboard::Escape :
                        window.close();
                    case Keyboard::Up :
                        MoveUp(buttonMenu, numberButton);
                        break;
                    case Keyboard::Down :
                        MoveDown(buttonMenu, numberButton);
                        break;
                    case Keyboard::Enter :
                        if (numberButton == 0) {
                            checkMode = 1;
                            ev.key.code = Keyboard::Unknown;
                        }
                        else if (numberButton == 1)
                            window.close();
                }
        }
    }
}

void modeDifficult (RenderWindow &window, Event &ev, Text buttonDifficult[], RectangleShape logOutButton)
{
    if (checkMode == 1) {
        buttonBack(logOutButton, window);
```

```
        buttonEasyCondition(buttonDifficult);
        buttonNormalCondition(buttonDifficult);
        buttonHardCondition(buttonDifficult);
        switch (ev.type) {
            case Event::KeyPressed :
                switch (ev.key.code) {
                    case Keyboard::Escape :
                        checkMode = 0;
                        numberDifficult = 0;
                        break;
                    case Keyboard::Up :
                        MoveUpDifficult(buttonDifficult, numberDifficult);
                        break;
                    case Keyboard::Down :
                        MoveDownDifficult(buttonDifficult, numberDifficult);
                        break;
                    case Keyboard::Enter :
                        checkMode = 2;
                }
        }
    }
}

void modeGame(RenderWindow &window, Event &ev, RectangleShape& logOutButton, RectangleShape&
refreshButton, Clock& clock, RectangleShape& cube, Text& letter) {
    if (checkMode == 2) {
        buttonBack(logOutButton, window);
        buttonRefresh(refreshButton, window, cube, letter);
        switch (ev.type) {
            case Event::KeyPressed :
                switch (ev.key.code) {
                    case Keyboard::Space :
                        flagStart = 1;
                        clock.restart();
                        break;
                    case Keyboard::Escape :
                        checkMode = 1;
                        flagStart = 0;
                        sumLetters = 0;
                        correctTypes = 0;
                        incorrectTypes = 0;
                        cube.setPosition(945, 110);
                        letter.setPosition(945 + 12, 110);
                        break;
                    default :
                        checkCorrect(ev);
                        break;
                }
        }
    }
}

void modeResult(RenderWindow& window, Event& ev, RectangleShape& logOutButton) {
    if (checkMode == 3) {
        buttonBack(logOutButton, window);
        switch (ev.type) {
            case Event::KeyPressed :
                switch (ev.key.code) {
                    case Keyboard::Escape :
                        sumLetters = 0;
```

```
                correctTypes = 0;
                incorrectTypes = 0;
                checkMode = 0;
                break;
            }
        }
    }
}
```

//file window.cpp

```
#include "window.h"
#include "menu.h"

void windowMenu(RenderWindow& window, Text& title, Text& button1, Text& button2) {
    window.draw(title);     //Заголовок
    window.draw(button1);   //Кнопка "Начать"
    window.draw(button2);   //Кнопка "Выход"
}

void windowDifficult (RenderWindow& window, RectangleShape& logOutButton, Text& button1, Text&
button2, Text& button3) {
    window.draw(logOutButton);
    window.draw(button1);   //Кнопка "Легко"
    window.draw(button2);   //Кнопка "Нормально"
    window.draw(button3);   //Кнопка "Сложно"
}

void windowGame(RenderWindow& window, RectangleShape& board, Text typeGamelvl[], Text&
noticeMessage,
            Text& timeMessage, RectangleShape& logOutButton, RectangleShape& refreshButton, Clock&
clock, int& timer,
            RectangleShape& cube, Text& letter, std::string *letters) {
    window.draw(board);
    window.draw(typeGamelvl[numberDifficult]);
    if (flagStart != 1) window.draw(noticeMessage);
    else if (flagStart == 1)  {
        startTimer(timeMessage, clock, cube, letter);
        gameKey(window, cube, letter, letters);
        window.draw(timeMessage);
    }
    window.draw(logOutButton);
    window.draw(refreshButton);
}

void windowResult(RenderWindow& window, Font& font, Text& titleResult, Text& textSumLetters, Text&
textCorrectTypes, Text& textIncorrectTypes, RectangleShape& logOutButton) {
    initText(textSumLetters, font, 80, L"Количество букв : " + std::to_string(sumLetters), 960, 540,
Color::White);
    initText(textCorrectTypes, font, 80, L"Правильных нажатий : " + std::to_string(correctTypes), 960, 640,
Color::White);
    initText(textIncorrectTypes, font, 80, L"Неправильных нажатий : " + std::to_string(incorrectTypes), 960,
740, Color::White);
    window.draw(titleResult);
    window.draw(textSumLetters);
```

```
    window.draw(textCorrectTypes);
    window.draw(textIncorrectTypes);
    window.draw(logOutButton);
}
```

//makefile

```
APP_NAME = main
LIB_NAME = lib
TEST_NAME = testmain

TESTFLAGS = -I thirdparty
CFLAGS = -lsfml-graphics -lsfml-system -lsfml-window -I src/lib
DEPSFLAGS = -MMD
CC = g++

BIN_DIR = bin
OBJ_DIR = obj
SRC_DIR = src
TEST_DIR = test

APP_PATH = $(BIN_DIR)/$(APP_NAME)
LIB_PATH = $(OBJ_DIR)/$(SRC_DIR)/$(LIB_NAME)/$(LIB_NAME).a
TEST_PATH = $(BIN_DIR)/$(TEST_NAME)

APP_SOURCES = $(wildcard $(SRC_DIR)/$(APP_NAME)/*.cpp)
APP_OBJECTS = $(patsubst %.cpp, $(OBJ_DIR)/%.o, $(APP_SOURCES))

LIB_SOURCES = $(wildcard $(SRC_DIR)/$(LIB_NAME)/*.cpp)
LIB_OBJECTS = $(patsubst %.cpp, $(OBJ_DIR)/%.o, $(LIB_SOURCES))

TEST_SOURCES = $(wildcard $(TEST_DIR)/*.cpp)
TEST_OBJECTS = $(patsubst %.cpp, $(OBJ_DIR)/%.o, $(TEST_SOURCES))

DEPS = $(APP_OBJECTS:.o=.d) $(LIB_OBJECTS:.o=.d)

all: $(APP_PATH)

-include $(DEPS)

$(APP_PATH): $(APP_OBJECTS) $(LIB_PATH)
	$(CC) $(CFLAGS) -o $@ $^

$(LIB_PATH): $(LIB_OBJECTS)
	ar rcs $@ $^

$(OBJ_DIR)/%.o: %.cpp
	$(CC) $(CFLAGS) $(DEPSFLAGS) -c -o $@ $<

test: $(LIB_PATH) $(TEST_PATH)
	$(TEST_PATH)

$(TEST_PATH): $(TEST_OBJECTS) $(LIB_PATH)
	$(CC) $(TESTFLAGS) $(CFLAGS) -o $@ $^

$(OBJ_DIR)/test/main.o: test/main.cpp
	$(CC) $(TESTFLAGS) $(CFLAGS) $(DEPSFLAGS) -c -o $@ $<
```

```
$(OBJ_DIR)/test/parser_test.o: test/parser_test.cpp
	$(CC) $(TESTFLAGS) $(CFLAGS) $(DEPSFLAGS) -c -o $@ $<

run : $(APP_PATH)
	$(APP_PATH)

clean:
	$(RM) $(APP_PATH) $(TEST_PATH) $(OBJ_DIR)/*/*/*.[aod] $(OBJ_DIR)/test/*.[aod]
```