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.

Содержание

Введение и постановка задачи …………………………………………………………………… 3

Задание на курсовую работу ...…………………………………………………………………… 4

[Техническое](#__RefHeading___Toc134_1766323042) задание ……………………………………………………………………………… 4

Описание выполненного проекта ....……………………………………………………………... 5

Тестирование программы ..……………………………………………………………………….. 7

Метод сборки проекта …………………………………………………………………………… 11

Результаты работы программы ………………………………………………………………….. 14

Личный вклад в проект ………………………………………………………………………….. 16

# Введение и постановка задачи

Передо мной и моей командной стояла цель: работая в команде, реализовать законченный продукт.

Задачи, которые были поставлены:

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. Практика использования CI.
7. Разбить проект на модули
8. Полностью подготовить проект к релизу.
9. Итоговая презентация.

**Задание на курсовую работу**

В качестве темы для курсовой работы нами была выбрана игра «KeyboardNinja», правила игры предельно просты:

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

**Техническое задание**

**Цель проекта**: Разработать интерактивную игру «KeyboardNinja».

1. **Стадии разработки и задачи на этих стадиях:**
2. **1. Разработать дизайн интерфейса игры**
3. - Создать игровое меню
4. - Определить цветовую схему и общий стиль игры
5. **2. Реализовать игровую логику**

- Сделать запуск игры по нажатию кнопки SPACE

- Реализовать падение букв в игровом окне

- Реализовать проверку на правильность нажатой клавиши на клавиатуре

**3. Реализовать несколько уровней сложности**

- Создать меню для выбора уровня сложности

- Изменить скорость падения буквы на игровом окне в зависимости от уровня сложности

**4. Подсчёт и оценка результата**

**-** Добавить счётчик правильно и неправильно нажатых клавиш

**-** Добавить таймер для игры

- 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

Отдельно опишем содержимое файлов программы

Файл **difficult.cpp**:

В данном файле содержатся функции:

* «MoveUpDifficult», «MoveDownDifficult» - Переключение кнопок уровня сложности при помощи стрелок на клавиатуре.
* «buttonEasyCondition», «buttonNormalCondition», «buttonHardCondition» - Кнопки «Easy», «Normal», «Hard».

Файл **fileload.cpp**:

Данный файл содержит функции:

* «loadTextureFromFile», «loadFontFromFile» - Проверка загрузки текстур и шрифта.

Файл **game.cpp**:

В данном файле содержатся функции:

* «initFrame» - Отрисовка рамки
* «buttonBack» - Кнопка «Назад»
* «buttonRefresh» - Кнопки «Обновить»
* «startTimer» - Таймер игры
* «selectDifficult» - уровень сложности
* «startPositionCube», «gameKey» - Игровой процесс
* «checkCorrect» - Счётчик правильных и неправильных нажатий

Файл **menu.cpp**:

В данном файле содержатся функции:

* «initText» - Отрисовка текста в меню
* «initButton» - Отрисовка кнопки
* «MoveUp», «MoveDown» - Переключение между кнопками «Начать» и «Выйти» при помощи стрелок на клавиатуре
* «buttonStartCondition», «buttonExitCondition» - Кнопки «Начать» и «Выйти»

Файл **mode.cpp**:

В данном файле содержатся функции:

* «modeMenu» - Стартовое меню
* «modeDifficult» - Меню выбора уровня сложности
* «modeGame» - Игровое окно
* «modeResult» - Окно результатов

Файл **window.cpp**:

В данном файле содержатся функции:

* «windowMenu» - Вывод стартового меню
* «windowDifficult» - Вывод меню выбора уровня сложности
* «windowGame» - Вывод игрового окна
* «windowResult» - Вывод окна результатов

**Тестирование программы**

**parser\_test.cpp**

#include <ctest.h>

#include <fileLoad.h>

#include <difficult.h>

#include <menu.h>

#include <game.h>

//проверка загрузки шрифта

CTEST (ctest, loadFontFromFile) {

Font fontTrue;

bool checkFontTrue = loadFontFromFile(fontTrue, "Resource/Font/Raleway/static/Raleway-Thin.ttf") ;

ASSERT\_TRUE(checkFontTrue);

Font fontFalse;

bool checkFontFalse = loadFontFromFile(fontFalse, "Resource/Font/Raleway/static/Ralewy-Thin.ttf") ;

ASSERT\_FALSE(checkFontFalse);

}

//проверка загрузки заднего фона

CTEST (ctest, loadTextureFromFile) {

Texture screenTrue;

bool checkScreenTrue = loadTextureFromFile(screenTrue, "Resource/Pictures/Background/3.jpg") ;

ASSERT\_TRUE(checkScreenTrue);

Texture screenFalse;

bool checkScreenFalse = loadTextureFromFile(screenFalse, "Resource/Pictures/Background/3333.jpg") ;

ASSERT\_FALSE(checkScreenFalse);

}

//проверка переключения клавиш

CTEST (ctest, MoveUp) {

Text button[2];

int numberTrue = 0;

bool MoveUpTrue = MoveUp(button, numberTrue);

ASSERT\_TRUE(MoveUpTrue);

int numberFalse = -4;

bool MoveUpFalse = MoveUp(button, numberFalse);

ASSERT\_FALSE(MoveUpFalse);

}

//проверка переключения клавиш

CTEST (ctest, MoveDown) {

Text button[2];

int numberTrue = 1;

bool MoveUpTrue = MoveUp(button, numberTrue);

ASSERT\_TRUE(MoveUpTrue);

int numberFalse = -1;

bool MoveUpFalse = MoveUp(button, numberFalse);

ASSERT\_FALSE(MoveUpFalse);

}

//проверка переключения клавиш

CTEST (ctest, MoveUpDifficult) {

Text button[3];

int numberTrue = 0;

bool MoveUpDifficultTrue = MoveUpDifficult(button, numberTrue);

ASSERT\_TRUE(MoveUpDifficultTrue);

int numberFalse = -5;

bool MoveUpDifficultFalse = MoveUpDifficult(button, numberFalse);

ASSERT\_FALSE(MoveUpDifficultFalse);

}

//проверка переключения клавиш

CTEST (ctest, MoveDownDifficult) {

Text button[3];

int numberTrue = 0;

bool MoveDownDifficultTrue = MoveDownDifficult(button, numberTrue);

ASSERT\_TRUE(MoveDownDifficultTrue);

int numberFalse = -5;

bool MoveDownDifficultFalse = MoveDownDifficult(button, numberFalse);

ASSERT\_FALSE(MoveDownDifficultFalse);

}

//проверка выбора уровня сложности

CTEST (ctest, selectDifficult) {

int numberDifficultTrue = 0;

float speedFall;

bool selectDifficultTrue = selectDifficult(numberDifficultTrue, speedFall);

ASSERT\_TRUE(selectDifficultTrue );

int numberDifficultFalse = -4;

bool selectDifficultFalse = selectDifficult(numberDifficultFalse, speedFall);

ASSERT\_FALSE(selectDifficultFalse );

}

//проверка игрового процесса

CTEST (ctest, startPositionCube) {

float xPos = 600;

RectangleShape cube;

int numberLetter;

Text letter;

std::string letters[26];

startPositionCube(xPos, cube, numberLetter, letter, letters);

ASSERT\_TRUE(cube.getPosition().x == xPos);

}

**Метод сборки проекта**

**Для сборки проекта был использован Makefile:**

APP\_NAME = main

LIB\_NAME = lib

TEST\_NAME = testmain

TESTFLAGS = -I thirdparty

SFML\_FLAGS = -lsfml-graphics -lsfml-system -lsfml-window

CFLAGS = -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 $@ $^ $(SFML\_FLAGS)

$(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 $@ $^ $(SFML\_FLAGS)

$(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)

./bin/main

clean:

$(RM) $(APP\_PATH) $(TEST\_PATH) $(OBJ\_DIR)/\*/\*/\*.[aod] $(OBJ\_DIR)/test/\*.[aod]

**Результаты работы программы:**



Рис. 1. Главное меню



Рис. 2. Меню выбора уровня сложности

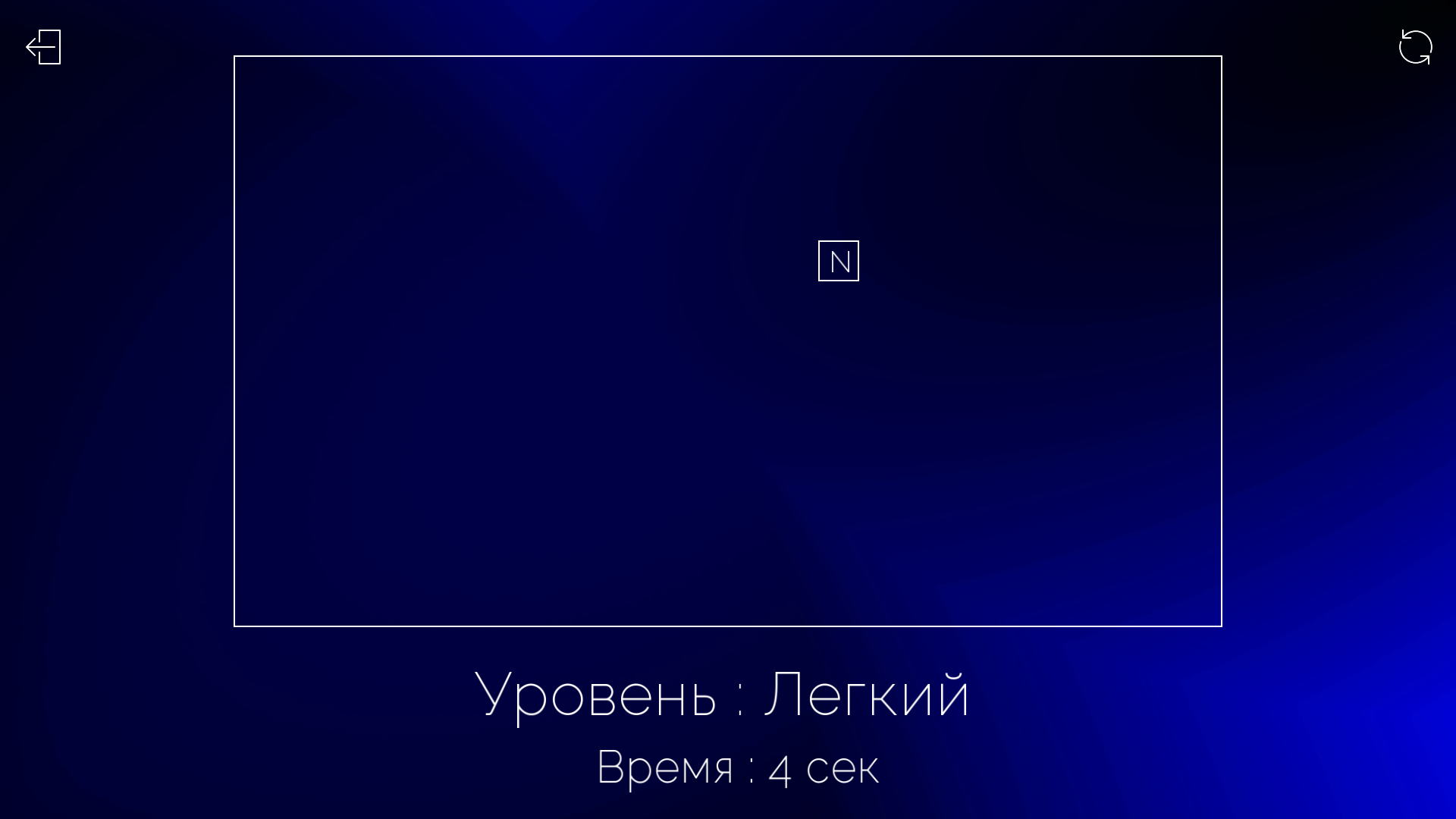


Рис. 3. Игровое меню и процесс

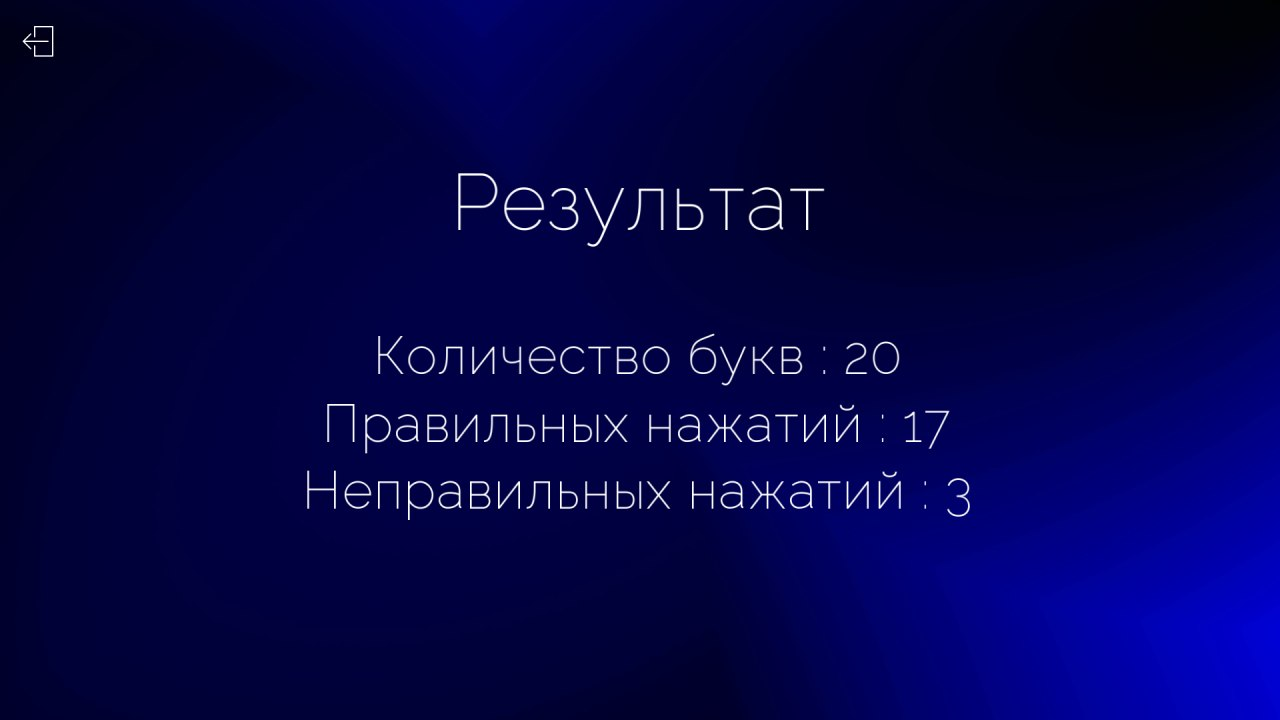


Рис. 4. Результаты

# Личный вклад в проект

* + Составление ТЗ и презентации продукта
  + Разработка интерфейса
  + Разработка логики и механики игры
  + Участие в разработке игрового процесса
  + Помощь в написании MakeFile
  + Разбиение проекта на модули
  + Частичное написание тестов