*Государственное образовательное учреждение высшего профессионального образования*



ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Системы обработки информации и управления» (ИУ-5)

**ДОМАШНЕЕ ЗАДАНИЕ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа ИУ5-35Б

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **16.12.2024** /**А. А. Торопыгин/** 

(Подпись, дата) (И.О.Фамилия)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /**Ю. Е. Гапанюк/**

(Подпись, дата) (И.О.Фамилия)

2024

**Задание:**

Разработать программу на языке Rust.

Реализовать классический вариант игры “2048”:

* Игра запускается в отдельном окне
* Матрица 4х4
* Плитки появляются, двигаются и складываются по обычным правилам
* Есть возможность отменить последнее действие
* Отображается изменяющийся счет
* Рекорд сохраняется при новом запуске
* Есть кнопка перезапуска

**Текст программы:**

**(для работы fltk может быть нужно установить** [**зависимости**](https://fltk-rs.github.io/fltk-book/Setup.html)**)**

extern crate rand;

use std::fs::File;

use std::io::{Read, Write};

use button::Button;

use group::Grid;

use frame::Frame;

use app::App;

use window::Window;

use rand::{rngs::ThreadRng, Rng};

use enums::Color;

use fltk::{enums::{self, Font, Key}, prelude::\*, \*};

// Палитра для плиток

const COLORS: [Color; 12] = [

Color::from\_rgb(238, 228, 218),

Color::from\_rgb(237, 224, 200),

Color::from\_rgb(242, 177, 121),

Color::from\_rgb(245, 149, 99),

Color::from\_rgb(246, 124, 95),

Color::from\_rgb(246, 94, 59),

Color::from\_rgb(237, 207, 114),

Color::from\_rgb(237, 204, 97),

Color::from\_rgb(237, 200, 80),

Color::from\_rgb(237, 197, 63),

Color::from\_rgb(60, 58, 55),

Color::from\_rgb(62, 57, 51),

];

fn main() {

let a = App::default(); // объект приложения

let mut win = Window::default(); // объект окна

win.set\_pos(100, 100);

win.set\_size(1600, 900);

win.set\_label("2049");

win.set\_color(Color::White);

let mut table = Grid::default(); // объект для отображения игрового поля

table.set\_pos(50, 50);

table.set\_size(800, 800);

table.show\_grid\_with\_color(true,Color::Black);

table.set\_layout(4, 4);

table.set\_color(Color::BackGround);

for i in 0..16 {

let \_ = table.set\_widget(&mut Frame::default()

.with\_size(200, 200), i/4, i%4);

} // заполняем поле плитками

table.end();

let mut top\_score = pull\_top\_score();

let mut top\_scoreboard = Frame::default(); // объект для отображения рекорда

top\_scoreboard.set\_pos(1100, 100);

top\_scoreboard.set\_size(300, 100);

top\_scoreboard.set\_label(format!("Top score:\n{}", top\_score).as\_str());

top\_scoreboard.set\_label\_size(32);

top\_scoreboard.set\_label\_font(Font::CourierBold);

let mut scoreboard = Frame::default(); // объект для отображения счета

scoreboard.set\_pos(1100, 300);

scoreboard.set\_size(300, 100);

scoreboard.set\_label("Your score:\n0");

scoreboard.set\_label\_size(32);

scoreboard.set\_label\_font(Font::CourierBold);

let mut rest\_btn = Button::default(); //кнопка перезапуска

rest\_btn.set\_pos(1150, 550);

rest\_btn.set\_size(200, 200);

rest\_btn.set\_label("@refresh");

rest\_btn.set\_label\_size(40);

win.end();

win.show();

let mut \_field = Field { // создаем игровое поле

table: table,

matrix: [[0; 4]; 4],

lst\_matrix: [[0; 4]; 4],

board: scoreboard,

score: 0,

lst\_score: 0,

random: rand::thread\_rng()

};

\_field.drop();

\_field.redraw();

win.handle({ // обработка событий

move |\_win, ev| {

match ev {

enums::Event::KeyUp => { // если была нажата кнопка

let key = app::event\_key();

match key { // определяем нажатую кнопку

Key::Up => \_field.swipe(Direction::Up),

Key::Down => \_field.swipe(Direction::Down),

Key::Right => \_field.swipe(Direction::Right),

Key::Left => \_field.swipe(Direction::Left),

Key::BackSpace => \_field.revert(),

\_ => ()

}

top\_score = top\_score.max(\_field.score); // обновляем рекорд

top\_scoreboard.set\_label(format!("Top score:\n{}", top\_score).as\_str());

push\_top\_score(top\_score);

}

enums::Event::Push => { // если было нажатие курсора

let coords = app::event\_coords(); // определяем координату

if coords.0 >= 1150 && coords.0 <= 1350 &&

coords.1 >= 550 && coords.1 <= 750 {

\_field.restart();

} // перезапуск если попали по кнопке

}

\_ => {}

}

true

}

});

a.run().unwrap();

}

// загрузка рекорда из файла

fn pull\_top\_score() -> i32 {

let mut file = match File::open("top.txt") {

Ok(f) => f,

Err (\_) => File::create("top.txt").unwrap()

};

let mut line = String::new();

\_ = file.read\_to\_string(&mut line);

match line.parse() {

Ok(value) => return value,

Err(\_) => return 0

};

}

// сохранение рекорда в файл

fn push\_top\_score(score: i32) {

let mut file = File::create("top.txt").unwrap();

\_ = file.write(score.to\_string().as\_bytes());

}

// объект игрового поля

struct Field {

table: Grid,

matrix: [[i32; 4]; 4],

lst\_matrix: [[i32; 4]; 4],

board: Frame,

score: i32,

lst\_score: i32,

random: ThreadRng

}

// направления движения плиток

enum Direction {

Up,

Down,

Right,

Left

}

impl Field{

// появление новой плитки в случайном незанятом месте

fn drop(&mut self) {

loop {

let cell = self.random.gen\_range(0..16);

if self.matrix[cell/4][cell%4] == 0 {

if self.random.gen\_range(0..10) == 0{

self.matrix[cell/4][cell%4] = 4;

}

else {

self.matrix[cell/4][cell%4] = 2;

}

return;

}

}

}

// отрисовка изменений

fn redraw(&mut self) {

for i in 0..16 {

let mut tile = self.table.child(i as i32).unwrap();

if self.matrix[i/4][i%4] == 0 {

tile.set\_label("");

tile.set\_frame(enums::FrameType::NoBox);

}

else {

tile.set\_label(self.matrix[i/4][i%4].to\_string().as\_str());

tile.set\_label\_font(Font::CourierBold);

tile.set\_label\_size(64);

let color\_index = match self.matrix[i/4][i%4] {

0 => 0,

v => (v as f64).log2() as usize + 1,

};

tile.set\_frame(enums::FrameType::RoundedBox);

tile.set\_color(COLORS[color\_index.min(11)]);

}

let \_ = self.table.set\_widget(&mut tile.clone(), i/4, i%4);

}

self.board.set\_label(format!("Your score:\n{}", self.score).as\_str());

}

// движение и сложение плиток в направлении

pub fn swipe(&mut self, dir: Direction) {

self.lst\_score = self.score;

for i in 0..16 {self.lst\_matrix[i/4][i%4] = self.matrix[i/4][i%4];}

let mut is\_changed = false;

for i in 0..4 {

let mut temp = [0; 4];

let mut ind = 0;

for j in 0..4 {

let val = match dir {

Direction::Up => self.matrix[j][i],

Direction::Down => self.matrix[3-j][i],

Direction::Right => self.matrix[i][3-j],

Direction::Left => self.matrix[i][j]

};

if val == 0 {continue;}

if temp[ind] == 0 {

temp[ind] = val;

}

else if temp[ind] == val {

temp[ind] += val;

self.score += val;

ind += 1;

}

else {

ind += 1;

temp[ind] = val;

}

}

for j in 0..4 {

match dir {

Direction::Up => {

if self.matrix[j][i] != temp[j] {

self.matrix[j][i] = temp[j];

is\_changed = true;

}

},

Direction::Down => {

if self.matrix[3-j][i] != temp[j] {

self.matrix[3-j][i] = temp[j];

is\_changed = true;

}

},

Direction::Right => {

if self.matrix[i][3-j] != temp[j] {

self.matrix[i][3-j] = temp[j];

is\_changed = true;

}

},

Direction::Left => {

if self.matrix[i][j] != temp[j] {

self.matrix[i][j] = temp[j];

is\_changed = true;

}

}

};

}

}

if is\_changed {

self.drop();

self.redraw();

}

}

// отмена последнего действия

pub fn revert(&mut self) {

self.score = self.lst\_score;

for i in 0..16 {self.matrix[i/4][i%4] = self.lst\_matrix[i/4][i%4];}

self.redraw();

}

// перезапуск игры

pub fn restart(&mut self) {

self.matrix = [[0; 4]; 4];

self.lst\_matrix = [[0; 4]; 4];

self.score = 0;

self.lst\_score = 0;

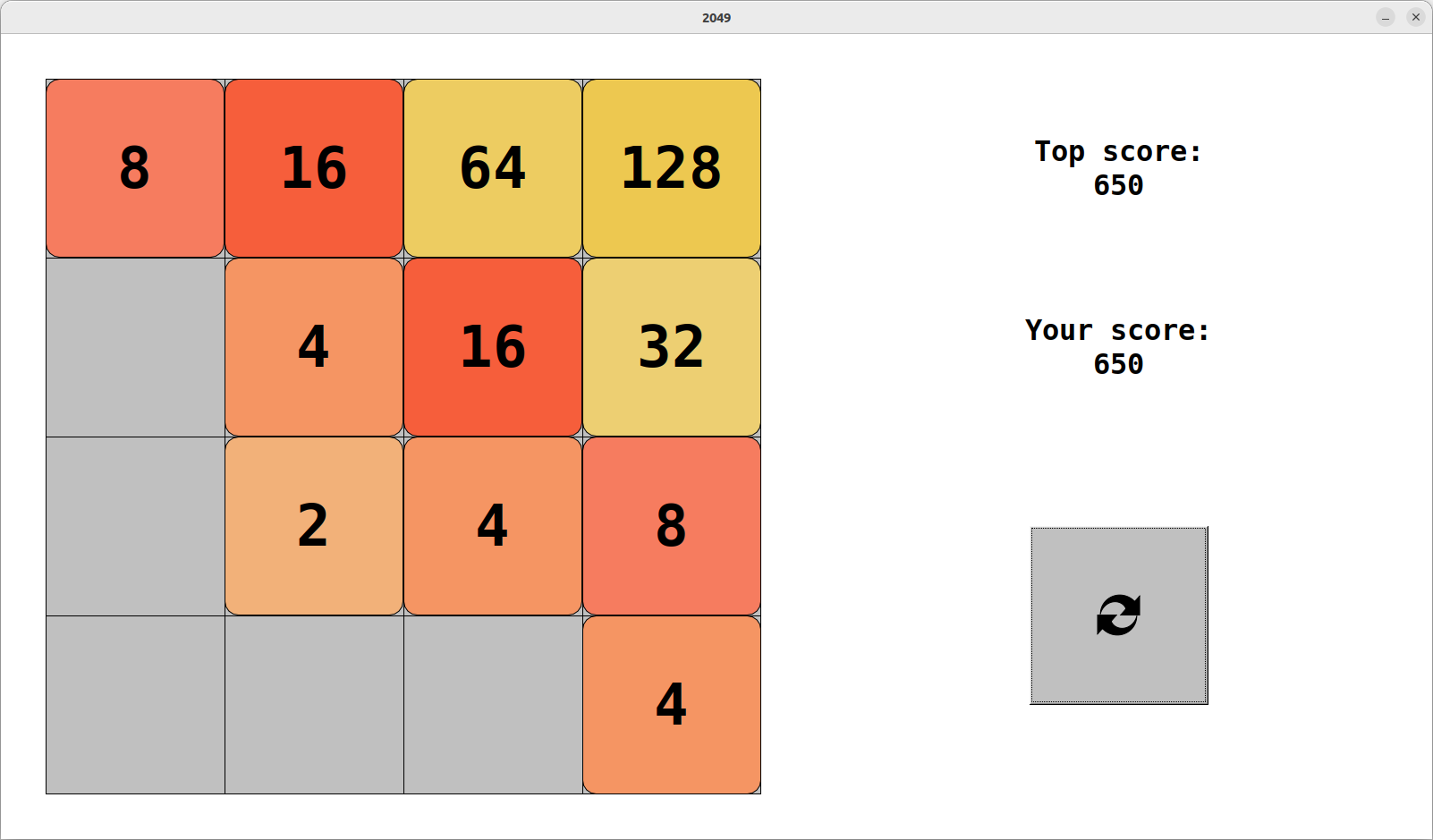
self.drop();

self.redraw();

}

}

**Результат:**

****