Министерство цифрового развития, связи и массовых коммуникаций РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский Государственный Университет Телекоммуникаций и Информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики (ПмиК)

09.03.01Программное обеспечение средств вычислительной техники и автоматизированных систем

(очная форма обучения)

«Приложение-симулятор логических схем»

ОТЧЕТ ПО РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЕ

По дисциплине «Визуальное программирование и человеко-машинное взаимодействие»

Выполнил:

студент гр. ИП-114 А.И. Амошенко

Проверил:

Ассистент каф. ПМиК И.А. Меркулов

Новосибирск 2023 г.

Оглавление

Задание:

Реализовать приложение-симулятор логических схем.

Работа состоит из следующих этапов:

1. Создание Use-Case диаграммы приложения. По окончании этапа должны быть построены Use-Case диаграммы.

2. Разработка графического интерфейса (схематичное изображение интерфейса и описание возможностей элементов, достижения сценариев описанных в Use-Case диаграмме посредством этих элементов). По окончании этапа должна быть построена схема интерфейса с подробным описанием элементов и достижения сценариев из use-case диаграммы.

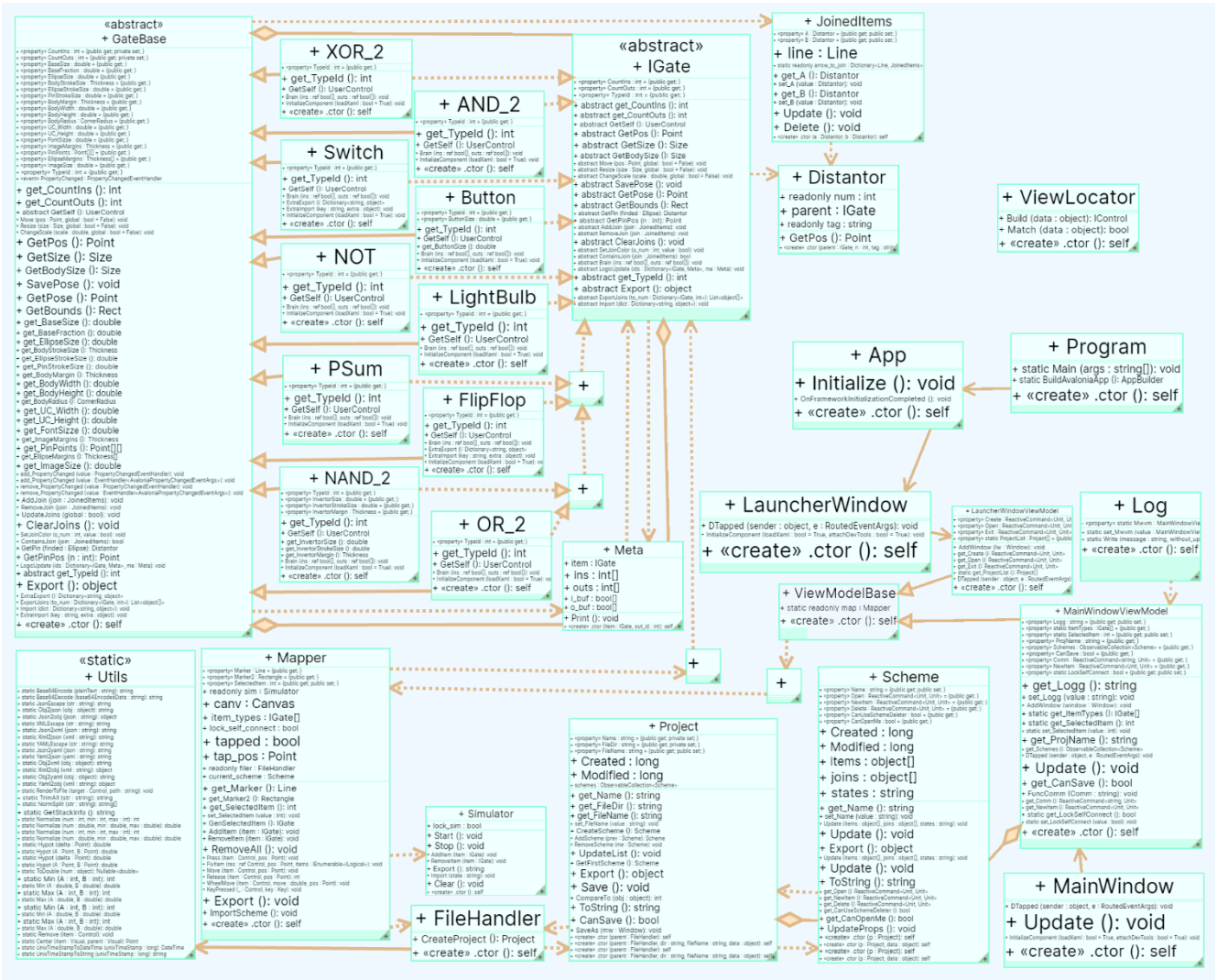
3. Проектирование приложения - создание ER-диаграмм, диаграмм классов. По окончании этапа должны быть построены диаграммы классов с описанием (обязательно), ER-диаграммы (необязательно).

4. Разработка. При разработке используется TDD и упрощённый git flow (одна функциональность - одна ветка, коммиты в логических точках).

В репозитории приложения должен находиться отчёт по первым трём пунктам и проекты с исходным кодом и юнит-тестами.

Мой вариант: формат хранения проекта – YAML, формат хранения списка открывавшихся проектов – YAML, дополнительные логические элементы – Полусумматор

Создание Use-Case диаграммы приложения:



Разработка графического интерфейса:

Было создано основное окно приложения, содержащее список проектов и три кнопки, с помощью которых вы можете создать или открыть проект, а также выйти из приложения.

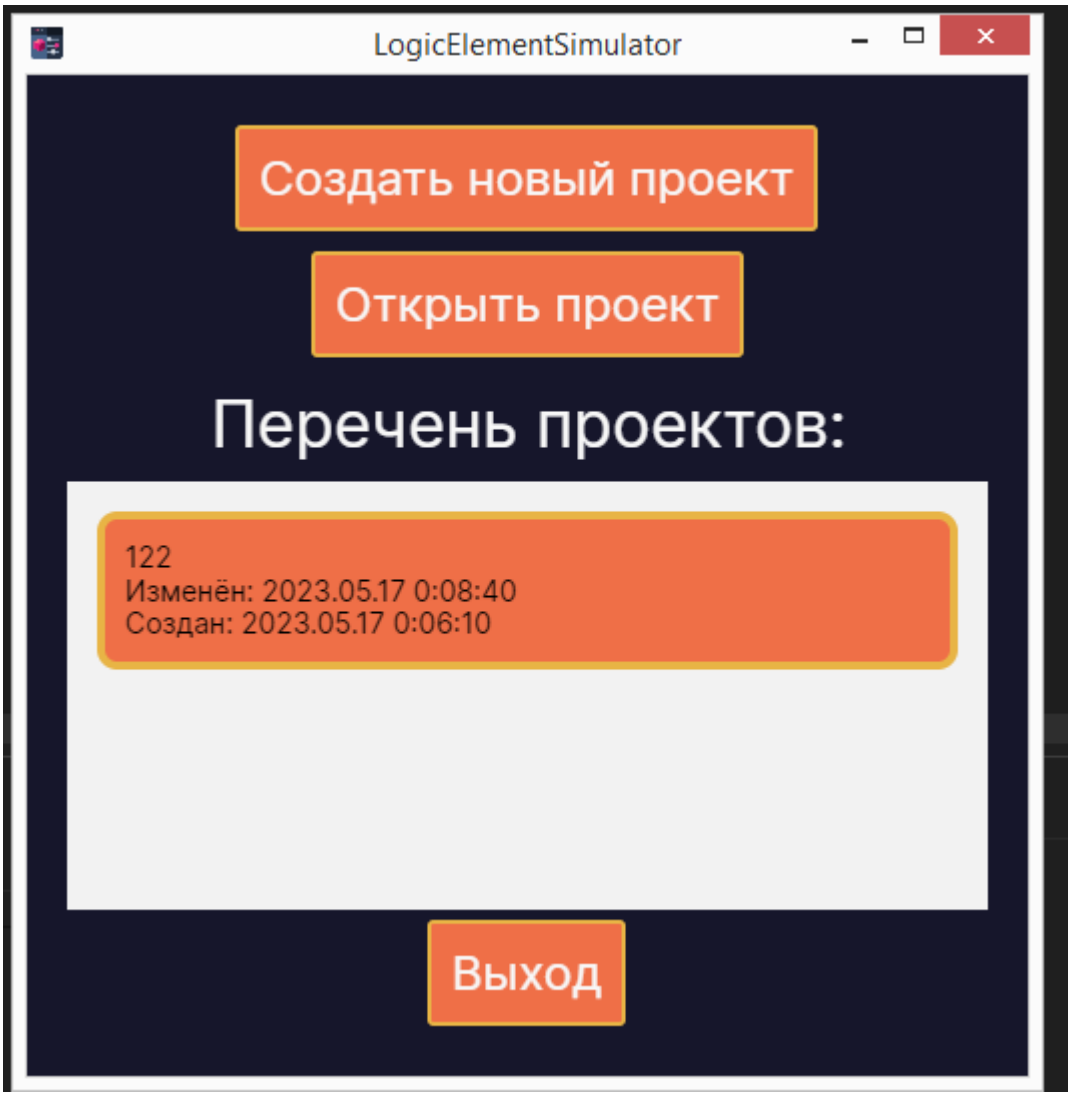


Рис. 1 – стартовое окно

При нажатии кнопки «Создать новый проект» открывается окно на котором размещены логические элементы, которые можно добавлять на схему. Также в верхнем левом углу расположено меню управления, имеющее 2 вкладки: файл и опции.

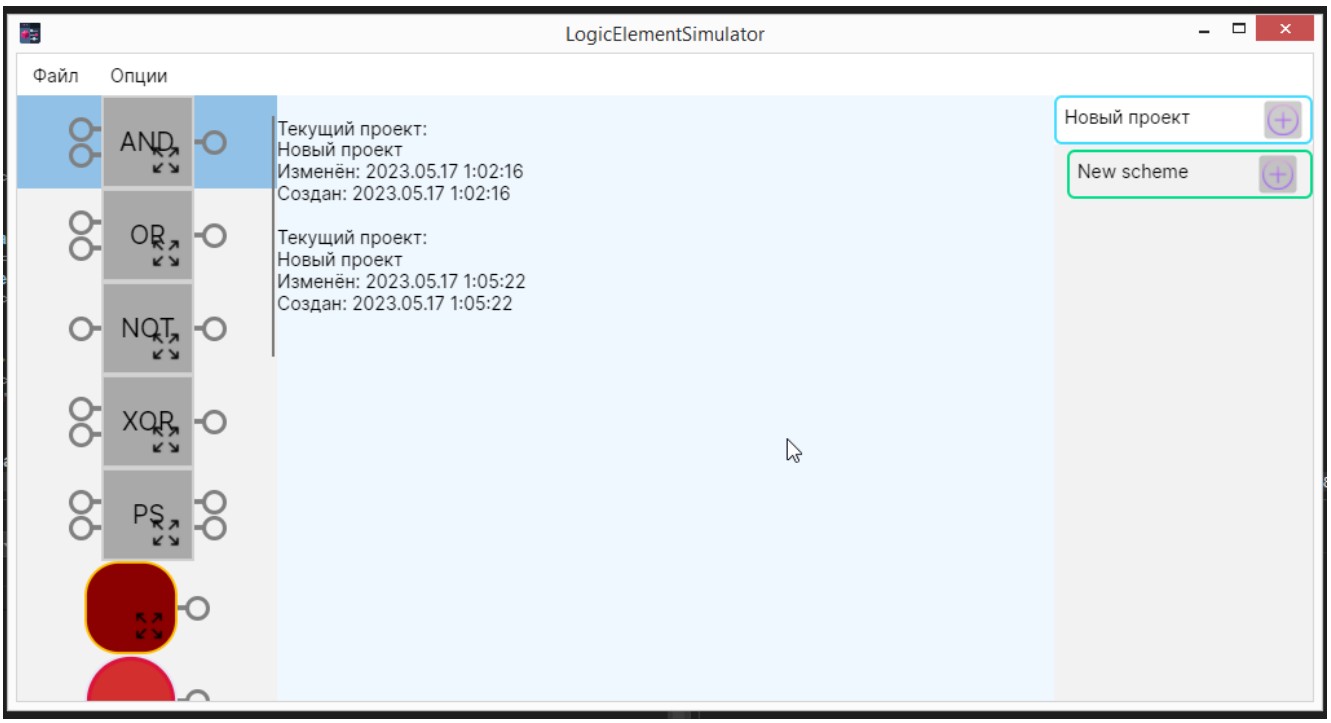


Рис. 2 – Окно создания проекта

Меню управления:

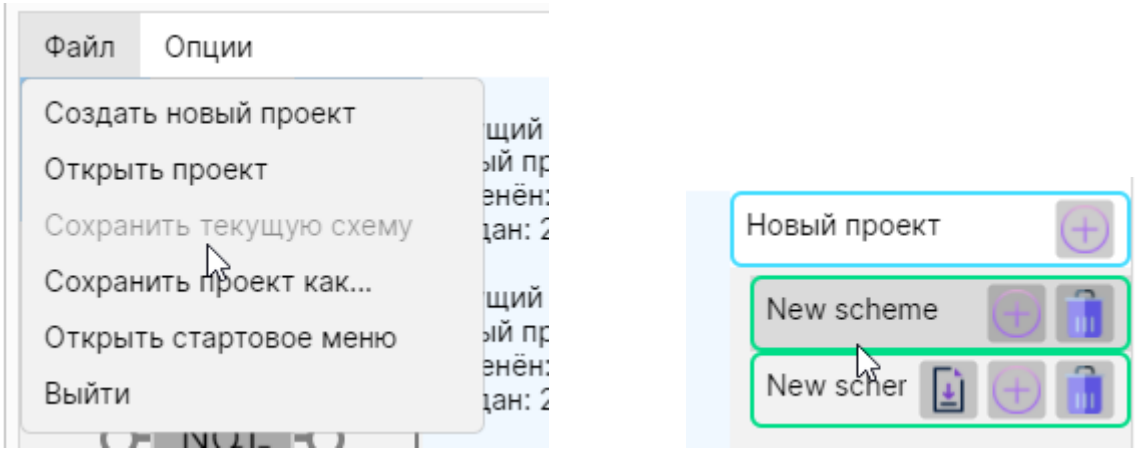


Рис. 3 и 4 – меню вкладок

Вкладка «Файл» включает 6 пунктов: "Создать новый проект", "Открыть проект", "Сохранить текущую схему", "Сохранить проект как", "Выйти в стартовое меню " и "Выйти".

Пункт "Создать новый проект" - создаёт новый проект.

Пункт "Открыть проект" - открывает диалоговое окно открытия файла проекта, при выборе файла проекта, проект подгружается в приложение - его схемы отображаются в списке схем проекта, на холсте появляется отображение первой схемы в проекте.

Пункт "Сохранить текущую схему" – сохраняет схему в файл, в котором вы находитесь.

Пункт "Сохранить проект как" - открывает диалоговое окно сохранения проекта в файл. При выборе файла содержимое проекта сохраняется в него.

Пункт " Выйти в стартовое меню " – выходит в главное меню.

Пункт "Выйти" - закрывает приложение.

Краткое руководство для использования редактора схем:

Для размещения элемента на схеме, нужно выбрать логический элемент на панели и кликнуть левой кнопкой мыши на холст в то место на которое нужно поместить логический элемент. При зажатой левой кнопкой мыши на логическом элементе, расположенном на холсте, его можно перетаскивать.

Для соединения логических элементов необходимо перетащить выход одного элемента на вход другого элемента или наоборот. При этом после соединения появляется линия между входом и выходом соединённых элементов. Для удаления соединения необходимо выбрать линию соединения левой кнопкой мыши и нажать клавишу delete. Для удаления элемента, необходимо выбрать удаляемый элемент левой кнопкой мыши и нажать клавишу delete. Выходные сигналы логических элементов должны рассчитываться в реальном времени.

Проект имеет вид списка с верхним элементом - названием проекта, все остальные элементы названия схем. Схемы можно добавлять и удалять, но в проекте всегда должна быть минимум одна схема. Чтобы отредактировать схему нужно кликнуть на неё в списке два раза левой кнопкой мыши. Название проекта можно отредактировать, кликнув два раза левой кнопкой мыши по нему.

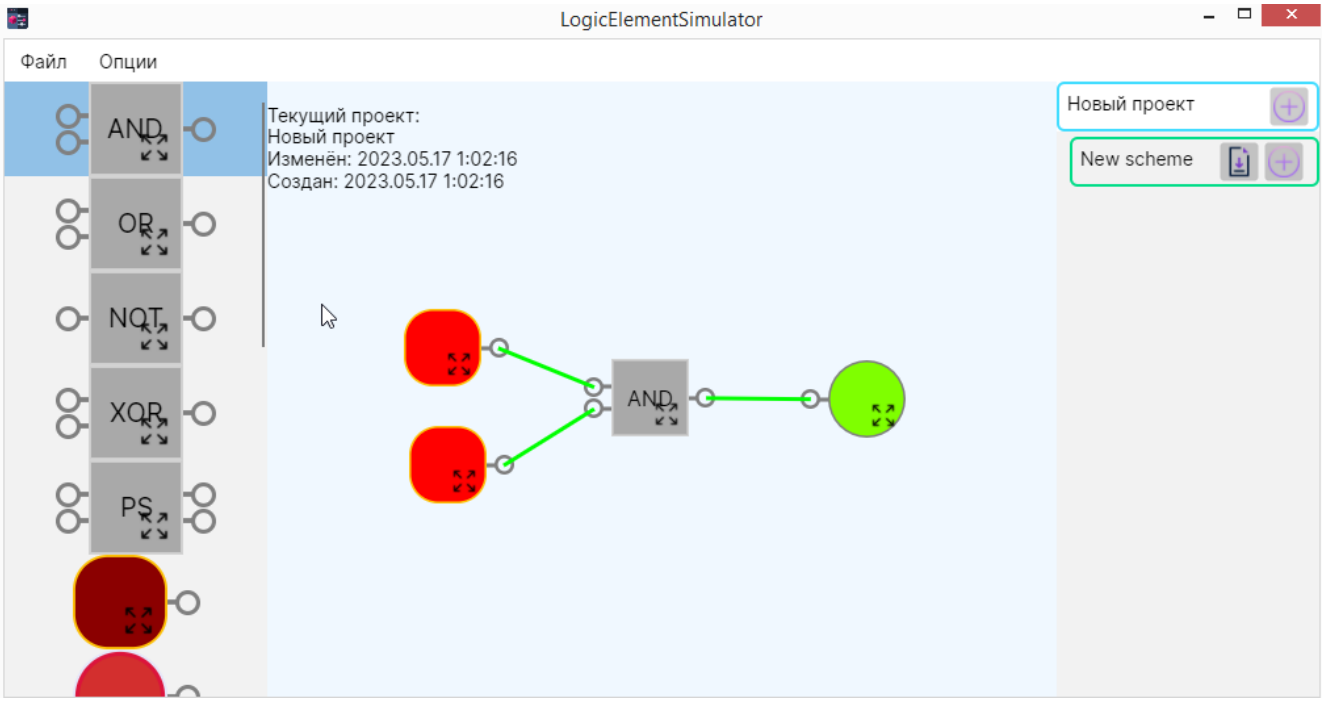


Рис. 5 – Вид схемы в сборке (пример)

Приложение поддерживает следующие логические элементы: И, ИЛИ, НЕ, ИСКЛ-ИЛИ, а также элементы ВХОД и ВЫХОД. ВХОД позволяет по клику на нему левой кнопкой мыши поменять

выходящее из него значение, ВЫХОД отображает значение сигнала, подающееся ему на вход.

Листинг

```
using Avalonia.Controls.Presenters;
using Avalonia.Controls;
using ReactiveUI;
using System.Reactive;
using LogicSimulator.Views;
using LogicSimulator.Models;

namespace LogicSimulator.ViewModels {
    public class LauncherWindowViewModel: ViewModelBase {
        Window? me;
        private static readonly MainWindow mw = new();

        public LauncherWindowViewModel() {
            Create = ReactiveCommand.Create<Unit, Unit>(_ => { FuncCreate(); return new Unit(); });
            Open = ReactiveCommand.Create<Unit, Unit>(_ => { FuncOpen(); return new Unit(); });
            Exit = ReactiveCommand.Create<Unit, Unit>(_ => { FuncExit(); return new Unit(); });
        }
```

```
public void AddWindow(Window lw) => me = lw;

void FuncCreate() {
    var newy = map.filer.CreateProject();
    CurrentProj = newy;
    mw.Show();
    mw.Update();
    me?.Close();
}
void FuncOpen() {
    if (me == null) return;

    var selected = map.filer.SelectProjectFile(me);
    if (selected == null) return;

    CurrentProj = selected;
    mw.Show();
    mw.Update();
    me?.Close();
}
void FuncExit() {
    me?.Close();
    mw.Close();
}

public ReactiveCommand<Unit, Unit> Create { get; }
public ReactiveCommand<Unit, Unit> Open { get; }
public ReactiveCommand<Unit, Unit> Exit { get; }


public static Project[] ProjectList { get => map.filer.GetSortedProjects(); }
```

```
        public void DTapped(object? sender, Avalonia.Interactivity.RoutedEventArgs e) {
            var src = (Control?) e.Source;

            if (src is ContentPresenter cp && cp.Child is Border bord) src = bord;
            if (src is Border bord2 && bord2.Child is TextBlock tb2) src = tb2;

            if (src is not TextBlock tb || tb.Tag is not Project proj) return;

            CurrentProj = proj;

            mw.Show();
            mw.Update();
            me?.Close();
        }

        public static MainWindow GetMW => mw;
    }
}

using Avalonia.Controls;
using Avalonia.Controls.Presenters;
using Avalonia.Input;
using LogicSimulator.Models;
using LogicSimulator.Views;
using LogicSimulator.Views.Shapes;
using ReactiveUI;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Reactive;

namespace LogicSimulator.ViewModels {
```

```
public class Log {
    static readonly List<string> logs = new();
    static readonly string path = "../../../Log.txt";
    static bool first = true;

    static readonly bool use_file = false;

    public static MainWindowViewModel? Mwvm { private get; set; }
    public static void Write(string message, bool without_update = false) {
        if (!without_update) {
            foreach (var mess in message.Split('\n')) logs.Add(mess);
            while (logs.Count > 45) logs.RemoveAt(0);

            if (Mwvm != null) Mwvm.Logg = string.Join('\n', logs);
        }

        if (use_file) {
            if (first) File.WriteAllText(path, message + "\n");
            else File.AppendAllText(path, message + "\n");
            first = false;
        }
    }
}

public class MainWindowViewModel: ViewModelBase {
    private string log = "";
    public string Logg { get => log; set => this.RaiseAndSetIfChanged(ref log, value); }

    public MainWindowViewModel() {
        Log.Mwvm = this;
        Comm = ReactiveCommand.Create<string, Unit>(n => { FuncComm(n); return new Unit(); });
        NewItem = ReactiveCommand.Create<Unit, Unit>(_ => { FuncNewItem(); return new Unit(); });
```

```
    }

    private Window? mw;
    public void AddWindow(Window window) {
        var canv = window.Find<Canvas>("Canvas");

        mw = window;
        map.canv = canv;
        if (canv == null) return;

        canv.Children.Add(map.Marker);
        canv.Children.Add(map.Marker2);

        var panel = (Panel?) canv.Parent;
        if (panel == null) return;

        panel.PointerPressed += (object? sender, PointerPressedEventArgs e) => {
            if (e.Source != null && e.Source is Control @control) map.Press(@control,
e.GetCurrentPoint(canv).Position);
        };
        panel.PointerMoved += (object? sender, PointerEventArgs e) => {
            if (e.Source != null && e.Source is Control @control) map.Move(@control,
e.GetCurrentPoint(canv).Position);
        };
        panel.PointerReleased += (object? sender, PointerReleasedEventArgs e) => {
            if (e.Source != null && e.Source is Control @control) {
                int mode = map.Release(@control, e.GetCurrentPoint(canv).Position);
                bool tap = map.tapped;
                if (tap && mode == 1) {
                    var pos = map.tap_pos;
                    if (canv == null) return;

                    var newy = map.GenSelectedItem();
```

```
                newy.Move(pos);
                map.AddItem(newy);
            }
        }
    };
    panel.PointerWheelChanged += (object? sender, PointerWheelEventArgs e) => {
        if (e.Source != null && e.Source is Control @control) map.WheelMove(@control, e.Delta.Y,
e.GetCurrentPoint(canv).Position);
    };
    mw.KeyDown += (object? sender, KeyEventArgs e) => {
        if (e.Source != null && e.Source is Control @control) map.KeyPressed(@control, e.Key);
    };
}

public static IGate[] ItemTypes { get => map.item_types; }
public static int SelectedItem { get => map.SelectedItem; set => map.SelectedItem = value; }

/*
 * Обработка панели со схемами проекта
 */

Grid? cur_grid;
TextBlock? old_b_child;
object? old_b_child_tag;
string? prev_scheme_name;

public static string ProjName { get => CurrentProj == null ? "???" : CurrentProj.Name; }

public static ObservableCollection<Scheme> Schemes { get => CurrentProj == null ? new() :
CurrentProj.schemes; }
```

```
public void DTapped(object? sender, Avalonia.Interactivity.RoutedEventArgs e) {
    var src = (Control?) e.Source;

    if (src is ContentPresenter cp && cp.Child is Border bord) src = bord;
    if (src is Border bord2 && bord2.Child is Grid g2) src = g2;
    if (src is Grid g3 && g3.Children[0] is TextBlock tb2) src = tb2;

    if (src is not TextBlock tb) return;

    var p = tb.Parent;
    if (p == null) return;

    if (old_b_child != null)
        if (cur_grid != null) cur_grid.Children[0] = old_b_child;

    if (p is not Grid g) return;
    cur_grid = g;

    old_b_child = tb;
    old_b_child_tag = tb.Tag;
    prev_scheme_name = tb.Text;

    var newy = new TextBox { Text = tb.Text };

    cur_grid.Children[0] = newy;

    newy.KeyUp += (object? sender, KeyEventArgs e) => {
        if (e.Key != Key.Return) return;

        if (newy.Text != prev_scheme_name) {
            if ((string?) tb.Tag == "p_name") CurrentProj?.ChangeName(newy.Text);
            else if (old_b_child_tag is Scheme scheme) scheme.ChangeName(newy.Text);
```

```
        }

        cur_grid.Children[0] = tb;
        cur_grid = null; old_b_child = null;
    };
}

public void Update() {
    Log.Write("\nТекущий проект:\n" + CurrentProj);

    map.ImportScheme();

    this.RaisePropertyChanged(new(nameof(ProjName)));
    this.RaisePropertyChanged(new(nameof(Schemes)));
    this.RaisePropertyChanged(new(nameof(CanSave)));
    if (mw != null) mw.Width++;
}

public static bool CanSave { get => CurrentProj != null && CurrentProj.CanSave(); }

/*
 * Кнопочки!
 */

public void FuncComm(string Comm) {
    switch (Comm) {
    case "Create":
        var newy = map.filer.CreateProject();
        CurrentProj = newy;
        Update();
        break;
    case "Open":
```

```
            if (mw == null) break;
            var selected = map.filer.SelectProjectFile(mw);
            if (selected != null) {
                CurrentProj = selected;
                Update();
            }
            break;
        case "Save":
            map.Export();
            File.WriteAllText("../../../for_test.json", Utils.Obj2json((map.current_scheme ?? throw new System.Exception("Что?!")).Export()));
            break;
        case "SaveAs":
            map.Export();
            if (mw != null) CurrentProj?.SaveAs(mw);
            this.RaisePropertyChanged(new(nameof(CanSave)));
            break;
        case "ExitToLauncher":
            new LauncherWindow().Show();
            mw?.Hide();
            break;
        case "Exit":
            mw?.Close();
            break;
        }
    }

    public ReactiveCommand<string, Unit> Comm { get; }

    private static void FuncNewItem() {
        CurrentProj?.AddScheme(null);
    }
```

```
        public ReactiveCommand<Unit, Unit> NewItem { get; }


        public static bool LockSelfConnect { get => map.lock_self_connect; set => map.lock_self_connect = value; }
    }
}
```