Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное образовательное бюджетное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики» (ФГБОУ «СибГУТИ»)

Кафедра ПМ и К

**Курсовая работа**

Дисциплина: Визуальное программирование и человеко-машинное взаимодействие

Тема: «Приложение-симулятор логических схем»

Вариант 244

Выполнил:

Студент группы ИП-117

Мешков А.А.

Проверил:

Ассистент кафедры ПМ и К

Меркулов И. А.

Новосибирск 2023

Оглавление

[Задание 3](#_Toc135501906)

[Общее задание 3](#_Toc135501907)

[Главное окно 3](#_Toc135501908)

[Работа со схемой 3](#_Toc135501909)

[Работа с проектом 3](#_Toc135501910)

[Меню 4](#_Toc135501911)

[Стартовое окно 4](#_Toc135501912)

[Поддерживаемые логические элементы 4](#_Toc135501913)

[Формат хранения проекта 4](#_Toc135501914)

[Формат хранения списка открывавшихся проектов 4](#_Toc135501915)

[Дополнительные логические элементы 4](#_Toc135501916)

[Пример работы программы 5](#_Toc135501917)

[Use-case диаграмма 6](#_Toc135501918)

[Заключение 7](#_Toc135501919)

[Листинг программы 8](#_Toc135501920)

Задание

Общее задание

Реализовать приложение-симулятор логических схем.

Работа состоит из следующих этапов:

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

Главное окно

Главное окно приложения состоит из холста, на котором размещаются логические элементы; дерева проекта, на котором отображается список схем проекта; панели элементов на котором располагаются логические элементы, которые можно добавлять на схему; меню управления из которого можно сохранить текущий проект, загрузить существующий проект, выйти из программы.

Работа со схемой

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.

Работа с проектом

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 на нём.

Меню

Меню должно включать четыре пункта: "Создать", "Открыть", "Сохранить", "выйти".

Пункт "Создать" - создаёт новый проект.

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

Пункт "Выйти" - закрывает приложение

Стартовое окно

При старте приложения появляется окно в котором отображается список недавно открывавшихся проектов, отсортированных по дате открытия. Также есть кнопки "Создать новый проект", "Открыть проект", "Выйти", которые дублируют пункты меню "Создать", "Открыть", "Выйти" основного окна. При выборе проекта из списка стартового окна, он открывается в главном окне.

Поддерживаемые логические элементы

Приложение должно иметь поддержку логических элементов И, ИЛИ, НЕ, ИСКЛ-ИЛИ, а также элементов 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.

Варианты

Формат хранения проекта

В соответствии с номером варианта: JSON

Формат хранения списка открывавшихся проектов

В соответствии с номером варианта: SQLite

Дополнительные логические элементы

В соответствии с номером варианта: дешифратор

Пример работы программы

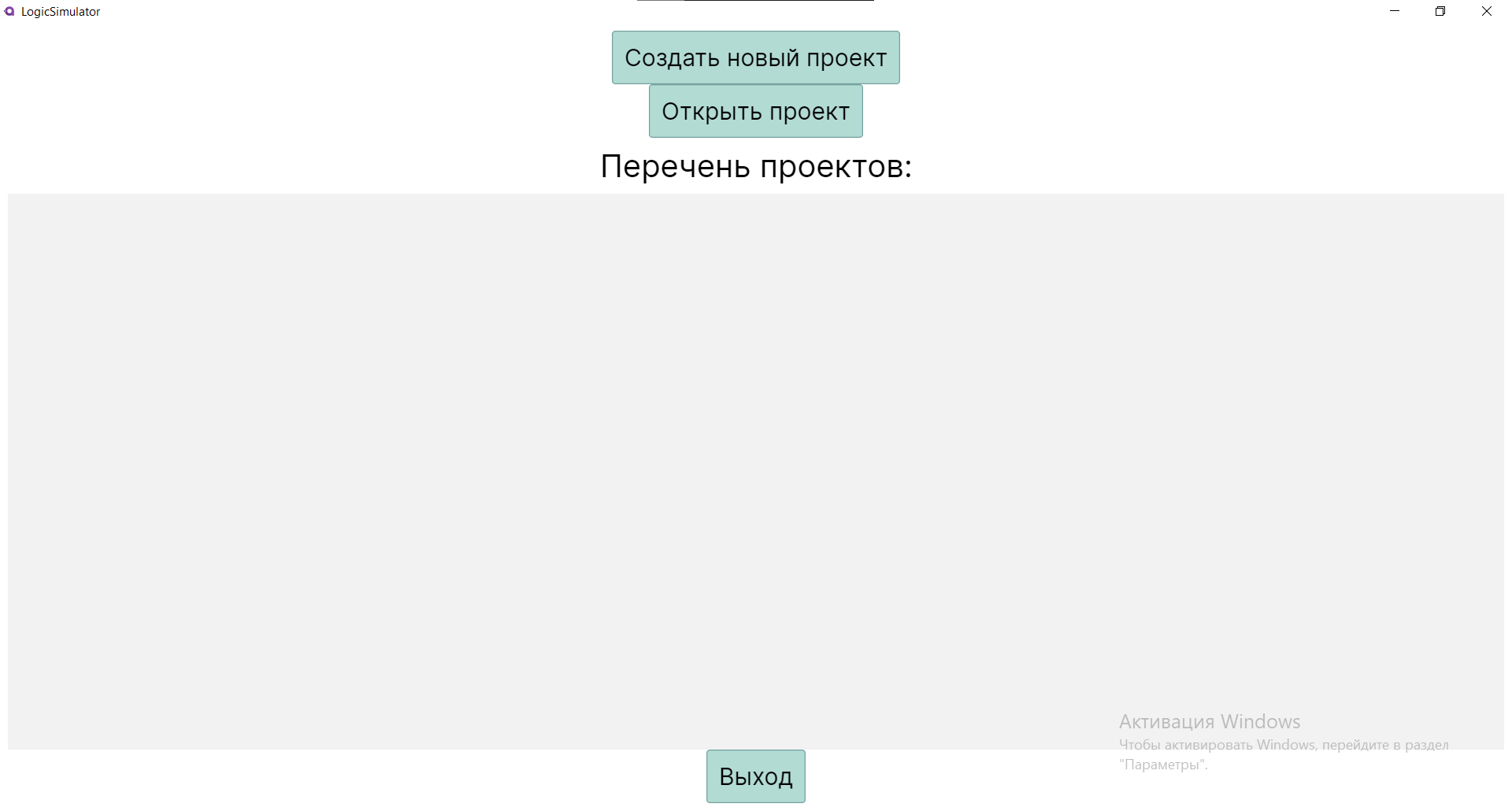


Рисунок 1 Стартовое окно программы

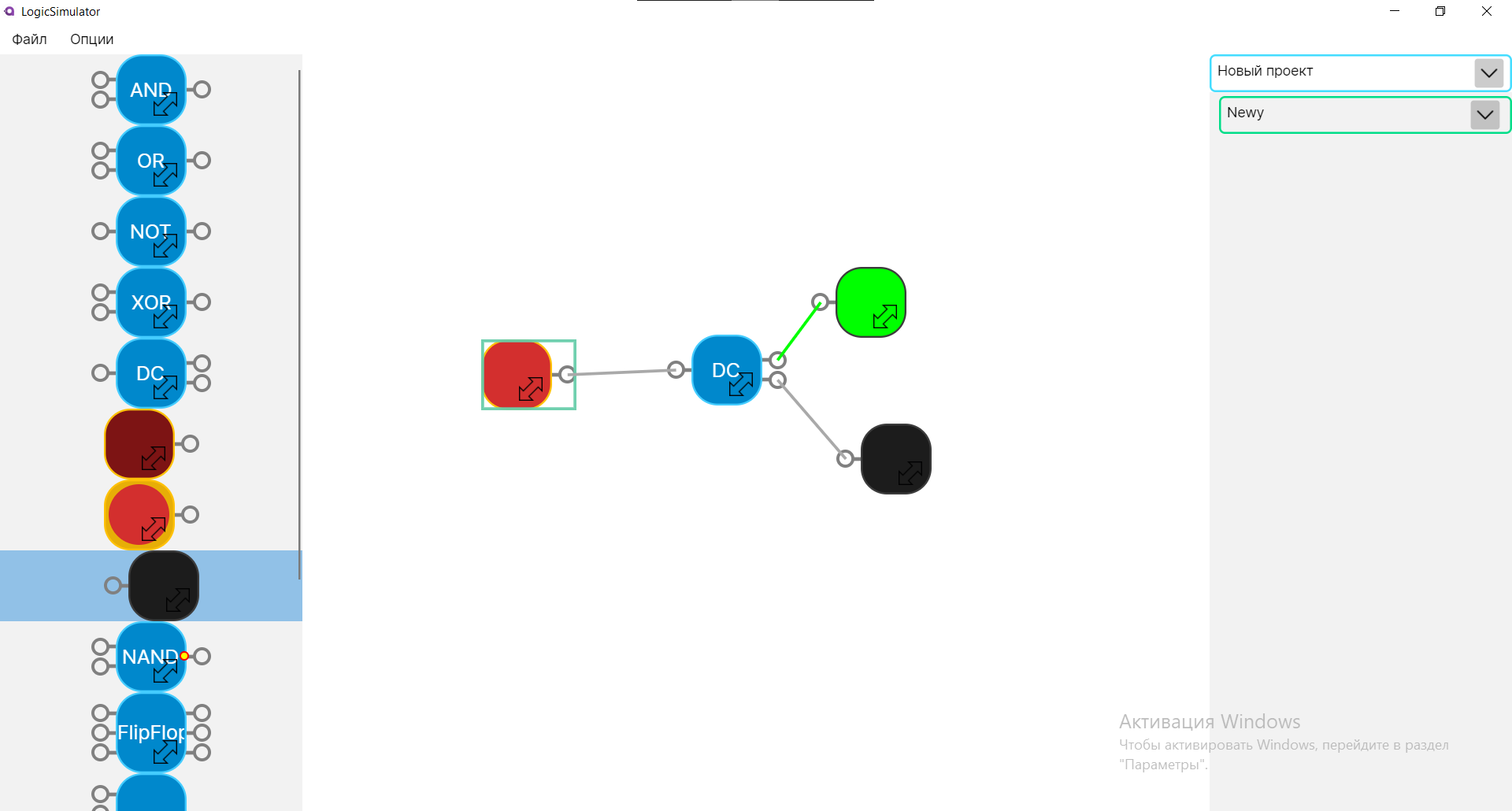


Рисунок 2 Окно редактирования схемы

Use-case диаграмма

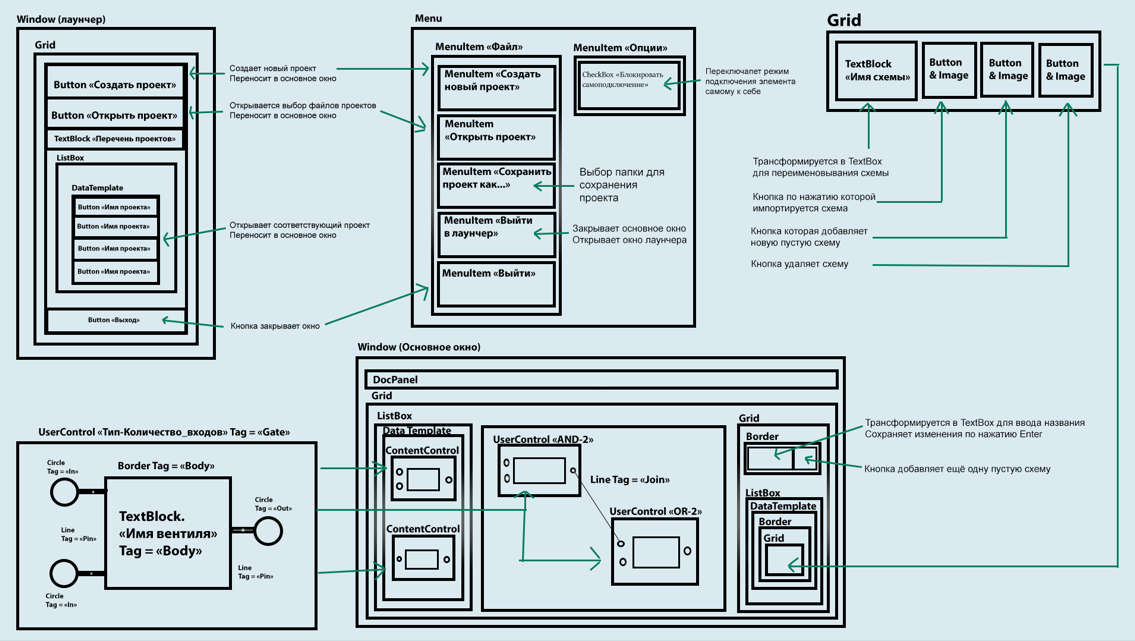
****

Рисунок 3Use-Case диаграмма

Заключение

При работе над проектом использовались навыки, полученные в курсе "Визуальное программирование и человеко-машинное взаимодействие", такие как создание пользовательских интерфейсов, тестирование графических интерфейсов, создание многостраничных приложений, механизмы связывания и шаблоны отображения, графическое преобразование и анимация, обработка событий с устройства ввода, создание собственных элементов управления, работа с разными типами хранения и представления данных.

Были отработаны навыки работы с GUI Framework Avalonia UI на платформе .NET с использованием языка программирования C#.

В результате работы было создано приложение для работы с логическими схемами, которое работает с базовыми и дополнительными логическими вентилями. Проекты со схемами можно сохранять, а также приложение хранит список проектов, с которыми вы работали ранее, поэтому проекты можно открывать позже для внесения изменений.

Листинг программы

<Distantor.cs>

using Avalonia;

using LogicSimulator.Views.Shapes;

namespace LogicSimulator.Models {

public class Distantor {

public readonly int num;

public IGate parent;

public readonly string tag;

public Distantor(IGate parent, int n, string tag) {

this.parent = parent;

num = n;

this.tag = tag;

}

public Point GetPos() => parent.GetPinPos(num);

}

}

<FileHandler.cs >

using Avalonia.Controls;

using LogicSimulator.ViewModels;

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

namespace LogicSimulator.Models

{

public class FileHandler

{

readonly string AppData;

readonly List<Project> projects = new();

readonly List<string> project\_paths = new();

readonly Dictionary<string, Project> proj\_dict = new();

public FileHandler()

{

string app\_data = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);

app\_data = Path.Combine(app\_data, "LogicSimulator");

if (!Directory.Exists(app\_data)) Directory.CreateDirectory(app\_data);

AppData = app\_data;

LoadProjectList();

}

private void AddProject(Project proj)

{

if (proj.FileDir == null || proj.FileName == null) return;

var path = Path.Combine(proj.FileDir, proj.FileName);

if (proj\_dict.ContainsKey(path)) return;

proj\_dict[path] = proj;

projects.Add(proj);

}

private static string GetProjectFileName(string dir)

{

int n = 0;

while (true)

{

string name = "proj\_" + ++n + ".json";

if (!File.Exists(Path.Combine(dir, name))) return name;

}

}

public Project CreateProject()

{

var proj = new Project(this);

// projects.Add(proj); Перенесено в AppendProject

return proj;

}

private Project? LoadProject(string dir, string fileName)

{

try

{

var path = Path.Combine(dir, fileName);

if (!File.Exists(path)) return null;

var obj = Utils.Json2obj(File.ReadAllText(path)) ?? throw new DataException("Не верная структура JSON-файла проекта!");

var proj = new Project(this, dir, fileName, obj);

AddProject(proj);

return proj;

}

catch (Exception e) { Log.Write("Неудачная попытка загрузить проект:\n" + e); }

return null;

}

private Project? LoadProject(string path)

{

var s\_arr = path.Split(Path.DirectorySeparatorChar).ToList();

var name = s\_arr[^1];

s\_arr.RemoveRange(s\_arr.Count - 1, 1);

var dir = Path.Combine(s\_arr.ToArray());

return LoadProject(dir, name);

}

private void LoadProjectList()

{

var file = Path.Combine(AppData, "project\_list.db");

if (!File.Exists(file)) return;

string[] data;

try { data = Utils.SQLite\_proj\_list2obj(file) ?? throw new DataException("Не верная структура SQLite (.db)-файла списка проектов!"); } catch (Exception e) { Log.Write("Неудачная попытка загрузить список проектов:\n" + e); return; }

foreach (var path in data)

{

project\_paths.Add(path);

LoadProject(path);

}

}

internal static void SaveProject(Project proj)

{

var dir = proj.FileDir;

if (dir == null) return;

var data = Utils.Obj2json(proj.Export());

var name = proj.FileName;

name ??= GetProjectFileName(dir);

proj.FileName = name;

var path = Path.Combine(dir, name);

File.WriteAllText(path, data);

}

private void SaveProjectList()

{

var file = Path.Combine(AppData, "project\_list.db");

if (Path.Exists(file)) File.WriteAllBytes(file, Array.Empty<byte>());

Utils.Obj2sqlite\_proj\_list(project\_paths.ToArray(), file);

}

internal Project[] GetSortedProjects()

{

projects.Sort();

return projects.ToArray();

}

internal void AppendProject(Project proj)

{

if (proj.FileDir == null || proj.FileName == null) return;

var path = Path.Combine(proj.FileDir, proj.FileName);

if (project\_paths.Contains(path)) return;

project\_paths.Add(path);

AddProject(proj);

SaveProjectList();

}

internal static string? RequestProjectPath(Window parent)

{

var dlg = new OpenFolderDialog

{

Title = "Выберите папку, куда надо сохранить новый проект"

};

var task = dlg.ShowAsync(parent);

return task.GetAwaiter().GetResult();

}

internal Project? SelectProjectFile(Window parent)

{

var dlg = new OpenFileDialog

{

Title = "Выберите файл с проектом (proj\_\*.json), который нужно открыть"

};

dlg.Filters?.Add(new FileDialogFilter() { Name = "JSON Files", Extensions = { "json" } });

dlg.Filters?.Add(new FileDialogFilter() { Name = "All Files", Extensions = { "\*" } });

dlg.AllowMultiple = false;

var task = dlg.ShowAsync(parent);

var res = task.GetAwaiter().GetResult();

if (res == null) return null;

var path = res[0];

if (!proj\_dict.TryGetValue(path, out var proj))

{

proj = LoadProject(path);

if (proj != null) AppendProject(proj);

}

return proj;

}

}

}

<JoinedItems.cs >

using Avalonia.Controls.Shapes;

using Avalonia.Media;

using System.Collections.Generic;

namespace LogicSimulator.Models {

public class JoinedItems {

public static readonly Dictionary<Line, JoinedItems> arrow\_to\_join = new();

public JoinedItems(Distantor a, Distantor b) {

A = a; B = b; Update();

a.parent.AddJoin(this);

if (a.parent != b.parent) b.parent.AddJoin(this);

arrow\_to\_join[line] = this;

}

public Distantor A { get; set; }

public Distantor B { get; set; }

public Line line = new() { Tag = "Join", ZIndex = 2, Stroke = Brushes.DarkGray, StrokeThickness = 3 };

public void Update() {

line.StartPoint = A.GetPos();

line.EndPoint = B.GetPos();

}

public void Delete() {

arrow\_to\_join.Remove(line);

line.Remove();

A.parent.RemoveJoin(this);

B.parent.RemoveJoin(this);

}

}

}

< Mapper.cs>

using Avalonia.Controls;

using Avalonia;

using LogicSimulator.ViewModels;

using LogicSimulator.Views.Shapes;

using System;

using System.Collections.Generic;

using DynamicData;

using Avalonia.Controls.Shapes;

using Avalonia.Media;

using Avalonia.LogicalTree;

using System.Linq;

using Button = LogicSimulator.Views.Shapes.Button;

using Avalonia.Input;

namespace LogicSimulator.Models {

public class Mapper {

readonly Line marker = new() { Tag = "Marker", ZIndex = 2, IsVisible = false, Stroke = Brushes.YellowGreen, StrokeThickness = 3 };

readonly Rectangle marker2 = new() { Tag = "Marker", Classes = new("anim"), ZIndex = 2, IsVisible = false, Stroke = Brushes.MediumAquamarine, StrokeThickness = 3 };

public Line Marker { get => marker; }

public Rectangle Marker2 { get => marker2; }

public readonly Simulator sim = new(); // забавно, но без public рефлексия вообще не видит этот параметр, от чего ER-diagram\_exTRACTOR теряет одну стрелочку зависимости...

public Canvas canv = new();

/\*

\* Маркер

\*/

private IGate? marked\_item;

private JoinedItems? marked\_line;

private void UpdateMarker() {

marker2.IsVisible = marked\_item != null || marked\_line != null;

if (marked\_item != null) {

var bound = marked\_item.GetBounds();

marker2.Margin = new(bound.X, bound.Y);

marker2.Width = bound.Width;

marker2.Height = bound.Height;

marked\_line = null;

}

if (marked\_line != null) {

var line = marked\_line.line;

var A = line.StartPoint;

var B = line.EndPoint;

marker2.Margin = new(Math.Min(A.X, B.X), Math.Min(A.Y, B.Y));

marker2.Width = Math.Abs(A.X - B.X);

marker2.Height = Math.Abs(A.Y - B.Y);

}

}

/\*

\* Выборка элементов

\*/

private int selected\_item = 0;

public int SelectedItem { get => selected\_item; set => selected\_item = value; }

private static IGate CreateItem(int n) {

return n switch {

0 => new AND\_2(),

1 => new OR\_2(),

2 => new NOT(),

3 => new XOR\_2(),

4 => new DC(),

5 => new Switch(),

6 => new Button(),

7 => new LightBulb(),

8 => new NAND\_2(),

9 => new FlipFlop(),

10 => new OR\_8(),

11 => new AND\_8(),

\_ => new AND\_2(),

};

}

public IGate[] item\_types = Enumerable.Range(0, 12).Select(CreateItem).ToArray();

public IGate GenSelectedItem() => CreateItem(selected\_item);

/\*

\* Хранилище

\*/

readonly List<IGate> items = new();

// readonly MatrixTransform general\_transform = new() { Matrix = new(1.0, 0.0, 0.0, 1.0, 0, 0) };

// Canvas? itemer;

private void AddToMap(IControl item) {

/\*if (itemer == null) { Снова мимо :///

itemer = new Canvas();

var layout = new LayoutTransformControl() {

LayoutTransform = general\_transform,

Child = itemer,

};

canv.Children.Add(layout);

}

itemer.Children.Add(item);\*/

canv.Children.Add(item);

}

public void AddItem(IGate item) {

items.Add(item);

sim.AddItem(item);

AddToMap(item.GetSelf());

}

public void RemoveItem(IGate item) {

if (marked\_item != null) {

marked\_item = null;

UpdateMarker();

}

if (marked\_line != null && item.ContainsJoin(marked\_line)) {

marked\_line = null;

UpdateMarker();

}

items.Remove(item);

sim.RemoveItem(item);

item.ClearJoins();

((Control) item).Remove();

}

public void RemoveAll() {

foreach (var item in items.ToArray()) RemoveItem(item);

sim.Clear();

}

private void SaveAllPoses() {

foreach (var item in items) item.SavePose();

}

/\*

\* Определение режима перемещения

\*/

int mode = 0;

/\*

\* Режимы:

\* 0 - ничего не делает

\* 1 - двигаем камеру

\* 2 - двигаем элемент

\* 3 - тянем элемент

\* 4 - вышвыриваем элемент

\* 5 - тянем линию от входа (In)

\* 6 - тянем линию от выхода (Out)

\* 7 - тянем линию от узла (IO)

\* 8 - тянем уже существующее соединение - переподключаем

\*/

private static int CalcMode(string? tag) {

if (tag == null) return 0;

return tag switch {

"Scene" => 1,

"Body" => 2,

"Resizer" => 3,

"Deleter" => 4,

"In" => 5,

"Out" => 6,

"IO" => 7,

"Join" => 8,

"Pin" or \_ => 0,

};

}

private void UpdateMode(Control item) => mode = CalcMode((string?) item.Tag);

private static bool IsMode(Control item, string[] mods) {

var name = (string?) item.Tag;

if (name == null) return false;

return mods.IndexOf(name) != -1;

}

private static UserControl? GetUC(Control item) {

while (item.Parent != null) {

if (item is UserControl @UC) return @UC;

item = (Control) item.Parent;

}

return null;

}

private static IGate? GetGate(Control item) {

var UC = GetUC(item);

if (UC is IGate @gate) return @gate;

return null;

}

/\*

\* Обработка мыши

\*/

Point moved\_pos;

IGate? moved\_item;

Point item\_old\_pos;

Size item\_old\_size;

Ellipse? marker\_circle;

Distantor? start\_dist;

int marker\_mode;

Line? old\_join;

bool join\_start;

bool delete\_join = false;

public bool lock\_self\_connect = true;

public void Press(Control item, Point pos) {

// Log.Write("PointerPressed: " + item.GetType().Name + " pos: " + pos);

UpdateMode(item);

// Log.Write("new\_mode: " + mode);

moved\_pos = pos;

moved\_item = GetGate(item);

tapped = true;

if (moved\_item != null) item\_old\_pos = moved\_item.GetPos();

switch (mode) {

case 1:

SaveAllPoses();

break;

case 3:

if (moved\_item == null) break;

item\_old\_size = moved\_item.GetBodySize();

break;

case 5 or 6 or 7:

if (marker\_circle == null) break;

var gate = GetGate(marker\_circle) ?? throw new Exception("ERROR"); // Такого не бывает

start\_dist = gate.GetPin(marker\_circle);

var circle\_pos = start\_dist.GetPos();

marker.StartPoint = marker.EndPoint = circle\_pos;

marker.IsVisible = true;

marker\_mode = mode;

break;

case 8:

if (item is not Line @join) break;

JoinedItems.arrow\_to\_join.TryGetValue(@join, out var @join2);

if (@join2 == null) break;

if (marked\_line == @join2) {

marked\_line = null;

UpdateMarker();

}

var dist\_a = @join.StartPoint.Hypot(pos);

var dist\_b = @join.EndPoint.Hypot(pos);

join\_start = dist\_a > dist\_b;

old\_join = @join;

marker.StartPoint = join\_start ? @join.StartPoint : pos;

marker.EndPoint = join\_start ? pos : @join.EndPoint;

marker\_mode = CalcMode(join\_start ? @join2.A.tag : @join2.B.tag);

marker.IsVisible = true;

@join.IsVisible = false;

break;

}

Move(item, pos);

}

public void FixItem(ref Control res, Point pos, IEnumerable<ILogical> items) {

foreach (var logic in items) {

// if (item.IsPointerOver) { } Гениальная вещь! ;'-} Хотя не, всё равно блокируется после Press и до Release, чего я впринципе хочу избежать ;'-}

var item = (Control) logic;

var tb = item.TransformedBounds;

// if (tb != null && new Rect(tb.Value.Clip.TopLeft, new Size()).Sum(item.Bounds).Contains(pos) && (string?) item.Tag != "Join") res = item; // Гениально! ;'-} НАКОНЕЦ-ТО ЗАРАБОТАЛО! (Так было в 8 лабе)

if (tb != null && tb.Value.Bounds.TransformToAABB(tb.Value.Transform).Contains(pos) && (string?) item.Tag != "Join") res = item; // Гениально! Апгрейд прошёл успешно :D

FixItem(ref res, pos, item.GetLogicalChildren());

}

}

public void Move(Control item, Point pos, bool use\_fix = true) {

// Log.Write("PointerMoved: " + item.GetType().Name + " pos: " + pos);

if (use\_fix && (mode == 5 || mode == 6 || mode == 7 || mode == 8)) {

var tb = canv.TransformedBounds;

if (tb != null) {

item = new Canvas() { Tag = "Scene" };

var bounds = tb.Value.Bounds.TransformToAABB(tb.Value.Transform);

FixItem(ref item, pos + bounds.TopLeft, canv.Children);

// Log.Write("tag: " + item.Tag);

}

}

string[] mods = new[] { "In", "Out", "IO" };

var tag = (string?) item.Tag;

if (IsMode(item, mods) && item is Ellipse @ellipse

&& !(marker\_mode == 5 && tag == "In" || marker\_mode == 6 && tag == "Out" ||

lock\_self\_connect && moved\_item == GetGate(item))) { // То самое место, что не даёт подключить вход ко входу, либо выход к выходу

if (marker\_circle != null && marker\_circle != @ellipse) { // На случай моментального перехода курсором с одного кружка на другой

marker\_circle.Fill = new SolidColorBrush(Color.Parse("#0000"));

marker\_circle.Stroke = Brushes.Gray;

}

marker\_circle = @ellipse;

@ellipse.Fill = Brushes.Lime;

@ellipse.Stroke = Brushes.Green;

} else if (marker\_circle != null) {

marker\_circle.Fill = new SolidColorBrush(Color.Parse("#0000"));

marker\_circle.Stroke = Brushes.Gray;

marker\_circle = null;

}

if (mode == 8) delete\_join = (string?) item.Tag == "Deleter";

/\* if (mode == 0 && (string?) item.Tag == "Join") { DEBUG

JoinedItems.arrow\_to\_join.TryGetValue((Line) item, out var @join);

if (@join != null) Log.Write("J a->b: id" + items.IndexOf(@join.A.parent) + " n:" + @join.A.num + " id" + items.IndexOf(@join.B.parent) + " n:" + @join.B.num);

}\*/

var delta = pos - moved\_pos;

if (delta.X == 0 && delta.Y == 0) return;

if (Math.Pow(delta.X, 2) + Math.Pow(delta.Y, 2) > 9) tapped = false;

switch (mode) {

case 1:

foreach (var item\_ in items) {

var pose = item\_.GetPose();

item\_.Move(pose + delta, true);

}

UpdateMarker();

break;

case 2:

if (moved\_item == null) break;

var new\_pos = item\_old\_pos + delta;

moved\_item.Move(new\_pos);

UpdateMarker();

break;

case 3:

if (moved\_item == null) break;

var new\_size = item\_old\_size + new Size(delta.X, delta.Y);

moved\_item.Resize(new\_size);

UpdateMarker();

break;

case 5 or 6 or 7:

var end\_pos = marker\_circle == null ? pos : marker\_circle.Center(canv);

marker.EndPoint = end\_pos;

break;

case 8:

if (old\_join == null) break;

var p = marker\_circle == null ? pos : marker\_circle.Center(canv);

if (join\_start) marker.EndPoint = p;

else marker.StartPoint = p;

break;

}

}

public bool tapped = false; // Обрабатывается после Release

public Point tap\_pos; // Обрабатывается после Release

public int Release(Control item, Point pos, bool use\_fix = true) {

Move(item, pos, use\_fix);

// Log.Write("PointerReleased: " + item.GetType().Name + " pos: " + pos);

switch (mode) {

case 5 or 6 or 7:

if (start\_dist == null) break;

if (marker\_circle != null) {

var gate = GetGate(marker\_circle) ?? throw new Exception("ERROR"); // Такого не бывает

var end\_dist = gate.GetPin(marker\_circle);

// Log.Write("Стартовый элемент: " + start\_dist.parent + " (" + start\_dist.GetPos() + ")");

// Log.Write("Конечный элемент: " + end\_dist.parent + " (" + end\_dist.GetPos() + ")");

var newy = new JoinedItems(start\_dist, end\_dist);

AddToMap(newy.line);

}

marker.IsVisible = false;

marker\_mode = 0;

break;

case 8:

if (old\_join == null) break;

JoinedItems.arrow\_to\_join.TryGetValue(old\_join, out var @join);

if (marker\_circle != null && @join != null) {

var gate = GetGate(marker\_circle) ?? throw new Exception("ERROR"); // Такого не бывает

var p = gate.GetPin(marker\_circle);

@join.Delete();

var newy = join\_start ? new JoinedItems(@join.A, p) : new JoinedItems(p, @join.B);

AddToMap(newy.line);

} else old\_join.IsVisible = true;

marker.IsVisible = false;

marker\_mode = 0;

old\_join = null;

if (delete\_join) @join?.Delete();

delete\_join = false;

break;

}

if (tapped) Tapped(item, pos);

int res\_mode = mode;

mode = 0;

moved\_item = null;

return res\_mode;

}

private void Tapped(Control item, Point pos) {

// Log.Write("Tapped: " + item.GetType().Name + " pos: " + pos);

tap\_pos = pos;

switch (mode) {

/\* case 4:

if (moved\_item != null) RemoveItem(moved\_item);

break; \*/

case 2 or 8:

if (item is Line @line) {

if (!JoinedItems.arrow\_to\_join.TryGetValue(@line, out var @join)) break;

marked\_item = null;

marked\_line = @join;

UpdateMarker();

break;

}

if (moved\_item == null) break;

marked\_item = moved\_item;

UpdateMarker();

break;

}

}

public void WheelMove(Control item, double move, Point pos) {

// Log.Write("WheelMoved: " + item.GetType().Name + " delta: " + (move > 0 ? 1 : -1));

int mode = CalcMode((string?) item.Tag);

double scale = move > 0 ? 1.1 : 1 / 1.1;

double inv\_scale = 1 / scale;

switch (mode) {

case 1:

foreach (var gate in items) {

gate.ChangeScale(scale, true);

var item\_pos = gate.GetPos();

var delta = item\_pos - pos;

delta \*= scale;

var new\_pos = delta + pos;

gate.Move(new\_pos, false);

}

UpdateMarker();

break;

case 2:

var gate2 = GetGate(item);

if (gate2 == null) return;

gate2.ChangeScale(inv\_scale);

UpdateMarker();

break;

}

}

public void KeyPressed(Control \_, Key key) {

// Log.Write("KeyPressed: " + item.GetType().Name + " key: " + key);

switch (key) {

case Key.Up:

case Key.Left:

case Key.Right:

case Key.Down:

int dx = key == Key.Left ? -1 : key == Key.Right ? 1 : 0;

int dy = key == Key.Up ? -1 : key == Key.Down ? 1 : 0;

marked\_item?.Move(marked\_item.GetPos() + new Point(dx \* 10, dy \* 10));

UpdateMarker();

break;

case Key.Delete:

if (marked\_item != null) RemoveItem(marked\_item);

if (marked\_line != null) {

marked\_line.Delete();

marked\_line = null;

UpdateMarker();

}

break;

}

}

/\*

\* Экспорт и импорт

\*/

public readonly FileHandler filer = new();

public Scheme? current\_scheme;

public void Export() {

if (current\_scheme == null) return;

var arr = items.Select(x => x.Export()).ToArray();

Dictionary<IGate, int> item\_to\_num = new();

int n = 0;

foreach (var item in items) item\_to\_num.Add(item, n++);

List<object[]> joins = new();

foreach (var item in items) joins.Add(item.ExportJoins(item\_to\_num));

sim.Clean();

string states = sim.Export();

try { current\_scheme.Update(arr, joins.ToArray(), states); }

catch (Exception e) { Log.Write("Save error:\n" + e); }

/\* Log.Write("Items: " + Utils.Obj2json(arr));

Log.Write("Joins: " + Utils.Obj2json(joins));

Log.Write("States: " + Utils.Obj2json(states)); \*/

}

public void ImportScheme(bool start = true) {

if (current\_scheme == null) return;

sim.Stop();

sim.lock\_sim = true;

RemoveAll();

List<IGate> list = new();

foreach (var item in current\_scheme.items) {

if (item is not Dictionary<string, object> @dict) { Log.Write("Не верный тип элемента: " + item); continue; }

if (!@dict.TryGetValue("id", out var @value)) { Log.Write("id элемента не обнаружен"); continue; }

if (@value is not int @id) { Log.Write("Неверный тип id: " + @value); continue; }

var newy = CreateItem(@id);

newy.Import(@dict);

AddItem(newy);

list.Add(newy);

}

var items\_arr = list.ToArray();

List<JoinedItems> joinz = new();

foreach (var obj in current\_scheme.joins) {

object[] join;

if (obj is List<object> @j) join = @j.ToArray();

else if (obj is object[] @j2) join = @j2;

else { Log.Write("Одно из соединений не того типа: " + obj + " " + Utils.Obj2json(obj)); continue; }

if (join.Length != 6 ||

join[0] is not int @num\_a || join[1] is not int @pin\_a || join[2] is not string @tag\_a ||

join[3] is not int @num\_b || join[4] is not int @pin\_b || join[5] is not string @tag\_b) { Log.Write("Содержимое списка соединения ошибочно"); continue; }

var newy = new JoinedItems(new(items\_arr[@num\_a], @pin\_a, tag\_a), new(items\_arr[@num\_b], @pin\_b, tag\_b));

AddToMap(newy.line);

joinz.Add(newy);

}

foreach (var join in joinz) join.Update();

sim.Import(current\_scheme.states);

sim.lock\_sim = false;

if (start) sim.Start(); // Во время тестирования лучше и близко не прикасаться к этой функции XD

}

}

}

< Project.cs>

using Avalonia.Controls;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

namespace LogicSimulator.Models {

public class Project: IComparable {

public string Name { get; private set; }

public long Created;

public long Modified;

public ObservableCollection<Scheme> schemes = new();

public string? FileDir { get; private set; }

public string? FileName { get; set; }

private readonly FileHandler parent;

public Project(FileHandler parent) { // Новый проект

this.parent = parent;

Name = "Новый проект";

Created = Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

FileDir = null;

FileName = null; // FileHandler.GetProjectFileName();

CreateScheme();

}

public Project(FileHandler parent, string dir, string fileName, object data) { // Импорт

this.parent = parent;

FileDir = dir;

FileName = fileName;

if (data is not Dictionary<string, object> dict) throw new Exception("Ожидался словарь в корне проекта");

if (!dict.TryGetValue("name", out var value)) throw new Exception("В проекте нет имени");

if (value is not string name) throw new Exception("Тип имени проекта - не строка");

Name = name;

if (!dict.TryGetValue("created", out var value2)) throw new Exception("В проекте нет времени создания");

if (value2 is not int create\_t) throw new Exception("Время создания проекта - не строка");

Created = create\_t;

if (!dict.TryGetValue("modified", out var value3)) throw new Exception("В проекте нет времени изменения");

if (value3 is not int mod\_t) throw new Exception("Время изменения проекта - не строка");

Modified = mod\_t;

if (!dict.TryGetValue("schemes", out var value4)) throw new Exception("В проекте нет списка схем");

if (value4 is not List<object> arr) throw new Exception("Списко схем проекта - не массив строк");

foreach (var s\_data in arr) {

if (s\_data == null) throw new Exception("Одно из файловых имёт списка схем проекта - null");

var scheme = new Scheme(this, s\_data);

schemes.Add(scheme);

}

}

public Scheme CreateScheme() {

var scheme = new Scheme(this);

schemes.Add(scheme);

Save();

return scheme;

}

public Scheme AddScheme(Scheme? prev) {

var scheme = new Scheme(this);

int pos = prev == null ? 0 : schemes.IndexOf(prev) + 1;

schemes.Insert(pos, scheme);

Save();

return scheme;

}

public void RemoveScheme(Scheme me) {

schemes.Remove(me);

Save();

}

public void UpdateList() {

foreach (var scheme in schemes) scheme.UpdateProps();

}

public Scheme GetFirstScheme() => schemes[0];

public object Export() {

return new Dictionary<string, object> {

["name"] = Name,

["created"] = Created,

["modified"] = Modified,

["schemes"] = schemes.Select(x => x.Export()).ToArray(),

};

}

public void Save() => FileHandler.SaveProject(this);

public int CompareTo(object? obj) {

if (obj is not Project proj) throw new ArgumentNullException(nameof(obj));

return (int)(proj.Modified - Modified);

}

public override string ToString() {

return Name + "\nИзменён: " + Modified.UnixTimeStampToString() + "\nСоздан: " + Created.UnixTimeStampToString();

}

internal void ChangeName(string name) {

Name = name;

Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

Save();

}

public bool CanSave() => FileDir != null;

public void SaveAs(Window mw) {

FileDir = FileHandler.RequestProjectPath(mw);

Save();

parent.AppendProject(this);

}

/\*

\* Для тестирования

\*/

public void SetDir(string path) => FileDir = path;

}

}

< Scheme.cs>

using LogicSimulator.ViewModels;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Reactive;

namespace LogicSimulator.Models {

public class Scheme : ReactiveObject {

public string Name { get; set; }

public long Created;

public long Modified;

public object[] items;

public object[] joins;

public string states;

private readonly Project parent;

public Scheme(Project p) { // Новая схема

Created = Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

Name = "Newy";

items = joins = Array.Empty<object>();

states = "0";

parent = p;

Open = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncOpen(); return new Unit(); });

NewItem = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncNewItem(); return new Unit(); });

Delete = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncDelete(); return new Unit(); });

}

public Scheme(Project p, object data) { // Импорт

parent = p;

if (data is not Dictionary<string, object> dict) throw new Exception("Ожидался словарь в корне схемы");

if (!dict.TryGetValue("name", out var value)) throw new Exception("В схеме нет имени");

if (value is not string name) throw new Exception("Тип имени схемы - не строка");

Name = name;

if (!dict.TryGetValue("created", out var value2)) throw new Exception("В схеме нет времени создания");

if (value2 is not int create\_t) throw new Exception("Время создания схемы - не строка");

Created = create\_t;

if (!dict.TryGetValue("modified", out var value3)) throw new Exception("В схеме нет времени изменения");

if (value3 is not int mod\_t) throw new Exception("Время изменения схемы - не строка");

Modified = mod\_t;

if (!dict.TryGetValue("items", out var value4)) throw new Exception("В схеме нет списка элементов");

if (value4 is not List<object> arr) throw new Exception("Список элементов схемы - не массив объектов");

items = arr.ToArray();

if (!dict.TryGetValue("joins", out var value5)) throw new Exception("В схеме нет списка соединений");

if (value5 is not List<object> arr2) throw new Exception("Список соединений схемы - не массив объектов");

joins = arr2.ToArray();

if (!dict.TryGetValue("states", out var value6)) throw new Exception("В схеме нет списка состояний");

if (value6 is not string arr3) throw new Exception("Список состояний схемы - не строка");

states = arr3;

Open = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncOpen(); return new Unit(); });

NewItem = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncNewItem(); return new Unit(); });

Delete = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncDelete(); return new Unit(); });

}

public void Update(object[] items, object[] joins, string states) {

this.items = items;

this.joins = joins;

this.states = states;

Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

Update();

}

public object Export() {

return new Dictionary<string, object> {

["name"] = Name,

["created"] = Created,

["modified"] = Modified,

["items"] = items,

["joins"] = joins,

["states"] = states,

};

}

public void Update() {

Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

parent.Modified = Modified;

parent.Save();

}

public override string ToString() => Name;

internal void ChangeName(string name) {

Name = name;

Update();

}

/\*

\* Кнопоки

\*/

void FuncOpen() {

ViewModelBase.map.current\_scheme = this;

ViewModelBase.map.ImportScheme();

parent.UpdateList();

}

void FuncNewItem() {

parent.AddScheme(this);

parent.UpdateList();

}

void FuncDelete() {

parent.RemoveScheme(this);

parent.UpdateList();

}

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

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

public ReactiveCommand<Unit, Unit> Delete { get; }

public bool CanUseSchemeDeleter { get => parent.schemes.Count > 1; }

public bool CanOpenMe { get => ViewModelBase.map.current\_scheme != this; }

public void UpdateProps() {

this.RaisePropertyChanged(nameof(CanUseSchemeDeleter));

this.RaisePropertyChanged(nameof(CanOpenMe));

}

}

}

<Simulator.cs >

using LogicSimulator.ViewModels;

using LogicSimulator.Views.Shapes;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicSimulator.Models {

public class Meta {

public IGate? item;

public int[] ins;

public int[] outs;

public bool[] i\_buf;

public bool[] o\_buf;

public Meta(IGate item, int out\_id) {

this.item = item;

ins = Enumerable.Repeat(0, item.CountIns).ToArray();

outs = Enumerable.Range(out\_id, item.CountOuts).ToArray();

i\_buf = Enumerable.Repeat(false, item.CountIns).ToArray();

o\_buf = Enumerable.Repeat(false, item.CountOuts).ToArray();

}

public void Print() {

Log.Write("Элемент: " + item + " | Ins: " + Utils.Obj2json(ins) + " | Outs: " + Utils.Obj2json(outs));

}

}

public class Simulator {

public Simulator() {

Start();

}

private Task? task;

private bool stop\_sim = false;

public bool lock\_sim = false;

public void Start() {

if (task != null || lock\_sim) return;

stop\_sim = false;

task = Task.Run(async () => {

for (; ; ) {

await Task.Delay(1000 / 1000);

try { Tick(); } catch (Exception e) { Log.Write("Logical crush: " + e); continue; }

if (stop\_sim) return;

}

});

}

public void Stop() {

if (task == null) return;

stop\_sim = true;

task.GetAwaiter().GetResult();

task = null;

}

List<bool> outs = new() { false };

List<bool> outs2 = new() { false };

readonly List<Meta> items = new();

readonly Dictionary<IGate, Meta> ids = new();

public void AddItem(IGate item) {

Stop();

int out\_id = outs.Count;

for (int i = 0; i < item.CountOuts; i++) {

outs.Add(false);

outs2.Add(false);

}

// int id = items.Count;

Meta meta = new(item, out\_id);

items.Add(meta);

ids.Add(item, meta);

Start();

// meta.Print();

}

public void RemoveItem(IGate item) {

Stop();

Meta meta = ids[item];

meta.item = null;

foreach (var i in Enumerable.Range(0, meta.outs.Length)) {

int n = meta.outs[i];

outs[n] = outs2[n] = false;

}

ids.Remove(item);

Start();

}

private void Tick() {

foreach (var meta in items) {

var item = meta.item;

if (item == null) continue;

item.LogicUpdate(ids, meta);

int[] i\_n = meta.ins, o\_n = meta.outs;

bool[] ib = meta.i\_buf, ob = meta.o\_buf;

for (int i = 0; i < ib.Length; i++) ib[i] = outs[i\_n[i]];

item.Brain(ref ib, ref ob);

for (int i = 0; i < ob.Length; i++) {

bool res = ob[i];

outs2[o\_n[i]] = res;

item.SetJoinColor(i, res);

}

}

(outs2, outs) = (outs, outs2);

if (comparative\_test\_mode) {

prev\_state = cur\_state;

cur\_state = Export();

}

}

public void Clean() {

int n = 0;

int[] arr = Enumerable.Repeat(-1, outs.Count).ToArray();

StringBuilder sb = new();

sb.Append('0');

foreach (var meta in items)

if (meta.item != null)

foreach (var @out in meta.outs) {

arr[@out] = ++n;

sb.Append(outs[@out] ? '1' : '0');

}

arr[0] = 0;

foreach (var meta in items) {

meta.outs = meta.outs.Select(x => arr[x]).ToArray();

meta.ins = meta.ins.Select(x => arr[x]).ToArray();

}

Import(sb.ToString());

}

public string Export() => string.Join("", outs.Select(x => x ? '1' : '0'));

public void Import(string state) {

if (state.Length == 0) state = "0";

outs = state.Select(x => x == '1').ToList();

outs2 = outs.ToList(); // clone

}

public void Clear() {

outs = new() { false };

outs2 = new() { false };

items.Clear();

ids.Clear();

}

public void TopSecretPublicTickMethod() => Tick();

// Для комплесного решения:

public Switch[] GetSwitches() => items.Select(x => x.item).OfType<Switch>().ToArray();

public LightBulb[] GetLightBulbs() => items.Select(x => x.item).OfType<LightBulb>().ToArray();

private bool comparative\_test\_mode = false;

private string prev\_state = "0";

private string cur\_state = "0";

public bool ComparativeTestMode {

get => comparative\_test\_mode;

set {

comparative\_test\_mode = value;

if (value) prev\_state = cur\_state = Export();

}

}

public bool SomethingHasChanged => prev\_state != cur\_state;

}

}

<Utils.cs>

using System.Text;

using System.Xml.Linq;

using System.Collections.Generic;

using System.Linq;

using Avalonia.Controls;

using Avalonia.Media.Imaging;

using Avalonia;

using Avalonia.Media;

using System.Text.Json;

using LogicSimulator.ViewModels;

using System.Collections;

using System.Diagnostics;

using System;

using Avalonia.Controls.Shapes;

using System.Data.SQLite;

using System.Data;

namespace LogicSimulator.Models {

public static class Utils {

/\*

\* Base64 абилка

\*/

public static string Base64Encode(string plainText) {

var plainTextBytes = Encoding.UTF8.GetBytes(plainText);

return System.Convert.ToBase64String(plainTextBytes);

}

public static string Base64Decode(string base64EncodedData) {

var base64EncodedBytes = System.Convert.FromBase64String(base64EncodedData);

return Encoding.UTF8.GetString(base64EncodedBytes);

}

public static string JsonEscape(string str) {

StringBuilder sb = new();

foreach (char i in str) {

sb.Append(i switch {

'"' => "\\\"",

'\\' => "\\\\",

'$' => "{$",

\_ => i

});

}

return sb.ToString();

}

public static string Obj2json(object? obj) {

switch (obj) {

case null: return "null";

case string @str: return '"' + JsonEscape(str) + '"';

case bool @bool: return @bool ? "true" : "false";

case short @short: return @short.ToString();

case int @int: return @int.ToString();

case long @long: return @long.ToString();

case float @float: return @float.ToString().Replace(',', '.');

case double @double: return @double.ToString().Replace(',', '.');

case Point @point: return "\"$p$" + (int) @point.X + "," + (int) @point.Y + '"';

case Size @size: return "\"$s$" + (int) @size.Width + "," + (int) @size.Height + '"';

case Points @points: return "\"$P$" + string.Join("|", @points.Select(p => (int) p.X + "," + (int) p.Y)) + '"';

case SolidColorBrush @color: return "\"$C$" + @color.Color + '"';

case Thickness @thickness: return "\"$T$" + @thickness.Left + "," + @thickness.Top + "," + @thickness.Right + "," + @thickness.Bottom + '"';

case Dictionary<string, object?> @dict: {

StringBuilder sb = new();

sb.Append('{');

foreach (var entry in @dict) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(Obj2json(entry.Key));

sb.Append(": ");

sb.Append(Obj2json(entry.Value));

}

sb.Append('}');

return sb.ToString();

}

case IEnumerable @list: {

StringBuilder sb = new();

sb.Append('[');

foreach (object? item in @list) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(Obj2json(item));

}

sb.Append(']');

return sb.ToString();

}

default: return "(" + obj.GetType() + " ???)";

}

}

private static object JsonHandler(string str) {

if (str.Length < 3 || str[0] != '$' || str[2] != '$') return str.Replace("{$", "$");

string data = str[3..];

string[] thick = str[1] == 'T' ? data.Split(',') : System.Array.Empty<string>();

return str[1] switch {

'p' => Point.Parse(data),

's' => Size.Parse(data),

// 'P' => new SafePoints(data.Replace('|', ' ')).Points,

'C' => new SolidColorBrush(Color.Parse(data)),

'T' => new Thickness(double.Parse(thick[0]), double.Parse(thick[1]), double.Parse(thick[2]), double.Parse(thick[3])),

\_ => str,

};

}

private static object? JsonHandler(object? obj) {

if (obj == null) return null;

if (obj is List<object?> @list) return @list.Select(JsonHandler).ToList();

if (obj is Dictionary<string, object?> @dict) {

return new Dictionary<string, object?>(@dict.Select(pair => new KeyValuePair<string, object?>(pair.Key, JsonHandler(pair.Value))));

}

if (obj is JsonElement @item) {

switch (@item.ValueKind) {

case JsonValueKind.Undefined: return null;

case JsonValueKind.Object:

Dictionary<string, object?> res = new();

foreach (var el in @item.EnumerateObject()) res[el.Name] = JsonHandler(el.Value);

return res;

case JsonValueKind.Array:

List<object?> res2 = @item.EnumerateArray().Select(item => JsonHandler((object?) item)).ToList();

return res2;

case JsonValueKind.String:

var s = JsonHandler(@item.GetString() ?? "");

// Log.Write("JS: '" + @item.GetString() + "' -> '" + s + "'");

return s;

case JsonValueKind.Number:

if (@item.ToString().Contains('.')) return @item.GetDouble();

// Иначе это целое число

long a = @item.GetInt64();

int b = @item.GetInt32();

// short c = @item.GetInt16();

if (a != b) return a;

// if (b != c) return b;

return b;

case JsonValueKind.True: return true;

case JsonValueKind.False: return false;

case JsonValueKind.Null: return null;

}

}

Log.Write("JT: " + obj.GetType());

return obj;

}

public static object? Json2obj(string json) {

json = json.Trim();

if (json.Length == 0) return null;

object? data;

if (json[0] == '[') data = JsonSerializer.Deserialize<List<object?>>(json);

else if (json[0] == '{') data = JsonSerializer.Deserialize<Dictionary<string, object?>>(json);

else return null;

return JsonHandler(data);

}

/\*

\* XML абилка

\*/

public static string XMLEscape(string str) {

StringBuilder sb = new();

foreach (char i in str) {

sb.Append(i switch {

'"' => "&quot;",

'\'' => "&apos;",

'>' => "&gt;",

'<' => "&lt;",

'&' => "&amp;",

\_ => i

});

}

return sb.ToString();

}

private static bool IsComposite(object? obj) {

if (obj == null) return false;

if (obj is List<object?> || obj is Dictionary<string, object?> || obj is not JsonElement @item) return true;

var T = @item.ValueKind;

return T == JsonValueKind.Object || T == JsonValueKind.Array;

}

private static string Dict2XML(Dictionary<string, object?> dict, string level) {

StringBuilder attrs = new();

StringBuilder items = new();

foreach (var entry in dict)

if (IsComposite(entry.Value))

items.Append(level + "\t<" + entry.Key + ">" + ToXMLHandler(entry.Value, level + "\t\t") + level + "\t</" + entry.Key + ">");

else attrs.Append(" " + entry.Key + "=\"" + ToXMLHandler(entry.Value, "{err}") + "\"");

if (items.Length == 0) return level + "<Dict" + attrs.ToString() + "/>";

return level + "<Dict" + attrs.ToString() + ">" + items.ToString() + level + "</Dict>";

}

private static string List2XML(List<object?> list, string level) {

StringBuilder attrs = new();

StringBuilder items = new();

int num = 0;

foreach (var entry in list) {

if (IsComposite(entry)) items.Append(ToXMLHandler(entry, level + "\t"));

else attrs.Append($" \_{num}='" + ToXMLHandler(entry, "{err}") + "'");

num++;

}

if (items.Length == 0) return level + "<List" + attrs.ToString() + "/>";

return level + "<List" + attrs.ToString() + ">" + items.ToString() + level + "</List>";

}

private static string ToXMLHandler(object? obj, string level) {

if (obj == null) return "null";

if (obj is List<object?> @list) return List2XML(@list, level);

if (obj is Dictionary<string, object?> @dict) return Dict2XML(@dict, level);

if (obj is JsonElement @item) {

switch (@item.ValueKind) {

case JsonValueKind.Undefined: return "undefined";

case JsonValueKind.Object:

return Dict2XML(new Dictionary<string, object?>(@item.EnumerateObject().Select(pair => new KeyValuePair<string, object?>(pair.Name, pair.Value))), level);

case JsonValueKind.Array:

return List2XML(@item.EnumerateArray().Select(item => (object?) item).ToList(), level);

case JsonValueKind.String:

var s = XMLEscape(@item.GetString() ?? "null");

// Log.Write("XS: '" + @item.GetString() + "' -> '" + s + "'");

return s;

case JsonValueKind.Number: return "$" + @item.ToString(); // escape NUM

case JsonValueKind.True: return "\_BOOL\_yeah";

case JsonValueKind.False: return "\_BOOL\_nop";

case JsonValueKind.Null: return "null";

}

}

Log.Write("XT: " + obj.GetType());

return "<UnknowType>" + obj.GetType() + "</UnknowType>";

}

public static string? Json2xml(string json) {

json = json.Trim();

if (json.Length == 0) return null;

object? data;

if (json[0] == '[') data = JsonSerializer.Deserialize<List<object?>>(json);

else if (json[0] == '{') data = JsonSerializer.Deserialize<Dictionary<string, object?>>(json);

else return null;

return "<?xml version=\"1.0\" encoding=\"utf-8\"?>" + ToXMLHandler(data, "\n");

}

private static string ToJSONHandler(string str) {

if (str.Length > 1 && str[0] == '$' && str[1] <= '9' && str[1] >= '0') return str[1..]; // unescape NUM

str = str.Replace("\\", "\\\\");

return str switch {

"null" => "null",

"undefined" => "undefined",

"\_BOOL\_yeah" => "true",

"\_BOOL\_nop" => "false",

\_ => '"' + str + '"',

};

}

private static string ToJSONHandler(XElement xml) {

var name = xml.Name.LocalName;

StringBuilder sb = new();

if (name == "Dict") {

sb.Append('{');

foreach (var attr in xml.Attributes()) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(ToJSONHandler(attr.Name.LocalName));

sb.Append(": ");

sb.Append(ToJSONHandler(attr.Value));

}

foreach (var el in xml.Elements()) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(ToJSONHandler(el.Name.LocalName));

sb.Append(": ");

sb.Append(ToJSONHandler(el.Elements().ToArray()[0]));

}

sb.Append('}');

} else if (name == "List") {

var attrs = xml.Attributes().ToArray();

var els = xml.Elements().ToArray();

int count = attrs.Length + els.Length;

var res = new string[count];

var used = new bool[count];

int num;

foreach (var attr in attrs) {

num = int.Parse(attr.Name.LocalName[1..]);

res[num] = ToJSONHandler(attr.Value);

used[num] = true;

}

num = 0;

foreach (var el in els) {

while (used[num]) num++;

res[num++] = ToJSONHandler(el);

}

sb.Append('[');

foreach (var item in res) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(item);

}

sb.Append(']');

} else sb.Append("Type??" + name);

return sb.ToString();

}

public static string Xml2json(string xml) => ToJSONHandler(XElement.Parse(xml));

public static string YAMLEscape(string str) {

string[] arr = new[] { "true", "false", "null", "undefined", "" };

if (arr.Contains(str)) return '"' + str + '"';

string black\_list = " -:\"\n\t";

bool escape = "0123456789[{".Contains(str[0]);

if (!escape)

foreach (char i in str)

if (black\_list.Contains(i)) { escape = true; break; }

if (!escape) return str;

StringBuilder sb = new();

sb.Append('"');

foreach (char i in str) {

sb.Append(i switch {

'"' => "\\\"",

'\\' => "\\\\",

\_ => i

});

}

sb.Append('"');

return sb.ToString();

}

private static string Dict2YAML(Dictionary<string, object?> dict, string level) {

if (dict.Count == 0) return " {}";

StringBuilder res = new();

foreach (var entry in dict)

res.Append(level + YAMLEscape(entry.Key) + ":" + (IsComposite(entry.Value) ? "" : " ") + ToYAMLHandler(entry.Value, level + "\t"));

return res.ToString();

}

private static string List2YAML(List<object?> list, string level) {

if (list.Count == 0) return " []";

StringBuilder res = new();

foreach (var entry in list)

res.Append(level + "-" + (IsComposite(entry) ? "" : " ") + ToYAMLHandler(entry, level + "\t"));

return res.ToString();

}

private static string ToYAMLHandler(object? obj, string level) {

if (obj == null) return "null";

if (obj is List<object?> @list) return List2YAML(@list, level);

if (obj is Dictionary<string, object?> @dict) return Dict2YAML(@dict, level);

if (obj is JsonElement @item) {

switch (@item.ValueKind) {

case JsonValueKind.Undefined: return "undefined";

case JsonValueKind.Object:

return Dict2YAML(new Dictionary<string, object?>(@item.EnumerateObject().Select(pair => new KeyValuePair<string, object?>(pair.Name, pair.Value))), level);

case JsonValueKind.Array:

return List2YAML(@item.EnumerateArray().Select(item => (object?) item).ToList(), level);

case JsonValueKind.String:

var s = YAMLEscape(@item.GetString() ?? "null");

// Log.Write("YS: '" + @item.GetString() + "' -> " + s);

return s;

case JsonValueKind.Number: return @item.ToString();

case JsonValueKind.True: return "true";

case JsonValueKind.False: return "false";

case JsonValueKind.Null: return "null";

}

}

Log.Write("YT: " + obj.GetType());

throw new Exception("Чё?!");

}

public static string? Json2yaml(string json) {

json = json.Trim();

if (json.Length == 0) return null;

object? data;

if (json[0] == '[') data = JsonSerializer.Deserialize<List<object?>>(json);

else if (json[0] == '{') data = JsonSerializer.Deserialize<Dictionary<string, object?>>(json);

else return null;

return "---" + ToYAMLHandler(data, "\n") + "\n"; // Конец будет обязателен, как в питоне!

}

private static void YAML\_Log(string mess, int level = 0) {

if (level >= 4) Log.Write(mess);

}

private static string YAML\_ParseString(ref string yaml, ref int pos) {

char first = ' ';

while (" \n\t".Contains(first)) first = yaml[pos++];

bool quote = first == '"';

StringBuilder sb = new();

if (quote) {

char c = yaml[pos++];

while (c != '"') {

sb.Append(c);

c = yaml[pos++];

}

c = yaml[pos++];

if (c != ':' && c != '\n') throw new Exception("После '\"' может быть только ':', либо '\n'");

if (c == ':') pos--;

} else {

sb.Append(first);

char c = yaml[pos++];

while (c != ':' && c != '\n') {

sb.Append(c);

c = yaml[pos++];

}

if (c == ':') pos--;

}

YAML\_Log("Parsed str: " + sb.ToString(), 1);

return sb.ToString();

}

private static string YAML\_ParseNum(ref string yaml, ref int pos) {

char c = yaml[pos++];

StringBuilder sb = new();

while ("0123456789.".Contains(c)) {

sb.Append(c);

c = yaml[pos++];

}

if (c != '\n') throw new Exception("После числа всяко должен быть '\n");

YAML\_Log("Parsed num: " + sb.ToString(), 1);

return sb.ToString();

}

private static string YAML\_ParseItem(ref string yaml, ref int pos) {

char first = ' ';

while (" \n\t".Contains(first)) first = yaml[pos++];

pos--;

if (first == '"')

return '"' + YAML\_ParseString(ref yaml, ref pos) + '"';

if ("0123456789".Contains(first))

return YAML\_ParseNum(ref yaml, ref pos);

string str = YAML\_ParseString(ref yaml, ref pos);

string[] arr = new[] { "true", "false", "null", "undefined", "", "[]", "{}" };

if (arr.Contains(str)) return str;

return '"' + str + '"';

}

private static string YAML\_ParseLayer(ref string yaml, ref int pos) {

if (pos == yaml.Length) return ""; // Конец файла

StringBuilder sb = new();

char first = yaml[pos++];

while (" \t".Contains(first)) {

sb.Append(first);

first = yaml[pos++];

}

pos--;

return sb.ToString();

}

private static string YAML\_ToJSONHandler(ref string yaml, ref int pos) {

var layer = YAML\_ParseLayer(ref yaml, ref pos);

if (pos == yaml.Length) return ""; // Конец файла

char first = yaml[pos++];

switch (first) {

case '[':

if (yaml[pos++] != ']' || yaml[pos++] != '\n') throw new Exception("После [ ожидалось ]\\n");

return "[]";

case '{':

if (yaml[pos++] != '}' || yaml[pos++] != '\n') throw new Exception("После { ожидалось }\\n");

return "{}";

case '-': {

StringBuilder res = new();

res.Append('[');

bool First = true;

pos--;

while (true) {

if (pos == yaml.Length) break; // Конец файла

if (First) First = false;

else {

var saved\_pos2 = pos;

var layer3 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("DOWN\_LAYER: '" + layer + "', '" + layer3 + "'");

if (layer != layer3) {

if (layer3.Length > layer.Length) throw new Exception("Ожидался элемент списка вместо подъёма");

if (!layer.StartsWith(layer3)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos2; break;

}

res.Append(", ");

}

if (yaml[pos++] != '-') throw new Exception("Ожидалось '-' в следующем элементе списка");

char c = yaml[pos++];

if (c == ' ') {

var value = YAML\_ParseItem(ref yaml, ref pos);

res.Append(value);

} else if (c == '\n') {

} else throw new Exception("После '-' ожидалось ' ', либо '\n'");

int saved\_pos = pos;

var layer2 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("LAYER: '" + layer + "', '" + layer2 + "'");

if (layer2.Length < layer.Length) {

if (!layer.StartsWith(layer2)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos; break;

}

if (!layer2.StartsWith(layer)) throw new Exception("Странность в следующем layer'е");

if (layer == layer2) { YAML\_Log("Сохранение"); pos = saved\_pos; continue; }

YAML\_Log("Подъём");

if (c == '\n') {

pos = saved\_pos;

var value = YAML\_ToJSONHandler(ref yaml, ref pos);

res.Append(value);

} else throw new Exception("Здесь не может быть подъёма");

}

res.Append(']');

YAML\_Log("Список рождён: " + res.ToString(), 2);

return res.ToString(); }

case '"':

default: {

pos--;

StringBuilder res = new();

res.Append('{');

bool First = true;

while (true) {

if (pos == yaml.Length) break; // Конец файла

if (First) First = false;

else {

var saved\_pos2 = pos;

var layer3 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("DICT\_LAYER: '" + layer + "', '" + layer3 + "'");

if (layer != layer3) {

if (layer3.Length > layer.Length) throw new Exception("Ожидался элемент словаря вместо подъёма");

if (!layer.StartsWith(layer3)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos2; break;

}

res.Append(", ");

}

var key = YAML\_ParseString(ref yaml, ref pos);

res.Append('"');

res.Append(key);

res.Append("\": ");

if (yaml[pos++] != ':') throw new Exception("После ключа ожидалось ':'");

char c = yaml[pos++];

if (c == ' ') {

var value = YAML\_ParseItem(ref yaml, ref pos);

res.Append(value);

} else if (c == '\n') {

} else throw new Exception("После ключа и ':' ожидалось ' ', либо '\n'");

int saved\_pos = pos;

var layer2 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("LAYER: '" + layer + "', '" + layer2 + "'");

if (layer2.Length < layer.Length) {

if (!layer.StartsWith(layer2)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos; break;

}

if (!layer2.StartsWith(layer)) throw new Exception("Странность в следующем layer'е");

if (layer == layer2) { YAML\_Log("Сохранение"); pos = saved\_pos; continue; }

YAML\_Log("Подъём");

if (c == '\n') {

pos = saved\_pos;

var value = YAML\_ToJSONHandler(ref yaml, ref pos);

res.Append(value);

} else throw new Exception("Здесь не может быть подъёма");

}

res.Append('}');

YAML\_Log("Словарь рождён: " + res.ToString(), 2);

return res.ToString(); }

}

}

public static string Yaml2json(string yaml) {

try {

yaml = yaml.Replace("\r", "");

if (!yaml.StartsWith("---\n")) throw new Exception("Это не YAML");

int pos = 4;

var res = YAML\_ToJSONHandler(ref yaml, ref pos);

YAML\_Log("data: " + res, 3);

return res;

} catch (Exception e) { Log.Write("Ошибка YAML парсера: " + e); throw; }

}

public static string? Obj2xml(object? obj) => Json2xml(Obj2json(obj));

public static object? Xml2obj(string xml) => Json2obj(Xml2json(xml));

public static string? Obj2yaml(object? obj) => Json2yaml(Obj2json(obj));

public static object? Yaml2obj(string xml) => Json2obj(Yaml2json(xml));

public static void RenderToFile(Control target, string path) {

// var target = (Control?) tar.Parent;

// if (target == null) return;

double w = target.Bounds.Width, h = target.Bounds.Height;

var pixelSize = new PixelSize((int) w, (int) h);

var size = new Size(w, h);

using RenderTargetBitmap bitmap = new(pixelSize);

target.Measure(size);

target.Arrange(new Rect(size));

bitmap.Render(target);

bitmap.Save(path);

}

public static string TrimAll(this string str) { // Помимо пробелов по бокам, убирает повторы пробелов внутри

StringBuilder sb = new();

for (int i = 0; i < str.Length; i++) {

if (i > 0 && str[i] == ' ' && str[i - 1] == ' ') continue;

sb.Append(str[i]);

}

return sb.ToString().Trim();

}

public static string[] NormSplit(this string str) => str.TrimAll().Split(' ');

public static string GetStackInfo() {

var st = new StackTrace();

var sb = new StringBuilder();

for (int i = 1; i < 11; i++) {

var frame = st.GetFrame(i);

if (frame == null) continue;

var method = frame.GetMethod();

if (method == null || method.ReflectedType == null) continue;

sb.Append(method.ReflectedType.Name + " " + method.Name + " | ");

if (i == 5) sb.Append("\n ");

}

return sb.ToString();

}

public static int Normalize(this int num, int min, int max) {

if (num < min) return min;

if (num > max) return max;

return num;

}

public static double Normalize(this double num, double min, double max) {

if (num < min) return min;

if (num > max) return max;

return num;

}

public static double Hypot(this Point delta) {

return Math.Sqrt(Math.Pow(delta.X, 2) + Math.Pow(delta.Y, 2));

}

public static double Hypot(this Point A, Point B) {

Point delta = A - B;

return Math.Sqrt(Math.Pow(delta.X, 2) + Math.Pow(delta.Y, 2));

}

public static double? ToDouble(this object num) {

return num switch {

int @int => @int,

long @long => @long,

double @double => @double,

\_ => null,

};

}

public static int Min(this int A, int B) => A < B ? A : B;

public static int Max(this int A, int B) => A > B ? A : B;

public static double Min(this double A, double B) => A < B ? A : B;

public static double Max(this double A, double B) => A > B ? A : B;

public static void Remove(this Control item) {

var p = (Panel?) item.Parent;

p?.Children.Remove(item);

}

public static Point Center(this Visual item, Visual? parent) {

var tb = item.TransformedBounds;

if (tb == null) return new(); // Обычно так не бывает

var bounds = tb.Value.Bounds.TransformToAABB(tb.Value.Transform);

var res = bounds.Center;

if (parent == null) return res; // parent в качестве точки отсчёта, например холст

var tb2 = parent.TransformedBounds;

if (tb2 == null) return res; // Обычно так не бывает

var bounds2 = tb2.Value.Bounds.TransformToAABB(tb2.Value.Transform);

return res - bounds2.TopLeft;

}

public static DateTime UnixTimeStampToDateTime(this long unixTimeStamp) {

DateTime dateTime = new(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);

dateTime = dateTime.AddSeconds(unixTimeStamp).ToLocalTime();

return dateTime;

}

public static string UnixTimeStampToString(this long unixTimeStamp) {

return UnixTimeStampToDateTime(unixTimeStamp).ToString("yyyy/MM/dd H:mm:ss");

}

/\*

\* SQLite\_proj\_list абилка

\*/

internal static void Obj2sqlite\_proj\_list(string[] proj\_list, string path) {

// Log.Write(Obj2yaml(proj.Export()) + "");

using var con = new SQLiteConnection("Data Source=" + path);

con.Open();

if (con.State != ConnectionState.Open) { Log.Write("Не удалось открыть SQLite: " + con.State); return; }

string comm = @"

CREATE TABLE header (

num INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

path TEXT NOT NULL

);";

new SQLiteCommand(comm, con).ExecuteReader().Close();

var arr = proj\_list.Select(x => $"('{x}')");

comm = $"INSERT INTO header (path) VALUES {string.Join(", ", arr)};";

new SQLiteCommand(comm, con).ExecuteReader().Close();

con.Dispose();

}

internal static string[] SQLite\_proj\_list2obj(string path) {

using var con = new SQLiteConnection("Data Source=" + path);

con.Open();

if (con.State != ConnectionState.Open) throw new Exception("Не удалось открыть SQLite: " + con.State);

List<string> res = new();

var sql\_comm = new SQLiteCommand("SELECT \* FROM header", con);

using (var reader = sql\_comm.ExecuteReader()) {

if (!reader.HasRows) throw new Exception("Не вышло считать заголовочную таблицу SQLite :/");

if (!reader.Read()) throw new Exception("Заголовочная таблица пустует");

var row = Enumerable.Range(0, reader.VisibleFieldCount).Select(x => reader[x]).ToArray();

Log.Write("row: " + Obj2json(row));

res.Add((string) row[1]);

}

con.Dispose();

return res.ToArray();

}

}

}

< Animation.axaml>

<Styles xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<Design.PreviewWith>

<Border Padding="20">

<Rectangle Classes="anim" Width="100" Height="100" Fill="Red"/>

</Border>

</Design.PreviewWith>

<Style Selector="Line.anim">

<Style.Animations>

<Animation Duration="0:0:1" IterationCount="INFINITE" Easing="SineEaseInOut" PlaybackDirection="Alternate">

<KeyFrame Cue="0%">

<Setter Property="Opacity" Value="1" />

</KeyFrame>

<KeyFrame Cue="50%">

<Setter Property="Opacity" Value="0" />

</KeyFrame>

</Animation>

</Style.Animations>

</Style>

<Style Selector="Polyline.anim">

<Style.Animations>

<Animation Duration="0:0:1" IterationCount="INFINITE" Easing="SineEaseInOut" PlaybackDirection="Alternate">

<KeyFrame Cue="0%">

<Setter Property="Opacity" Value="1" />

</KeyFrame>

<KeyFrame Cue="50%">

<Setter Property="Opacity" Value="0" />

</KeyFrame>

</Animation>

</Style.Animations>

</Style>

<Style Selector="Rectangle.anim">

<Style.Animations>

<Animation Duration="0:0:1" IterationCount="INFINITE" Easing="SineEaseInOut" PlaybackDirection="Alternate">

<KeyFrame Cue="0%">

<Setter Property="Opacity" Value="1" />

</KeyFrame>

<KeyFrame Cue="50%">

<Setter Property="Opacity" Value="0" />

</KeyFrame>

</Animation>

</Style.Animations>

</Style>

<Style Selector="Ellipse.anim">

<Style.Animations>

<Animation Duration="0:0:1" IterationCount="INFINITE" Easing="SineEaseInOut" PlaybackDirection="Alternate">

<KeyFrame Cue="0%">

<Setter Property="Opacity" Value="1" />

</KeyFrame>

<KeyFrame Cue="50%">

<Setter Property="Opacity" Value="0" />

</KeyFrame>

</Animation>

</Style.Animations>

</Style>

<Style Selector="Polygon.anim">

<Style.Animations>

<Animation Duration="0:0:1" IterationCount="INFINITE" Easing="SineEaseInOut" PlaybackDirection="Alternate">

<KeyFrame Cue="0%">

<Setter Property="Opacity" Value="1" />

</KeyFrame>

<KeyFrame Cue="50%">

<Setter Property="Opacity" Value="0" />

</KeyFrame>

</Animation>

</Style.Animations>

</Style>

<Style Selector="Path.anim">

<Style.Animations>

<Animation Duration="0:0:1" IterationCount="INFINITE" Easing="SineEaseInOut" PlaybackDirection="Alternate">

<KeyFrame Cue="0%">

<Setter Property="Opacity" Value="1" />

</KeyFrame>

<KeyFrame Cue="50%">

<Setter Property="Opacity" Value="0" />

</KeyFrame>

</Animation>

</Style.Animations>

</Style>

</Styles>

< LauncherWindowViewModel.cs>

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();

}

/\*

\* Для тестирования

\*/

public static MainWindow GetMW => mw;

}

}

<MainWindowViewModel.cs >

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() { // Если я буду Window mw передавать через этот конструктор, то предварительный просмотр снова порвёт смачно XD

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; // Такого не бывает

canv.Children.Add(map.Marker);

canv.Children.Add(map.Marker2);

var panel = (Panel?) canv.Parent;

if (panel == null) return; // Такого не бывает

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; // Такого не бывает

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 той самой 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 }; // Изи блиц-транcформация в одну строчку ;'-}

// Log.Write("Tag: " + tb.Tag);

cur\_grid.Children[0] = newy;

//Log.Write("Tag: " + tb.Tag); // КААААК?!?!?!? Почему пропажа предка удаляет Tag?!

newy.KeyUp += (object? sender, KeyEventArgs e) => {

if (e.Key != Key.Return) return;

if (newy.Text != prev\_scheme\_name) {

// tb.Text = newy.Text;

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("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n Текущий 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();

// Для создания тестов:

File.WriteAllText("../../../for\_test.yaml", Utils.Obj2yaml((map.current\_scheme ?? throw new System.Exception("ERROR")).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; }

}

}

< ViewModelBase.cs>

using LogicSimulator.Models;

using ReactiveUI;

namespace LogicSimulator.ViewModels {

public class ViewModelBase: ReactiveObject {

public readonly static Mapper map = new();

private static Project? current\_proj;

protected static Project? CurrentProj {

get => current\_proj;

set {

if (value == null) return;

current\_proj = value;

map.current\_scheme = value.GetFirstScheme();

}

}

/\*

\* Для тестирования

\*/

public static Project? TopSecretGetProj() => current\_proj;

}

}

< AND\_2.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.AND\_2"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">AND</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< AND\_2.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class AND\_2: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 0;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = ins[0] && ins[1];

}

}

< AND\_8.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.AND\_8"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">AND</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< AND\_8.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class AND\_8: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 11;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0, 0, 0, 0, 0, 0, 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = ins[0] && ins[1] && ins[2] && ins[3] && ins[3] && ins[5] && ins[6] && ins[7];

}

}

< Button.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.Button"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#e7ae07" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#fec107" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}" PointerPressed="Press" PointerReleased="Release">

<Panel>

<Ellipse Tag="Body" Width="{Binding ButtonSize}" Height="{Binding ButtonSize}" Fill="#d32f2e"/>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< Button.axaml.cs>

using Avalonia.Controls;

using Avalonia.Controls.Shapes;

using Avalonia.Input;

using Avalonia.Media;

using LogicSimulator.Models;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class Button: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 6;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

System.Array.Empty<int>(),

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Обработка размеров внутренностей

\*/

public double ButtonSize => width.Min(height) - BodyStrokeSize.Left \* 5.5;

/\*

\* Мозги

\*/

bool my\_state = false;

private void Press(object? sender, PointerPressedEventArgs e) {

if (e.Source is not Ellipse button) return;

my\_state = true;

button.Fill = new SolidColorBrush(Color.Parse("#7d1414"));

}

private void Release(object? sender, PointerReleasedEventArgs e) {

if (e.Source is not Ellipse button) return;

my\_state = false;

button.Fill = new SolidColorBrush(Color.Parse("#d32f2e"));

}

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = my\_state;

}

}

< DC.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.DC"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">DC</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< DC.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class DC: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 4;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0 },

new int[] { 1, 1},

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) {

outs[0] = ins[0] ? false : true;

outs[1] = ins[0];

}

}

}

< FlipFlop.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.FlipFlop"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">FlipFlop</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

<FlipFlop.axaml.cs >

using Avalonia.Controls;

using LogicSimulator.ViewModels;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

namespace LogicSimulator.Views.Shapes {

public partial class FlipFlop: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 9;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0, 0 },

new int[] { 1, 1, 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

private readonly bool[] prev = new bool[3];

private readonly bool[] out\_d = new bool[3];

public void Brain(ref bool[] ins, ref bool[] outs) {

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

if (prev[i] && !ins[i]) out\_d[i] = !out\_d[i];

outs[i] = out\_d[i];

prev[i] = ins[i];

}

}

/\*

\* Кастомный экспорт и импорт

\*/

public override Dictionary<string, object> ExtraExport() =>

new() {

["state"] = string.Join('.', prev.Select(x => x ? '1' : '0')) + "." +

string.Join('.', out\_d.Select(x => x ? '1' : '0'))

};

public override void ExtraImport(string key, object extra) {

if (key != "state") { Log.Write(key + "-запись элемента не поддерживается"); return; }

if (extra is not string @state) { Log.Write("Неверный тип state-записи элемента: " + extra); return; }

var arr = @state.Split('.');

prev[0] = arr[0] == "1";

prev[1] = arr[1] == "1";

prev[2] = arr[2] == "1";

out\_d[0] = arr[3] == "1";

out\_d[1] = arr[4] == "1";

out\_d[2] = arr[5] == "1";

}

}

}

<GateBase.cs >

using Avalonia;

using Avalonia.Controls;

using Avalonia.Controls.Shapes;

using Avalonia.Media;

using Avalonia.Threading;

using LogicSimulator.Models;

using LogicSimulator.ViewModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

namespace LogicSimulator.Views.Shapes {

public abstract class GateBase: UserControl {

public int CountIns { get; private set; }

public int CountOuts { get; private set; }

public abstract UserControl GetSelf();

protected abstract IGate GetSelfI { get; }

protected abstract void Init();

protected abstract int[][] Sides { get; }

protected readonly Line[] line\_arr;

protected readonly Ellipse[] pins;

protected readonly Border border;

protected bool use\_top;

protected bool use\_left;

protected bool use\_right;

protected bool use\_bottom;

private int[][] pin\_data;

public GateBase() {

var sides = Sides;

use\_top = sides[0].Length > 0;

use\_left = sides[1].Length > 0;

use\_right = sides[2].Length > 0;

use\_bottom = sides[3].Length > 0;

int ins = 0, outs = 0, ios = 0, n = 0;

List<int[]> pin\_d = new();

foreach (var side in sides)

foreach (var type in side) {

switch (type) {

case 0: pin\_d.Add(new int[] { 0, ins }); ins++; break;

case 1: pin\_d.Add(new int[] { 1, outs }); outs++; break;

case 2: pin\_d.Add(new int[] { 2, ios }); ios++; break;

}

if (type != -1) n++;

}

pin\_data = pin\_d.ToArray();

CountIns = ins;

CountOuts = outs + ios;

/\* double sizer = sides.Select(x => x.Length).Max();

double vert\_sizer = Math.Max(Math.Max(sides[0].Length, sides[3].Length), 3);

width = 30 \* (2 + Math.Min(sizer, vert\_sizer) / 2);

height = Math.Max(30 \* (2 + sizer / 2), (9 + 32) \* 2 / 3 \* (1.5 + 0.75 \* CountIns.Max(CountOuts)));\*/

width = MinW; height = MinH;

if (height < width) height = width;

// AvaloniaXamlLoader.Load(GetSelf()); // InitializeComponent();

DataContext = GetSelf();

Init(); // :///

var canv = (Canvas) LogicalChildren[0];

List<Line> list = new();

List<Ellipse> list2 = new();

if (canv.Children[0] is not Border b) throw new Exception("Такого не бывает");

border = b;

border.ZIndex = 2;

foreach (var side in sides)

foreach (var type in side) {

if (type < 0) continue;

var newy = new Line() { Tag = "Pin", ZIndex = 1, Stroke = Brushes.Gray };

list.Add(newy);

canv.Children.Add(newy);

var newy2 = new Ellipse() { Tag = type == 0 ? "In" : type == 1 ? "Out" : "IO", ZIndex = 2, Stroke = Brushes.Gray, Fill = new SolidColorBrush(Color.Parse("#0000")) };

list2.Add(newy2);

canv.Children.Add(newy2);

}

line\_arr = list.ToArray();

pins = list2.ToArray();

joins\_in = new JoinedItems?[ins];

joins\_out = new List<JoinedItems>[outs];

for (int i = 0; i < outs; i++) joins\_out[i] = new();

MyRecalcSizes();

}

public void Move(Point pos, bool global = false) {

Margin = new(pos.X - UC\_Width / 2, pos.Y - UC\_Height / 2, 0, 0);

// Log.Write("Пришла позиция: " + pos + " | а вышла: " + GetPos());

UpdateJoins(global);

}

private double MinW => BodyRadius.TopLeft \* 1.5 + (EllipseSize + BaseFraction \* 2) \* (Sides[0].Length.Max(Sides[3].Length).Max(2) - 0.8);

private double MinH => BodyRadius.TopLeft \* 1.5 + (EllipseSize + BaseFraction \* 2) \* (Sides[1].Length.Max(Sides[2].Length).Max(2) - 0.8);

public void Resize(Size size, bool global = false) {

width = global ? size.Width : size.Width.Max(MinW);

height = global ? size.Height : size.Height.Max(MinH);

RecalcSizes();

UpdateJoins(global);

}

public void ChangeScale(double scale, bool global = false) {

var fix = GetPos();

base\_size \*= scale;

width \*= scale;

height \*= scale;

Move(fix, global);

RecalcSizes();

UpdateJoins(global);

}

public Point GetPos() => new(Margin.Left + UC\_Width / 2, Margin.Top + UC\_Height / 2);

public Size GetSize() => new(Width, Height);

public Size GetBodySize() => new(width, height);

private Point pose;

public void SavePose() => pose = GetPos();

public Point GetPose() => pose;

public Rect GetBounds() => new(Margin.Left, Margin.Top, UC\_Width, UC\_Height);

/\*

\* Обработка размеров внутренностей

\*/

protected double base\_size = 25;

protected double width = 30 \* 3; // Размеры тела, а не всего UserControl

protected double height = 30 \* 3;

public double BaseSize => base\_size;

public double BaseFraction => base\_size / 40;

public double EllipseSize => BaseFraction \* 30;

public Thickness BodyStrokeSize => new(BaseFraction \* 3);

public double EllipseStrokeSize => BaseFraction \* 5;

public double PinStrokeSize => BaseFraction \* 6;

public Thickness BodyMargin => new(use\_left ? base\_size : 0, use\_top ? base\_size : 0, 0, 0);

public double BodyWidth => width;

public double BodyHeight => height;

public CornerRadius BodyRadius => new(width.Min(height) / 3 + BodyStrokeSize.Top);

public double UC\_Width => (use\_left ? base\_size : 0) + width + (use\_right ? base\_size : 0);

public double UC\_Height => (use\_top ? base\_size : 0) + height + (use\_bottom ? base\_size : 0);

public double FontSizze => BodyRadius.TopLeft / 1.3;

public Thickness ImageMargins { get {

double R = BodyRadius.BottomLeft;

double num = R - R / Math.Sqrt(2);

return new(0, 0, num, num); // Картинка с переместителем

// Картинка с удалителем ... устранена ;'-}

} }

public Point[][] PinPoints { get {

List<Point[]> res = new();

int n = -1;

double R = BodyRadius.TopLeft;

double min = EllipseSize + BaseFraction \* 2;

double pin\_start = EllipseSize - EllipseStrokeSize / 2;

double pin\_width = base\_size - EllipseSize + PinStrokeSize;

// .1.

// .1..2.

// .1..2..3.

foreach (var side in Sides) {

n++;

double count = side.Length;

if (count == 0) continue;

double body\_len = n == 0 || n == 3 ? height : width;

double body\_len2 = n == 0 || n == 3 ? width : height;

double delta = n < 2 ? pin\_start : (n == 2 ? (use\_left ? base\_size : 0) : (use\_top ? base\_size : 0)) + body\_len - EllipseStrokeSize / 2;

double left = R, mid = body\_len2 / 2, right = body\_len2 - R;

bool overflow = count > 1 && (right - left) / count < min;

int n2 = 0;

foreach (int type in side) {

double delta2 = overflow ?

mid + min \* (n2 - (count - 1) / 2) :

left + (right - left) / (count \* 2) \* (n2 \* 2 + 1);

if (type >= 0) res.Add(n == 0 || n == 3 ?

new Point[] { new(delta2 + (use\_left ? base\_size : 0), delta), new(0, pin\_width) } :

new Point[] { new(delta, delta2 + (use\_top ? base\_size : 0)), new(pin\_width, 0) }

);;

n2++;

}

}

return res.ToArray();

} }

public Thickness[] EllipseMargins { get {

Point[][] pins = PinPoints;

double R2 = EllipseSize / 2;

double X = UC\_Width - EllipseSize;

double Y = UC\_Height - EllipseSize;

int n = 0, side\_n = 0;

List<Thickness> list = new();

foreach (var side in Sides) {

foreach (var type in side) {

if (type == -1) continue;

var pin\_line = pins[n++];

switch (side\_n) {

case 0: list.Add(new(pin\_line[0].Y - R2, 0, 0, 0)); break;

case 1: list.Add(new(0, pin\_line[0].Y - R2, 0, 0)); break;

case 2: list.Add(new(X, pin\_line[0].Y - R2, 0, 0)); break;

case 3: list.Add(new(pin\_line[0].X - R2, Y, 0, 0)); break;

}

}

side\_n++;

}

return ellipse\_margins = list.ToArray();

} }

public double ImageSize => base\_size / 25 \* 24;

#pragma warning disable CS0108

public event PropertyChangedEventHandler? PropertyChanged;

#pragma warning restore CS0108

protected void RecalcSizes() {

// Log.Write("Size: " + width + " " + height);

PropertyChanged?.Invoke(this, new(nameof(BodyStrokeSize)));

PropertyChanged?.Invoke(this, new(nameof(BodyMargin)));

PropertyChanged?.Invoke(this, new(nameof(BodyWidth)));

PropertyChanged?.Invoke(this, new(nameof(BodyHeight)));

PropertyChanged?.Invoke(this, new(nameof(BodyRadius)));

PropertyChanged?.Invoke(this, new(nameof(UC\_Width)));

PropertyChanged?.Invoke(this, new(nameof(UC\_Height)));

PropertyChanged?.Invoke(this, new(nameof(FontSizze)));

PropertyChanged?.Invoke(this, new(nameof(ImageMargins)));

PropertyChanged?.Invoke(this, new(nameof(ImageSize)));

PropertyChanged?.Invoke(this, new("ButtonSize"));

PropertyChanged?.Invoke(this, new("InvertorSize"));

PropertyChanged?.Invoke(this, new("InvertorStrokeSize"));

PropertyChanged?.Invoke(this, new("InvertorMargin"));

MyRecalcSizes();

}

protected void MyRecalcSizes() {

var pin\_points = PinPoints;

var pin\_stroke\_size = PinStrokeSize;

int n = 0;

foreach (var line in line\_arr) {

var A = pin\_points[n][0];

var B = pin\_points[n++][1];

line.StrokeThickness = pin\_stroke\_size;

// line.StartPoint = A;

line.Margin = new(A.X, A.Y, 0, 0);

line.EndPoint = B;

}

n = 0;

var ellipse\_margin = EllipseMargins;

var ellipse\_size = EllipseSize;

var ellipse\_stroke\_size = EllipseStrokeSize;

foreach (var pin in pins) {

pin.Margin = ellipse\_margin[n++];

pin.Width = ellipse\_size;

pin.Height = ellipse\_size;

pin.StrokeThickness = ellipse\_stroke\_size;

}

}

/\*

\* Обработка соединений

\*/

protected JoinedItems?[] joins\_in;

protected List<JoinedItems>[] joins\_out;

public void AddJoin(JoinedItems join) {

for (int i = 0; i < 2; i++) {

var dist = i == 0 ? join.A : join.B;

if (dist.parent == this) {

int[] data = pin\_data[dist.num];

int n = data[1];

if (data[0] == 0) {

joins\_in[n]?.Delete();

joins\_in[n] = join;

// Log.Write("AddIn: " + n);

} else {

joins\_out[n].Add(join);

// Log.Write("AddOut: " + n);

}

}

}

skip\_upd = false;

}

public void RemoveJoin(JoinedItems join) {

for (int i = 0; i < 2; i++) {

var dist = i == 0 ? join.A : join.B;

if (dist.parent == this) {

int[] data = pin\_data[dist.num];

int n = data[1];

if (data[0] == 0) joins\_in[n] = null;

else joins\_out[n].Remove(join);

}

}

skip\_upd = false;

}

public void UpdateJoins(bool global) {

foreach (var join in joins\_in) join?.Update();

if (!global)

foreach (var joins in joins\_out)

foreach (var join in joins) join.Update();

}

public void ClearJoins() {

foreach (var join in joins\_in) join?.Delete();

foreach (var joins in joins\_out)

foreach (var join in joins.ToArray()) join.Delete();

}

public void SetJoinColor(int o\_num, bool value) {

var joins = joins\_out[o\_num];

Dispatcher.UIThread.InvokeAsync(() => {

foreach(var join in joins)

join.line.Stroke = value ? Brushes.Lime : Brushes.DarkGray;

});

}

public bool ContainsJoin(JoinedItems join) {

foreach (var join2 in joins\_in) if (join == join2) return true;

foreach (var joins in joins\_out)

foreach (var join2 in joins) if (join == join2) return true;

return false;

}

/\*

\* Обработка пинов

\*/

public Distantor GetPin(Ellipse finded) {

int n = 0;

foreach (var pin in pins) {

if (pin == finded) return new(GetSelfI, n, (string?) finded.Tag ?? "");

n++;

}

throw new Exception("Так не бывает");

}

Thickness[] ellipse\_margins = Array.Empty<Thickness>();

public Point GetPinPos(int n) {

var m = ellipse\_margins[n];

double R2 = EllipseSize / 2;

return new Point(Margin.Left + m.Left + R2, Margin.Top + m.Top + R2);

}

/\*

\* Мозги

\*/

public int[][] GetPinData() => pin\_data;

bool skip\_upd = true;

public void LogicUpdate(Dictionary<IGate, Meta> ids, Meta me) {

if (skip\_upd) return;

skip\_upd = true;

int ins = CountIns;

for (int i = 0; i < ins; i++) {

var join = joins\_in[i];

if (join == null) { me.ins[i] = 0; continue; }

if (join.A.parent == this) {

var item = join.B;

if (item.tag == "Out" || item.tag == "IO") {

var p = item.parent;

Meta meta = ids[p];

int[] data = p.GetPinData()[item.num];

me.ins[i] = meta.outs[data[1]];

// Log.Write("ins: " + Utils.Obj2json(me.ins) + " | " + data[1]);

}

}

if (join.B.parent == this) {

var item = join.A;

if (item.tag == "Out" || item.tag == "IO") {

var p = item.parent;

Meta meta = ids[p];

int[] data = p.GetPinData()[item.num];

me.ins[i] = meta.outs[data[1]];

// Log.Write("ins: " + Utils.Obj2json(me.ins) + " | " + data[1]);

}

}

}

}

/\*

\* Экспорт

\*/

public abstract int TypeId { get; }

public object Export() {

var res = new Dictionary<string, object> {

["id"] = TypeId,

["pos"] = GetPos(),

["size"] = GetBodySize(),

["base\_size"] = base\_size

};

var res2 = ExtraExport();

if (res2 != null) foreach (var item in res2) res.Add(item.Key, item.Value);

return res;

}

public virtual Dictionary<string, object>? ExtraExport() => null;

public List<object[]> ExportJoins(Dictionary<IGate, int> to\_num) {

List<object[]> res = new();

foreach (var joins in joins\_out) foreach (var join in joins) {

Distantor a = join.A, b = join.B;

res.Add(new object[] {

to\_num[a.parent], a.num, a.tag,

to\_num[b.parent], b.num, b.tag,

});

}

return res;

}

public void Import(Dictionary<string, object> dict) {

double new\_b\_size = base\_size;

Point new\_pos = GetPos();

Size new\_size = GetSize();

foreach (var item in dict) {

object value = item.Value;

switch (item.Key) {

case "id":

if (value is int @id) {

if (@id != TypeId) throw new ArgumentException("ВНИМАНИЕ! Пришёл не верный id: " + @id + " Ожидалось: " + TypeId);

} else Log.Write("Неверный тип id-записи элемента: " + value);

break;

case "pos":

if (value is Point @pos) new\_pos = @pos;

else Log.Write("Неверный тип pos-записи элемента: " + value);

break;

case "base\_size":

double? b\_size = value.ToDouble();

if (b\_size != null) new\_b\_size = (double) b\_size;

else Log.Write("Неверный тип base\_size-записи элемента: " + value);

break;

case "size":

if (value is Size @size) new\_size = @size;

else Log.Write("Неверный тип size-записи элемента: " + value);

break;

default:

ExtraImport(item.Key, value);

break;

}

}

base\_size = new\_b\_size;

Resize(new\_size, true);

Move(new\_pos);

}

public virtual void ExtraImport(string key, object extra) {

Log.Write(key + "-запись элемента не поддерживается");

}

/\* Для тестирования \*/

public Ellipse SecretGetPin(int n) => pins[n];

}

}

< IGate.cs>

using Avalonia;

using Avalonia.Controls;

using Avalonia.Controls.Shapes;

using LogicSimulator.Models;

using System.Collections.Generic;

namespace LogicSimulator.Views.Shapes {

public interface IGate {

public int CountIns { get; }

public int CountOuts { get; }

public UserControl GetSelf();

public Point GetPos();

public Size GetSize();

public Size GetBodySize();

public void Move(Point pos, bool global = false);

public void Resize(Size size, bool global = false);

public void ChangeScale(double scale, bool global = false);

public void SavePose();

public Point GetPose();

public Rect GetBounds();

public Distantor GetPin(Ellipse finded);

public Point GetPinPos(int n);

public void AddJoin(JoinedItems join);

public void RemoveJoin(JoinedItems join);

public void ClearJoins();

public void SetJoinColor(int o\_num, bool value);

public bool ContainsJoin(JoinedItems join);

public void Brain(ref bool[] ins, ref bool[] outs);

public int[][] GetPinData();

public void LogicUpdate(Dictionary<IGate, Meta> ids, Meta me);

public int TypeId { get; }

public object Export();

public List<object[]> ExportJoins(Dictionary<IGate, int> to\_num);

public void Import(Dictionary<string, object> dict);

public Ellipse SecretGetPin(int n);

}

}

< LightBulb.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.LightBulb"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#1c1c1c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#3c3c3c" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< LightBulb.axaml.cs>

using Avalonia.Controls;

using Avalonia.Media;

using Avalonia.Threading;

using LogicSimulator.ViewModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class LightBulb: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 7;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

Array.Empty<int>(),

new int[] { 0 },

Array.Empty<int>(),

Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

readonly SolidColorBrush ColorA = new(Color.Parse("#00ff00")); // On

readonly SolidColorBrush ColorB = new(Color.Parse("#1c1c1c")); // Off

public void Brain(ref bool[] ins, ref bool[] outs) {

var value = state = ins[0];

Dispatcher.UIThread.InvokeAsync(() => {

border.Background = value ? ColorA : ColorB;

});

}

/\*

\* Для тестирования

\*/

bool state;

public bool GetState() => state;

/\*

\* Кастомный экспорт и импорт

\*/

public override Dictionary<string, object> ExtraExport() => new() { ["state"] = state };

public override void ExtraImport(string key, object extra) {

if (key != "state") { Log.Write(key + "-запись элемента не поддерживается"); return; }

if (extra is not bool @st) { Log.Write("Неверный тип state-записи элемента: " + extra); return; }

state = @st;

if (state) border.Background = ColorA;

}

}

}

<NAND\_2.axaml >

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.NAND\_2"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">NAND</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

<Ellipse Tag="Body" Margin="{Binding InvertorMargin}" Width="{Binding InvertorSize}" Height="{Binding InvertorSize}" Stroke="Red" StrokeThickness="{Binding InvertorStrokeSize}" Fill="Yellow"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< NAND\_2.axaml.cs>

using Avalonia;

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class NAND\_2: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 8;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Обработка размеров внутренностей

\*/

public double InvertorSize => EllipseSize / 2;

public double InvertorStrokeSize => EllipseStrokeSize / 2;

public Thickness InvertorMargin => new(width + BaseFraction \* 2 - InvertorSize / 2, 0, 0, 0);

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = !(ins[0] && ins[1]);

}

}

<NOT.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.NOT"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">NOT</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< NOT.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class NOT: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 2;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = !ins[0];

}

}

<OR\_2.axaml >

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.OR\_2"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">OR</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< OR\_2.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class OR\_2: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 1;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = ins[0] || ins[1];

}

}

<OR\_8.axaml >

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.OR\_8"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">OR</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< OR\_8.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class OR\_8: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 10;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0, 0, 0, 0, 0, 0, 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = ins[0] || ins[1] || ins[2] || ins[3] || ins[4] || ins[5] || ins[6] || ins[7];

}

}

<Switch.axaml >

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.Switch"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#d32f2e" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#fec107" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}" PointerPressed="Press" PointerReleased="Release">

<Panel>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< Switch.axaml.cs>

using Avalonia;

using Avalonia.Controls;

using Avalonia.Input;

using Avalonia.Media;

using LogicSimulator.Models;

using LogicSimulator.ViewModels;

using System;

using System.Collections.Generic;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class Switch: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 5;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

Array.Empty<int>(),

Array.Empty<int>(),

new int[] { 1 },

Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

bool my\_state = false;

Point? press\_pos;

// Данная схема работает гораздо быстрее, чем событие Tapped ;'-} Из-за того, что не обрабатывается дополнительно DoubleTapped, что гасит второй Tapped + некоторые задержки

private static Point GetPos(PointerEventArgs e) {

if (e.Source is not Control src) return new();

while ((string?) src.Tag != "scene" && src.Parent != null) src = (Control) src.Parent;

return e.GetCurrentPoint(src).Position;

}

private void Press(object? sender, PointerPressedEventArgs e) {

if (e.Source == border) press\_pos = GetPos(e);

}

private void Release(object? sender, PointerReleasedEventArgs e) {

if (e.Source != border) return;

if (press\_pos == null || GetPos(e).Hypot((Point) press\_pos) > 5) return;

press\_pos = null;

my\_state = !my\_state;

border.Background = new SolidColorBrush(Color.Parse(my\_state ? "#7d1414" : "#d32f2e"));

}

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = my\_state;

/\*

\* Кастомный экспорт и импорт

\*/

public override Dictionary<string, object> ExtraExport() => new() { ["state"] = my\_state };

public override void ExtraImport(string key, object extra) {

if (key != "state") { Log.Write(key + "-запись элемента не поддерживается"); return; }

if (extra is not bool @state) { Log.Write("Неверный тип state-записи элемента: " + extra); return; }

my\_state = @state;

if (my\_state) border.Background = new SolidColorBrush(Color.Parse("#7d1414"));

}

/\*

\* Для тестирования

\*/

public void SetState(bool state) {

my\_state = state;

border.Background = new SolidColorBrush(Color.Parse(state ? "#7d1414" : "#d32f2e"));

}

}

}

< XOR\_2.axaml>

<UserControl xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="{Binding UC\_Width}" d:DesignHeight="{Binding UC\_Height}"

Width="{Binding UC\_Width}" Height="{Binding UC\_Height}"

x:Class="LogicSimulator.Views.Shapes.XOR\_2"

Tag="Gate">

<Canvas Tag="Gate">

<Border Tag="Body" Margin="{Binding BodyMargin}" Background="#08c" BorderThickness="{Binding BodyStrokeSize}" BorderBrush="#4cf" Width="{Binding BodyWidth}" Height="{Binding BodyHeight}" CornerRadius="{Binding BodyRadius}">

<Panel>

<TextBlock Tag="Body" FontSize="{Binding FontSizze}" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="White">XOR</TextBlock>

<Image Tag="Resizer" Width="{Binding ImageSize}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="{Binding ImageMargins}" Height="{Binding ImageSize}" Source="avares://LogicSimulator/Assets/Resizer.png"/>

</Panel>

</Border>

</Canvas>

</UserControl>

< XOR\_2.axaml.cs>

using Avalonia.Controls;

using System.ComponentModel;

namespace LogicSimulator.Views.Shapes {

public partial class XOR\_2: GateBase, IGate, INotifyPropertyChanged {

public override int TypeId => 3;

public override UserControl GetSelf() => this;

protected override IGate GetSelfI => this;

protected override int[][] Sides => new int[][] {

System.Array.Empty<int>(),

new int[] { 0, 0 },

new int[] { 1 },

System.Array.Empty<int>()

};

protected override void Init() => InitializeComponent();

/\*

\* Мозги

\*/

public void Brain(ref bool[] ins, ref bool[] outs) => outs[0] = ins[0] ^ ins[1];

}

}

< LauncherWindow.axaml>

<Window xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:vm="using:LogicSimulator.ViewModels"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="500" d:DesignHeight="800"

Width="500" Height="800"

x:Class="LogicSimulator.Views.LauncherWindow"

Icon="/Assets/avalonia-logo.ico"

Title="LogicSimulator"

Padding="8" Background="White">

<Design.DataContext>

<vm:LauncherWindowViewModel/>

</Design.DataContext>

<Window.Styles>

<Style Selector="ListBoxItem">

<Setter Property="Padding" Value="0"/>

<Setter Property="Margin" Value="0 0 0 10"/>

</Style>

<Style Selector="Button">

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="BorderBrush" Value="#75A29F"/>

<Setter Property="Background" Value="#B2DBD3"/>

<Setter Property="CornerRadius" Value="3"/>

<Setter Property="Padding" Value="12"/>

<Setter Property="FontSize" Value="24"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

</Style>

<Style Selector="Border.b">

<Setter Property="BorderThickness" Value="4"/>

<Setter Property="BorderBrush" Value="#75A29F"/>

<Setter Property="CornerRadius" Value="3"/>

<Setter Property="Padding" Value="10"/>

</Style>

<Style Selector="TextBlock.tb">

<Setter Property="Margin" Value="5"/>

<Setter Property="Padding" Value="4"/>

<Setter Property="FontSize" Value="32"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

</Style>

</Window.Styles>

<Grid RowDefinitions="auto,auto,auto,\*,auto">

<Button Command="{Binding Create}">Создать новый проект</Button>

<Button Grid.Row="1" Command="{Binding Open}">Открыть проект</Button>

<TextBlock Grid.Row="2" Classes="tb">Перечень проектов:</TextBlock>

<ListBox Grid.Row="3" Items="{Binding ProjectList}" DoubleTapped="DTapped" Padding="10">

<ListBox.ItemTemplate>

<DataTemplate>

<Border Classes="b">

<TextBlock Text="{Binding}" Tag="{Binding}"/>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Button Grid.Row="4" Command="{Binding Exit}">Выход</Button>

</Grid>

</Window>

< LauncherWindow.axaml.cs>

using Avalonia.Controls;

using LogicSimulator.ViewModels;

namespace LogicSimulator.Views {

public partial class LauncherWindow: Window {

readonly LauncherWindowViewModel lwvm;

public LauncherWindow() {

InitializeComponent();

lwvm = new LauncherWindowViewModel();

DataContext = lwvm;

lwvm.AddWindow(this);

}

public void DTapped(object? sender, Avalonia.Interactivity.RoutedEventArgs e) {

lwvm.DTapped(sender, e);

}

}

}

< MainWindow.axaml>

<Window xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:vm="using:LogicSimulator.ViewModels"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="1400" d:DesignHeight="800"

Width="1400" Height="800"

x:Class="LogicSimulator.Views.MainWindow"

Icon="/Assets/avalonia-logo.ico"

Title="LogicSimulator">

<Design.DataContext>

<vm:MainWindowViewModel/>

</Design.DataContext>

<Window.Styles>

<Style Selector="ListBoxItem">

<Setter Property="Padding" Value="0"/>

</Style>

<Style Selector="Border.b">

<Setter Property="BorderThickness" Value="2"/>

<Setter Property="BorderBrush" Value="#0d8"/>

<Setter Property="CornerRadius" Value="5"/>

<Setter Property="Padding" Value="6"/>

</Style>

<Style Selector="TextBox">

<Setter Property="Margin" Value="-5"/>

<Setter Property="Padding" Value="4"/>

<Setter Property="MinHeight" Value="0"/>

</Style>

<StyleInclude Source="/Styles/Animation.axaml"/>

</Window.Styles>

<DockPanel>

<Menu DockPanel.Dock="Top" ZIndex="10" Background="White">

<MenuItem Header="Файл">

<MenuItem Header="Создать новый проект" Command="{Binding Comm}" CommandParameter="Create"/>

<MenuItem Header="Открыть проект" Command="{Binding Comm}" CommandParameter="Open"/>

<MenuItem Header="Сохранить текущую схему" Command="{Binding Comm}" CommandParameter="Save" IsEnabled="{Binding CanSave}"/>

<MenuItem Header="Сохранить проект как..." Command="{Binding Comm}" CommandParameter="SaveAs"/>

<MenuItem Header="Выйти в лаунчер" Command="{Binding Comm}" CommandParameter="ExitToLauncher"/>

<MenuItem Header="Выйти" Command="{Binding Comm}" CommandParameter="Exit"/>

</MenuItem>

<MenuItem Header="Опции">

<CheckBox IsChecked="{Binding LockSelfConnect}">Блокировать самоподключение</CheckBox>

</MenuItem>

</Menu>

<Grid ColumnDefinitions="\*,3\*,\*">

<Panel Grid.Column="1">

<!--

<TextBlock Text="{Binding Logg}" Background="AliceBlue"/>-->

<Canvas Tag="Scene" Name="Canvas" Background="#0000"/>

</Panel>

<ListBox Items="{Binding ItemTypes}" SelectedIndex="{Binding SelectedItem}" Name="Gates" VirtualizationMode="None">

<ListBox.ItemTemplate>

<DataTemplate>

<ContentControl Content="{Binding}"/>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Grid Grid.Column="2" RowDefinitions="auto,\*" DoubleTapped="DTapped">

<Border Classes="b" BorderBrush="#4df" Background="White">

<Grid ColumnDefinitions="\*,auto">

<TextBlock Tag="p\_name" Text="{Binding ProjName}"/>

<Button Grid.Column="1" Padding="2" Margin="6 -4 0 -4" Command="{Binding NewItem}">

<Image Width="24" Height="24" Source="/Assets/NewItem.png"/>

</Button>

</Grid>

</Border>

<ListBox Grid.Row="1" Padding="10 4 0 0" Items="{Binding Schemes}" VirtualizationMode="None">

<ListBox.ItemTemplate>

<DataTemplate>

<Border Classes="b">

<Grid ColumnDefinitions="\*,auto,auto,auto">

<TextBlock Text="{Binding}" Tag="{Binding}"/>

<Button Grid.Column="1" Padding="2" Margin="6 -4 4 -4" Command="{Binding Open}" IsVisible="{Binding CanOpenMe}" Name="OpenScheme">

<Image Width="24" Height="24" Source="/Assets/Open.png"/>

</Button>

<Button Grid.Column="2" Padding="2" Margin="0 -4 4 -4" Command="{Binding NewItem}" Name="NewScheme">

<Image Width="24" Height="24" Source="/Assets/NewItem.png"/>

</Button>

<Button Grid.Column="3" Padding="2" Margin="0 -4 0 -4" Command="{Binding Delete}" IsVisible="{Binding CanUseSchemeDeleter}">

<Image Width="24" Height="24" Source="/Assets/TrashCan.png"/>

</Button>

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Grid>

</Grid>

</DockPanel>

</Window>

< MainWindow.axaml.cs>

using Avalonia.Controls;

using LogicSimulator.ViewModels;

namespace LogicSimulator.Views {

public partial class MainWindow: Window {

readonly MainWindowViewModel mwvm;

public MainWindow() {

InitializeComponent();

mwvm = new MainWindowViewModel();

DataContext = mwvm;

mwvm.AddWindow(this);

}

public void DTapped(object? sender, Avalonia.Interactivity.RoutedEventArgs e) {

mwvm.DTapped(sender, e);

}

public void Update() => mwvm.Update();

}

}

< App.axaml>

<Application xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:LogicSimulator"

x:Class="LogicSimulator.App">

<Application.DataTemplates>

<local:ViewLocator/>

</Application.DataTemplates>

<Application.Styles>

<FluentTheme Mode="Light"/>

</Application.Styles>

</Application>

< App.axaml.cs>

using Avalonia;

using Avalonia.Controls.ApplicationLifetimes;

using Avalonia.Markup.Xaml;

using LogicSimulator.Views;

using System.IO;

namespace LogicSimulator {

public partial class App: Application {

public override void Initialize() {

AvaloniaXamlLoader.Load(this);

}

public override void OnFrameworkInitializationCompleted() {

if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime desktop)

desktop.MainWindow = new LauncherWindow();

base.OnFrameworkInitializationCompleted();

IncrementBuildNum();

}

private static void IncrementBuildNum() {

if (lock\_inc\_build) return;

string path = "../../../../build.num";

int num;

try { num = int.Parse(File.ReadAllText(path)); }

catch (FileNotFoundException) { num = 0; }

num++;

File.WriteAllText(path, num.ToString());

}

/\*

\* Для тестирования

\*/

public static bool lock\_inc\_build = false;

}

}

< app.manifest>

<?xml version="1.0" encoding="utf-8"?>

<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">

<!-- This manifest is used on Windows only.

Don't remove it as it might cause problems with window transparency and embeded controls.

For more details visit https://learn.microsoft.com/en-us/windows/win32/sbscs/application-manifests -->

<assemblyIdentity version="1.0.0.0" name="AvaloniaTest.Desktop"/>

<compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">

<application>

<!-- A list of the Windows versions that this application has been tested on

and is designed to work with. Uncomment the appropriate elements

and Windows will automatically select the most compatible environment. -->

<!-- Windows 10 -->

<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}" />

</application>

</compatibility>

</assembly>

< for\_test.json>

{"name": "Для тестирования #4", "created": 1683857166, "modified": 1683994865, "items": [{"id": 5, "pos": "$p$81,313", "size": "$s$54,53", "base\_size": 15.523033076478937, "state": false}, {"id": 5, "pos": "$p$79,367", "size": "$s$51,51", "base\_size": 15.523033076478937, "state": false}, {"id": 5, "pos": "$p$75,424", "size": "$s$54,53", "base\_size": 15.523033076478937, "state": false}, {"id": 5, "pos": "$p$81,478", "size": "$s$54,53", "base\_size": 15.523033076478937, "state": false}, {"id": 5, "pos": "$p$82,533", "size": "$s$54,53", "base\_size": 15.523033076478937, "state": false}, {"id": 2, "pos": "$p$160,279", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$249,279", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$252,312", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 2, "pos": "$p$187,320", "size": "$s$18,18", "base\_size": 5.440728394753729}, {"id": 0, "pos": "$p$248,342", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$246,373", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$375,281", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$375,310", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$376,341", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$376,373", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$375,403", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$375,436", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$507,276", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$506,306", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$504,334", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$505,364", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$504,394", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$504,426", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$502,457", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$502,487", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$645,276", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$646,306", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$647,336", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$646,367", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$645,397", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$647,425", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$647,455", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$646,486", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$646,516", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 0, "pos": "$p$646,547", "size": "$s$30,30", "base\_size": 8.762347487034825}, {"id": 1, "pos": "$p$284,326", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$411,325", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$417,388", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$544,319", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$545,379", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$544,442", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$690,319", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$692,381", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$694,441", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 1, "pos": "$p$694,500", "size": "$s$22,22", "base\_size": 6.58328135765202}, {"id": 2, "pos": "$p$305,400", "size": "$s$18,18", "base\_size": 5.440728394753729}, {"id": 2, "pos": "$p$423,466", "size": "$s$18,18", "base\_size": 5.440728394753729}, {"id": 2, "pos": "$p$573,524", "size": "$s$18,18", "base\_size": 5.440728394753729}, {"id": 1, "pos": "$p$305,418", "size": "$s$18,18", "base\_size": 5.440728394753729}, {"id": 1, "pos": "$p$423,486", "size": "$s$18,18", "base\_size": 5.440728394753729}, {"id": 1, "pos": "$p$574,545", "size": "$s$18,18", "base\_size": 5.440728394753721}, {"id": 7, "pos": "$p$766,260", "size": "$s$54,53", "base\_size": 15.523033076478878, "state": true}, {"id": 7, "pos": "$p$766,314", "size": "$s$54,53", "base\_size": 15.523033076478878, "state": false}, {"id": 7, "pos": "$p$765,367", "size": "$s$54,53", "base\_size": 15.523033076478878, "state": false}, {"id": 7, "pos": "$p$765,421", "size": "$s$54,53", "base\_size": 15.523033076478878, "state": false}, {"id": 7, "pos": "$p$769,475", "size": "$s$54,53", "base\_size": 15.523033076478878, "state": false}, {"id": 7, "pos": "$p$773,529", "size": "$s$54,53", "base\_size": 15.523033076478878, "state": false}], "joins": [[0, 0, "Out", 5, 0, "In"], [0, 0, "Out", 9, 0, "In"], [0, 0, "Out", 10, 0, "In"], [1, 0, "Out", 8, 0, "In"], [1, 0, "Out", 7, 1, "In"], [1, 0, "Out", 10, 1, "In"], [2, 0, "Out", 48, 1, "In"], [3, 0, "Out", 49, 1, "In"], [4, 0, "Out", 50, 1, "In"], [5, 1, "Out", 6, 0, "In"], [5, 1, "Out", 7, 0, "In"], [6, 2, "Out", 11, 0, "In"], [6, 2, "Out", 12, 0, "In"], [35, 0, "In", 7, 2, "Out"], [8, 1, "Out", 6, 1, "In"], [8, 1, "Out", 9, 1, "In"], [35, 1, "In", 9, 2, "Out"], [10, 2, "Out", 15, 0, "In"], [10, 2, "Out", 16, 0, "In"], [11, 2, "Out", 17, 0, "In"], [11, 2, "Out", 18, 0, "In"], [36, 0, "In", 12, 2, "Out"], [13, 2, "Out", 36, 1, "In"], [37, 0, "In", 14, 2, "Out"], [15, 2, "Out", 37, 1, "In"], [16, 2, "Out", 23, 0, "In"], [16, 2, "Out", 24, 0, "In"], [17, 2, "Out", 25, 0, "In"], [26, 0, "In", 17, 2, "Out"], [18, 2, "Out", 38, 0, "In"], [38, 1, "In", 19, 2, "Out"], [20, 2, "Out", 39, 0, "In"], [39, 1, "In", 21, 2, "Out"], [22, 2, "Out", 40, 0, "In"], [40, 1, "In", 23, 2, "Out"], [33, 0, "In", 24, 2, "Out"], [24, 2, "Out", 34, 0, "In"], [25, 2, "Out", 51, 0, "In"], [41, 0, "In", 26, 2, "Out"], [27, 2, "Out", 41, 1, "In"], [42, 0, "In", 28, 2, "Out"], [29, 2, "Out", 42, 1, "In"], [43, 0, "In", 30, 2, "Out"], [31, 2, "Out", 43, 1, "In"], [44, 0, "In", 32, 2, "Out"], [33, 2, "Out", 44, 1, "In"], [56, 0, "In", 34, 2, "Out"], [35, 2, "Out", 13, 0, "In"], [35, 2, "Out", 14, 0, "In"], [36, 2, "Out", 19, 0, "In"], [36, 2, "Out", 20, 0, "In"], [37, 2, "Out", 21, 0, "In"], [37, 2, "Out", 22, 0, "In"], [27, 0, "In", 38, 2, "Out"], [38, 2, "Out", 28, 0, "In"], [29, 0, "In", 39, 2, "Out"], [39, 2, "Out", 30, 0, "In"], [31, 0, "In", 40, 2, "Out"], [40, 2, "Out", 32, 0, "In"], [52, 0, "In", 41, 2, "Out"], [42, 2, "Out", 53, 0, "In"], [54, 0, "In", 43, 2, "Out"], [44, 2, "Out", 55, 0, "In"], [45, 1, "Out", 11, 1, "In"], [13, 1, "In", 45, 1, "Out"], [15, 1, "In", 45, 1, "Out"], [46, 1, "Out", 17, 1, "In"], [19, 1, "In", 46, 1, "Out"], [21, 1, "In", 46, 1, "Out"], [23, 1, "In", 46, 1, "Out"], [47, 1, "Out", 25, 1, "In"], [27, 1, "In", 47, 1, "Out"], [29, 1, "In", 47, 1, "Out"], [31, 1, "In", 47, 1, "Out"], [33, 1, "In", 47, 1, "Out"], [48, 2, "Out", 45, 0, "In"], [48, 2, "Out", 16, 1, "In"], [14, 1, "In", 48, 2, "Out"], [12, 1, "In", 48, 2, "Out"], [46, 0, "In", 49, 2, "Out"], [49, 2, "Out", 24, 1, "In"], [49, 2, "Out", 22, 1, "In"], [20, 1, "In", 49, 2, "Out"], [18, 1, "In", 49, 2, "Out"], [50, 2, "Out", 47, 0, "In"], [50, 2, "Out", 34, 1, "In"], [32, 1, "In", 50, 2, "Out"], [30, 1, "In", 50, 2, "Out"], [28, 1, "In", 50, 2, "Out"], [26, 1, "In", 50, 2, "Out"]], "states": "0000001101001000001000000010000000000000000000111000"}

< for\_test.yaml>

---

name: Newy

created: 1684584469

modified: 1684584494

items:

-

id: 0

pos: $p$352,172

size: $s$71,71

base\_size: 25

joins: []

states: "00"

< LogicSimulator.csproj>

using Avalonia;

using LogicSimulator.Views.Shapes;

namespace LogicSimulator.Models {

public class Distantor {

public readonly int num;

public IGate parent;

public readonly string tag;

public Distantor(IGate parent, int n, string tag) {

this.parent = parent;

num = n;

this.tag = tag;

}

public Point GetPos() => parent.GetPinPos(num);

}

}

< Program.cs>

using Avalonia;

using Avalonia.ReactiveUI;

using System;

namespace LogicSimulator {

public class Program {

// Initialization code. Don't use any Avalonia, third-party APIs or any

// SynchronizationContext-reliant code before AppMain is called: things aren't initialized

// yet and stuff might break.

[STAThread]

public static void Main(string[] args) => BuildAvaloniaApp()

.StartWithClassicDesktopLifetime(args);

// Avalonia configuration, don't remove; also used by visual designer.

public static AppBuilder BuildAvaloniaApp()

=> AppBuilder.Configure<App>()

.UsePlatformDetect()

.LogToTrace()

.UseReactiveUI();

}

}

< Roots.xml>

<linker>

<!-- Can be removed if CompiledBinding and no reflection are used -->

<assembly fullname="LogicSimulator" preserve="All" />

<assembly fullname="Avalonia.Themes.Fluent" preserve="All" />

</linker>

< ViewLocator.cs>

using Avalonia.Controls;

using Avalonia.Controls.Templates;

using LogicSimulator.ViewModels;

using System;

namespace LogicSimulator {

public class ViewLocator: IDataTemplate {

public IControl Build(object data) {

var name = data.GetType().FullName!.Replace("ViewModel", "View");

var type = Type.GetType(name);

if (type != null) {

return (Control) Activator.CreateInstance(type)!;

}

return new TextBlock { Text = "Not Found: " + name };

}

public bool Match(object data) {

return data is ViewModelBase;

}

}

}