Что нам стоит /GUI\ построить?
Отрисуем — будем жить.
От общего ландшафта к практике с Tcl::Tk

Вадим Коновалов

CPAN ID: VKON

Perl-Conf.Ru/26

24 октября 2026г.

План доклада

- Общий обзор имеющихся технологий GUI
- В конкретике - Perl и tcl/tk

Поехали

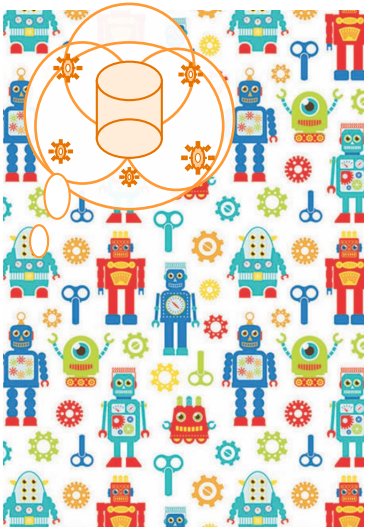
Часть 1

Общий обзор

Макрокосм ПО

Мир современного ПО разнообразен:

- **CLI:** hello-world, скрипты автоматизации, утилиты, zip, конвертеры, …
- **TUI:** mc, nc, редакторы, утилиты.
- **GUI:** редакторы (текстовые, графич), игры.
- **Web:** браузерные решения.
- **Backend:** демоны, СУБД, SaaS, Шмаас, фигас, прочие докеры с кубернетесами.



Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python , Ruby
- JS (nodejs)
- Haskell
- C/C++

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python, Ruby
- JS (nodejs)
- Haskell
- C/C++

Интересный «перехлёст» — GUI-библиотека считает себя полноценным ЯП (а это так и есть), что делает гибкость GUI просто невероятной

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python, Ruby
- JS (nodejs)
- Haskell
- C/C++

PyQt, PySide

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python, Ruby
- JS (nodejs)
- Haskell
- C/C++

биндинги есть, но 1. очень старые и 2. очень странные (pipe). ИИшка сказала что она лучше напрямую через foreign call накидает код, и - вуаля - заработало со второго раза! (1.бинд на tcl 2.бинд на tk)

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python , Ruby
- JS (nodejs)
- Haskell
- C/C++

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой ком-пьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python , Ruby
- JS (nodejs)
- Haskell
- C/C++

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python , Ruby
- JS (nodejs)
- Haskell
- C/C++

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python , Ruby
- JS (nodejs)
- Haskell
- C/C++

Рассматриваемые аспекты и библиотеки

аспекты

- внешний вид
- фичи
- простота кода
- кроссплатф-сть
- переносимость на другой компьютер

библиотеки

- Tcl/Tk
- Wx (wxWidgets)
- Qt
- GTK
- FLTK
- HTML5/Web
- Java (Swing)
- Java (JavaFX)
- Нативный (Win32::GUI, ...)

языки

- Tcl/Tk
- Perl, Python , Ruby
- JS (nodejs)
- Haskell
- C/C++

Сравнение GUI-библиотек

Библиотеки	Внешний вид	Фичи	Простота кода	Кросс-платф.	Переноси-мость	Размер рантайма, Мб (софт)
Tcl/Tk	■■□□□ *	■■■■■ +	■■■■■	■■■■■	■■■■■	∼5
Wx (wxWidgets)	■■■□□	■■■■□	■■■□□	■■■■□	■■■□□	∼10
Qt	■■■■□	■■■■■	■■□□□	■■■■■	■■■■□	∼30–50
GTK	■■■□□	■■■■□	■■□□□	■■■□□	■■■□□	∼20
FLTK	■□□□□	■■□□□	■■■■□	■■■■□	■■■■■	<2
HTML5/Web	■■■■■	■■■■□	■■■□□	■■■■■	■■■■□	>120 (Electron)
Java (Swing)	■■□□□	■■■□□	■■■□□	■■■■■	■■■■■	∼50 (JRE)
Java (JavaFX)	■■■■□	■■■■□	■■■□□	■■■■■	■■■■■	∼80
ImGui	■■■□□	■■■■■	■■■■□	■■■■■	■■■■□	<1 (Инжект.)
Нативный	■■■■■	■■■□□	■■□□□	□□□□□	■□□□□	0 (Встроен)

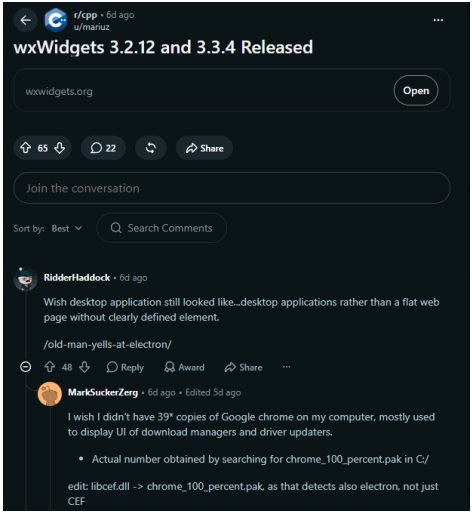
* С использованием пакета `ttk::*` (которому уже под 15 лет) имеем твёрдую ■■■■■ и нативный системный look-and-feel.

HTML5-GUI: Сравнительный анализ актуальных фреймворков

Фреймворк	Бэкенд-язык	Размер ф.	RAM	Архитектура и Рендеринг
Electron	JS / Node.js	∼100-150МБ	150+МБ	2+пр, свой Chromium
Tauri (v2)	Rust + JS	∼3–8 МБ	∼30-50МБ	2пр, системный WebView
Wails	Go + JS	∼5–15 МБ	∼30-50МБ	2пр, системный WebView
Neutralino.js	C++ / чистый JS	<5МБ	Минимальное	2пр, Localhost + WebView
NW.js	JS / Node.js	∼100-150МБ	100+МБ	1пр (единый конт.), Chromium
Sciter	C++/свой JS	∼5-8МБ	Мин(<30МБ)	1пр, свой легкий движок

Резюме: Для тяжелого Enterprise по-прежнему лидирует Electron. Для легковесных современных приложений стандартом стал Tauri. В специфических соло-задачах монолитный NW.js или ультра-легкий Sciter предлагают удобную однопроцессную архитектуру.

Боль комьюнити: Эра Web-GUI на десктопе



Суровая реальность HTML5/Web

Каждое «простое» приложение тащит с собой полноценный тяжелый движок Chromium.

К чему это приводит:

- **Десятки копий Chrome** на одном ПК ради интерфейса простых утилит.
- Поиск по `chrome_100_percent.pak` в `C:/` открывает глаза на масштаб.
- Размытие границ: вместо десктоп-приложений мы получаем плоские веб-страницы.

Нативные виджеты

Выбирая GUI-стек, мы выбираем одну из двух философий общения с ОС:

Лагерь А: Враперы (Нативные)

- Каждая кнопка — честный объект ОС.
- На Windows имеет собственный `HWND`.
- Полная системная интеграция.
- *Примеры:* **Win32::GUI**, **Wx**.

плюсы: можно дебажить виджеты средствами ОС
минусы: мало фич

Лагерь Б: Отрисовщики (Эмуляция)

- Окно приложения — пустое поле.
- Виджеты рисуются кодом фреймворка.
- ОС «не видит» кнопок внутри окна.
- *Примеры:* **Qt**, **GTK**, **FLTK**, **Prima**.

Ландшафт современных GUI-технологий

Фреймворк / Платформа	Подход под капотом	Поддержка BiDi (RTL)
Tcl/Tk	Эмуляция (Темы ttk)	–
Wx (wxWidgets)	Нативный (`HWND`)	**+** (Через ОС)
Qt	Эмуляция (Свой движок)	**+**
GTK	Эмуляция (Свой движок)	**+**
FLTK	Эмуляция (Кастомная)	–
HTML5 / Web[1]	Эмуляция (Браузерный движок)	**++** (Эталонная)
Java (Swing)	Эмуляция (Свой движок)	**+**
Java (JavaFX)	Эмуляция (Свой движок)	**+**
Win32::GUI	Нативный (`HWND`)	**+** (Через ОС)

[1] Включает Electron, CEF, NW.js, Tauri, WebView2, UltraLight, Sciter.

Дилемма деплоя: Кому и как мы отдаем утилиту?

Когда интерфейс написан, возникает главный вопрос: **как программа полетит на чужой компьютер?** Здесь сценарии разработки жестко разделяются на три ветви:

1. **«Для себя и серверов» (Полный контроль)**
   - Запуск на разных ОС, где мы сами хозяева окружения. Нам не трудно поставить рантаймы, настроить зависимости или доустановить библиотеки.
2. **«Для коллеги» (Монолитный EXE)**
   - Нужно отдать один файл человеку, у которого нет ни интерпретаторов, ни библиотек. Всё должно запуститься «из коробки» кликом мыши.
3. **«Гибридный путь» (Скрипты в архиве)**
   - Подход в духе Python-экосистемы: тяжелое ядро или интерпретатор поставляется отдельно, а логика приложения упакована в динамический `.zip`.

Языки и GUI: Цена независимости исполняемого файла

Как разные языковые стеки справляются с GUI при создании отчуждаемых утилит:

- **C++ / Go:** На выходе настоящий, чистый бинарник.
- **Python (PyQt / PySide / Tkinter):** Упаковка через `PyInstaller`.
- **HTML5 (Electron, NW.js, …):** Огромный exe-шник, через сокеты
- **Tcl / Perl:** Tcl хорошо приспособлен упаковываться в монолитный exe или 2 файла - технологии вроде `Tcl starkit/starpack`. В эру ActivePerl tcl/tk либо входил либо мог быть добавлен одной командой. Сейчас - в зависимости от продолжительности танцев с бубном - можно сделать аналогично, но не проверялось.

Perl и tcl/tk

Часть 2

Perl и tcl/tk

Что есть для GUI в Perl?

- **Prima** — родная, активно развивающаяся перловая библиотека. Великолепна, но свой мир.
- **Wx / Gtk3 / Qt** — биндинги к тяжелым монстрам. Иногда сложно собирать (CPAN-hell при компиляции XS).
- **Tcl/Tk** — живая классика, на которой написано пол-CPAN'а и миллион внутренних enterprise-утилит.

Экосистема Tk: Как разные языки работают с движком?

Язык	Основной модуль / Библиотека	Принцип интеграции
Tcl	Родной язык движка (Tk)	Нативная интеграция
Python	`tkinter`	Обертка над встроенным Tcl
Perl	Perl/Tk Tcl::pTk Tcl::Tk Tkx Tcl	Зависит от выбранного модуля
C++	`cpptk` / Родной C API	Компиляция через C-биндинги
Go	`go-tk`	Обертка через CGo
Node.js	`node-tk` / `tcltk`	Бинарные биндинги
Haskell	`HTk`	Порт через потоки ввода-вывода

Важное примечание: `Perl/Tk` (*Tk.pm*) — это **единственный в мире** уникальный случай, когда весь C-код графической библиотеки Tk был полностью вырезан из Tcl и физически вшит внутрь Perl-модуля как XS-расширение.

Экосистема Tk: Как разные языки работают с движком?

Язык	Основной модуль / Библиотека	Принцип интеграции
Tcl	Родной язык движка (Tk)	Нативная интеграция
Python	`tkinter`	Обертка над встроенным Tcl
Perl	Perl/Tk Tcl::pTk Tcl::Tk Tkx Tcl	Зависит от выбранного модуля
C++	`cpptk` / Родной C API	Компиляция через C-биндинги
Go	`go-tk`	Обертка через CGo
Node.js	`node-tk` / `tcltk`	Бинарные биндинги
Haskell	`HTk`	Порт через потоки ввода-вывода

Важное примечание: `Perl/Tk` (*Tk.pm*) — это **единственный в мире** уникальный случай, когда весь C-код графической библиотеки Tk был полностью вырезан из Tcl и физически вшит внутрь Perl-модуля как XS-расширение.

Использование Tk из Perl - 5 способов

Модуль	Виджеты Tcl/Tk	Мегавиджеты	Синтаксис	Обновление Tk
Tcl.pm	Полное (все виджеты)	Tcl/Tk	Tcl/Tk	Автоматически
Tkx	Полное (все виджеты)	свои, Tcl/Tk	Tcl/Tk, свой	Автоматически
Tcl::Tk	Полное (все виджеты)	Tcl/Tk	Tcl/Tk, Perl/Tk	Автоматически
Tcl::pTk	Полное (все виджеты)	Tcl/Tk	Tcl/Tk, Perl/Tk	Автоматически
Perl/Tk	Ограниченное (без `ttk`)	свои	Родной классический	Заморожено

синтаксис

Устаревший Perl/Tk

```
my $mw = MainWindow->new;

# Старая плоская кнопка
my $btn = $mw->Button(
    -text    => "Жми",
    -command => \&do_something
)->pack;
```

Итог: Интерфейс в стиле Windows 95. Системные темы оформления не поддерживаются в принципе.

Современный Tcl::Tk

```
my $int = Tcl::Tk->new;

# Новая нативная кнопка ttk
my $btn = $int->Widget(
    "ttk::button",
    -text    => "Жми",
    -command => \&do_something
)->pack;
```

Итог: Нативный виджет ОС (Windows 10/11, macOS, GTK). Полная поддержка современных стилей.

Под капотом Tcl

Как устроен модуль `Tcl` изнутри.

- Tcl.xs
- Tcl.pm

Под капотом Tcl::Tk: Динамическая магия AUTOLOAD

Вместо тысяч строк хардкода под сотни виджетов — элегантная архитектура «ленивой генерации» (Lazy Compilation).

Как работает механизм `AUTOLOAD`:

- **Старт:** Классов и методов для виджетов изначально **нет** в памяти Perl.
- **Первый вызов:** При обращении к `$mw->TtkButton` срабатывает `AUTOLOAD`.
- **Генерация:** Модуль на лету компилирует метод и кеширует его в таблицу символов.
- **Повторный вызов:** Идёт напрямую со скоростью родной функции Perl.

Главный инженерный профит:

- **Мгновенный запуск:** Нет траты времени на парсинг сотен неиспользуемых функций.
- **Минимум памяти:** Потребляются ресурсы только под реально отрисованные виджеты.

3 случая

- **Календари** и ИИ реал**ии**
- **mied** MDI-редактор, накидаем туда фич
- **tcled** Консольный редактор, накидаем туда фич

Случай 1: Календари и ИИ реалии (Tcl/Tk)

- Календарь с `Tcl::Tk`, синтаксис tcl/tk

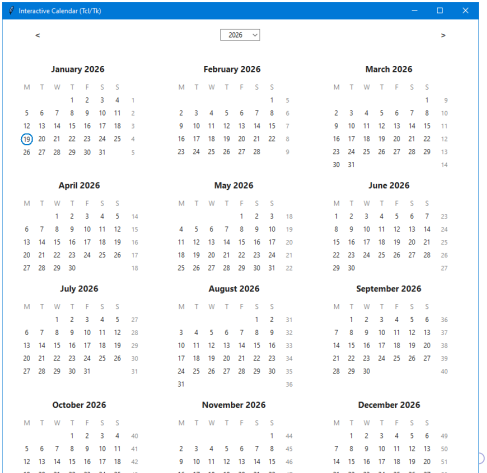
```
use strict;
use Tcl::Tk;

my $mw = Tcl::Tk::MainWindow->new();
my $int = $mw->interp;

$int->Eval(<<'EOS');
wm title . "Interactive␣Calendar␣(Tcl/Tk)"
wm geometry . "850x880"
. configure -bg "#ffffff"

# Active viewing year variable
set view_year 2026
...
draw_calendar $view_year
EOS

Tcl::Tk::MainLoop;
```



Случай 1: Календари и ИИ реалии (Perl/Tk)

- Календарь с `Tcl::Tk`, синтаксис perl/tk

```
use strict;
use Tcl::Tk;

my $mw = Tcl::Tk::MainWindow->new();
$mw->title("Calendar␣Demo␣(Tcl::Tk)");
$mw->geometry("850x880");
$mw->configure(-bg => "#ffffff");
my $int = $mw->interp;
...
# --- Header Section (Navigation controls) ---
my $header = $mw->Frame(-bg => "#ffffff", -pady => 10)->pack(-fill => 'x');

# Step backward button
my $btn_prev = $header->Button(
    -text => "<", -font => $font_title, -fg => $color_dark, -bg => "#ffffff",
    -activebackground => "#f0f0f0", -relief => "flat", -overrelief => "raised"
        ,
    -command => sub { $view_year_var--; draw_calendar($view_year_var) }
)->pack(-side => 'left', -padx => [50, 10]);
...
Tcl::Tk::MainLoop();
```

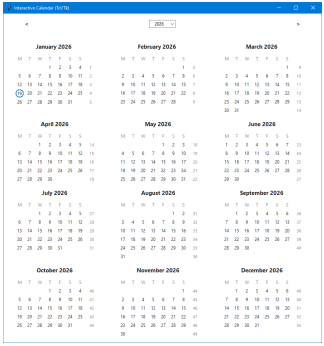


Рис.: Календарь

Случай 2: mied MDI-редактор — Запуск

В качестве основы взят готовый дистракшн-фри редактор:
`KikyTokamuro/mied` на `github.com`

Шаг 1: Чистый запуск Tcl-скрипта из-под Perl

```
use strict;
use Tcl::Tk;
my $int = Tcl::Tk->new();
$int->Eval('source␣mied.tcl');
Tcl::Tk::MainLoop;
```

- Окна редактора создаются как свободные MDI-буферы, которые можно перемещать и масштабировать.

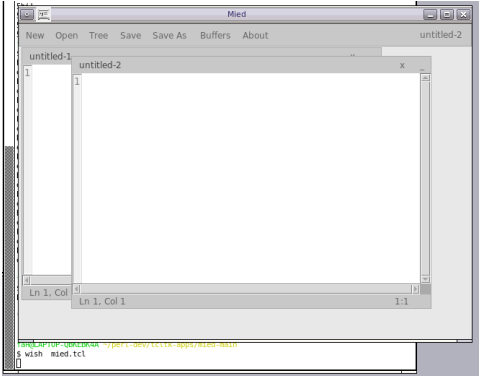


Рис.: Исходный интерфейс mied

Случай 2: mied — Внедрение плагинов

Шаг 3: Выделение ключевых компонентов

```
use strict;
use Tcl::Tk;
my $int = Tcl::Tk->new();
$int->Eval('source␣mied.tcl');

# on TCL: proc MakeToolbarButton {name width text cmd} {...}
$int->call('MakeToolbarButton', 'perl', 40, "Perl", sub {
   print "hi!\neval␣is␣" . eval($int->Eval('$Buffers(
         $ActiveBufferId,window).content.ctext␣get␣1.0␣end-1c')) .
         "\n";
});

Tcl::Tk::MainLoop;
```

- **Прекрасное внедрение:** Perl-коллбэк успешно интегрирован в Tcl/Tk GUI.
- **Интерактивный Eval:** Кнопка считывает текст из Tcl-виджета `ctext`, вычисляет его в Perl и выводит результат в консоль.

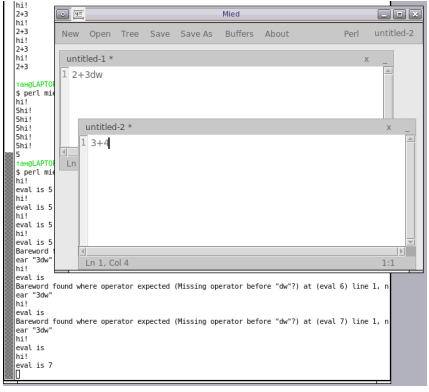


Рис.: Выполнение Perl-кода по кнопке

Консольный GUI: Связка Perl, Tcl и редактора tcled

Если нам нужно запустить инструмент строго в терминале (без X-сервера/Win32), Tcl снова приходит на помощь через концепцию **TUI (Text User Interface)**.

Кейс: Текстовый редактор tcled

tcled — это полноценный консольный текстовый редактор, написанный на чистом Tcl.

Как это работает в связке с Perl?

1. Perl выступает в роли «мозга» приложения (парсит логику, ходит в сеть/БД).
2. Через низкоуровневый мост Perl вызывает Tcl-скрипт редактора `tcled`.
3. Пользователь прямо внутри консоли получает интерактивные окна, меню и текстовые поля.

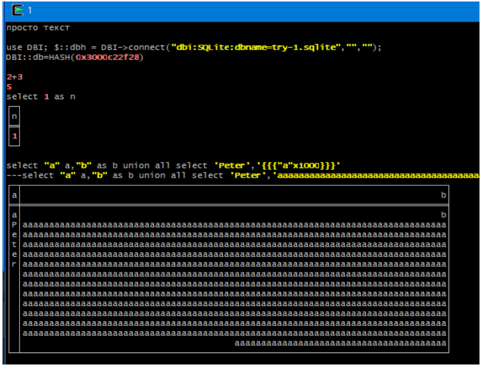
Случай 3: SQL-клиент на базе tcled и vispen

Ингредиенты:

- **tcled** `github.com/slebetman/tcled`
- **vispen** `github.com/vadrerko/vispen`
- **Tcl** со CPAN-а

перемешиваем до готовности

```
use utf8;
use strict;
use Tcl;
my $int = Tcl->new();
$int->Init();
...
sub vispen_execute_here {
    ...
}
# Register command within the Tcl interpreter space.
$int->CreateCommand('vispen_trigger', \&vispen_execute_here)
    ;
...
$int->Eval('source␣tcled');
```



Эволюция генерации кода: От Visual Tcl к ИИ

Как вписать громоздкий, чистый синтаксис Tcl/Tk в Perl-скрипт?

- **Вчера: Среды разработки WYSIWYG (например, vTcl)**
  - *Visual Tcl* позволял накликать интерфейс мышкой и экспортировать Tcl-код.
  - *Минусы:* Тяжелая поддержка, избыточный сгенерированный код, проект застыл в прошлом.
- **Сегодня: Генерация логики с помощью ИИ (Нейросети)**
  - Достаточно скормить LLM-модели текстовое описание (промпт) нужного интерфейса.
  - Нейросеть мгновенно выдает чистый, оптимизированный Tcl/Tk код.
  - Мы просто вставляем его в Perl через блок `$int->Eval(<<'EOS')`, как в демо-календаре.

Выводы

- tcl/tk - идеальный GUI toolkit в своей нише (дебагер (tkdb), disasm, словарик, do-png)
  - Шаг вправо - шаг влево - и ”идеальной” будет другая комбинация
- идеального инструмента не существует
- Tcl::Tk - идеальная подвязка tcl/tk для Perl-а