Software necromancy with Perl

Dave Lambley
dlambley@cpan.org

Accounting software

Informix C-ISAM

1987

- Perl 1.0
- GCC 1.0

TPS report example

```
50
  set 0 = tps_report_id
  readnext tps_report_id

100
  on io_code ne 0 goto 999

  on tps_status eq 1 : tps_num + 1 * 3 = tps_num \
         : tps_checked + 1 = tps_checked
  gosub 2000
  write tps_report_id

999
  return
```

Historic 4GL

- Variables bound to database fields.
- Schema defined outside code using UI.
- Weakish types.

Implied dualvars, predating Perl's?

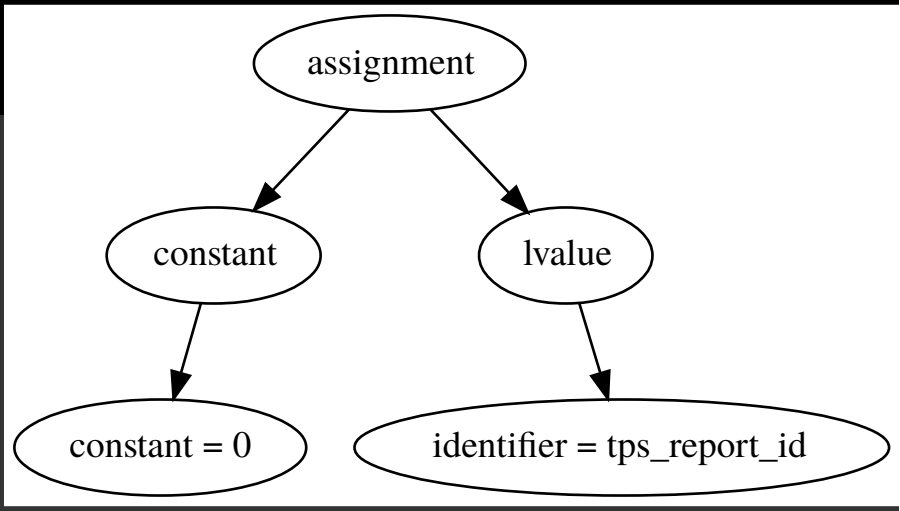
```
# Types are defined outside code in the UI.
300
  1 = enumeratation
  enumeration = string_var # assign string "Yes" to string_var
```

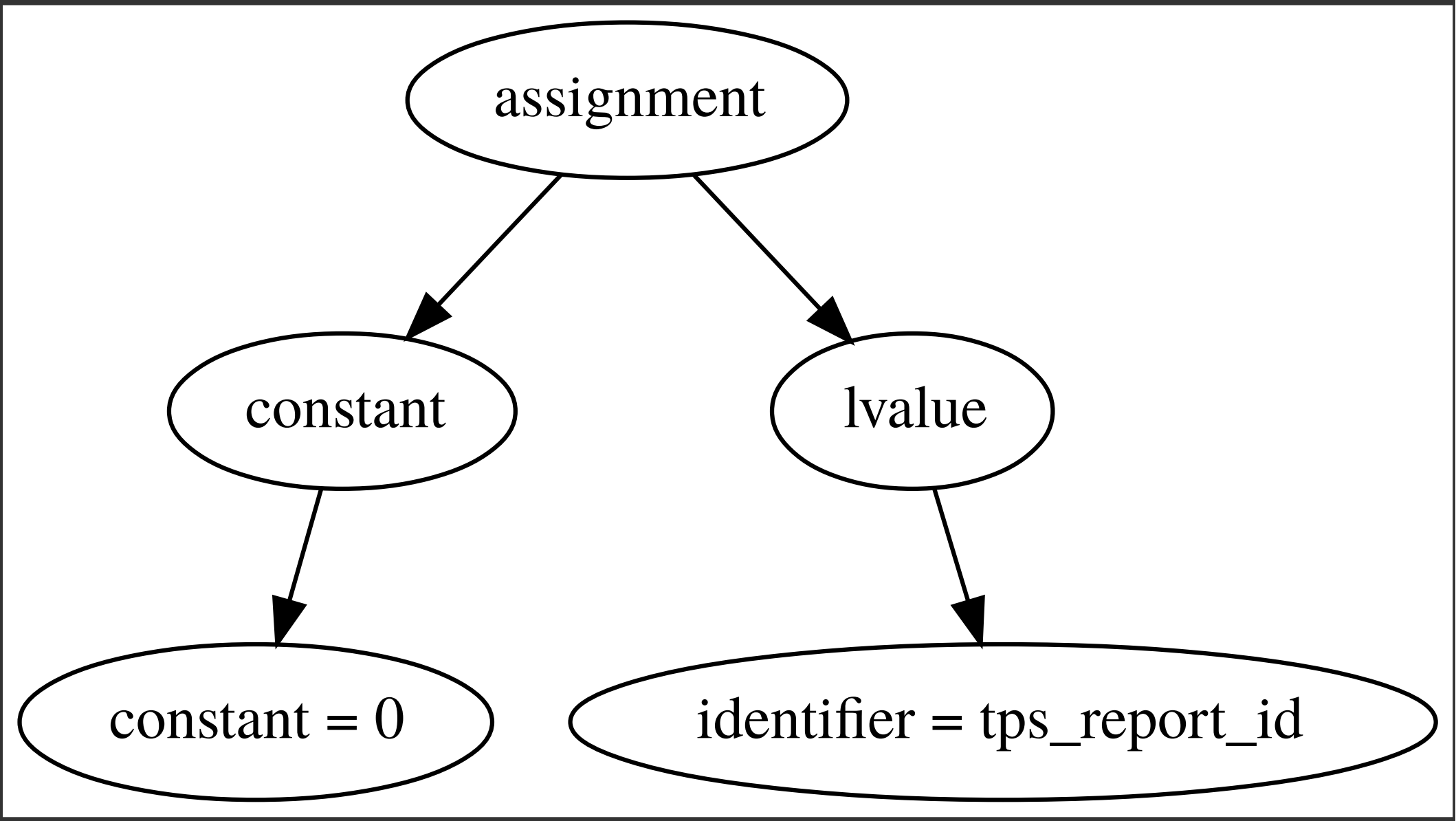
Regexp::Grammars

- Perl 5.10 recursive regexps.
- Build a parser from familiar regexps.
- No separate lexical analysis.
- Mis-parses are slow and hard to debug

```
50
  set 0 = tps_report_id
  readnext tps_report_id

100
  on io_code ne 0 goto 999

  on tps_status eq 1 : tps_num + 1 * 3 = tps_num \
        : tps_checked + 1 = tps_checked
  gosub 2000
  write tps_report_id

999
  return
```

```
<rule: stmt_set>
(?: set | on | : ) <Scalar> <[Operation]>*

<rule: Operation>
<Comparison> | <Arithmetic> | <Assignment>

<rule: Assignment>
<assignment_op> <LValue>

<rule: LValue>
<ArrayIndex> | <Identifier>

<token: assignment_op>
= | d= | =d
```

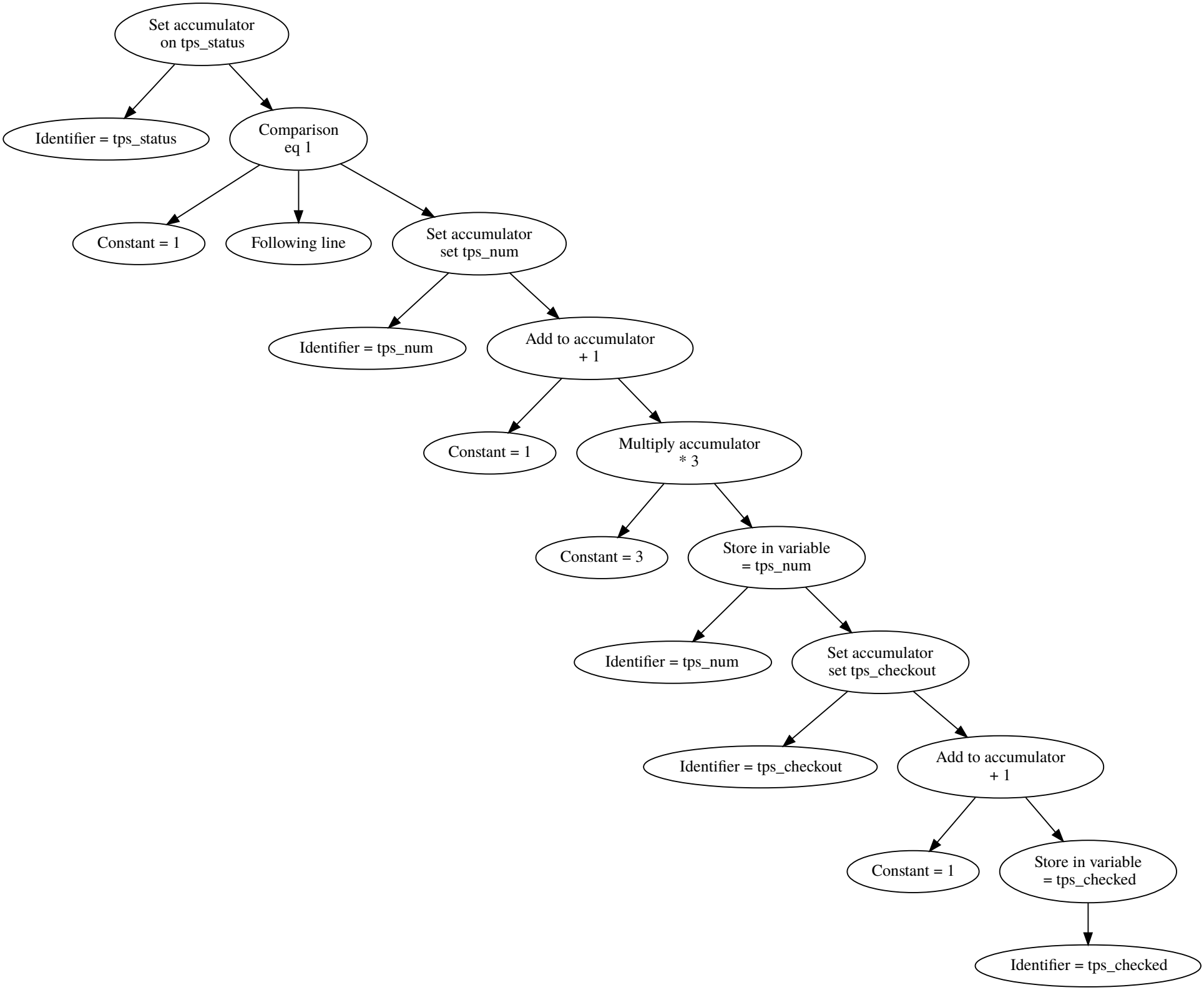



assignment
constant
lvalue
constant = 0
identifier = tps_report_id

Regexp::Grammars

```
<rule: stmt_set>
(?: set | on | : ) <Scalar> <[Operation]>*

<rule: Operation>
<Comparison> | <Arithmetic> | <Assignment>

<rule: Assignment>
<assignment_op> <LValue>

<rule: LValue>
<ArrayIndex> | <Identifier>

<token: assignment_op>
= | d= | =d
```

```
50
  set 0 = tps_report_id
  readnext tps_report_id

100
  on io_code ne 0 goto 999

  on tps_status eq 1 tps_num + 1 * 3 = tps_num \
        tps_checked + 1 = tps_checked
  gosub 2000
  write tps_report_id

999
  return
```


Set accumulator
on tps_status
Identifier = tps_status
Comparison
eq 1
Constant = 1
Following line
Set accumulator
set tps_num
Identifier = tps_num
Add to accumulator
+ 1
Constant = 1
Multiply accumulator
* 3
Constant = 3
Store in variable
= tps_num
Identifier = tps_num
Set accumulator
set tps_checkout
Identifier = tps_checkout
Add to accumulator
+ 1
Constant = 1
Store in variable
= tps_checked
Identifier = tps_checked

Phases

- Preprocess text (join line continuations, remove comments.
- Parse lines to parse tree.
- Flatten parse tree to op list.
- Execute op list.

```
if ($lines[$line] =~ $line_parser) {
    if (ref $/{Line}) {
        $/{Line}->{_line} = $line+1;
        $/{Line}->{_fn}   = $fn;
        push @parses, $/{Line};
    }
}
```

```
{
  '' => '  on tps_status eq 1 : tps_num + 1 * 3 = tps_num : tps_checked + 1 = tps_checked',
  'Line' => {
              '' => '  on tps_status eq 1 : tps_num + 1 * 3 = tps_num : tps_checked + 1 = tps_checked',
              'Statement' => [
                               {
                                 '' => 'on tps_status eq 1',
                                 'stmt_set' => {
                                                 '' => 'on tps_status eq 1',
                                                 'Operation' => [
                                                                  {
                                                                    '' => 'eq 1',
                                                                    'Comparison' => {
                                                                                      '' => 'eq 1',
                                                                                      'Scalar' => {
                                                                                                    '' => '1',
                                                                                                    'DecimalConstant' => '1'
                                                                                                  },
                                                                                      'comparison_op' => 'eq'
                                                                                    }
                                                                  }
                                                                ],
                                                 'Scalar' => {
                                                               '' => 'tps_status',
                                                               'Identifier' => 'tps_status'
                                                             }
                                               }
                               },
                               {
                                 '' => ': tps_num + 1 * 3 = tps_num',
                                 'stmt_set' => {
                                                 '' => ': tps_num + 1 * 3 = tps_num',
                                                 'Operation' => [
                                                                  {
                                                                    '' => '+ 1',
```

In summary

- Laziness as a virtue predates Perl 1.0.

But next

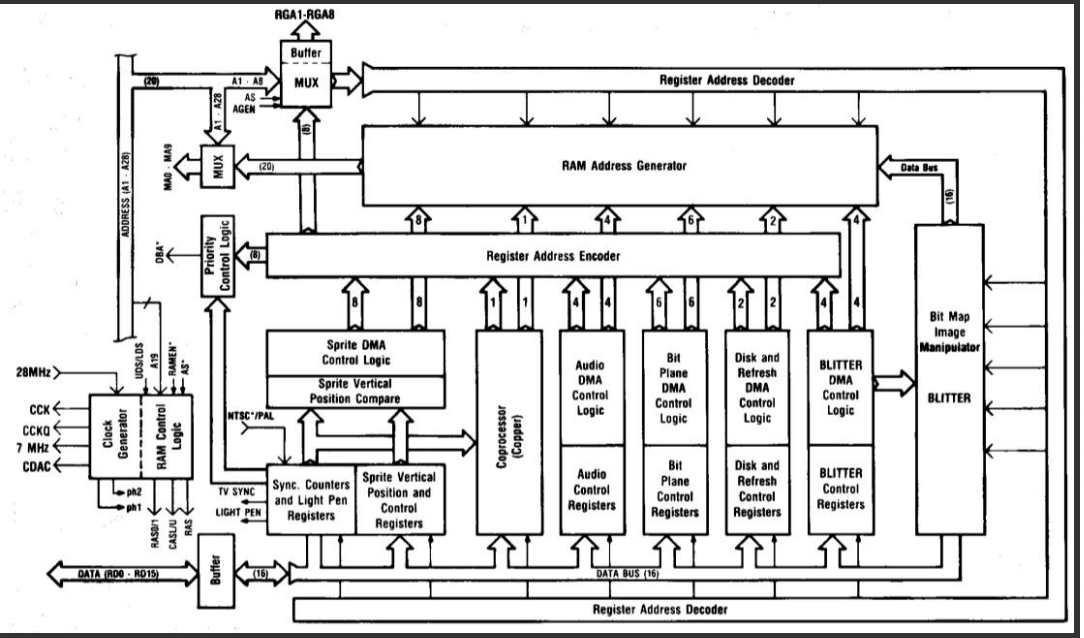
- Is Perl dead?
- Is the Perl demoscene dead?

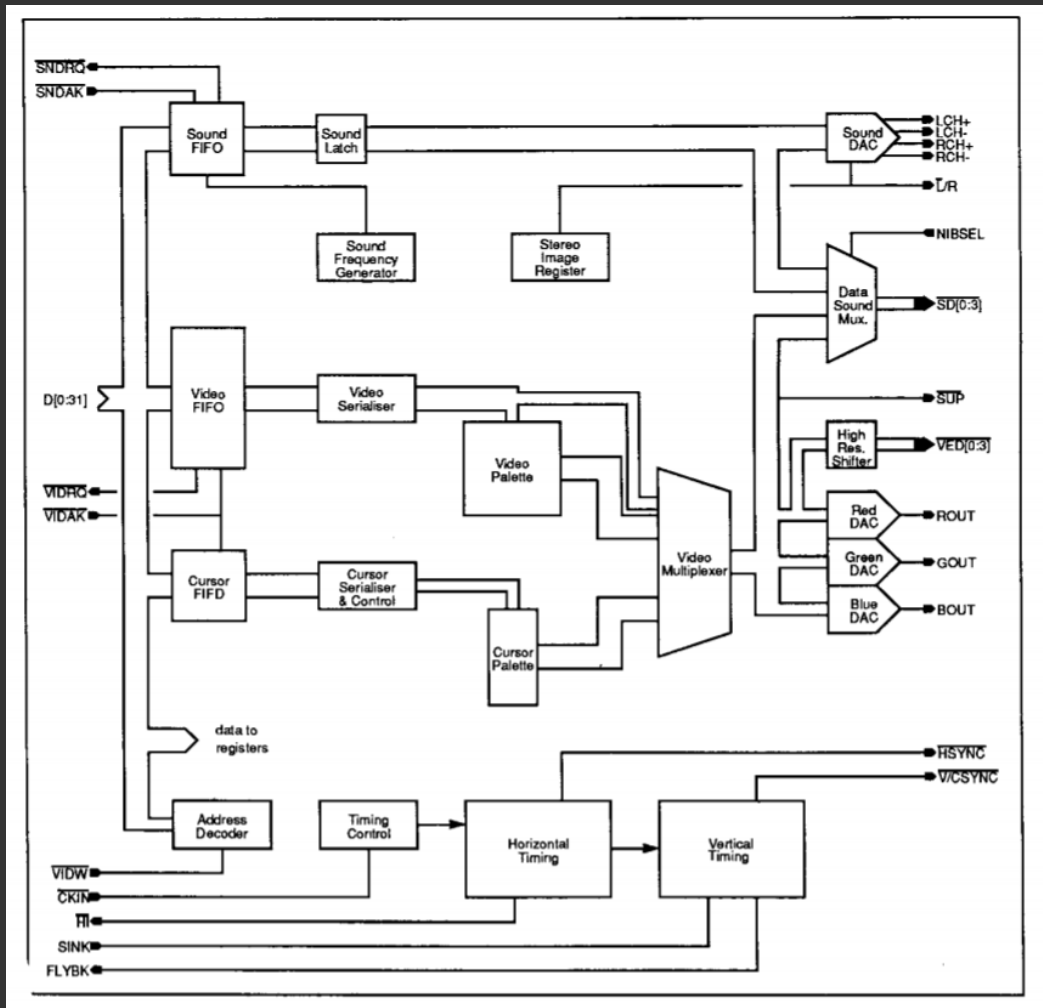
Presenting demo#1

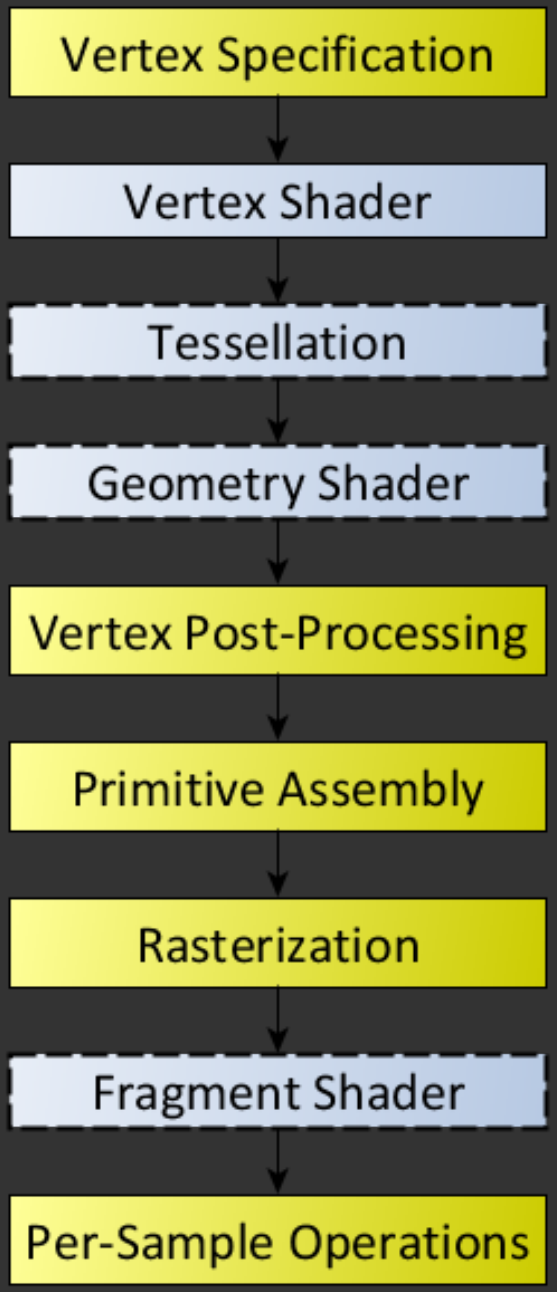
- Code by DLAMBLEY of CPAN.
- Grafix by .. no-one.
- Music by no-one.

Demos and games need hardware

Sources: CSG and Acorn Computer





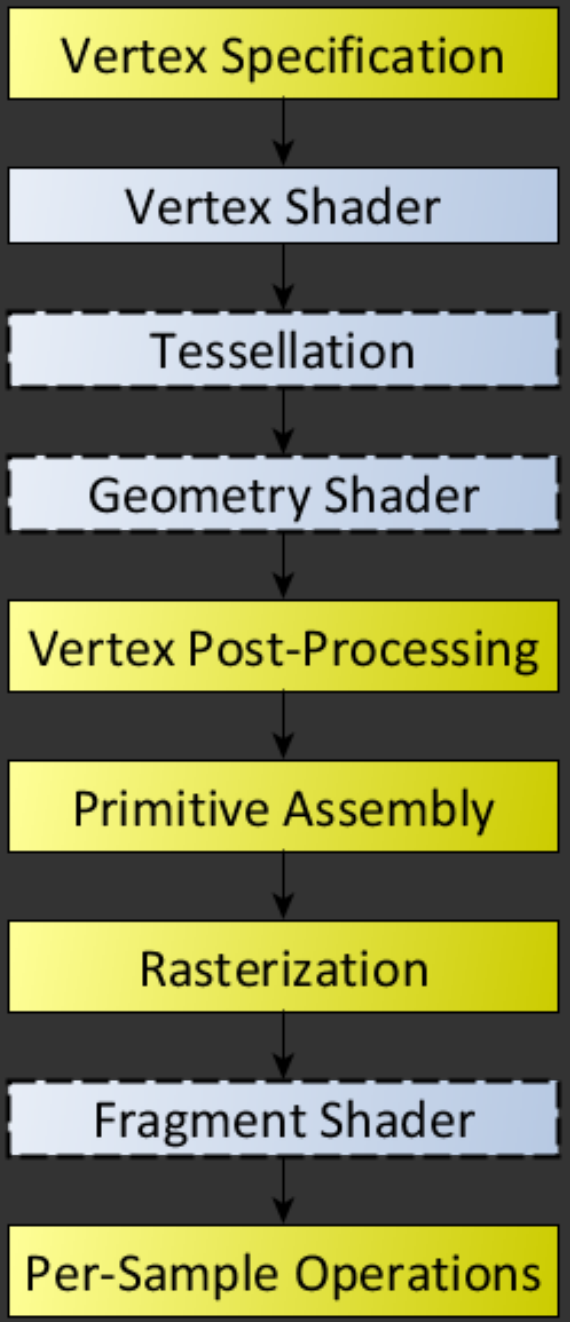


Source: https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview

OpenGL

- SDL.pm recommends OpenGL.pm.
- OpenGL.pm gives you most of OpenGL 1.2.

```
my $offset = 0;

while (1) {
    glutMainLoopEvent();
    $offset += 1/(2**11);
    $offset = $offset > 1 ? $offset-1 : $offset;
    glutPostRedisplay();
}

sub display {
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_POLYGON);
    glColor3f(0, 0, $offset);
    glVertex2f(-1, -1);
    glColor3f(0, 1, $offset);
    glVertex2f(-1,  1);
    glColor3f(1, 1, $offset);
    glVertex2f( 1,  1);
    glColor3f(1, 0, $offset);
    glVertex2f( 1, -1);
    glEnd();
    glutSwapBuffers();

    return;
}
```



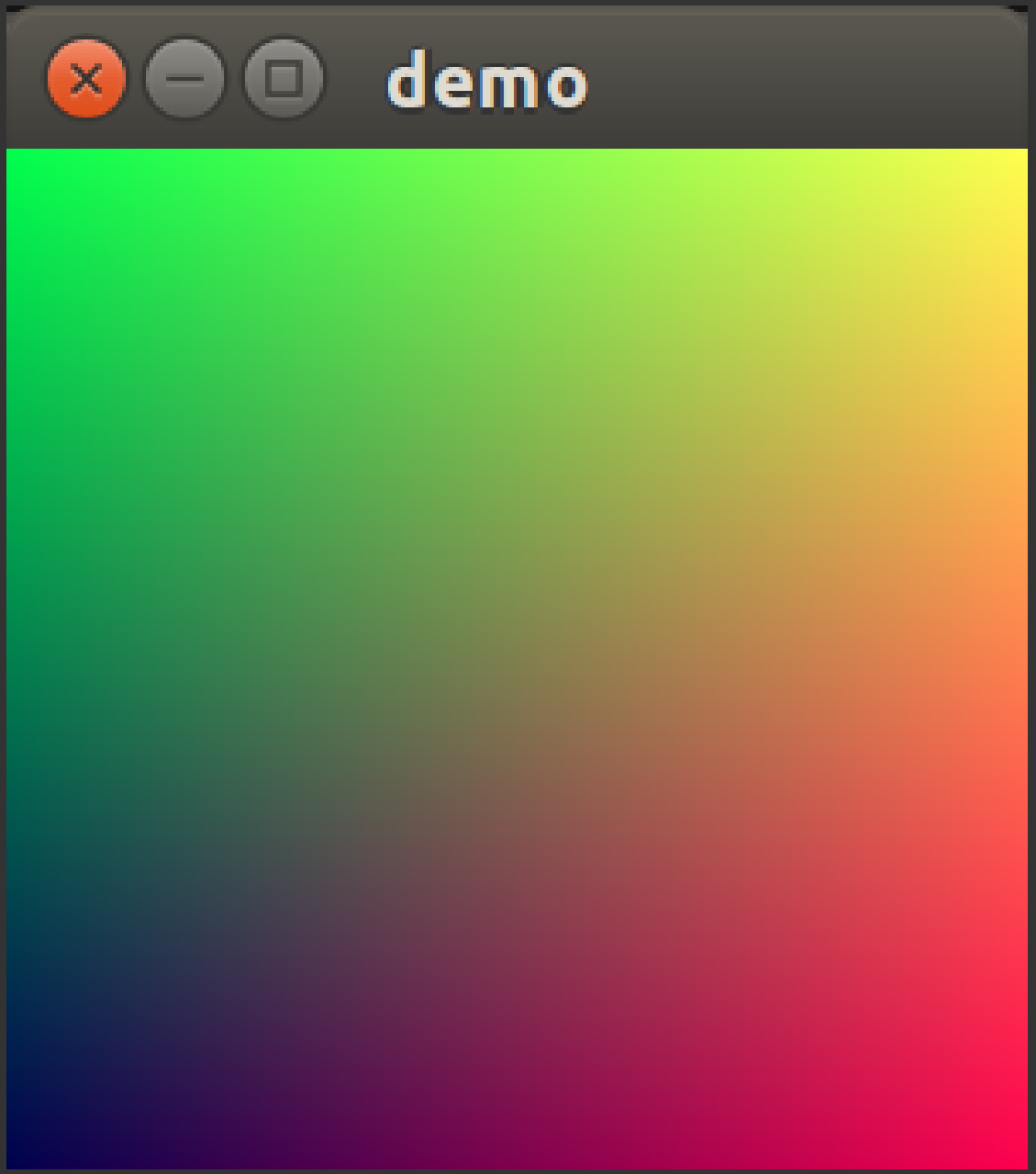
Source: https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview

A vertex shader

```
my $vertex_shader = glCreateShaderObjectARB( GL_VERTEX_SHADER );

glShaderSourceARB_p($vertex_shader, q{
void main() {
    gl_FrontColor = gl_Color;
    gl_Position = ftransform();
}
});
```

A fragment shader

```
glShaderSourceARB_p($frag_shader, qq{#version 120
void main() {
    gl_FragColor = gl_Color;
}
});
```


demo

```
my $star_data = join(", ", map {
    my $x = rand();
    my $y = rand();
    my $z = int(rand(5));
    "vec3($x, $y, $z)"
} 1..$stars);

glShaderSourceARB_p($frag_shader, qq{#version 120
    uniform vec3 stars[$stars] = vec3[$stars]( $star_data );

    void main() {
        float bright = 0;
        float shift = gl_Color.z;

        for (int n = 0; n<$stars; n++) {
            float d = distance(
                vec2(fract(gl_Color.x+10*shift / stars[n].z), gl_Color.y),
                vec2(stars[n].x, stars[n].y)
            );
            bright += clamp(
                (1.0-d*500)/stars[n].z,
                0.0, 1.0
            );
        }
        gl_FragColor = vec4(bright, bright, bright, bright);
    }
});
```


TURBO PASCAL
FOR WINDOWS
Version 1.5
Copyright © 1991,1992
Borland International, Inc.
All rights reserved.

Pascal?

- ObjectWindows ++
- Pointer arithmetic ++
- Compiler ++
- IDE ++
- System library --

FreePascal

- Author terrified of leaving DOS.
- https://www.freepascal.org/
- fpc -Mtp foo.pas
- fpc -Mtp -Ci -Co -CR -Cr -Ct -g foo.pas

Artistic license? DFSG??

```
{******************************************************************}
{                                                                  }
{       Turbo Pascal Textual and Graphical Windows                 }
{                    Runtime Library                               }
{                                                                  }
{       Written by Dr. I. Checkland 5/95                           }
{                                                                  }
{       Based on Windows CRT Interface Unit                        }
{       Copyright (c) 1992 Borland International                   }
{                                                                  }
{******************************************************************}
```

A new GRAPHWIN.PAS

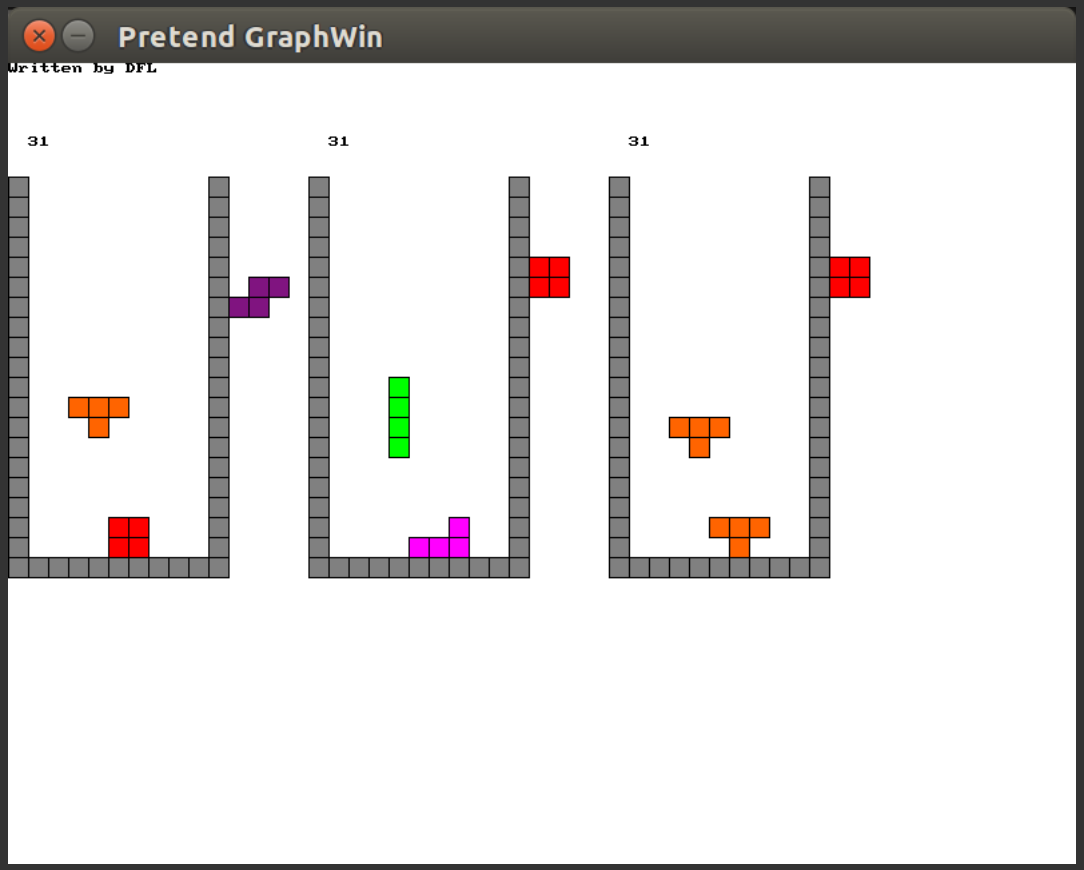
- “graphwin.pas”

```
procedure DrawOblong(x1,y1,x2,y2:integer);
begin
        WriteLn('{"action":"drawoblong",',
                '"x1":',x1,',',
                '"y1":',y1,',',
                '"x2":',x2,',',
                '"y2":',y2,',',
                '"style":',BrushStyle,',',
                '"penr":',PenR,',',
                '"peng":',PenG,',',
                '"penb":',PenB,',',
                '"brushr":',BrushR,',',
                '"brushg":',BrushG,',',
                '"brushb":',BrushB,'}'
        );
end;
```

Running it


```
{"action":"init"}
{"action":"writeln","string":"                    TRON"}
{"action":"drawoblong","x1":50,"y1":50,"x2":55,"y2":55,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":55,"x2":55,"y2":60,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":60,"x2":55,"y2":65,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":65,"x2":55,"y2":70,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":70,"x2":55,"y2":75,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":75,"x2":55,"y2":80,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":80,"x2":55,"y2":85,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
{"action":"drawoblong","x1":50,"y1":85,"x2":55,"y2":90,"style":1,"penr":0,"pen
g":0,"penb":0,"brushr":0,"brushg":0,"brushb":0}
```

Does it work?



Grab the code! It works!

- https://github.com/davel/perldemo
- https://github.com/davel/pascal-games
- **My customer needs more Perl, talk to me.**