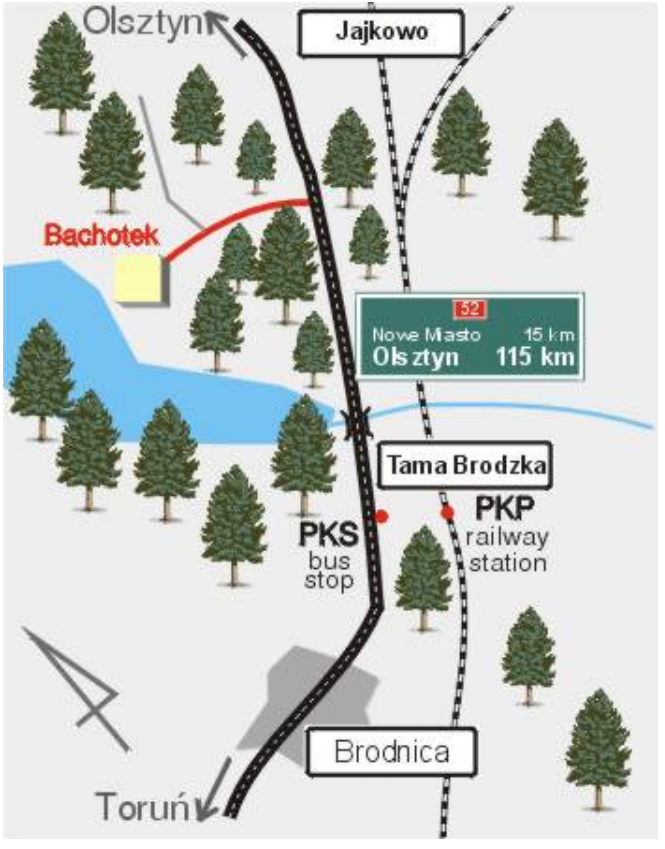
All roads lead to BachoTEX ...

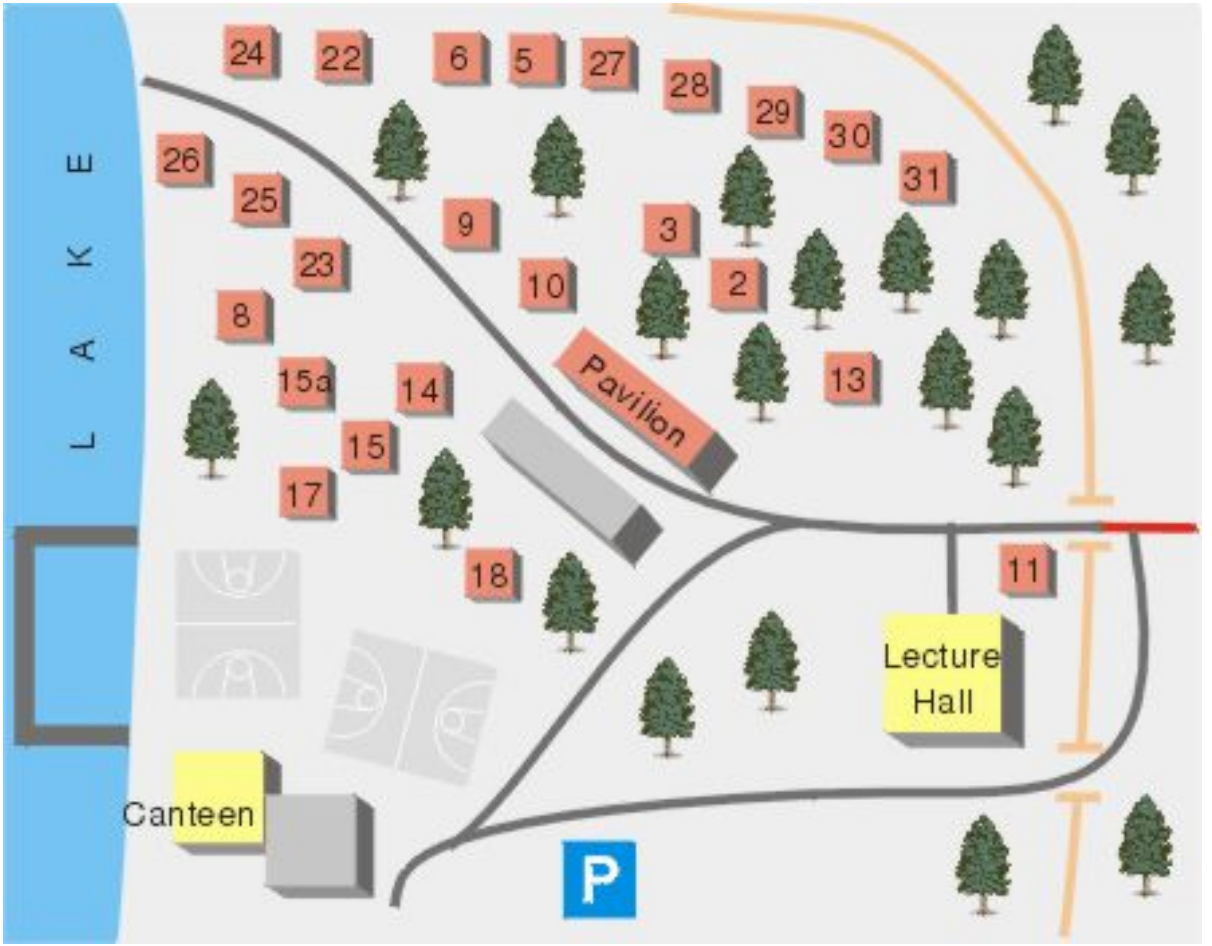
BachoTEX 2014

X14

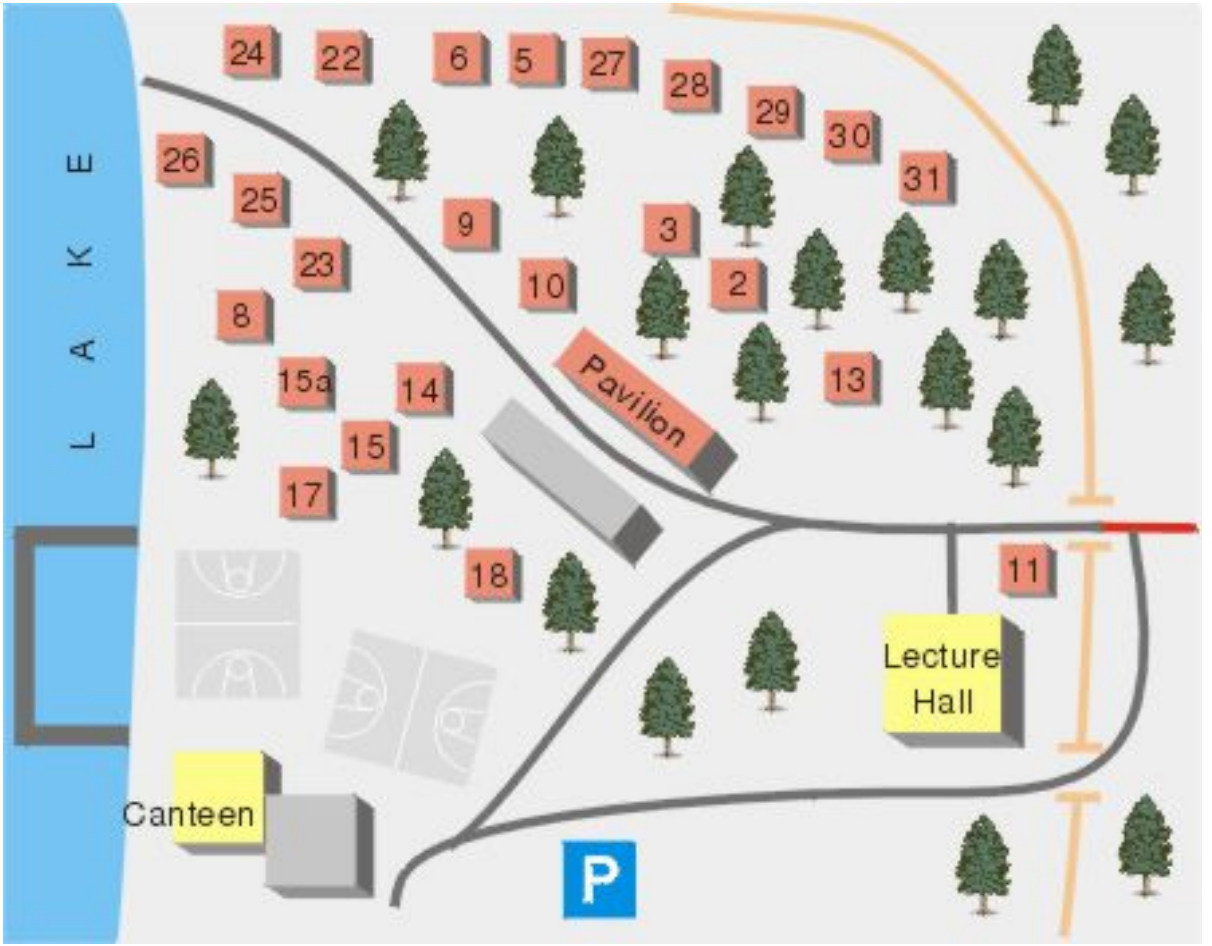
All roads lead to BachoTEX ...



Relatively easy to find your hut …

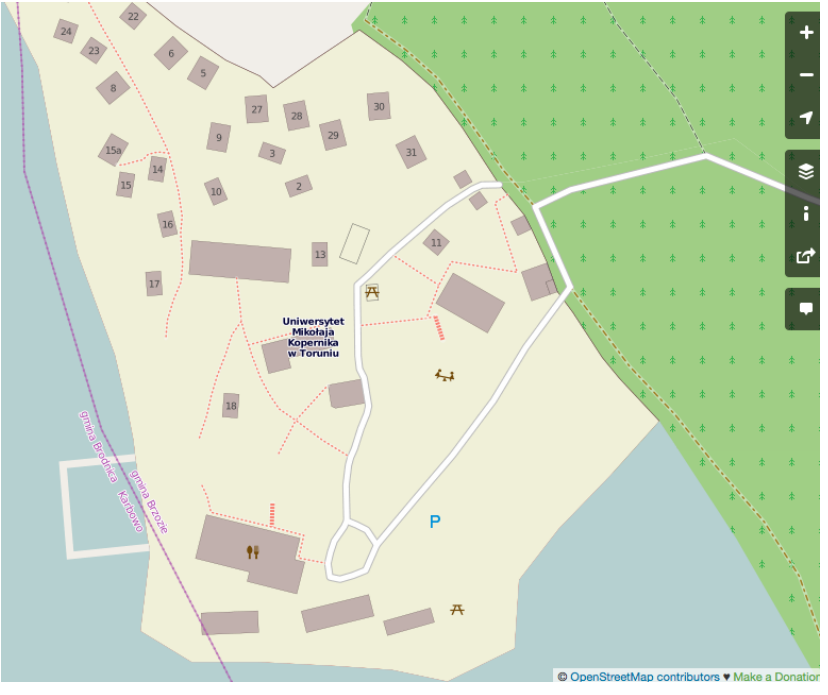


… except the hut numbers don't match



The OpenStreetMap Project

- measure coordinates with GPS
- upload tracks to www.openstreetmap.org



The OpenStreetMap Project

Credits:

- Harald König, Ulrik Vieth, Sebastian Krüger

Disadvantage:

- bitmap output, no nicely printable map

Advantage:

- easy to export data

Interesting talks @ BachoTEX

After being able to find the lecture room and the hut, we were ready to listen to some interesting talks.

Lua in MetaPost

Taco Hoekwater @ BachoTEX (3rd May 2014):

- **MetaPost v2.000**

Hans Hagen @ BachoTEX (3rd May 2014):

- **MetaPost 0.99..99, an experiment**

- "decimal" arbitrary precision system: more freedom
- enable Lua calls inside MetaPost
- enable Alan to reimplement core MetaPost packages
- new MetaPost and LuaTEX releases around BachoTEX

OpenStreetMap Export

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.3.3 (16030 thorn-01.openstreetmap.org)" copyright="OpenS
and contributors" attribution="http://www.openstreetmap.org/copyright" license="http://opendata
 <bounds minlat="53.2870100" minlon="19.4732000" maxlat="53.2908400" maxlon="19.4777500"/>
...
 <node id="2187747929" visible="true" version="1" changeset="15264405" timestamp="2013-03-05T22
user="FiligranFifak" uid="392955" lat="53.2888329" lon="19.4754375"/>
...
 <way id="208487403" visible="true" version="1" changeset="15264405" timestamp="2013-03-05T22:1
user="FiligranFifak" uid="392955">
  <nd ref="2187747929"/>
  <nd ref="2187747930"/>
  <nd ref="2187747917"/>
  <nd ref="2187747916"/>
  <nd ref="2187747929"/>
  <tag k="addr:housenumber" v="13"/>
  <tag k="building" v="hut"/>
 </way>
...
</osm>
```

TeX + MetaPost + XML + Lua

- read data from XML
- use Lua to join the puzzles together
- use MetaPost to draw the image
- typeset in ConTeXt

```
-- read the map and image bounds (minimum/maximum latitude and longitude)
local root = xml.load("bachotex.osm")

local b = xml.first(root,"/osm/bounds")
local minlat = b.at.minlat
local minlon = b.at.minlon
local maxlat = b.at.maxlat
local maxlon = b.at.maxlon
local midlat = 0.5 * (minlat + maxlat)

-- store coordinates
local coordinates = { }

for c in xml.collected(root,"/osm/node") do
    local a = c.at
    coordinates[a.id] = a
end

-- convert coordinates to (metapost) pairs
local deg_to_rad = math.pi / 180.0

local function f_pair(lon, lat)
    return string.formatters("(%.2fcm,%.2fcm)",
        (lon - minlon) * scale * math.cos(midlat * deg_to_rad), (lat-minlat) * scale)
end
```

```
-- functions to print metapost code
local f_pattern = string.formatters["/osm/(way|relation)[@visible='true']/tag[@k='%s']"]
local f_draw    = string.formatters["draw %--t withcolor %s withpen pencircle scaled 1;"]
local f_fill    = string.formatters["fill %--t -- cycle withcolor %s withpen pencircle scaled
1;"]
local f_draw_p  = string.formatters["path p ; p := %--t ; draw p withcolor %s withpen pencircle
scaled 1;"]
local f_fill_p  = string.formatters["path p ; p := %--t -- cycle ; fill p withcolor %s withpen
pencircle scaled 1;"]
local f_bounds  = string.formatters["setbounds currentpicture to %--t -- cycle ; addbackground
withcolor %s ;"]
local f_way     = string.formatters["/osm/way[@id='%s']"]
local f_textext = string.formatters['draw (textext("\\bf %s") scaled 0.35) shifted center p
withcolor white;']
```

```
local function drawshapes(what)
    local function filterpath(r,pattern,name,kind)
        local p = { }
        for c in xml.collected(r,pattern) do
            local coordinate = coordinates[c.at.ref]
            if coordinate then
                p[#p+1] = f_pair(coordinate.lon, coordinate.lat)
            end
        end
        local color = colors[kind] or "black"
        if #p == 0 then -- error
        elseif name then
            if rendering[what] then
                context(f_fill_p(p,color))
            else
                context(f_draw_p(p,color))
            end
            context(f_textext(name))
        else
            if rendering[what] then
                context(f_fill(p,color))
            else
                context(f_draw(p,color))
            end
        end
    end
```

```
    -- we're in tex so filters print to tex which is why we need the xml://
    for c in xml.collected(root,f_pattern(what)) do
        local parent = xml.parent(c)
        local tag    = parent.tg
        local name   = xml.filter(parent,"xml://tag[@k='addr:housenumber']/attribute('v')")
        local kind   = xml.filter(parent,"xml://tag[@k='amenity']/attribute('v')") or c.at.v
        if tag == "way" then
            filterpath(parent,"/nd",name,kind)
        elseif tag == "relation" then
            for m in xml.collected(parent,"/member[@type='way']") do
                local f = xml.first(root,f_way(m.at.ref))
                if f then
                    filterpath(f,"/nd",name,kind)
                end
            end
        end
    end
end
...
```

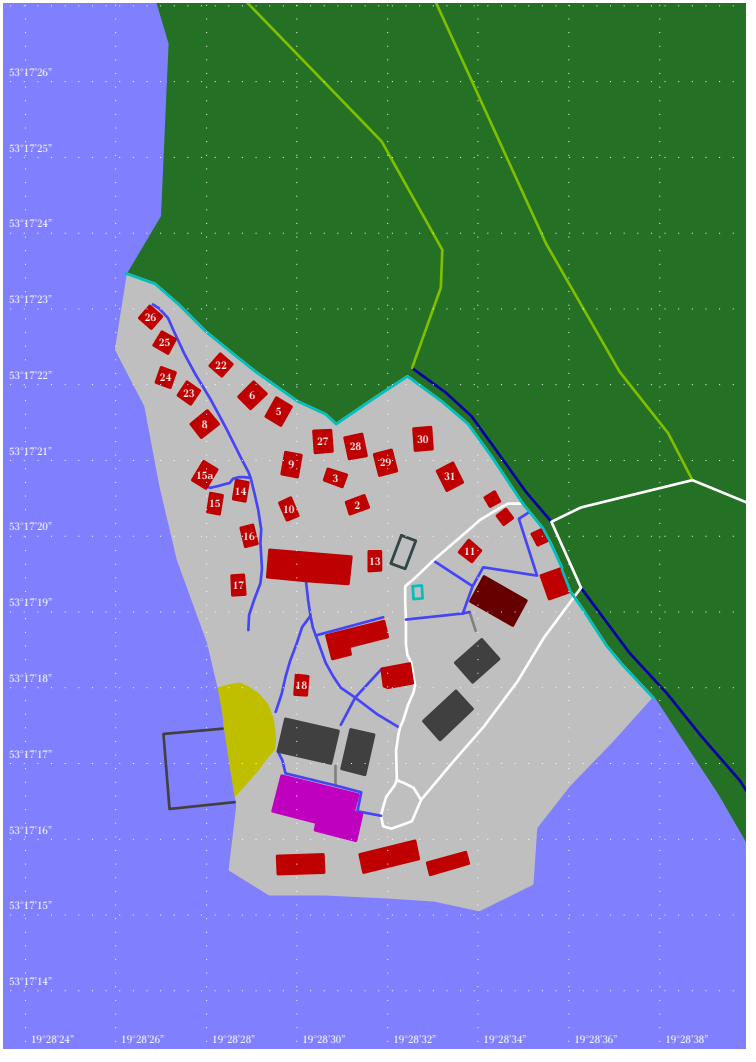
```
-- draw boundary
local function drawboundary()
    local p = {
        f_pair(minlon,minlat),
        f_pair(maxlon,minlat),
        f_pair(maxlon,maxlat),
        f_pair(minlon,maxlat),
    }
    context(f_bounds(p,colors.background))
end

-- finally draw the map
context.startTEXpage()
context.startMPcode("doublefun")

for i=1,#shown do
    drawshapes(shown[i])
end

draw_grid()
drawboundary()

context.stopMPcode()
context.stopTEXpage()
```


53°17'26"
53°17'25"
53°17'24"
53°17'23"
53°17'22"
53°17'21"
53°17'20"
53°17'19"
53°17'18"
53°17'17"
53°17'16"
53°17'15"
53°17'14"
19°28'24"
19°28'26"
19°28'28"
19°28'30"
19°28'32"
19°28'34"
19°28'36"
19°28'38"
26
25
24
23
22
8
6
5
27
28
29
30
31
9
3
2
15a
14
15
10
16
17
13
11
18