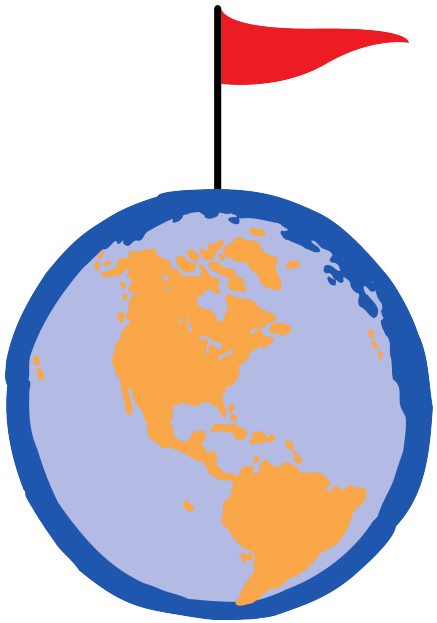
OpenOrienteering Mapper

an open source mapmaking program for orienteering

13.07.2012 **ICOM - 2012** Thomas Schöps

Orienteering mapmaking today

- By far the most maps are drawn with one proprietary program
- Can be quite expensive for small clubs — old, outdated versions are used to save money
- A free version is offered, but it is “from the Stone Age”
- No real competitor exists



Why should mapping programs be free?

- No worries about the number of available program licenses
  - Easier for beginners to try mapping, or for course setters to make map corrections
  - Ideal for school maps, for example
- You always get the latest & best version, without artificial limitations
- Every programmer can improve the program
- **Spend the money on maps, not on the tools!**

Some attempts to free o-mapping

- **O-Scape** - https://sourceforge.net/projects/o-scape/
  - Set of python extensions for Inkscape, an open source vector graphics program, to provide orienteering map symbols
- **COMO** - http://como.oxtract.se/
  - Based on OpenStreetMap and its editing tools
  - Apparently abandoned
- **OpenOrienteeringMap** - http://oobrien.com/oom/
  - Not a way to make orienteering maps directly, but a custom style applied to existing OpenStreetMap data

OpenOrienteering Mapper

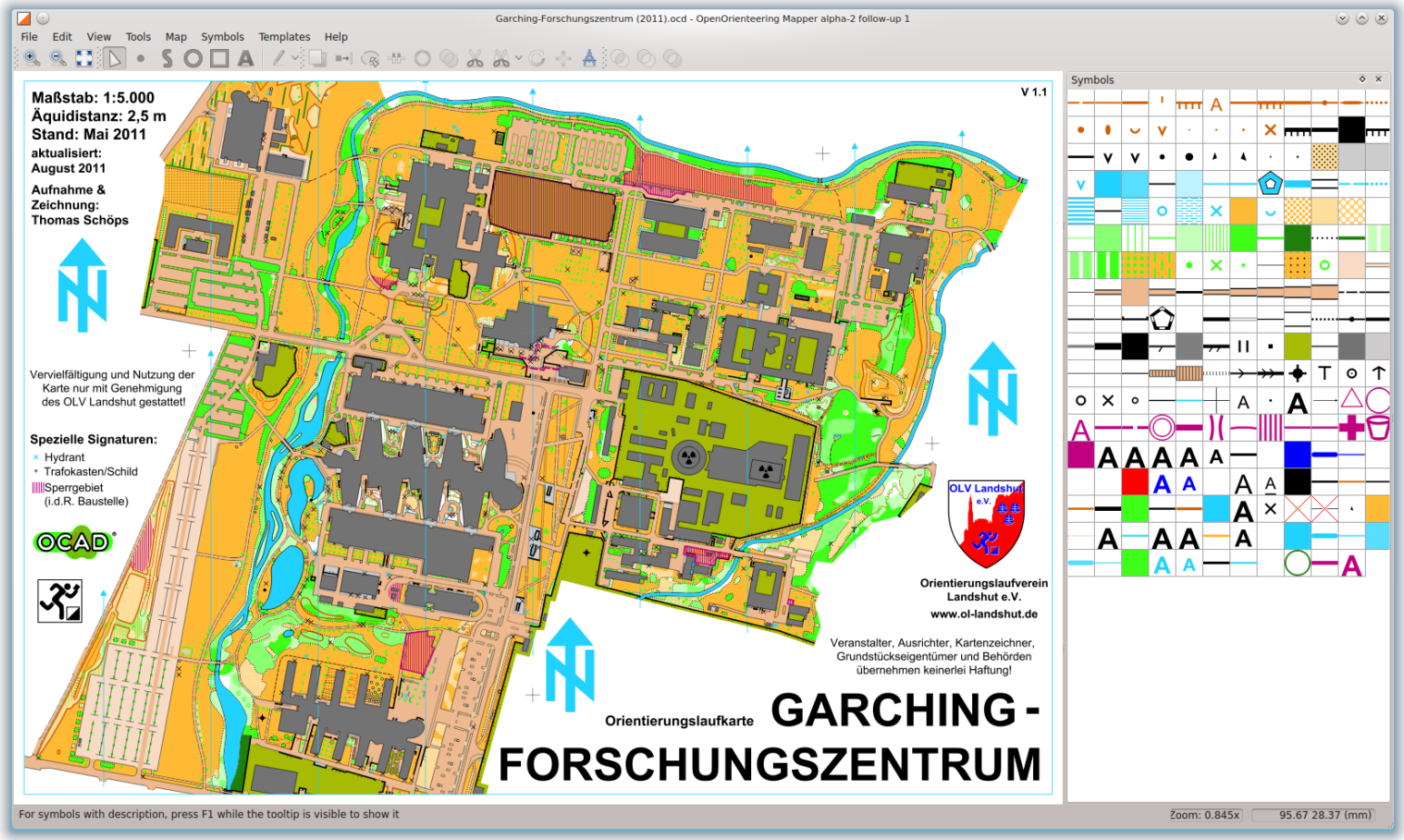
- Independent program, specifically for creating orienteering maps

Short history:

- Started as a one-man project in late 2011
- First public announcement on January 1st, 2012
- Releases up to now:
  - **Alpha 1**: February 10th, 2012
  - **Alpha 2**: March 24th, 2012
  - **Alpha 3**: July 6th, 2012 (**currently latest version**)
- Now contains contributions by more than 10 people and is translated into 5 languages already

Features

- Do not let yourself deceive by the term “Alpha” ...



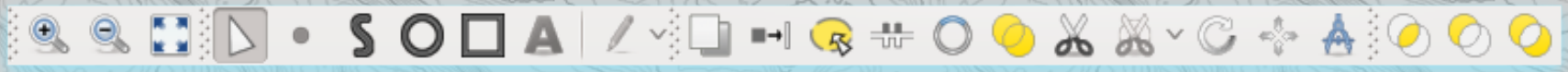
A complete .ocd map (version 8) loaded into Mapper

Some highlights

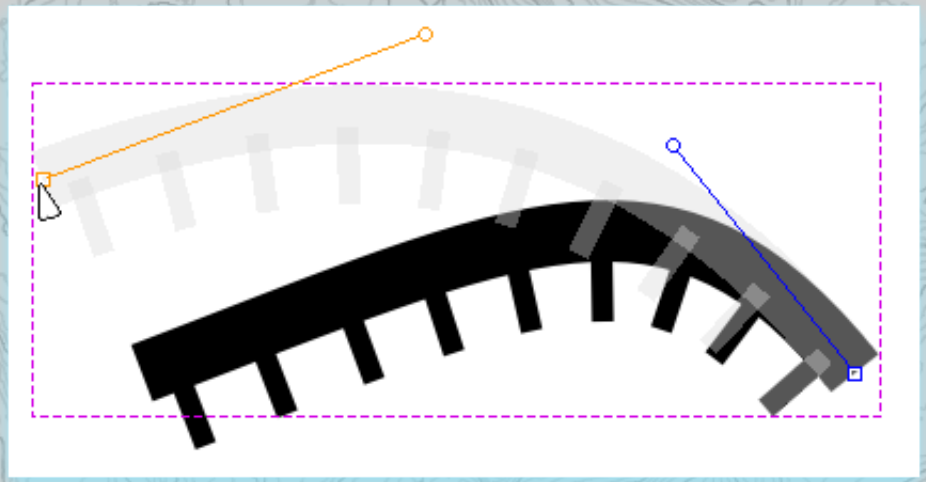
ATTENTION:

Start of advertisement mode!

Some highlights

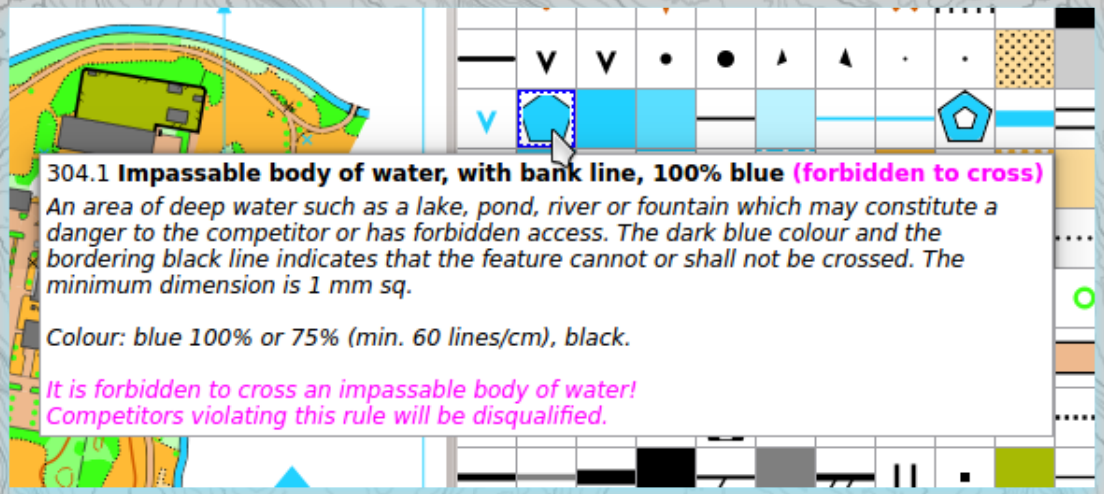


All basic editing tools needed to create orienteering maps



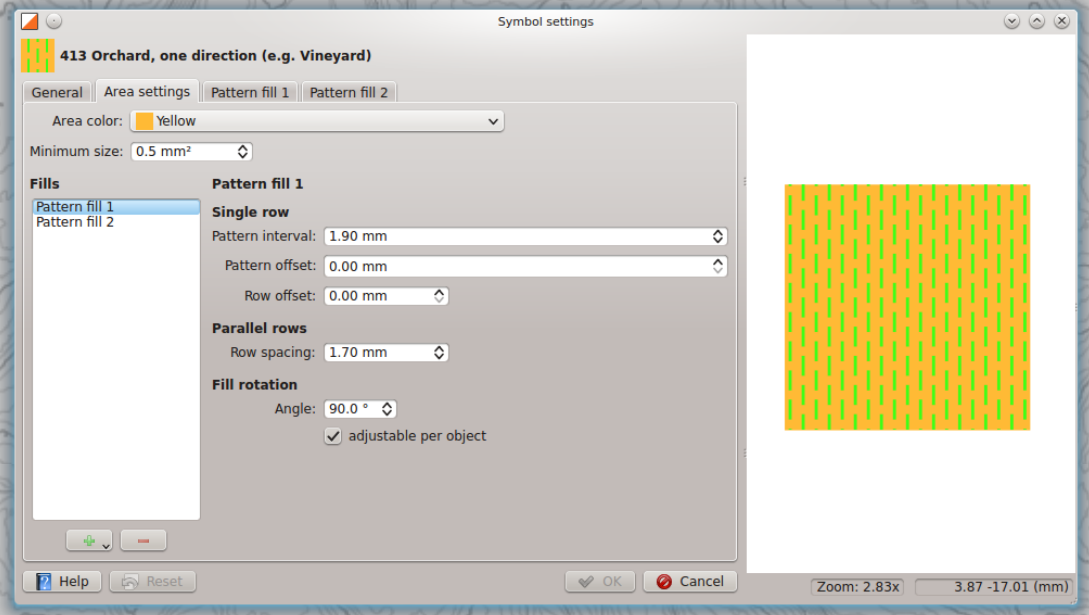
Accurate, transparent live preview for most editing actions

Some highlights

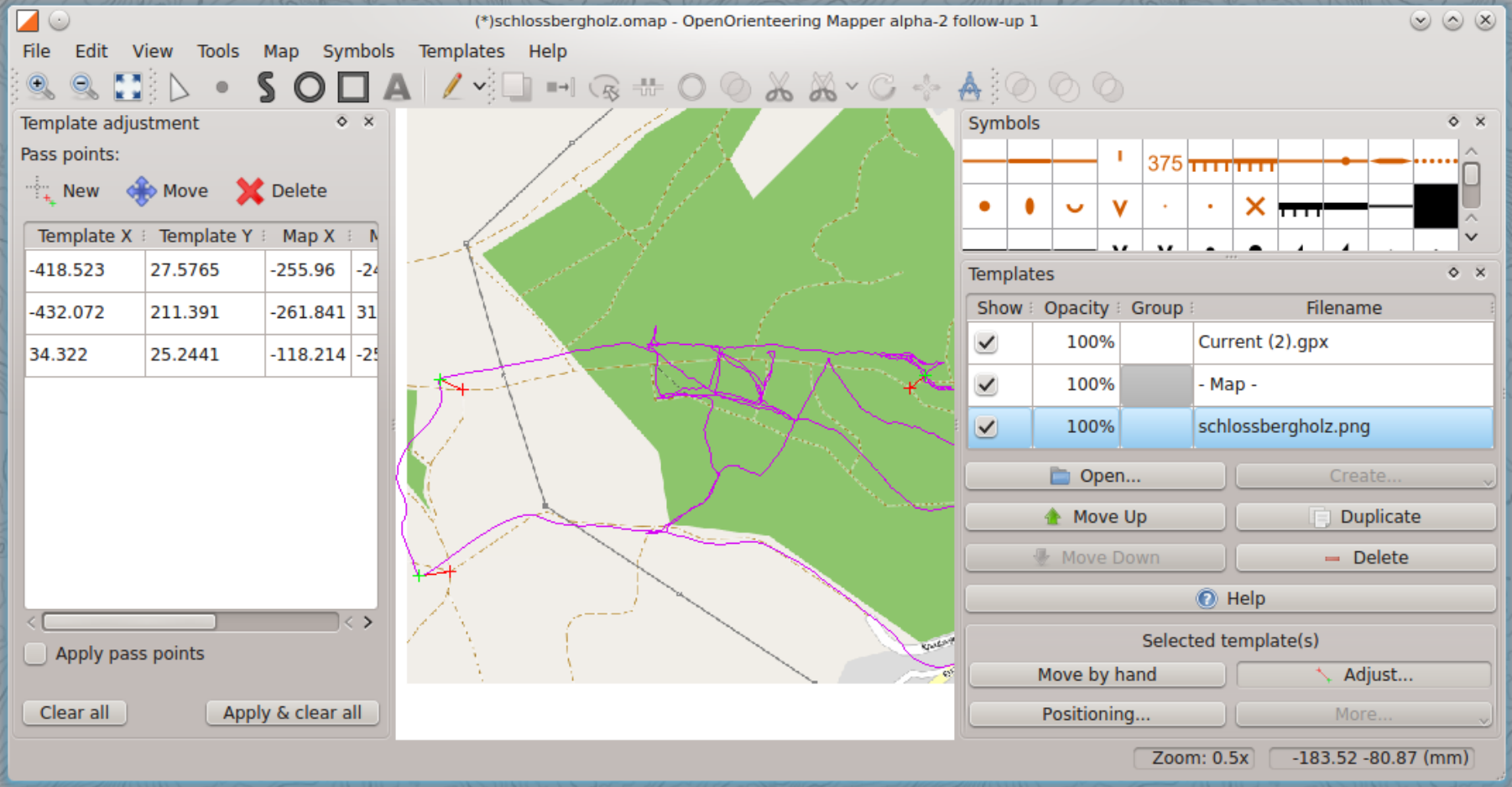


Symbol descriptions from map standards directly integrated into the program (press F1 to show)

Easy-to-use symbol editor with preview



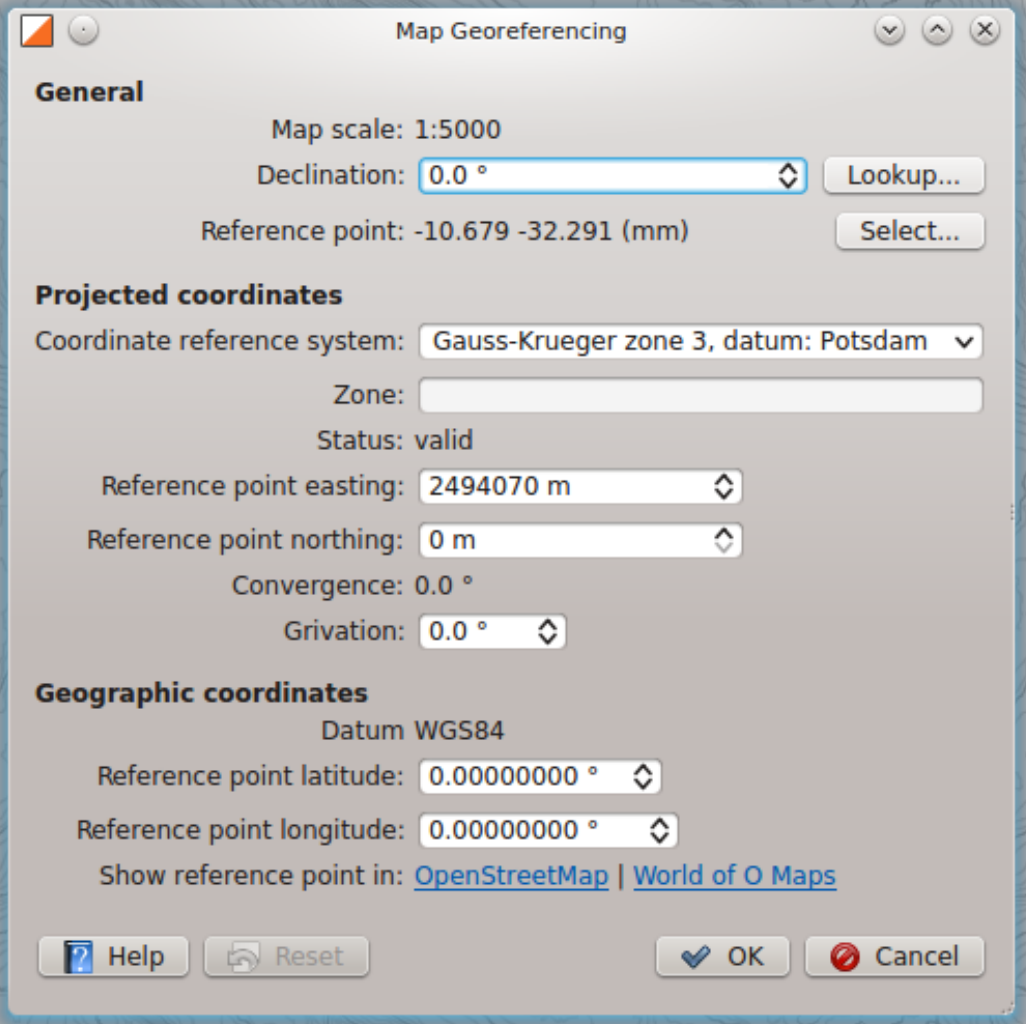
Some highlights



Loading of .gpx / .osm files and images as templates, below or above the map, and adjusting them to each other

(map data source: OpenStreetMap)

Some highlights



Map georeferencing

Features (without cool screenshots)

Import and export of .ocd version 8 map files

Direct PDF export

Comes with complete ISOM and ISSOM symbol sets

Available for Windows and Linux, Mac also possible

Some drawbacks

- .ocd <-> .omap conversion is not 100% accurate
- Was not used for any map project yet
- Still under development — some functionality instable or missing, for example:
  - Many convenience functions for map drawing missing
  - No GeoTiff support
  - No live GPS functionality
  - Export/Import of .ocd only from or to version 8
    - From version 6 on all .ocd file formats are documented, so could be added with some effort
  - No course planning

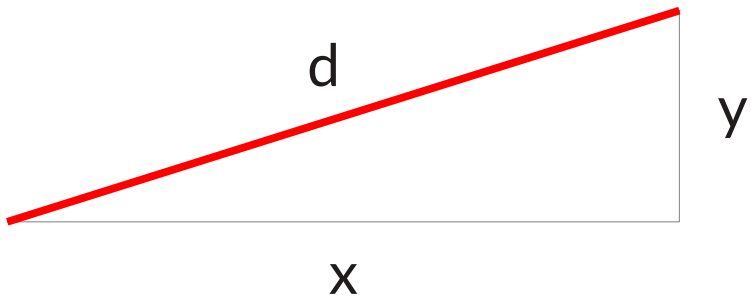
Errors in .ocd <-> .omap conversion

- **Don't panic**: map objects will all stay in their correct position
  - Cubic Bezier splines are used by both OCAD and Mapper
- Kind of errors are mostly changes to the **symbols**:
  - In theory, OCAD and Mapper symbol definitions have varying features, so only the smallest common denominator is convertible
    - But this of course includes the orienteering symbol sets
  - Special path nodes (dash and corner points) are handled differently
  - Line dash patterns are handled differently

Why not clone OCAD to 100%?

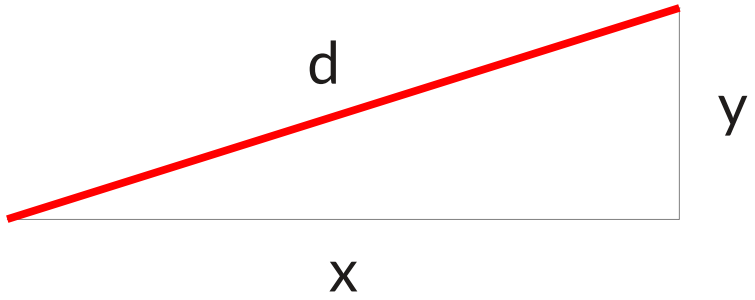
- **Example**: path length calculation quirk
- Applies to .ocd version 8, at least
- Taken from Purple Pen source code, written by Peter Golde

- Formula for calculating **path length for applying dash patterns** in OCAD 8 seems to be:



$$a = max\{|x|, |y|\}$$
$$b = min\{|x|, |y|\}$$
$$d \stackrel{?}{=} a + \frac{b}{2}$$

Why not clone OCAD to 100%?



$$a = max\{|x|, |y|\}$$
$$b = min\{|x|, |y|\}$$
$$d \stackrel{?}{=} a + \frac{b}{2}$$

- As a reminder, the correct formula is:

$$d = \sqrt{x^2 + y^2}$$

- The incorrect approximation leads to **different numbers of dashes** in horizontal/vertical lines than in diagonal lines **of the same length**
- Should OO Mapper adapt to the quirk, or should mappers adapt their maps (slightly)?

Some drawbacks

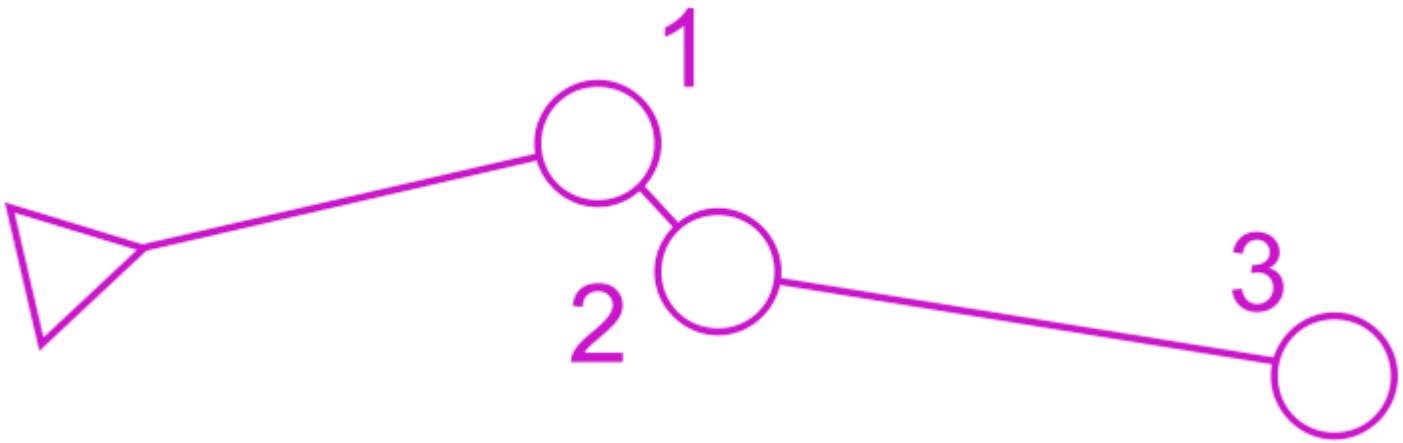
- .ocd <-> .omap conversion is not 100% accurate
- Was not used for any map project yet
- Still under development — some functionality instable or missing, for example:
  - Many convenience functions for map drawing missing
  - No GeoTiff support
  - No live GPS functionality
  - Export/Import of .ocd only from or to version 8
    - From version 6 on all .ocd file formats are documented, so could be added with some effort
  - No course planning

Some drawbacks

No course planning?

Course planning

- Export maps as .ocd, or with given dpi setting as image for exact representation
- Import into course setting program of your choice, for example Purple Pen (open source, http://purplepen.golde.org/)
- As there exists a good and open source possibility for course planning, implementing this in Mapper is of low priority



Technical details of the program (for the interested)

- **Programming language:** C++, using Qt
- **License:** GPLv3
- **Hosted on:** SourceForge
- **Version control:** git

```
    objectUndoManager().notifyOfSave();

    return true;
}
bool Map::loadFrom(const QString& path, MapEditorController* map_editor, bool load_symbols_only)
{
    MapView *view = new MapView(this);

    // Ensure the file exists and is readable.
    QFile file(path);
    if (!file.open(QIODevice::ReadOnly))
    {
        QMessageBox::warning(NULL, tr("Error"), tr("Cannot open file:\n%1\nfor reading.").arg(path));
        return false;
    }

    // Delete previous objects
    clear();

    // Read a block at the beginning of the file, that we can use for magic number checking.
    unsigned char buffer[256];
    size_t total_read = file.read((char *)buffer, 256);
    file.seek(0);

    bool import_complete = false;
    QString error_msg = tr("Invalid file type.");
    Q_FOREACH(const Format *format, FileFormats.formats())
    {
        // If the format supports import, and thinks it can understand the file header, then proceed.
        if (format->supportsImport() && format->understands(buffer, total_read))
        {
            Importer *importer = NULL;
            // Wrap everything in a try block, so we can gracefully recover if the importer balks.
            try {
                // Create an importer instance for this file and map.
```

Who are we?

- **Standing**: July 2012
- Initiator:
  - Thomas Schöps
- Active **C++** developers:
  - Kai Pastor
  - Thomas Schöps
- Other **contributors / translators** and currently **inactive developers**:
  - Jon Cundill
  - Peter Curtis
  - Jan Dalheimer
  - Eugeniy Fedirets
  - Peter Hoban
  - Henrik Johansson
  - Tojo Masaya
  - Russell Porter
  - Aivars Zogla


OpenOrienteering Mapper - Ukrainian translations team

Roadmap (very rough, may change)

- Finish basic map drawing functionality
- Use the program for a complete map project
- Port to Android
  - Can be done for C++/Qt, see QGIS mobile
- ...

Contributing

- Like the project? You can help — whether you can program or not.
- For example by ...
  - creating MTBO / Ski-O symbol sets
  - translating the program into a new language
  - reporting bugs and giving feedback
  - just telling other orienteers / mappers about the project

- **What you could do right now (if you have a laptop with you):**
  - Try out the program — it's free!
  - Download at: www.openorienteering.org
  - Or get it from me via USB stick in case of no internet access

Questions?