Refactoring JPSVis

What, Why and How

Outline

Today we will talk about:

- How we identified some key problems in JPSVis Code
- How we are attempting to address them (Ongoing effort)

This talk should be ~25min.

The rest of the time is reserved for your questions and discussion!

What

- Bugs!
  -> Crash on exit
  -> Trails not being rendered where they should be
  -> Broken video recording
- Very hard to modify substantially!
  -> Long standing request to improve memory consumption
  -> Slow parsing of large input files

Why

- Global State
- Parsing and Rendering code interleaves
- Large complicated functions
- Lots of duplicated code
- Lots of unused code

Why global state is a problem

```cpp
/// define the speed/rate/pace at which  the trajectories are displayed.
/// 1 is the normal default playing rate
int extern_update_step = 1;

/// visualizing 2D or 3 D
/// true for 3D, false for 2D
bool extern_is_3D = true;

bool extern_shutdown_visual_thread = false;
bool extern_recording_enable       = false;
bool extern_is_pause               = false;
bool extern_launch_recording       = false;
bool extern_fullscreen_enable      = false;
bool extern_take_screenshot        = false;
/// states whether a setting has been altered
/// and force the system to update
bool extern_force_system_update = false;

/// enables of disables tracking.
/// With this enable, moving pedestrians will leave a
/// trail behind them
PointPlotter * extern_trail_plotter = NULL;

/// relative scale from pedestrian to the geometry (environment)
double extern_scale            = 0.1;
double extern_scale_pedestrian = 0.1;

// At most three pedestrians groups can be loaded
/// The first pedestrian group
Pedestrian ** extern_pedestrians_firstSet        = NULL;
vtkSmartPointer<vtkSphereSource> extern_mysphere = nullptr;
std::map<std::string, std::shared_ptr<TrainType>> extern_trainTypes;
std::map<int, std::shared_ptr<TrainTimeTable>> extern_trainTimeTables;
vtkTensorGlyph * extern_glyphs_pedestrians    = NULL;
vtkTensorGlyph * extern_glyphs_pedestrians_3D = NULL;
vtkTensorGlyph * extern_glyphs_directions     = NULL;
vtkActor2D * extern_pedestrians_labels        = NULL;
vtkActor * extern_glyphs_pedestrians_actor_2D = NULL;
vtkActor * extern_glyphs_pedestrians_actor_3D = NULL;
vtkActor * extern_glyphs_directions_actor     = NULL;

// and here the corresponding dataset
/// The first dataset
SyncData extern_trajectories_firstSet;

// states if the datasets are loaded.
bool extern_first_dataset_loaded = false;

// states whether the loaded datasets are visible
bool extern_first_dataset_visible = false;
```

Why code with multiple responsibilities is a problem

```
bool MainWindow::addPedestrianGroup(int groupID,QString fileName)
{
     Debug::Messages("Enter MainWindow::addPedestrianGroup with filename <%s>", fileName.toStdString().c_str());

    statusBar()->showMessage(tr("Select a file"));
     if(fileName.isEmpty())
        fileName = QFileDialog::getOpenFileName(this,
                                                "Select the file containing the data to visualize",
                                                QDir::currentPath(),
                                                "JuPedSim Files (*.xml *.txt);;All Files (*.*)");

    //the action was cancelled
    if (fileName.isNull()) {
        return false;
    }

    //get and set the working dir
    QFileInfo fileInfo(fileName);
    QString wd=fileInfo.absoluteDir().absolutePath();
    Debug::Messages("MainWindow::addPedestrianGroup: wd:  <%s>", wd.toStdString().c_str());
    SystemSettings::setWorkingDirectory(wd);
    SystemSettings::setFilenamePrefix(QFileInfo ( fileName ).baseName()+"_");

    //the geometry actor
    GeometryFactory & geometry = _visualisationThread->getGeometry();
    QString geometry_file;
    //try to get a geometry filename
    if(fileName.endsWith(".xml",Qt::CaseInsensitive))
    {
         Debug::Messages("1. Extract geometry file from <%s>", fileName.toStdString().c_str());
         geometry_file=SaxParser::extractGeometryFilename(fileName);
    }
    else
    {
         Debug::Messages("Extract geometry file from <%s>", fileName.toStdString().c_str());
         geometry_file=SaxParser::extractGeometryFilenameTXT(fileName);
    }

    Debug::Messages("MainWindow::addPedestrianGroup: geometry name: <%s>", geometry_file.toStdString().c_str());
    if(geometry_file.isEmpty())
    {
         auto fileDir = fileInfo.path();
         if(fileName.endsWith(".txt",Qt::CaseInsensitive))
         {
              int res = QMessageBox::warning(this, "Did not find geometry name in TXT file",
                                             "Warning: Did not find geometry name in TXT file\nOpen geometry file?"
                                             , QMessageBox::Yes
                                             | QMessageBox::No, QMessageBox::Yes);
              if (res == QMessageBox::No) {
                   exit(EXIT_FAILURE);
                   //return false;
              }
              geometry_file = QFileDialog::getOpenFileName(this,
                                                           "Select a geometry file",
                                                           fileDir,
                                                           "Geometry (*.xml)");
              Debug::Messages("Got geometry file: <%s>", geometry_file.toStdString().c_str());
              QFileInfo check_file(geometry_file);
              if( !(check_file.exists() && check_file.isFile()) )
              {
```

Why code with multiple responsibilities is a problem

```
               Debug::Error("Geomery file does not exist.");
               //exit(EXIT_FAILURE);
               return(false);

          }
          //geometry_file =  check_file.fileName();
     }
     // @todo: check xml file too, although probably xml files
     // always have a geometry tag

}
std::cout << "---> geometry " << geometry_file.toStdString().c_str() << "\n" ;

// if xml is detected, just load and show the geometry then exit
if(geometry_file.endsWith(".xml",Qt::CaseInsensitive)) {

    //try to parse the correct way
    // fall back to this if it fails
    SystemSettings::CreateLogfile();
    Debug::Messages("Calling parseGeometryJPS with <%s>", geometry_file.toStdString().c_str());
    if(! SaxParser::parseGeometryJPS(geometry_file,geometry)) {
        int res = QMessageBox::warning(this, "Errors in Geometry. Continue Parsing?",
                                       "JuPedSim has detected an error in the supplied geometry.\n"
                                       "<"
                                       +geometry_file+
                                       ">"
                                       "The simulation will likely fail using this geometry.\n"
                                       "More information are provided in the log file:\n"
                                       +SystemSettings::getLogfile()+
                                       "\n\nShould I try to parse and display what I can?"
                                       , QMessageBox::Yes
                                       | QMessageBox::No, QMessageBox::No);
        if (res == QMessageBox::No) {
            return false;
        }
        SaxParser::parseGeometryXMLV04(wd+"/"+geometry_file,geometry);//@todo:
                                                                      //use
                                                                      //qt sep
    } else {
        //everything was fine. Delete the log file
         //std::cout << "won't delete logfile\n";
        SystemSettings::DeleteLogfile();
    }

    //SaxParser::parseGeometryXMLV04(fileName,geometry);
    //slotLoadParseShowGeometry(fileName);
    //return false;
}

//check if it is vtrk file containinf gradient
if(fileName.endsWith(".vtk",Qt::CaseInsensitive))
{
    if (false==SaxParser::ParseGradientFieldVTK(fileName,geometry))
        return false;

}

QFile file(fileName);
if (!file.open(QIODevice::ReadOnly)) {
    Debug::Error("parseGeometryJPS:  could not open the File: ",fileName.toStdString().c_str());
```

Why code with multiple responsibilities is a problem

```
        return false;

    }

    SyncData* dataset=NULL;

    extern_trajectories_firstSet.clearFrames();


    vtkSmartPointer<vtkSphereSource> org = vtkSphereSource::New();
    org->SetRadius(10);
    // extern_mysphere = org;


    switch(groupID) {
    case 1:
        Debug::Messages("handling first set");
        dataset=&extern_trajectories_firstSet;
        extern_first_dataset_loaded=true;
        extern_first_dataset_visible=true;
        ui.actionFirst_Group->setEnabled(true);
        ui.actionFirst_Group->setChecked(true);
        slotToggleFirstPedestrianGroup();
        break;

    default:
        Debug::Error("Only one dataset can be loaded at a time");
        //return false;
        break;
    }

    //no other geometry format was detected
    double frameRate=16; //default frame rate
    statusBar()->showMessage(tr("parsing the file"));



    //parsing the xml file
    if(fileName.endsWith(".xml",Qt::CaseInsensitive))
    {
        QXmlInputSource source(&file);
        QXmlSimpleReader reader;

        SaxParser handler(geometry,*dataset,&frameRate);
        reader.setContentHandler(&handler);
        reader.parse(source);
        file.close();
    }
    //parsing the vtk file
//    else if(fileName.endsWith(".vtk",Qt::CaseInsensitive))
//    {
//        if (false==SaxParser::ParseGradientFieldVTK(fileName,geometry))
//            return false;

//    }
    // try to parse the txt file
    else if(fileName.endsWith(".txt",Qt::CaseInsensitive))
    {
         QString source_file= wd + QDir::separator() + SaxParser::extractSourceFileTXT(fileName);
         QString ttt_file= wd + QDir::separator() + SaxParser::extractTrainTimeTableFileTXT(fileName);
```

Why code with multiple responsibilities is a problem

```
 QString tt_file= wd + QDir::separator() + SaxParser::extractTrainTypeFileTXT(fileName);
 QString goal_file=wd + QDir::separator() + SaxParser::extractGoalFileTXT(fileName);
 QFileInfo check_file(source_file);
 if( !(check_file.exists() && check_file.isFile()) )
{
     Debug::Messages("WARNING: MainWindow::addPedestrianGroup: source name: <%s> not found!", source_file.toStdString().c_str());
}
 else
      Debug::Messages("INFO: MainWindow::addPedestrianGroup: source name: <%s>", source_file.toStdString().c_str());

 check_file = goal_file;
 if( !(check_file.exists() && check_file.isFile()) )
{
     Debug::Messages("WARNING: MainWindow::addPedestrianGroup: goal name: <%s> not found!", goal_file.toStdString().c_str());
}
 else
      Debug::Messages("INFO: MainWindow::addPedestrianGroup: goal name: <%s>", goal_file.toStdString().c_str());


 check_file = ttt_file;
 if( !(check_file.exists() && check_file.isFile()) )
{
     Debug::Messages("WARNING: MainWindow::addPedestrianGroup: ttt name: <%s> not found!", ttt_file.toStdString().c_str());
}
 else
      Debug::Messages("INFO: MainWindow::addPedestrianGroup: ttt name: <%s>", ttt_file.toStdString().c_str());

 check_file = tt_file;
 if( !(check_file.exists() && check_file.isFile()) )
{
     Debug::Messages("WARNING: MainWindow::addPedestrianGroup: tt name: <%s> not found!", tt_file.toStdString().c_str());
}
 else
      Debug::Messages("INFO: MainWindow::addPedestrianGroup: tt name: <%s>", tt_file.toStdString().c_str());



// ------ parsing sources
QFile file(source_file);
QXmlInputSource source(&file);
QXmlSimpleReader reader;
SaxParser handler(geometry,*dataset,&frameRate);
reader.setContentHandler(&handler);
reader.parse(source);
file.close();
// -----
// // ---- parsing goals
// -----
QFile file2(goal_file);
QXmlInputSource source2(&file2);
reader.parse(source2);
file2.close();
// parsing trains
// train type
std::map<int, std::shared_ptr<TrainTimeTable> > trainTimeTable;
std::map<std::string, std::shared_ptr<TrainType> > trainTypes;
SaxParser::LoadTrainType(tt_file.toStdString(), trainTypes);
extern_trainTypes = trainTypes;
```

Why code with multiple responsibilities is a problem

```
            bool ret = SaxParser::LoadTrainTimetable(ttt_file.toStdString(), trainTimeTable);

            extern_trainTimeTables = trainTimeTable;
            QString geofileName = SaxParser::extractGeometryFilenameTXT(fileName);

            std::tuple<Point, Point>  trackStartEnd;
            double elevation;
            for(auto tab: trainTimeTable)
            {
                  int trackId = tab.second->pid;
                  trackStartEnd = SaxParser::GetTrackStartEnd(geofileName, trackId);
                  // todo:
                  // int roomId = SaxParser::GetRoomId(tab.second->pid)
                  // int subroomId = SaxParser::GetSubroomId(tab.second->pid)
                  // elevation = SaxParser::GetElevation(geofileName, roomId, subroomId);
                  //--------
                  elevation = 0;

                  Point trackStart = std::get<0>(trackStartEnd);
                  Point trackEnd = std::get<1>(trackStartEnd);

                  tab.second->pstart = trackStart;
                  tab.second->pend = trackEnd;
                  tab.second->elevation = elevation;

                  std::cout << "=======\n";
                  std::cout << "tab: " << tab.first << "\n";
                  std::cout << "Track start: " <<  trackStart._x << ", " << trackStart._y << "\n";
                  std::cout << "Track end: " <<  trackEnd._x << ", " << trackEnd._y << "\n";
                  std::cout << " room " << tab.second->rid << "\n";
                  std::cout << " subroom " << tab.second->sid << "\n";
                  std::cout << " elevation " << tab.second->elevation << "\n";
                  std::cout << "=======\n";
            }
            for(auto tab: trainTypes)
                 std::cout << "type: " << tab.first << "\n";

            if(false==SaxParser::ParseTxtFormat(fileName, dataset,&frameRate))
                return false;

        }

        QString frameRateStr=QString::number(frameRate);
        // set the visualisation window title
        _visualisationThread->setWindowTitle(fileName);
        _visualisationThread->slotSetFrameRate(frameRate);
        //visualisationThread->setGeometry(geometry);
        //visualisationThread->setWindowTitle(caption);
        labelFrameNumber->setText("fps: " + frameRateStr+"/"+frameRateStr);

        //shutdown the visio thread
        extern_shutdown_visual_thread=true;
        waitForVisioThread();

        statusBar()->showMessage(tr("file loaded and parsed"));

        return true;
    }
```

Why large complicated functions are a problem

I think we have seen that already ...

Why lots of unused code is a problem

```
...
 src/Parsing.cpp                    |  808 +++++++++++++++++++++++++++++++++++++++
 src/Parsing.h                      |   99 +++++
 src/Pedestrian.cpp                 | 1672 --------------------------------------------------------------------------------
 src/Pedestrian.h                   |  240 ------------
 src/RenderMode.h                   |    1 +
 src/SaxParser.cpp                  | 2151 ------------------------------------------------------------------------------------------------------
 src/SaxParser.h                    |  149 --------
 src/Settings.h                     |   26 ++
 src/SimpleVisualisationWindow.cpp  |  148 -------
 src/SimpleVisualisationWindow.h    |   66 ----
 src/SyncData.cpp                   |  376 ------------------
 src/SyncData.h                     |  177 ---------
 src/SystemSettings.cpp             |  490 ------------------------
 src/SystemSettings.h               |  250 ------------
 src/ThreadDataTransfert.cpp        |  450 ----------------------
 src/ThreadDataTransfert.h          |  139 -------
 src/ThreadVisualisation.cpp        |  910 --------------------------------------------
 src/ThreadVisualisation.h          |  209 ----------
 src/TimerCallback.cpp              |  729 -----------------------------------
 src/TimerCallback.h                |  180 ---------
 src/TrailPlotter.cpp               |   49 +--
 src/TrailPlotter.h                 |   17 +-
 src/TrajectoryData.cpp             |   72 ++++
 src/TrajectoryData.h               |   44 +++
 src/TrajectoryPoint.cpp            |  309 ++++++++-------
 src/TrajectoryPoint.h              |  202 +++++-----
 src/Visualisation.cpp              |  898 +++++++++++++++++++++++++++++++++++++++++++
 src/Visualisation.h                |  232 +++++++++++
 src/events/Event.cpp               |    2 +-
 src/events/Event.h                 |   10 +-
 src/events/EventManager.h          |   14 +-
 src/extern_var.h                   |  120 ------
 src/fix/osx_thread_fix.h           |    7 -
 src/fix/osx_thread_fix.mm          |   38 --
 src/general/Macros.h               |  244 ++----------
 src/geometry/Building.cpp          | 2391 ++++++++++++++++++++++++++++++++++++++++++++++++++++++------------------------------------------------------------
 src/geometry/Building.h            |  342 ++++++++---------
 src/geometry/Crossing.cpp          |  139 ++++---
 src/geometry/Crossing.h            |  163 ++++----
 src/geometry/FacilityGeometry.cpp  | 1939 ++++++++++++++++++++++++++++++++++++++++++++++----------------------------------------------
 src/geometry/FacilityGeometry.h    |  446 ++++++++++++----------
 src/geometry/GeometryFactory.cpp   |  193 ++++------
...
125 files changed, 14909 insertions(+), 26772 deletions(-)
```

Why - Commonalities

They all make the code harder to comprehend

How to address this

Write code for HUMAN comprehension

"Performace" is almost never an acceptable excuse

Because:

You will read and need to understand the code many more times after writing it!

Lets see that parsing again

```
void MainWindow::slotOpenFile()
{
    switch(_state) {
        case ApplicationState::Playing:
            [[fallthrough]];
        case ApplicationState::NoData:
            [[fallthrough]];
        case ApplicationState::Paused: {
            const auto path = selectFileToLoad();
            if(path) {
                stopRendering();
                clearDataSet(1);
                const bool could_load_data = tryParseFile(path.value());
                if(could_load_data) {
                    _state = ApplicationState::Paused;
                    enablePlayerControls();
                    startRendering();
                } else {
                    _state = ApplicationState::NoData;
                    disablePlayerControls();

            }
        }
    }
}}
```

Lets see that parsing again

```cpp
bool MainWindow::tryParseFile(const std::filesystem::path & path)
{
    Log::Info("Trying to parse %s", path.string().c_str());
    const auto file_type = Parsing::detectFileType(path);
    switch(file_type) {
        case Parsing::InputFileType::GEOMETRY_XML:
            return tryParseGeometry(path);
        case Parsing::InputFileType::TRAJECTORIES_TXT:
            return tryParseTrajectory(path);
        case Parsing::InputFileType::UNRECOGNIZED:
            return false;
    }
}

bool MainWindow::tryParseGeometry(const std::filesystem::path & path)
{
    return Parsing::readJpsGeometryXml(path, _visualisationThread->getGeometry());
}
```

Lets see that parsing again

```
bool MainWindow::tryParseTrajectory(const std::filesystem::path & path)
{
    const auto parent_path       = path.parent_path();
    auto fileName                = QString::fromStdString(path.string());
    const auto additional_inputs = Parsing::extractAdditionalInputFilePaths(path);

    const bool readTrainTimeTable =
        additional_inputs.train_time_table_path &&
        std::filesystem::is_regular_file(additional_inputs.train_time_table_path.value());
    if(readTrainTimeTable) {
        Log::Info(
            "Found train time table file: \"%s\"",
            additional_inputs.train_time_table_path.value().string().c_str());
    }

    const bool readTrainTypes =
        additional_inputs.train_type_path &&
        std::filesystem::is_regular_file(additional_inputs.train_type_path.value());
    if(readTrainTypes) {
        Log::Info(
            "Found train types file: \"%s\"",
            additional_inputs.train_type_path.value().string().c_str());
    }

    std::map<std::string, std::shared_ptr<TrainType>> trainTypes;
    if(readTrainTypes) {
        // TODO(kkratz): This just continues on error, fixup impl.
        Parsing::LoadTrainType(additional_inputs.train_type_path.value().string(), trainTypes);
    }

    std::map<int, std::shared_ptr<TrainTimeTable>> trainTimeTable;
    if(readTrainTimeTable) {
        // TODO(kkratz): This just continues on error, fixup impl.
        bool ret = Parsing::LoadTrainTimetable(
            additional_inputs.train_time_table_path.value().string(), trainTimeTable);
    }
    if(readTrainTimeTable && readTrainTypes) {
        _visualisationThread->setTrainData(std::move(trainTypes), std::move(trainTimeTable));
    }


    if(!additional_inputs.geometry_path ||
       !tryParseGeometry(additional_inputs.geometry_path.value())) {
        return false;
    }
```

Lets see that parsing again

```
    std::tuple<Point, Point> trackStartEnd;
    double elevation;
    for(auto tab : trainTimeTable) {
        int trackId   = tab.second->pid;
        trackStartEnd = Parsing::GetTrackStartEnd(
            QString::fromStdString(additional_inputs.geometry_path.value().string()), trackId);
        elevation = 0;

        Point trackStart = std::get<0>(trackStartEnd);
        Point trackEnd   = std::get<1>(trackStartEnd);

        tab.second->pstart    = trackStart;
        tab.second->pend      = trackEnd;
        tab.second->elevation = elevation;

        Log::Info("=======\n");
        Log::Info("tab: %d\n", tab.first);
        Log::Info("Track start: [%.2f, %.2f]\n", trackStart._x, trackStart._y);
        Log::Info("Track end: [%.2f, %.2f]\n", trackEnd._x, trackEnd._y);
        Log::Info("Room: %d\n", tab.second->rid);
        Log::Info("Subroom %d\n", tab.second->sid);
        Log::Info("Elevation %d\n", tab.second->elevation);
        Log::Info("=======\n");
    }
    for(auto tab : trainTypes)
        Log::Info("type: %s\n", tab.first.c_str());

    double fps;
    // TODO(kkratz): Figure out why this is required
    _trajectories.clearFrames();
    ui.actionFirst_Group->setEnabled(true);
    ui.actionFirst_Group->setChecked(true);
    if(false == Parsing::ParseTxtFormat(fileName, &_trajectories, &fps)) {
        return false;
    }

    QString frameRateStr = QString::number(fps);
    _visualisationThread->slotSetFrameRate(fps);
    labelFrameNumber.setText("fps: " + frameRateStr + "/" + frameRateStr);

    statusBar()->showMessage(tr("file loaded and parsed"));

    return true;
}
```

Discussion