# Indoor and outdoor channel simulations using Winprop API.

Name: Ren Zhenyu

Date: 2021.7.12

# Overview

- Simulation goal: simulate the channel variations and received power variations due to impacts of moving objects or moving RXs.
- Environment settings: `MSVC(VS 2019)` + `Clion 2021` + `WinProp API 2021 for windows`. Use `CMakeLists.txt` to compile the project.
- Two simulation schemes:
  - Time variant prediction.
    - Can only add time-variant components in `.idb`
    - Generate the prediction result on the whole "map" at each time instance.
  - Trajectory prediction.
    - Could use C++ command to control.
    - Generate the prediction result along the trajectory.

- Simulation codes: https://github.com/rzy0901/testWinprop (private access now.)

# Simulation settings

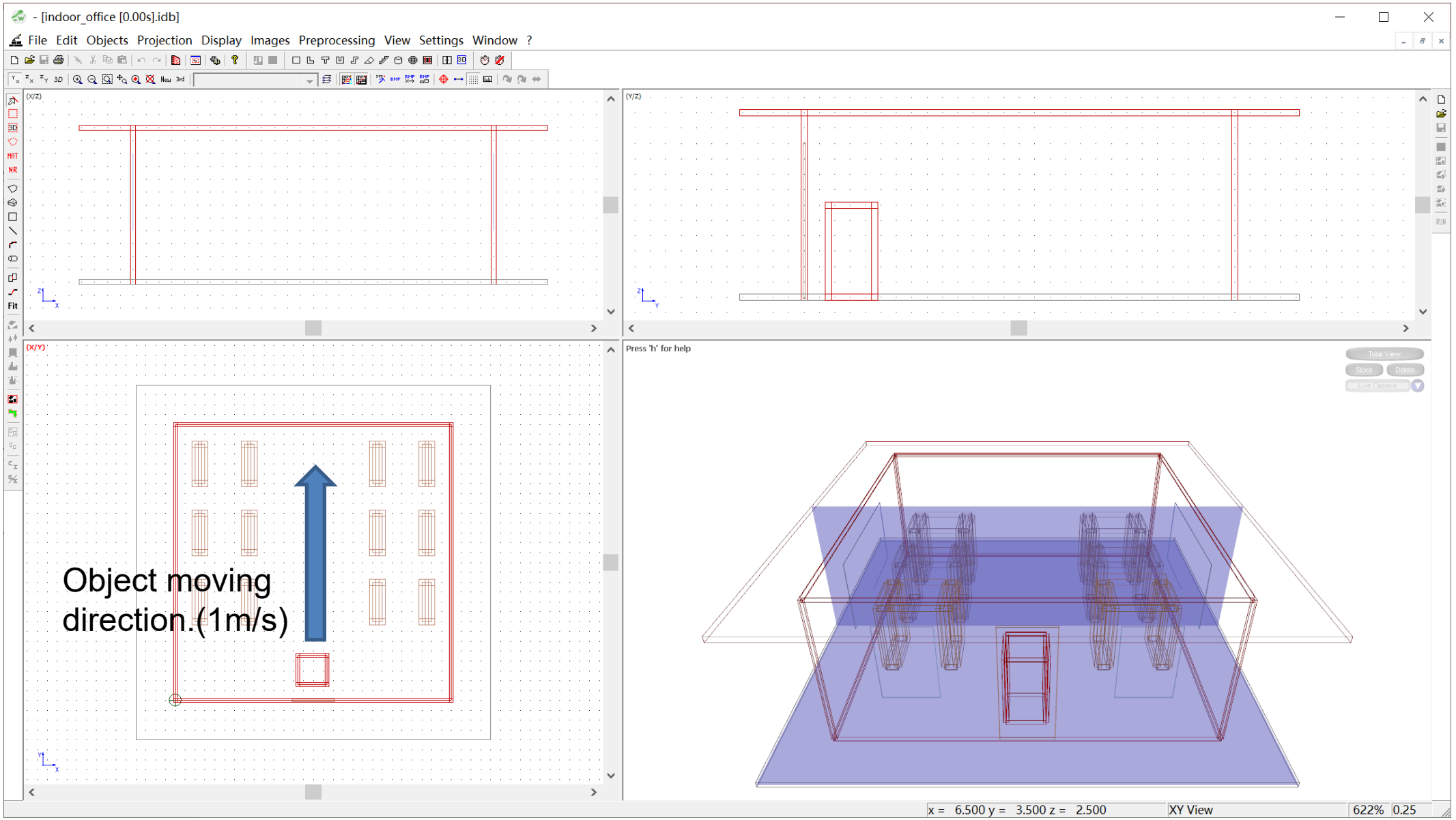
- Simulation scenarios:

| | Indoor | Outdoor (Urban) |
|---|---|---|
| Trajectory | indoor_trajectory.cpp | outdoor_trajectory.cpp |
| Time variant | indoor_time_variant.cpp | Not completed yet (Treated as indoor.) |
| Ray-Tracing | SRT | SRT & DPM & IRT |

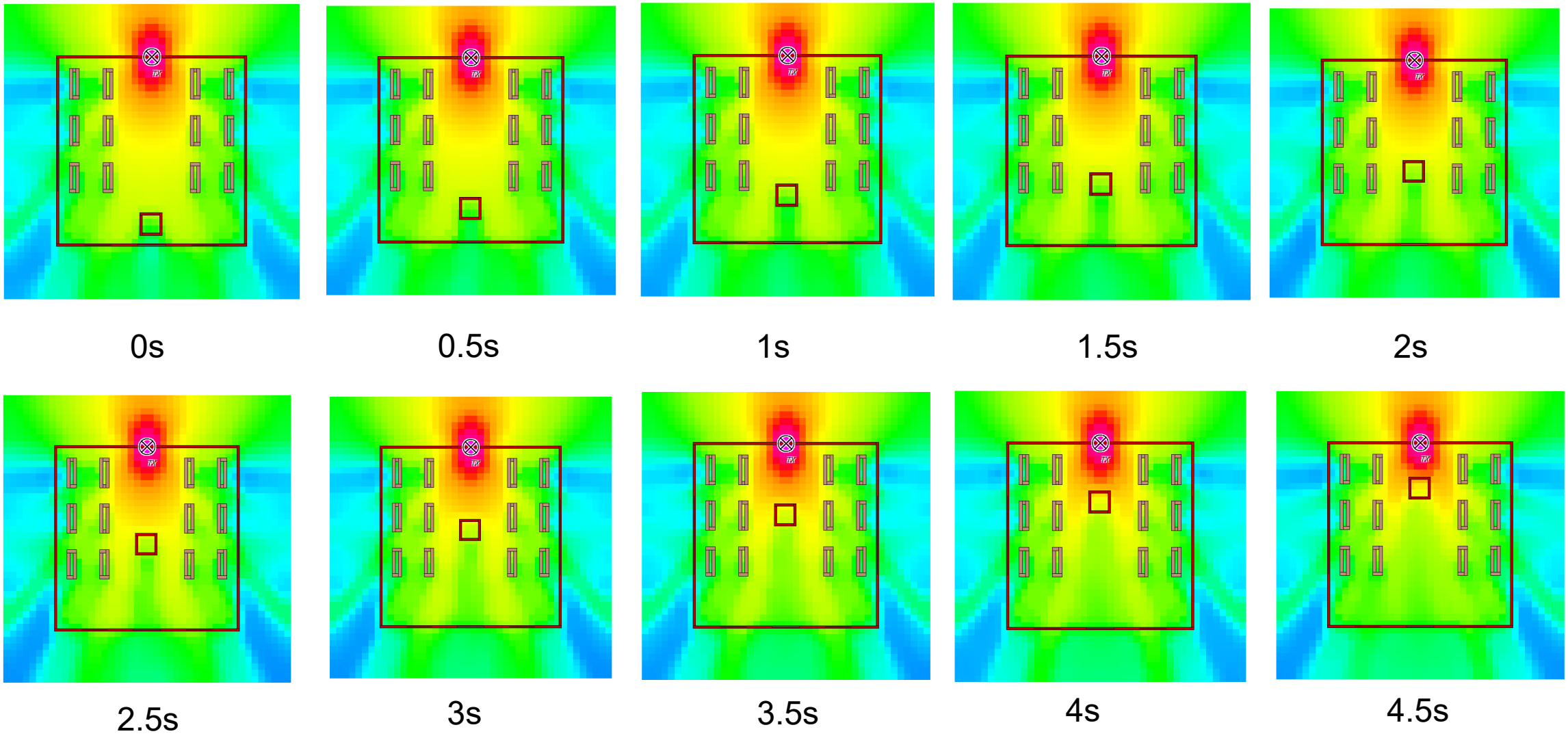
- Simulation settings:

| | TX (10W) | RX | Ray-Tracing | Specifications | Prediction Height |
|---|---|---|---|---|---|
| indoor_trajectory.cpp | omni, 1.25m, 2000 MHz | omni, 1.25m | SRT | 1m/s RX | 1.25m |
| indoor_time_variant.cpp | omni, 1.25m, 2000 MHz | omni, 1.25m | SRT | 1m/s Moving Object | 1.25m |
| outdoor_trajectory.cpp | omni, 15m, 2000 MHz | omni, 1.5m | DPM, SRT (Treated as indoor), IRT (Preprocess the database) | 10m/s moving RX | 1.5m |
| Time variant scenario in outdoor (Not completed yet.) | | | | | |

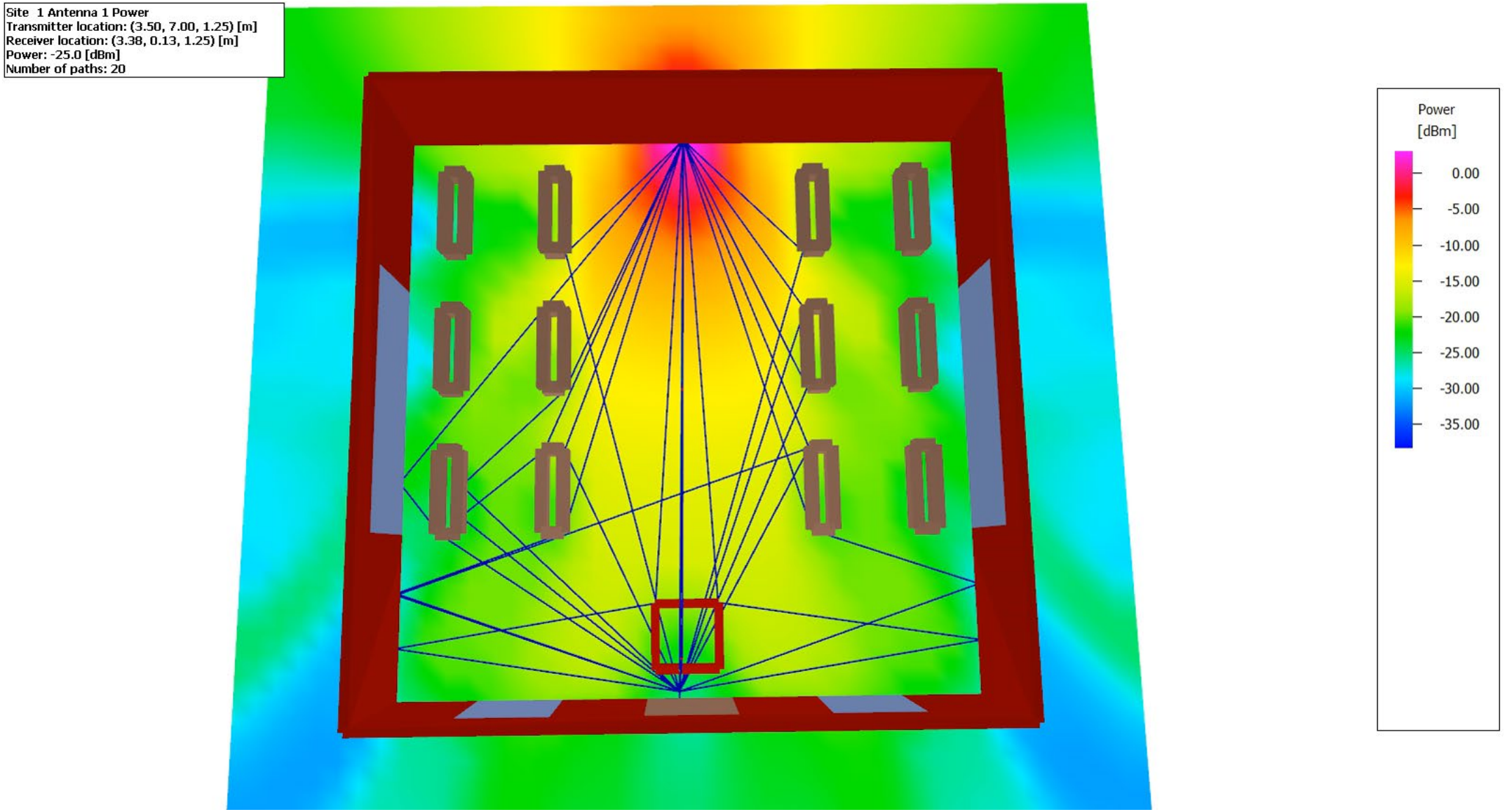
# Simulation database: indoor_time_invariant.cpp



# Result: indoor_time_invariant.cpp

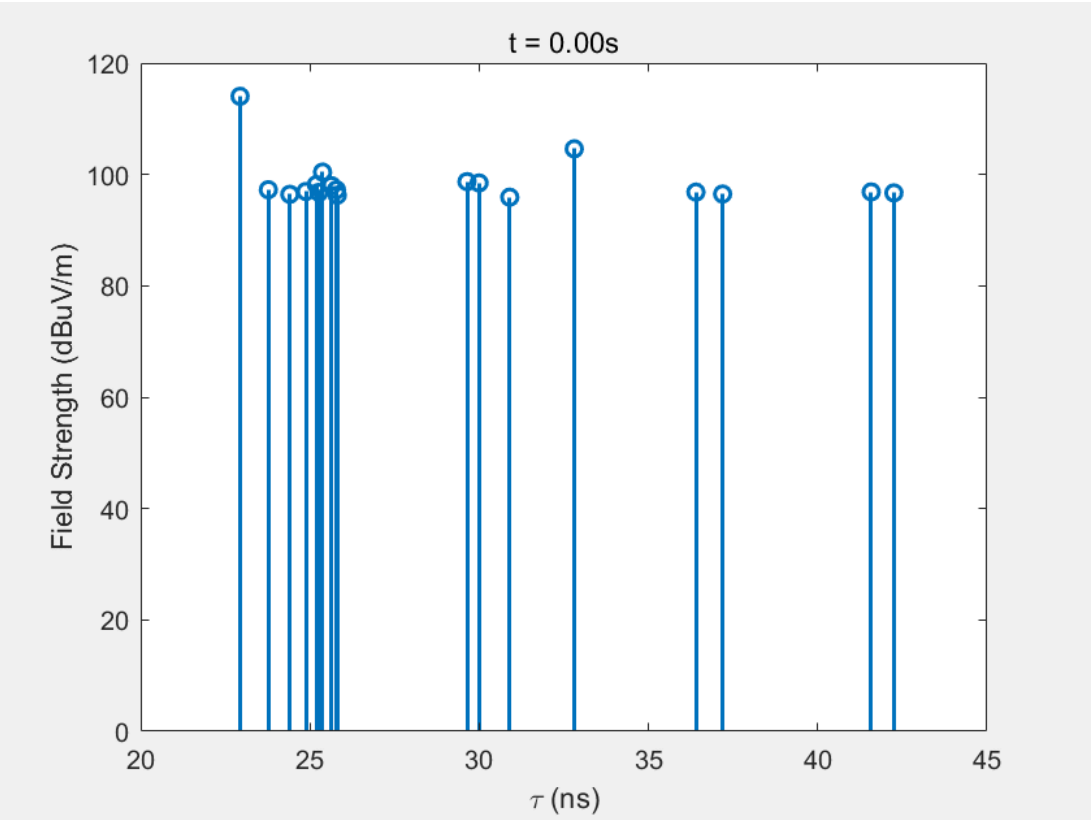


# Result: indoor_time_invariant.cpp （ at pixel [3.5m:3.75m,0m:0.25m] ）

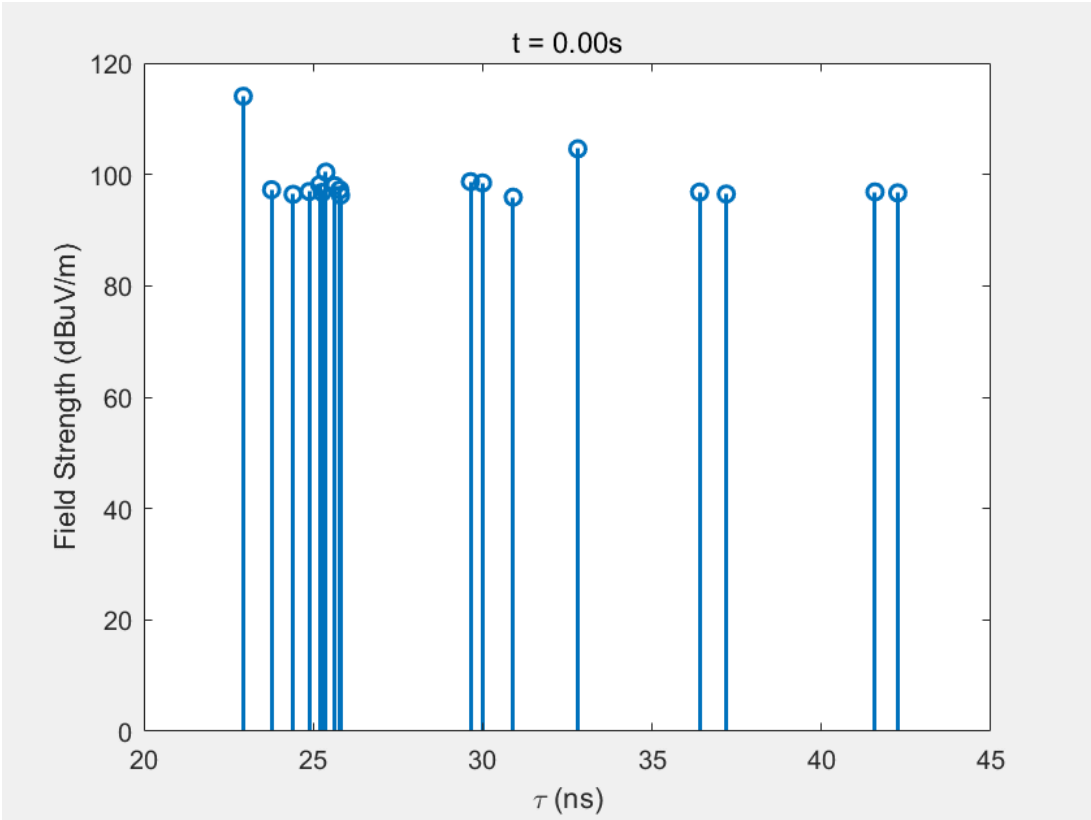


# CIR at pixel [3.5m:3.75m,0m:0.25m]

- Resolution here: 0.25m




Time step: 0s: 0.05s : 0.5s

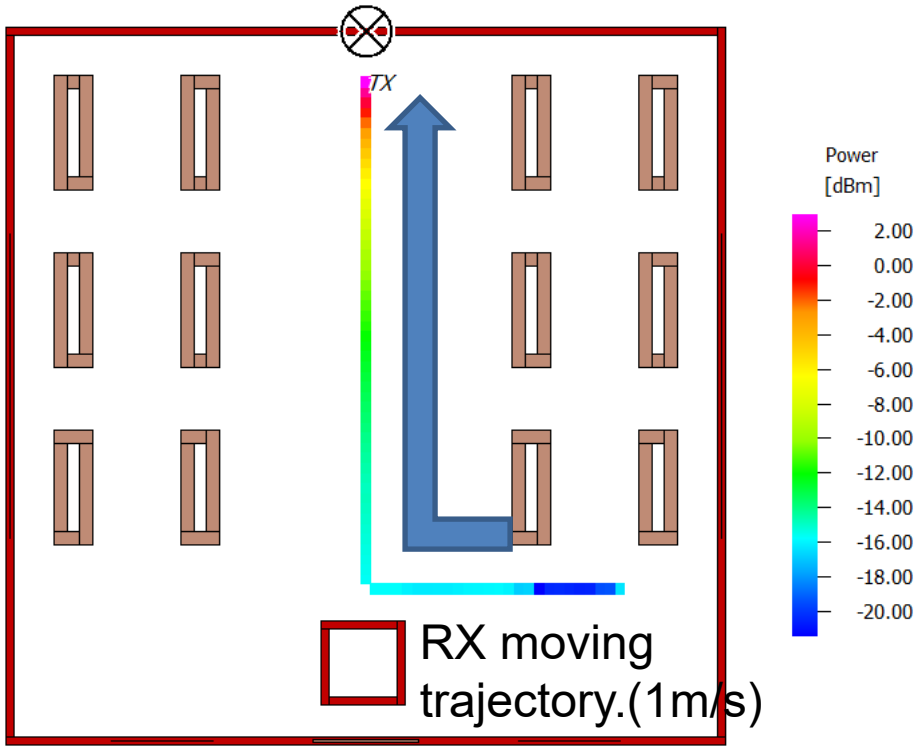


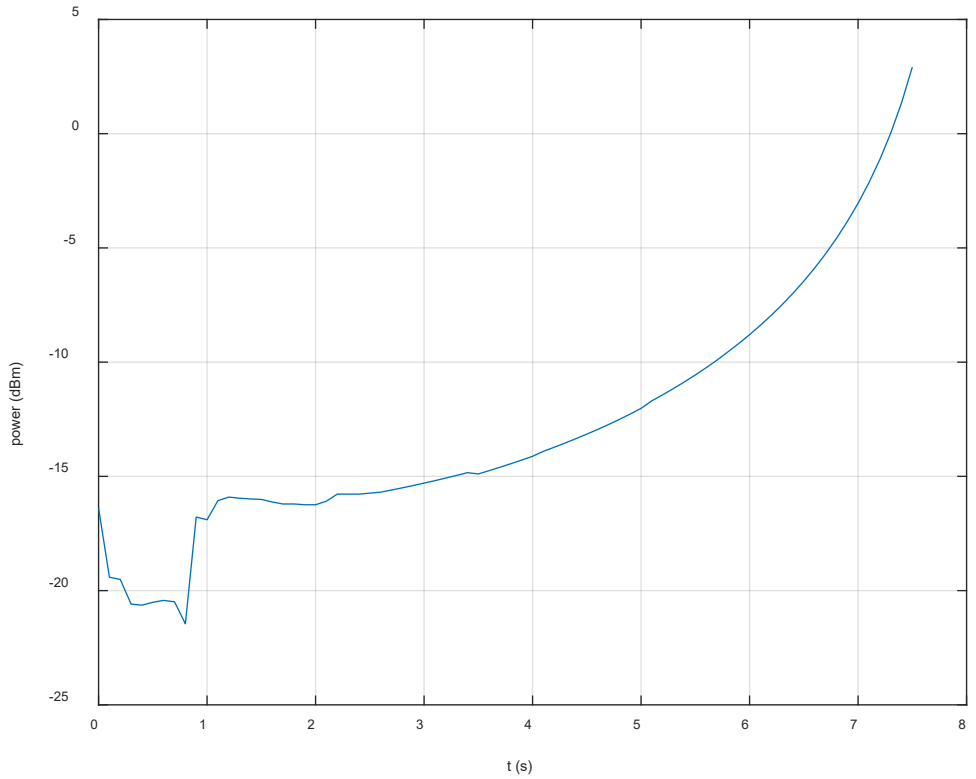

Time step: 0s: 0.01s : 0.1s

# Result: indoor_trajectory.cpp

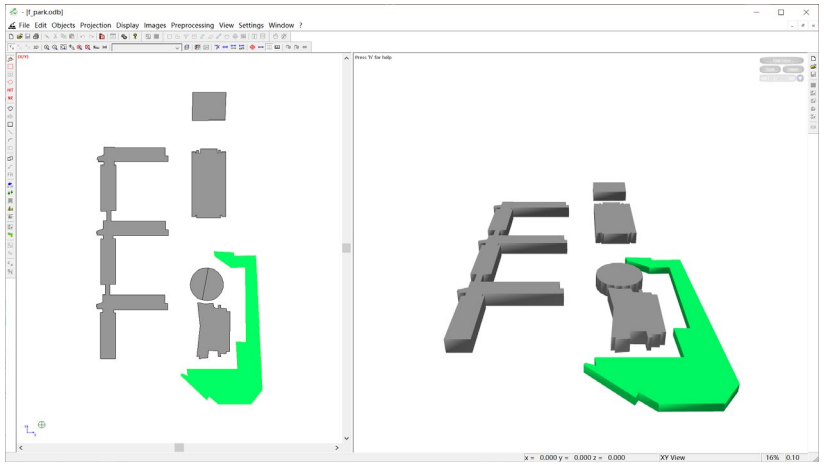
- No object moving here. Only consider the mobility of RX.
- Trajectory could be defined by locations of corners and velocity using C++ command. (Multiple RXs are also supported.)

```
WinPropTrajectory.NrPoints = 3; // Number of corners of the trajectory.
WinPropTrajectory.samplingResolution = 0.1;
WinProp_Trajectory_Point points[3];
points[0] = { .location: COORDPOINT{ .x: 6.00, .y: 1.50, .z: 1.25}, .velocity: 1.0, .pitch: 0, .roll: 0, .yaw: 0};
points[1] = { .location: COORDPOINT{ .x: 3.5, .y: 1.5, .z: 1.25}, .velocity: 1.0, .pitch: 0, .roll: 0, .yaw: 0};
points[2] = { .location: COORDPOINT{ .x: 3.5, .y: 6.5, .z: 1.25}, .velocity: 1.0, .pitch: 0, .roll: 0, .yaw: 0};
WinPropTrajectory.Points = points;
WinPropTrajectory.PointSize = 0.1; // Resolution for each sampling point, default by 1.
```



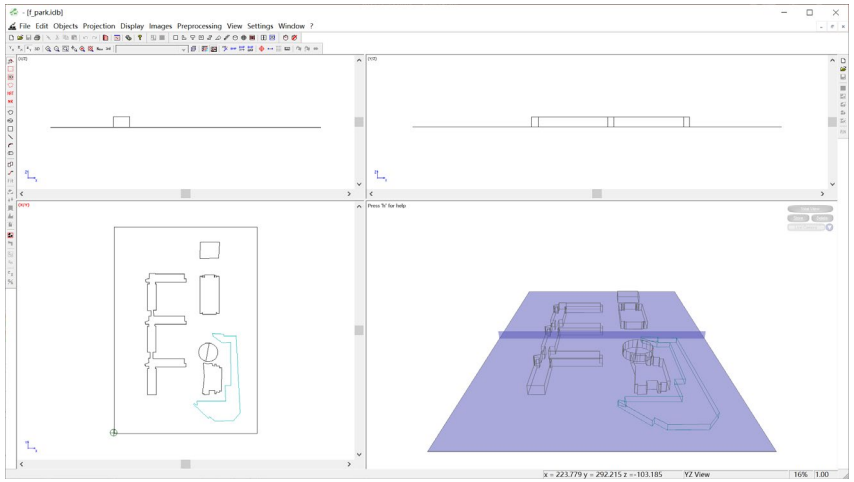



# Simulation database: outdoor_trajectory.cpp



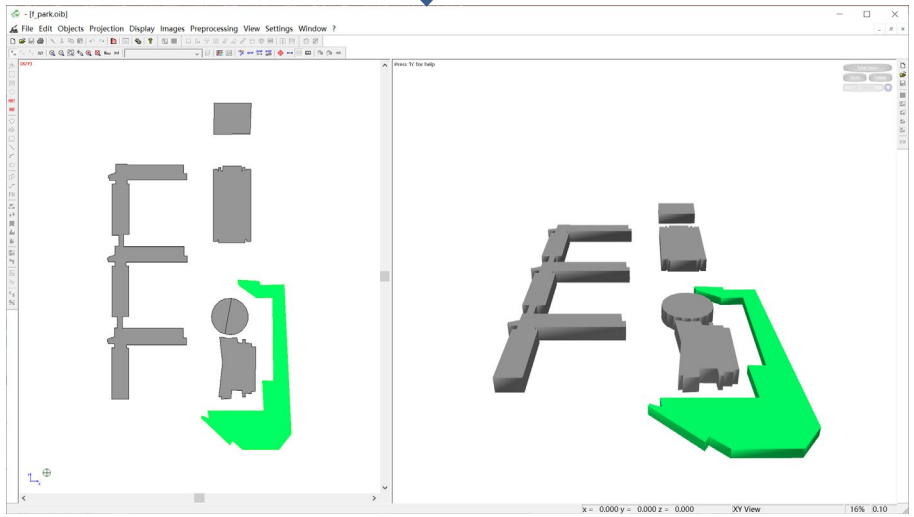
f_park.odb (outdoor database)

export



f_park.idb (indoor database)

preprocess



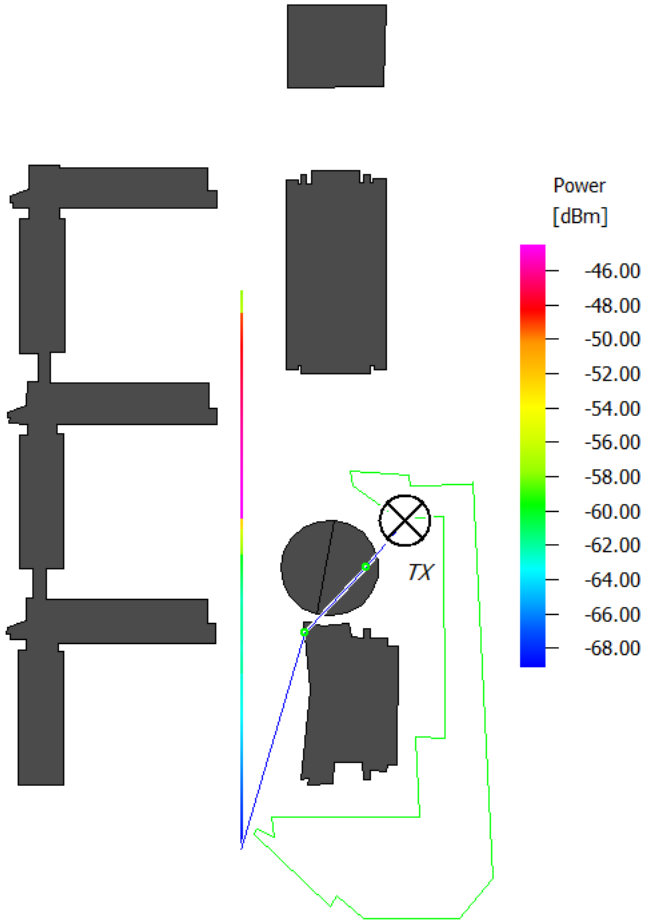
f_park.oib (preprocesed outdoor database)

Three Ray-Tracing method:

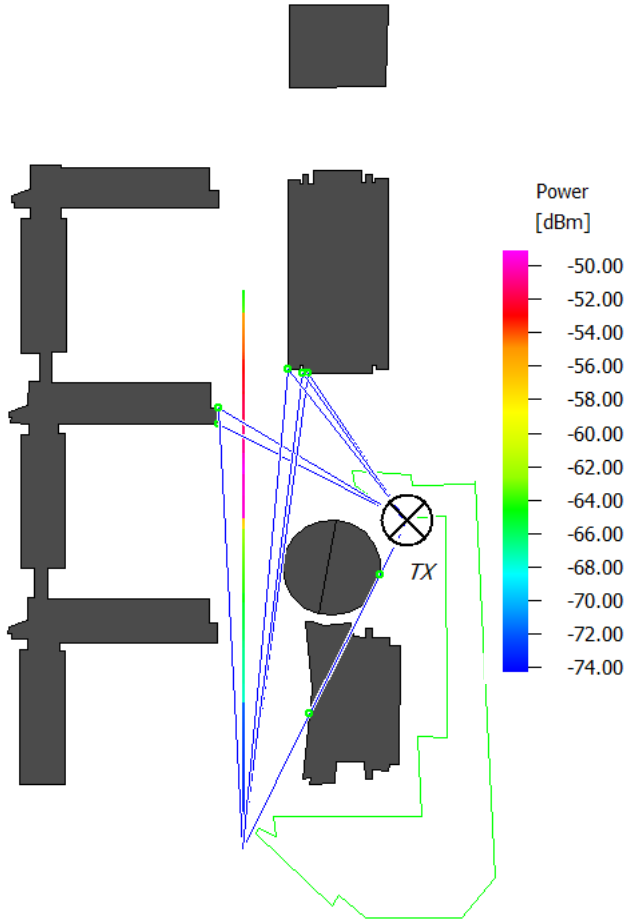
- DPM
  - f_park.odb
- IRT
  - f_park.oib
- SRT
  - f_park.idb

# Result: outdoor_trajectory.cpp

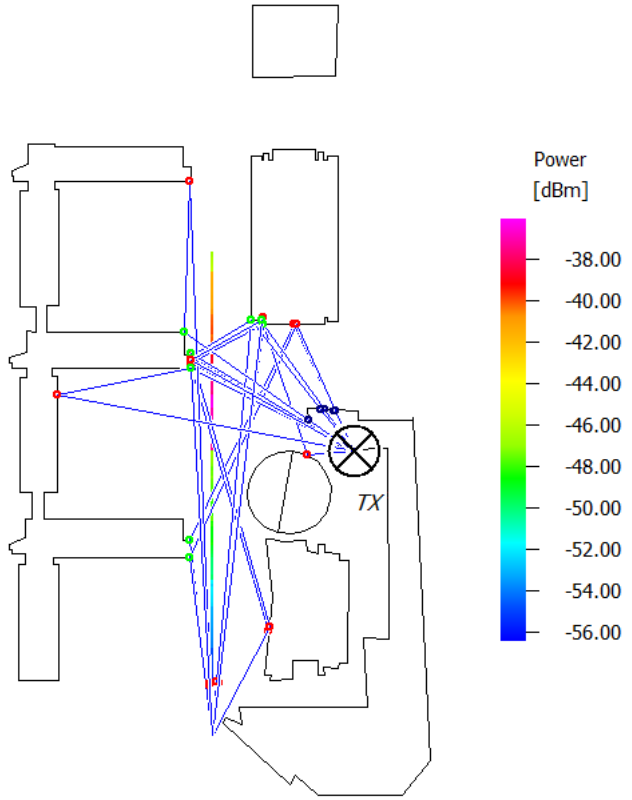
- RX moves from the South to the North with speed 10m/s.




DPM
(1 path predicted at start point)

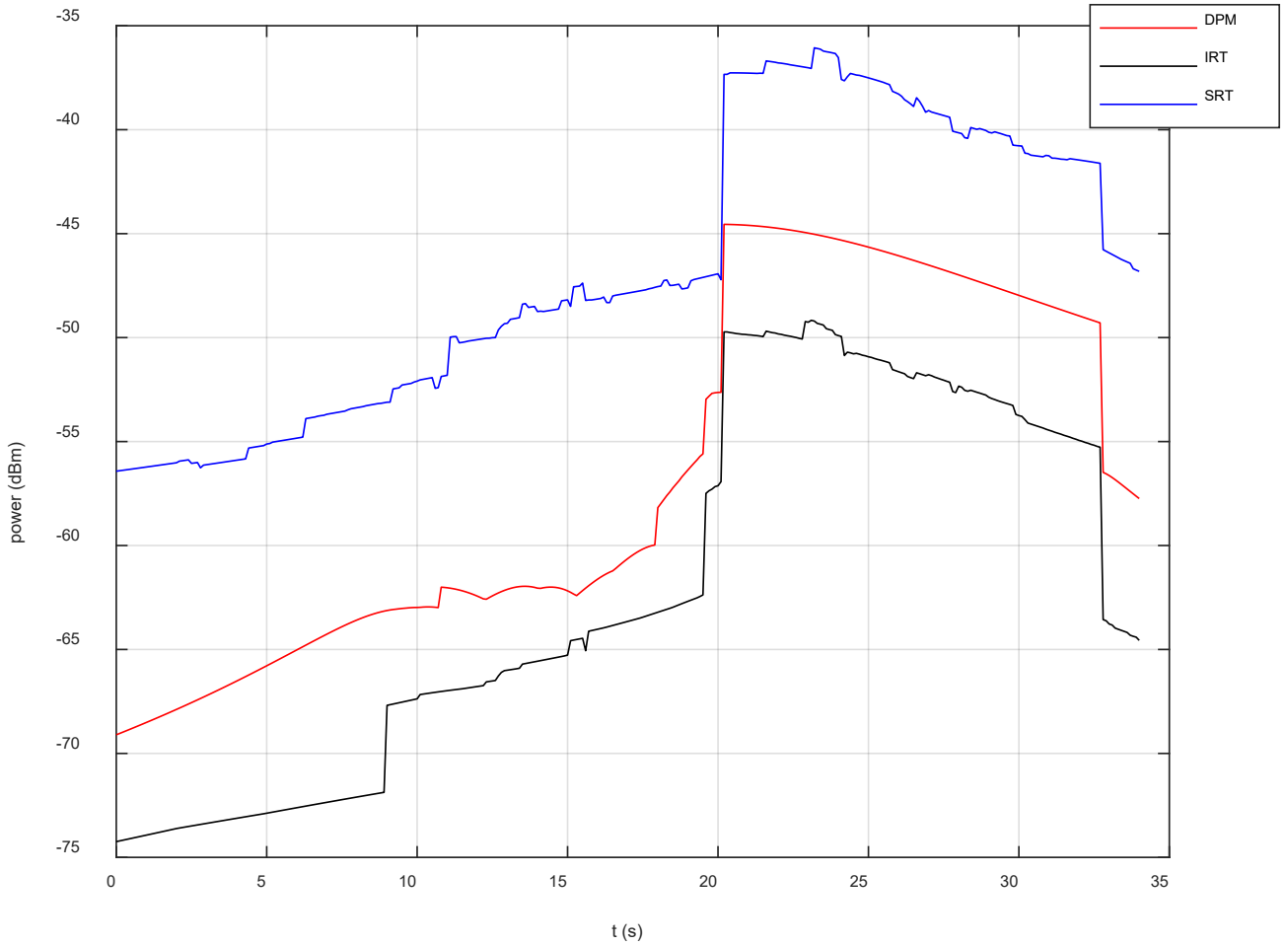



IRT
(5 paths predicted at start point)




SRT
(19 paths predicted at start point)

# Result: outdoor_trajectory.cpp (cont.)




Accuracy:
SRT > DPM > IRT.

## Remaining problems.

- Time-invariant scenario in outdoor (urban) database.
  - Winprop does not support time-variant simulations in outdoor database.
    - Winprop just support trajectory simulations in outdoor database.
  - One possible alternative approach:
    - Treat the outdoor scenario as indoor.
- Add directional antenna to RX.
- Material settings.
  - Physical properties of materials in this project are defined at 2000 MHz.
    - It is required to calculate these physical properties again at 60 GHz.
  - Design specific indoor models for cars and working humans rather than using a simple box.

# Appendix: C++ environment settings.

- Do not use C++ environment in Linux or virtual machines. (can not compile winprop API)
- Approach 1: install VS2019 directly. (Most straightforward but not recommended.)
  - poor portability: all the codes must be placed at a specific folder.
  - Use absolute path to include header files. (Not compatible with CMakeLists.txt.)
- Approach 2: use VS2019 to set the Clion.
  - Download feko2021 or feko2020.
  - Download VS2019 and Download Clion.
  - Add “%FEKO_HOME%/bin” to system environment variable “path”.
    - % FEKO_HOME % refers to the installation path of feko.
  - In Clion, enter File -> Settings-> Build, Execution, Deployment -> Toolchains, top the VS environment, Then select architecture with 64 bit.
  - Use CMakeLists.txt to build the project.