

KBSW180141 Win-32- Save and Load Composite Map

This document introduces the demo project of "composite_map_demo", including how to save and load maps of stcm format(composite map).

Content

- [IDE Preparation](#)
 - [Software](#)
 - [Hardware](#)
 - [Download](#)
 - [Compiling](#)
 - [Code](#)
-

IDE Preparation

• Software

- Visual Studio 2010 SP1
- Slamware Windows SDK:[Slamware Windows SDK](#)
- RoboStudio(for map display):[Robostudio installer](#)
- Sample Code:



Higher version of Visual Studio will cause errors. sometime you will need to upgrade SP1 package to make your VS compatible with .Net Framework.

• Hardware

Either one of following

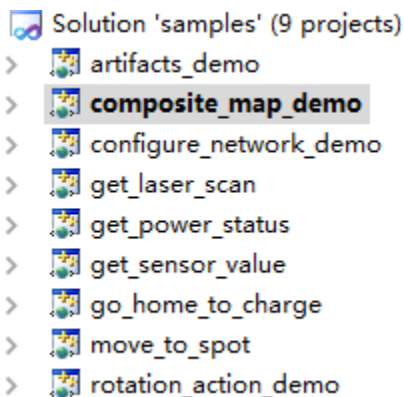
- Slamware SDP mini
 - Slamware SDP
 - Slamware Kit
 - Zeus/Apollo robot base
-

Download

[Win32-Demo](#)

Compiling

1. Right click on "composite_map_demo" project, set as StartUp project.



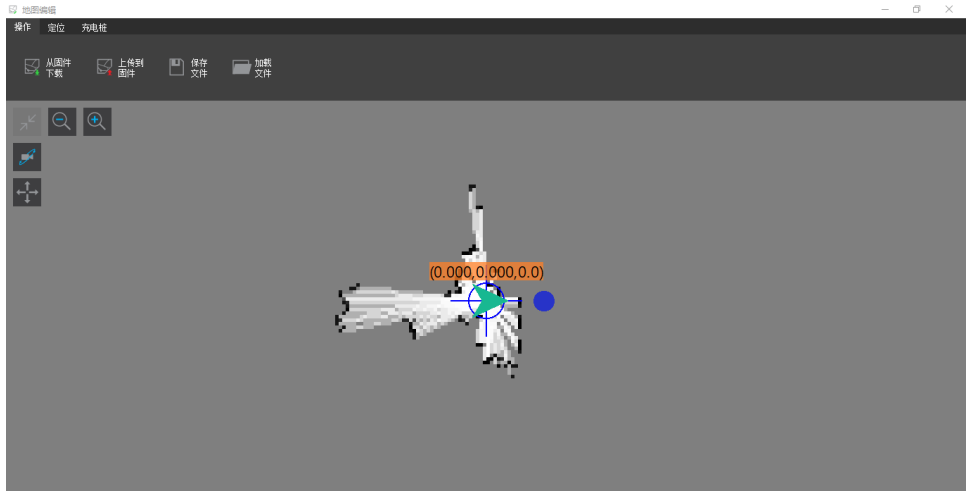
2. Right click on "composite_map_demo", then " Properties"configure "include" and "lib" directories to the corresponding folder path of Slamware SDK.



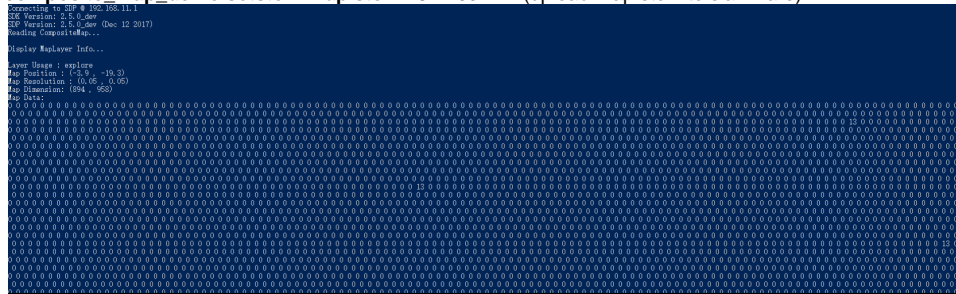
It's not necessary to copy files to the project directory, user will only need to configure the folder path of SDK.

[illegible]

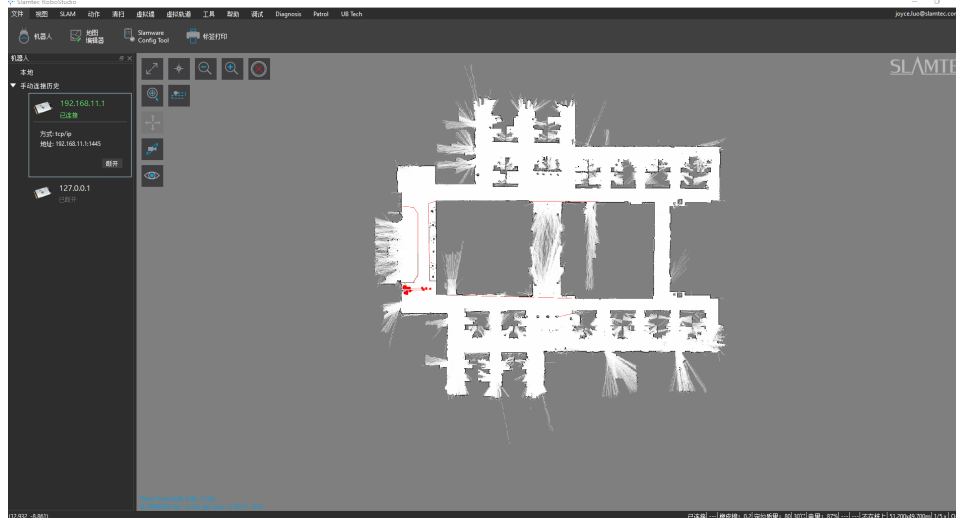
The generated maps could be opened with the "map editor" plug-in of the Robostudio



- **composite_map_demo setstcm map.stcm 192.168.11.1**(upload map.stcm to slamware)



if map is successfully loaded, it will be shown in Robostudio.



Code

- Save maps from Slamware to a local directory.

composite map

```
bool StcmMapWriter(const std::string file_name, SlamwareCorePlatform platform) {
    CompositeMap composite_map = platform.getCompositeMap();
    CompositeMapWriter composite_map_writer;
    std::string error_message;
    bool result = composite_map_writer.saveFile(error_message, file_name, composite_map);
    return result;
}
```

- Load composite map from a local directory.

composite map

```
bool StcmMapReader(const std::string file_path, rpos::core::Pose pose, SlamwareCorePlatform platform) {
    CompositeMapReader composite_map_reader;
    std::string error_message;
    boost::shared_ptr<CompositeMap> composite_map(composite_map_reader.loadFile(error_message, file_path));
    if (composite_map) {
        platform.setCompositeMap((*composite_map), pose);
        return true;
    }
    return false;
}
```

- Display data of different map layers from a composite map, including virtual walls and virtual tracks, etc.

map layer

```
CompositeMapReader composite_map_reader;
std::string error_message;
boost::shared_ptr<CompositeMap> composite_map(composite_map_reader.loadFile(error_message, file_path));
if (composite_map) {
    for (auto it = composite_map->maps().begin(); it != composite_map->maps().end(); ++it) {
        auto layer = *it;
        std::string usage = layer->getUsage();
        std::string type = layer->getType();
        std::cout << "Layer Usage : " << usage << std::endl;
        //get grid map layer
        if (type == GridMapLayer::Type) {
            auto grid_map = boost::dynamic_pointer_cast<GridMapLayer>(layer);
            std::cout << "Map Position : (" << grid_map->getOrigin().x() << " , " <<
                grid_map->getOrigin().y() << ")" <<std::endl;
            std::cout << "Map Resolution : (" << grid_map->getResolution().x() <<
                " , " << grid_map->getResolution().y() << ")" <<std::endl;
            std::cout << "Map Dimension: (" << grid_map->getDimension().x() <<
                " , " << grid_map->getDimension().y() << ")" <<std::endl;
            std::cout << "Map Data:" << std::endl;
            for (auto it = grid_map->mapData().begin(); it != grid_map->mapData().end();
++it) {
                std::cout << (int)*it << " " ;
            }
            std::cout << std::endl << std::endl;
        }
        //get line map layer
        else if (type == LineMapLayer::Type) {
            auto line_map = boost::dynamic_pointer_cast<LineMapLayer>(layer);
            for (auto it = line_map->lines().begin(); it != line_map->lines().end();
++it) {
                auto line = it->second;
                std::cout << "start: (" << line.start.x() << " , " << line.start.y()
<< ")" << std::endl;
                std::cout << "end: (" << line.end.x() << " , " << line.end.y() <<
")" << std::endl;
            }
            std::cout << std::endl;
        }
        //get pose map layer
        else if (type == PoseMapLayer::Type) {
            auto pose_map = boost::dynamic_pointer_cast<PoseMapLayer>(layer);
            for (auto it = pose_map->poses().begin(); it != pose_map->poses().end();
++it) {
                auto pos = it->second;
                std::cout << "Position : (" << pos.pose.x() << " , " << pos.pose.y()
<< ")" << std::endl;
            }
            std::cout << std::endl;
        }
        else if (type == PointsMapLayer::Type) {
            //TODO: get Points map layer
            std::cout << std::endl;
        }
        else {
            //TODO: get unknown map layer
            std::cout << std::endl;
        }
    }
}
```