Tampere University Unit of Computing Sciences

COMP.SE.610 Software Engineering Project 1

COMP.SE.620 Software Engineering Project 2

G12

# Simulation environment for point cloud scanning augmented reality application development

## Spider game port documentation

# Introduction

### Purpose and scope of project

We succeeded in porting the pre-existing spider game onto our simulation.
(Issue #25)  (Branch of port)
This document describes what steps were necessary to achieve this, with the aim of being a useful reference for future porting of games between the core application and our platform. The description may not be exact or exhaust all the actions that were done in the Unity editor, but it should be a sufficient reference for this purpose.

The port has happened in stages, due to the simulation platform being continuously developed during the project.

# Spider game port

- Export package from the core application (from context menu of the spider game Scene). You are exporting a package of the Scene of the spider game.

- Import into the simulation Unity project

- Exporting the package will miss the ConeCollider prefab object (Assets/ConeCollider/Resources/Prefab). Copy the whole ConeCollider folder by hand into the simulation project.

- fix possible errors (delete files or implement fix depending on situation)

    -errors to fix:
    - Delete various AR scripts not needed in the simulation which have errors, delete them
    - There is prompt for api changes, asking if you want to fix them. Select [yes, for these files only.]

- There are errors in some TextMesh Pro –scripts if you imported those from the core application,  about some UnityEngine Vector types. Fix those by hand by changing to appropriate types

- Copy-paste Terrain-scene objects into SpiderGame-scene

- Put vacuum_collider object from under XR Origin to under Player->Main Camera

- Remove XR Origin-object and AR Session-object

-Remove duplicate Directional light

- Activate EventSystem –object and press the Fix button (Replace with InputSystemUIInputModule) for the Input System UI Input Module component of the object in the Inspector

- change shader for spider material (URP - Lit), put spider texture in its basemap

- Add the Tags that the code needs (will prompt exceptions when missing)

- Attach ARMesh –Tag to the terrains wanted to be used for spider spawning (we dont have the point cloud mesh that the real application has, yet....)

- add hack for Default-ParticleSystem (conflicts with our pointcloud scanning)
(See spider_game_port –branch commit history)

- add hack for vacuum_collider functionality
(See spider_game_port –branch commit history)

- Add LayerMask "ARMesh" to spiderspawner.cs raycast to only spawn spiders on surfaces tagged as ARMesh

- Add asmdef file to SceneSelector to have StaticVariables available

- Add asmdef file to com.8bitgoose.asciifbxexporter Editor and Runtime folders and add the Runtime asmdef to the Editor asmdef's refrences. The editor asmdef should have "Editor" in includedPlatforms

- Do the same as above for jp.keijiro.pcx@1.0.1 Editor and Runtime folders.

- Add other missing dependencies to asmdef files to prevent errors

- See project gitlab issue #100 (https://gitlab.tuni.fi/cs/gamilidar/simulated_forest_scanning/-/issues/100): for some reason, our raycast for point scanning scanned points of the spiders if the spiders were in the air (walking on player collider / map edge). To prevent this from happening, spiderspawner.cs had a line of code added which sets a layer for the spawned spiders telling the raycast to ignore them: **createdObject.layer = LayerMask.NameToLayer("Ignore Raycast");**
This is the simplest solution, as at this point of the project the group shouldn't start refactoring the point scanning for this reason.

- The class StaticVariables from StaticVariables.cs must be given the public scope

- Copy the Keybind Manager object into the spider scene