ALTRUIST System Documentation

Authors Name Omitted

May 2, 2023

Abstract

This document describes how to install and use of ALTRUIST: A Multi-platform Tool for Conducting QoE Subjective Tests

Contents

1 Introduction

In the context of tools for conduction QoE experiments, we propose ALTRUIST a light-way multi-platform distributed system, that can automatically perform common tasks required to successfully conduct an experiment, collect and catalog the data. Its code is open source, and can be used to modify existent or include new features. ALTRUIST was built in Unity[1], a game engine that allows deploying applications in multiple platforms including PC, Linux, Mac, Android, and others. We leverage this feature, to crate a system that can be deployed in different devices each, as a single application, capable to communicate among them following a client/server network architecture.

In its current form, ALTRUIST is distributed as a unity package to be included in any project created within Unity environment. In addition, we also provide the source code from the where the package was created. However we advise using the Unity package, as the standard way of installing since it is boundless to a particular editor version. The steps described in this document assumes the reader has basic knowledge of Unity game engine and C# coding.

2 Package Installation

In order the setup a unity project and be able to see the ALTRUIST framework, follow t he necessary steps below:

1. To start, download install UnityHub [2].
2. Drag and drop Mirror-73.0.0.unitypackage
3. Drag and drop ALTRUIST.unitypackage
4. Open the Scene ../Assets/Scenes/New_System which is pre configured with prefabs to start using the the library.

[1] https://unity.com/
[2] https://unity.com/download

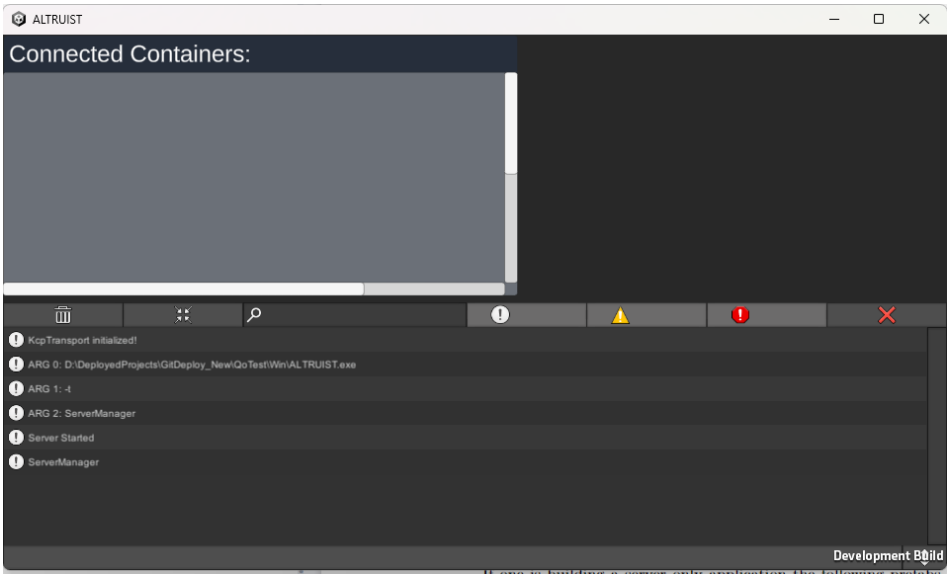

Figure 1: UI view of the standard ServerManager app, that supports connected containers.

3 Important Prefabs

Regardless of which agents one choose to include in a experiment settings, the following prefabs should always be included in a scene.

- /Assets/Prefabs/QoTestPrefabs/PrefabsToAdd/NetComponent.prefab: Responsible for either hosting or finding a server.
- /Assets/Prefabs/QoTestPrefabs/PrefabsToAdd/AutoTask_Manger.prefab: Scripts loader to automatically control a test. The inclusion of scripts in the list is optional. However the prefab must exist in the scene to avoid errors in the code.

As example, when we built our questionnaire application, we created it as separate unity project, import the package of one of many questionnaires tool-kits available online, and than included the necessary prefabs. Following this approach our Qt application was able to find our main testing tool, and communicate although they were originated from two different unity projects.

If one is building a server only application the following prefabs can be useful for including in a scene:

- ../Assets/Prefabs/QoTestPrefabs/PrefabsToAdd/ServerCtr.prefab : Contains canvas for listing which containers are currently connected to the server.

As an example the figure Fig. 1 shows the space name “Connected Containers” that will be designated to list all the connections.