Shima User Manual

Revision Sheet

Release No	Date	Revision Description
Rev. 1	11/01/2017	Initial Work

Table of contents

I. General overview

1. System overview

This software is a simulator for intelligent environment. It provides users with tools for:
- modeling intelligent environments,
- designing interaction scenario,
- generating datasets.

It is compatible with MacOS and Windows platforms.

2. Organization of the Manual

This manual is structured as the following:
- Chapter II provides installation instructions.
- Chapter III provides on overview of using instruction
- Chapter IV provides extension
-

3. Acronyms and Abbreviations

- RFID : Radio Frequency Identification
- PIR : Passive Infrared
- UI : User Interface
- BT: Behavior Tree

II. System installation

The simulator runs on Unity Game Engine. Unity is available on https://unity3d.com/. Unity is free for personal use, however, it requires to create an account to download. System requirements are available at https://unity3d.com/unity/system-requirements.

Python is required to generated datasets. Python is available at https://www.python.org.

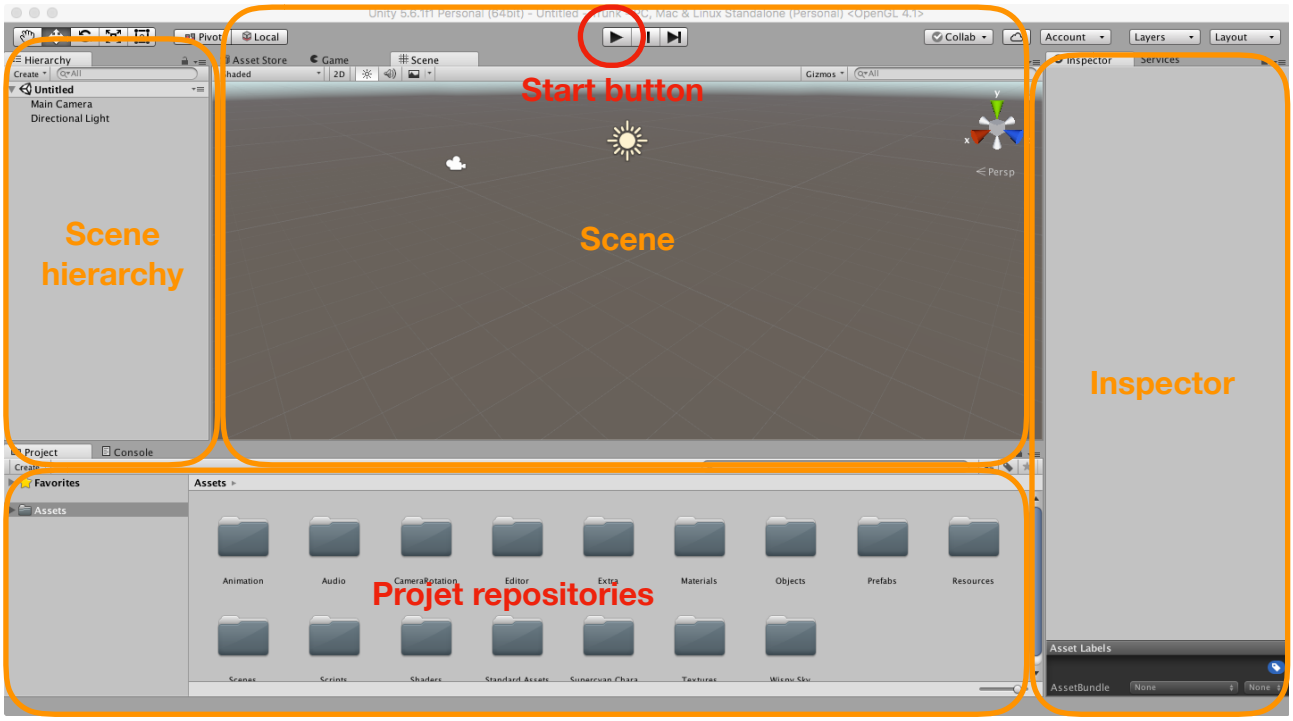
Once Unity is installed and configured, download the project at https://github.com/lannyck/shima and open it.

III. Getting Started

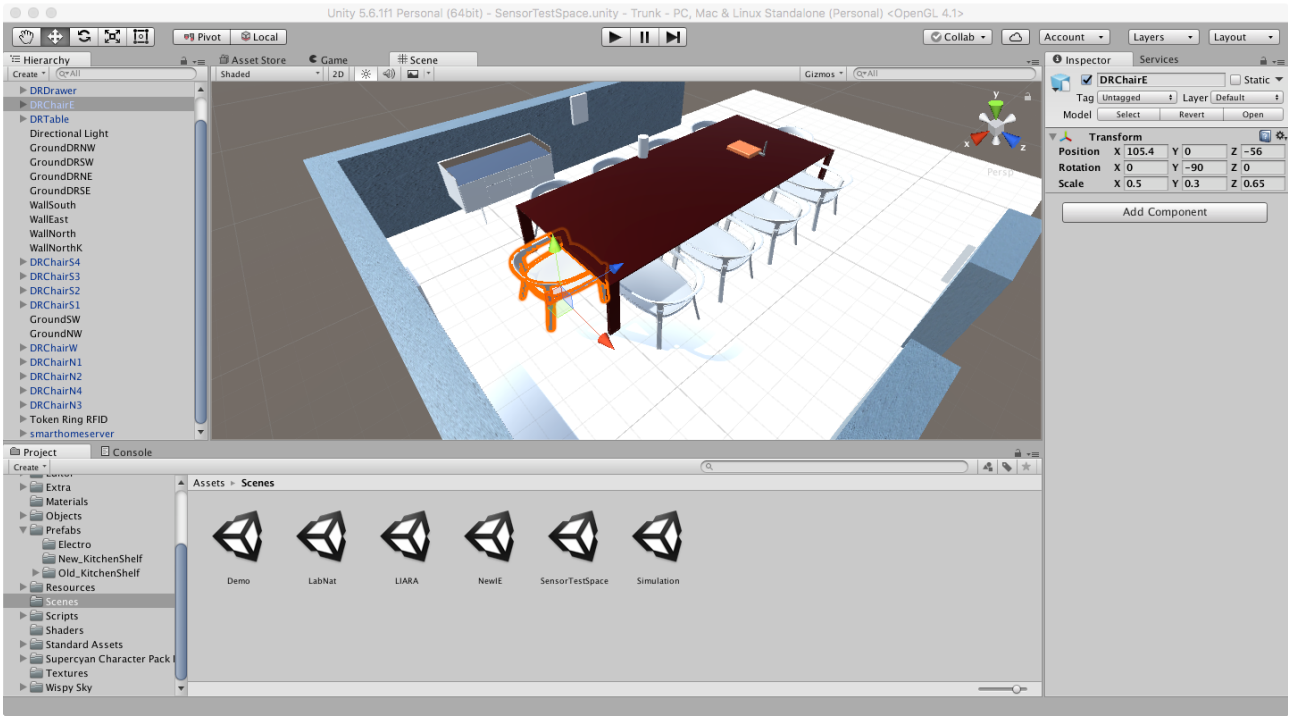
The simulator starts with a scene named ‘’Simulation”. The user has to put simulation elements and actors into this scene. This section will describe Unity view. Explanation of ‘’elements’' and repositories are in the next section.

We recommend users that want to use advanced features of Unity to read unity documentation at https://docs.unity3d.com/Manual/index.html. In this document, we provide readers with elements to build and use our simulator.

We can split the Unity UI in four main components. Figure 1 shows a screenshot of a Unity window. In this screenshot, we can see these elements:
- The scene panel
- Scene hierarchy
- The project repositories
- The inspector



The scene panel is the zone where the user build his/her virtual environment. Basically, he/she has to drag and drop elements from the project repositories. He can use the inspector to configure the



elements. Figure 2 shows a screenshot of the inspector.

FIGURE 1: SCREENSHOT OF UNITY WINDOW

FIGURE 2: SCREENSHOT OF AN OBJECT SELECTED

IV. Using the System

The simulator involves two main phrases:

- Designing phases: the user builds his/her intelligent environment and define the entities' behaviors.

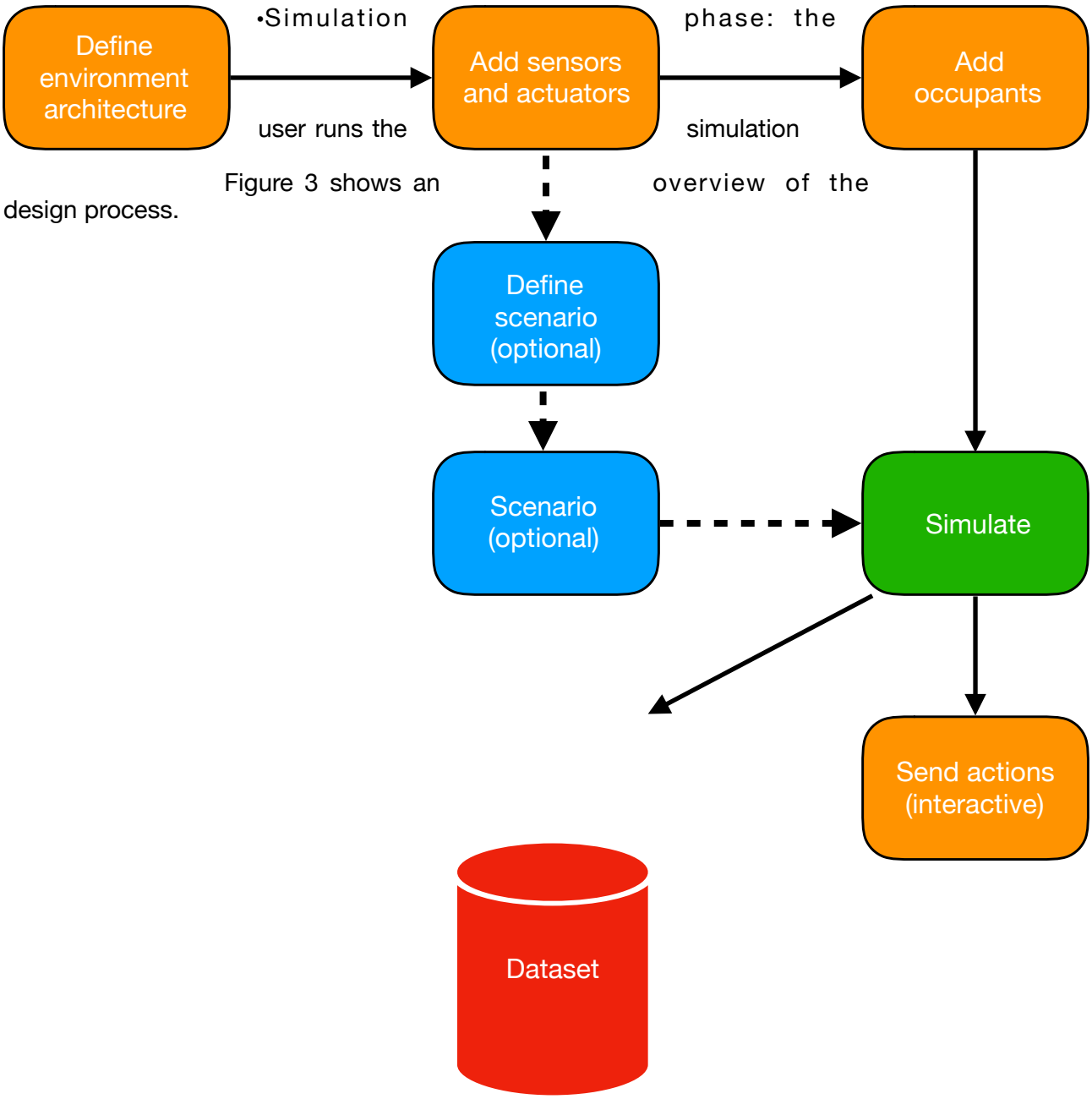


FIGURE 3: SCHEME GIVING AN OVERVIEW OF THE DESIGN PROCESS.

1. Project repository organization

Main repositories of the project are:
- Resources
- Objects
- Scripts
- Extra

‘’Resources” repository contains prefabs. The users have to use these prefabs in order to build his/her intelligent/environment. A prefab is an entity ‘’ready to use”. It contains the 3D model of an element and scripts related to its function.

‘’Objects” repository contains 3D models. The user can also use these entities to build systems. However, he may have to add script to these models to define a behavior.

‘’Scripts” repository contains all scripts. The scripts define behavior of all entities of the system. The users can drag and drop a script on an entity to add a behavior to the entity.

‘’Extra” repository contains elements related to the database server.

2. Run the simulation

Once, the system is ready for simulation, the user has to start the database server. He has to launch the python script ‘’database.py”. The python command is ‘’python database.py”. The script will be generated in the repository a database named ‘’database”. This database will contain the dataset. Then, the user can press the start button in Unity.

V. Extend the System

The user can extend the simulator in several ways. He can add new objects, sensors, actuators and behavior three components.

1. Objects

To add simple objects is very simple. Just drag and drop your 3D models object into the object repository. Unity will refresh the collection of objects and your object will be ready to use.

2. Sensors

To add new kinds of sensors the user has to add the sensor 3D model (just fellow process to add 3D objects) and his behavior. To implement the behavior, just add a new script that extends the script ‘’IESensor”.

3. Atomic Behavior Tree Components

To add new behavior tree components, the user has to extend the script ‘’EntityBehaviour”. He has to implement the sensor behavior into the method ‘’EBUpdate()”. This method must return a BTState. BTState are:

- Stop: the BT does not run.
- Running: the BT is running.
- Succeeded: the BT has ended with success.
- Failed: the BT has ended with failure.