Tangible Tabletop Experience Toolkit

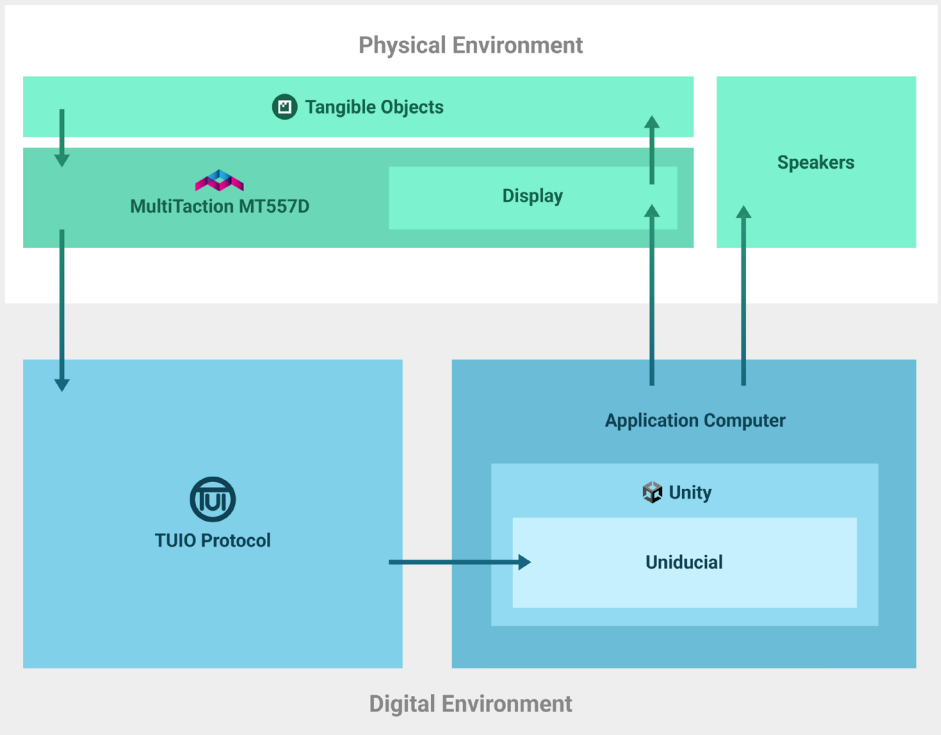
MultiTaction MT557D & Unity - Manual

Table of Contents

Introduction

This toolkit is intended to make the design and development of tangible tabletop-based experiences more accessible to future researchers and designers. This manual will cover how to set up a development environment using the MultiTaction MT557D Cell and Unity.

A tangible tabletop experience is made up of two parts; the physical and the digital. This manual will guide the setup of both the hardware and software so that information can be sent back and forth, and prototypes can be developed with Unity. While manuals for setting up the MT557D cell, and similar products to it exist, they often are missing parts, or are too vague. This manual aims to fill in for any missing parts which are relevant to this particular setup.



Setup

Hardware

The first step towards designing and developing experiences for the MultiTaction MT557D is to get the hardware set up. The setup consists of the MT557D itself, as well as a range of peripheral devices and cables.

What you will need:

- MultiTaction MT557D (Or a similar MultiTaction product)
- An application computer to run the experience
- A mouse and keyboard (We recommend two mice for the convenience of switching back and forth between the application computer and MT557D OSD)
- Two ethernet cables
- A displayport cable
- Any amount of physical objects to be used in the experience
- **Optional:** A router (required to run experience without access to ethernet ports)
- **Optional:** A second monitor to allow for development directly on the MT557D

Additionally, devices such as speakers can be connected to the application computer if needed.

For assembly of the MT557D we refer to the original manual by MultiTaction. This manual covers the assembly process in-depth, as well as the basic power connection set up. Once the MT557D has been assembled and its power supply has been connected, you are ready to start connecting it to the application computer.

Step 1: Connect MT557D and application computer with a displayport cable. The MT557D has ports available on the opposite side of the power supply.

Step 2: Connect the MT557D and the application computer to the same internet using the two ethernet cables. This is important, as the TUIO protocol sends the tracking data via the internet. We recommend connecting both to the internet provided by your Internet Service Provider through ethernet sockets in the wall. This way you will have access to the internet directly from the application computer and you will be able to utilize Git to work on the experience from separate devices.

If you do not have access to ethernet sockets in the wall, which is often the case when needing to run the experience in the field, a router is required. We used a D-Link Wireless N 300 Easy Router. Connect the router's power supply, and plug the two ethernet cables into the back of the router. This will create a small local network between the MT557D and the application computer, by automatically providing each with an IP.

Step 3: Set up the rest of the application computer by connecting its power supply and plugging in mouse and keyboard. By default, the MT557D will act as the main display, however a second monitor can be connected to the application computer for ease of control.

Creating Tokens

The physical objects which act as the primary method of control in a tangible tabletop-based experience are often referred to as tokens. The MT557D tracks something called fiducial markers using a grid of infrared cameras. Fiducial markers are essentially simple QR-code like symbols, each with their own ID. Any object can be a token, as long as it has a fiducial marker attached to it. The only requirement of a token is that it needs to have a flat bottom area of approximately 4cm x 4cm for the fiducial marker to fit.

The fiducial markers are hardcoded to the MT557D and are provided by MultiTaction. However, we included the markers in this toolkit for convenience. To make a token, simply print a marker and glue it to an object.

Software

With the hardware set up, you are ready to have the MT557D track tokens and communicate the tracking data to the application computer.

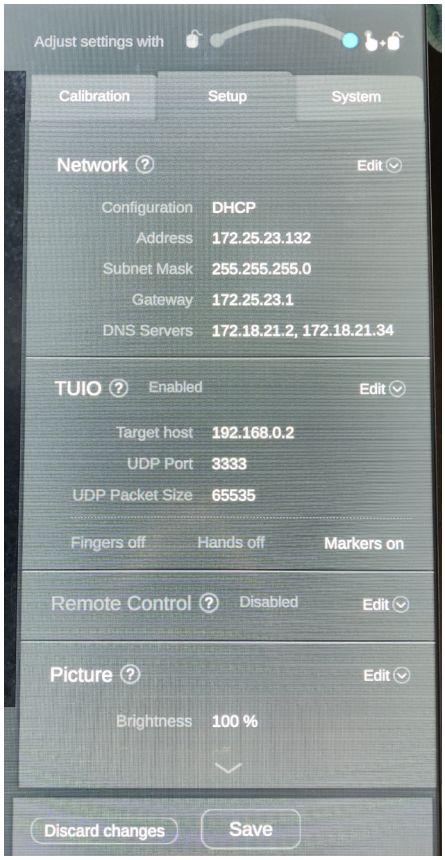
Configuring the OSD

When booting up the MT557D you will be met with a user interface referred to as the on-screen display (OSD). This is where you configure the settings of the MT557D, primarily its own IP and the target IP (The application computer). You can also choose which items to track, such as objects, fingers and pens. We recommend disabling everything but objects initially to get started. Enabling pen tracking has shown to cause freezes when people lean on the edge of the table, as it mistakes shirts for pens.

Additionally, If you have created your tokens, you can place them on the camera feed shown in the OSD to test whether it tracks them properly.

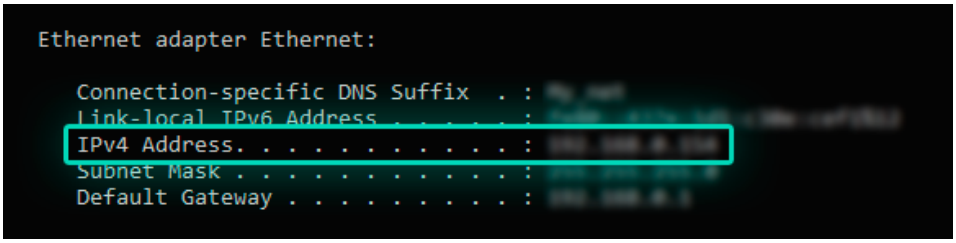
Step 1: Set up the network settings of the MT557D by navigating to the setup tab on the right side of the screen. Set the configuration mode to DHCP. This allows the MT557D to automatically assign an IP, assuming the ethernet cable is properly connected. In certain cases, the assignment process can be a bit slow or buggy. We found that simply switching the configuration mode to manual, saving the settings, switching back to DHCP and then resaving will make it attempt to assign an IP again. Once an IP has been assigned, it will show both on the address bar in the right under the configuration mode as well as on the bottom left of the screen. Write down the assigned IP address as you will need it later.

Step 2: Set up the TUIO settings in the section right below the network settings. Make sure the UDP port is set to 3333 as this is the default. The target address is the IP of the application computer, so leave this for now. Turn off finger and hand data initially, to make sure they don't interfere. They can be enabled later if necessary.



Step 3: Go to the calibration tab and tap the edit settings in the marker section. Find the slider called Keep-Alive, and increase it to about 35. This will create a 35 frame delay when a token is lifted from the table till the tracking stops for that specific marker. This largely fixes an issue where inconsistencies in tracking will cause unwanted events to occur in the experience. For example, if playing an animation when placing a token on the table, this removes the possibility of that animation occurring randomly due to packet loss. The higher the delay, the more room for tracking errors, but the experience also becomes slightly less responsive when lifting tokens.

Step 4: The last thing required in the OSD is for the TUIO protocol to know where to send the tracking data. This is what we define in the target host setting in the TUIO section. First, tap on the external source button in the top left corner of the OSD. This will display the application computer OS, similarly to a regular computer setup. On the application computer, open up the command prompt by searching for it. Once open, type "**`ipconfig`**" and press enter. This will provide you with a basic overview of the network settings:



The address that you are interested in, is the IPv4 address. When found, write it down. Before returning to the OSD, you can compare the two IP addresses you have written down. If the first three numbers are identical it means that they are on the same network. Once you have made sure both the application computer and the MT557D are on the same network, return to the TUIO settings in the OSD. You can return by connecting a mouse to the MT557D and simply moving it around. This is where having a second mouse is convenient.

When in the TUIO settings, set the target host address to the application computer IP that you just wrote down. Save the changes, and the OSD has now been fully set up.

Receiving the Tracking Data

The next step is to configure the application computer to properly receive the tracking data and make it available for Unity to pick up. Navigate back to the application computer display by tapping the external source button in the top left of the OSD.

Step 1: Install the Cornerstone SDK as provided by MultiTaction.

The download can be found on the MultiTaction developer website, which requires an account to log in. Simply follow the link, sign up and download the Cornerstone SDK:

https://apps.multitaction.com/mt-utils/cornerstone-runtime.html

Once downloaded, install the SDK to the default location similarly to how any other piece of software is installed.

Step 2: Open up the explorer, type "**%appdata%**" in the path bar and press enter. Alternatively, go to your **C: Drive** and navigate to **user › AppData › Roaming**.

Scroll through, and look for a folder called MultiTouch. In this folder you will find a **config.txt** file. This file is where we establish the connection to the MT557D by defining the sending and receiving IP addresses. When opening the config file, it should contain three settings blocks: **Globals**, **NetBridge** and **TUIOSender.**

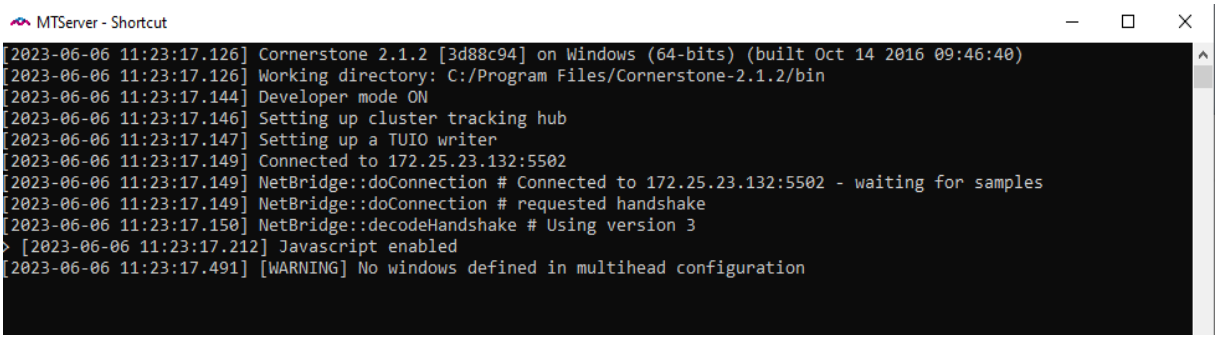
If it does not contain those, either add them manually, or replace the file with the config.txt file we provide in the toolkit. The config file should look like this :

```
Globals {
  binary-server-port = "0"
  xml-handshake-port = "0"
  xml-server-port = "0"
}

NetBridge {
  host =  "172.25.23.132"  /*  This is the IP of the cell */
  port = "5502" /* This is should be whatever the default port is set to - it is not
related to the TUIO port */
}

TUIOSender {
    /* The features that should be enabled, options are:
       "fingers" - Send finger tip locations
       "hands" - Enable the TUIO Hand Extension and send palm / finger connectivity
information.
       "objects" - Send marker information
       "silent" - Do not print debug messages to console
       "verbose" - More verbose debug messages printed to console
       "objects_as_fingers" - Markers are sent over TUIO as if they were fingers
       To enable multiple features, just put multiple keywords into the field.
       The default value is "fingers objects" */
    features = "objects"
    /* address: The TUIO stream receiver IP network address, default = 127.0.0.1 */
     address = "172.25.23.104"
    /* port: The TUIO stream receiver IP port, default = 3333 */
    port = "3333"
}
```

To configure the settings, you can ignore the **Globals** block entirely. In the **NetBridge** block, replace the host address (highlighted in red) with the IP address of the MT557D that you wrote down earlier. The port can be left untouched.

In the **TUIOSender** block, replace the address (highlighted in green) with the IP address of the application computer that you wrote down, and make sure that the port is set to the default of 3333. Save and close the file.

Step 3: Now that the settings have been configured, all that is left to do is run the server which is responsible for receiving the tracking data. Navigate to the Cornerstone installation folder, which will likely be in the top-level of the **C: drive** assuming the default installation procedure was followed. In the Cornerstone folder, open the bin folder and look for a .exe file called **MTServer**. Run it, and if everything was configured properly it should look like this:



The **MTServer** executable receives and interprets the tracking data, and therefore needs to be running at all times. We recommended creating a file shortcut on the desktop, as you will need to access it often.

Prototyping

After having set up the system, so that the application computer is receiving TUIO information from the cell, we can begin designing and creating prototypes. This section will describe how to do this using the Unity game development engine, using the Uniducial library, as well as the practicalities of developing for a tangible tabletop remotely.

Unity & Uniducial

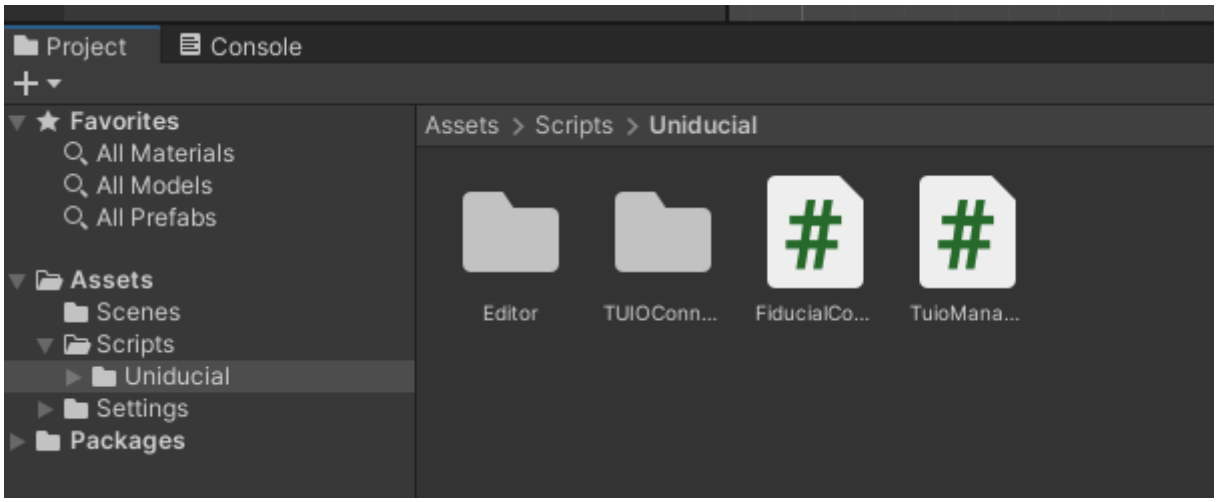
The first thing to do is to install Unity on the application computer. This is as simple as going to the official Unity website and clicking the download button:

https://unity.com/download

Once installed, open up the Unity Hub application as this is where you create and manage your Unity projects. When creating a new project we recommend making it a 2D (URP) project to make sure things work. If your experience is 3D based, you can make another project afterwards. URP is the Universal Render Pipeline which allows you to use proper lighting. Once created, boot up the project.

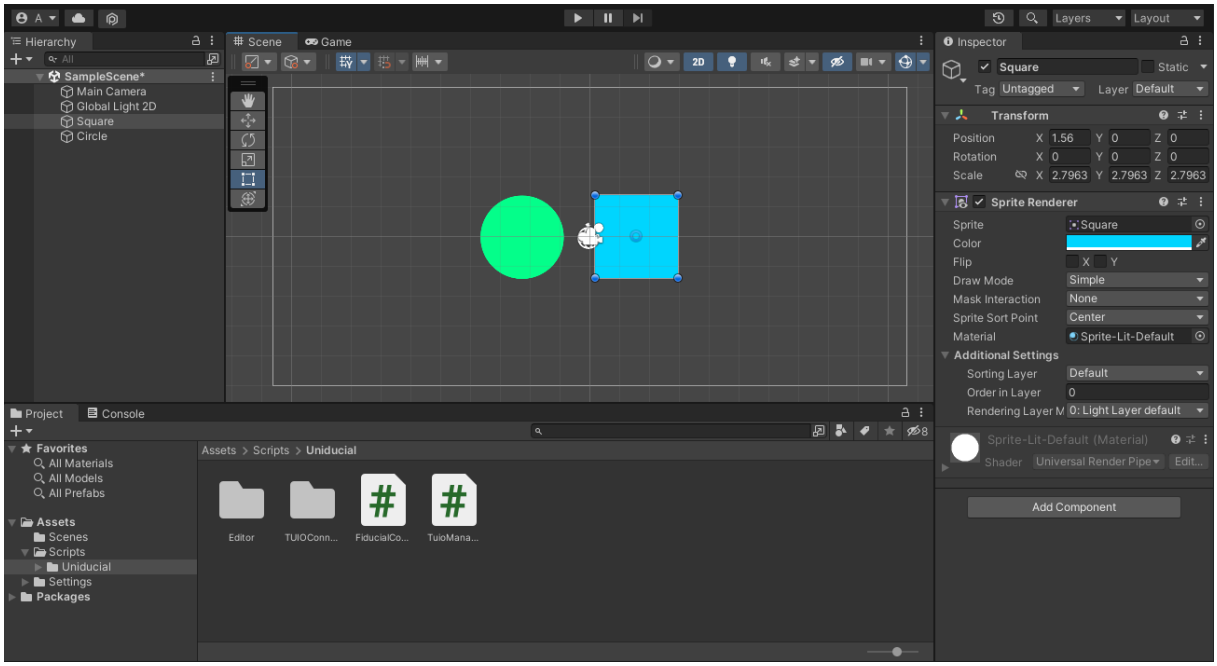
Go to the project folder structure in the bottom of Unity's user interface. The asset folder is where all your project files are stored. This folder will likely be open by default. To keep things neat, create a new folder inside the assets folder called Scripts. Now, drag the Uniducial folder provided in this toolkit, into the

Scripts folder. The Uniducial library found on the TUIO.org page is outdated and does not work with the current version of Unity. Therefore, the Uniducial library provided in this toolkit has been updated to work with the current version of Unity as of June, 2023. Additionally, we extended the library by adding a couple of new features that people might find handy.



The uniducial library contains multiple files and folders, but the only file you care about is the **`Fiducial-Controller.cs`** script, which can be found in the top level of the Uniducial folder. This script is what allows us to control GameObjects in Unity with the TUIO tracking data, by referencing the other scripts of the library.

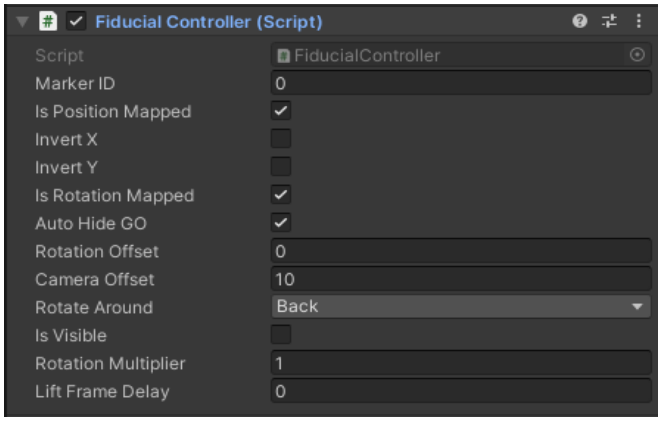
Now you are ready to start creating **`GameObjects`** that can be manipulated by your tokens. Let's set up a simple example with a square and a circle that can be moved by two different tokens. In the game hierarchy on the left, right click and create two 2D objects: a square and a circle. You can click on each and change their color in the inspector on the right side, if you please.



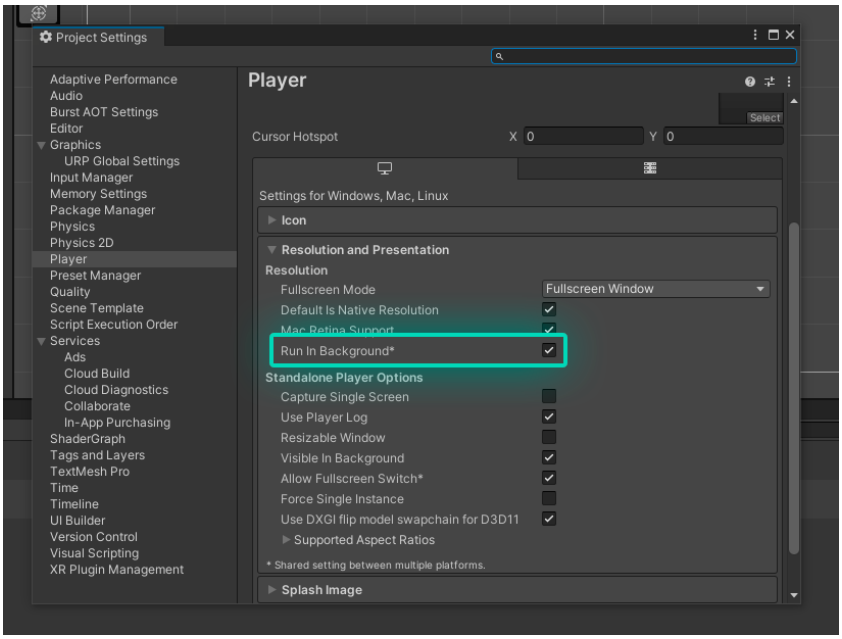
Next step is to add the **`FiducialController.cs`** script to each of the objects. Any object that needs to be moved with the TUIO tracking data, requires this script to be attached. To add the script, simply click the square object in the hierarchy, and drag the script into the Inspector panel on the right. You can also drag it directly onto the object in the hierarchy, or onto the object within the scene.

Once added, you should see the script appear as a component in the **Inspector**. You may have to click the small arrow next to its name to see the component properties. From these properties, only a handful are important. The **Marker ID** property is where you define which fiducial marker will be controlling this specific **GameObject**. This will depend on which marker you attached to your token. If you are unsure what ID the marker on your token has, return to the MultiTaction OSD and place the token on the camera feed. This will tell you the ID.

Once the **Marker ID** has been set, you will likely want to enable **Is Position Mapped**, this is what makes the object move based on the token position. If you want the **GameObjects** to rotate with the token, enable **Is Rotation Mapped**. **Auto Hide GO (GameObject)** is used to hide the **GameObject** in the game by disabling its sprite whenever its associated token is lifted from the table. We found having this enabled to be the most natural, and would only disable it in specific cases. **Lift Frame Delay** has the same purpose as the Keep-Alive feature in the OSD, but provides more control as it can be changed programmatically and during runtime. The rest of the properties can be left as is. Repeat the process for the circle.When complete, each of the **FiducialController** components should look something like this:



The last thing you need to do before running the experience, is to allow Unity to run in the background when other windows are in focus. This is required if using the TUIO simulator, which we describe in the next section. To set this up, go to **Edit** in the top navigation and click **Project Settings**. Navigate to the **Player** section on the left side of the settings window. In the player settings, scroll down and extend the **Resolution and Presentation** bar and enable **Run In Background**.

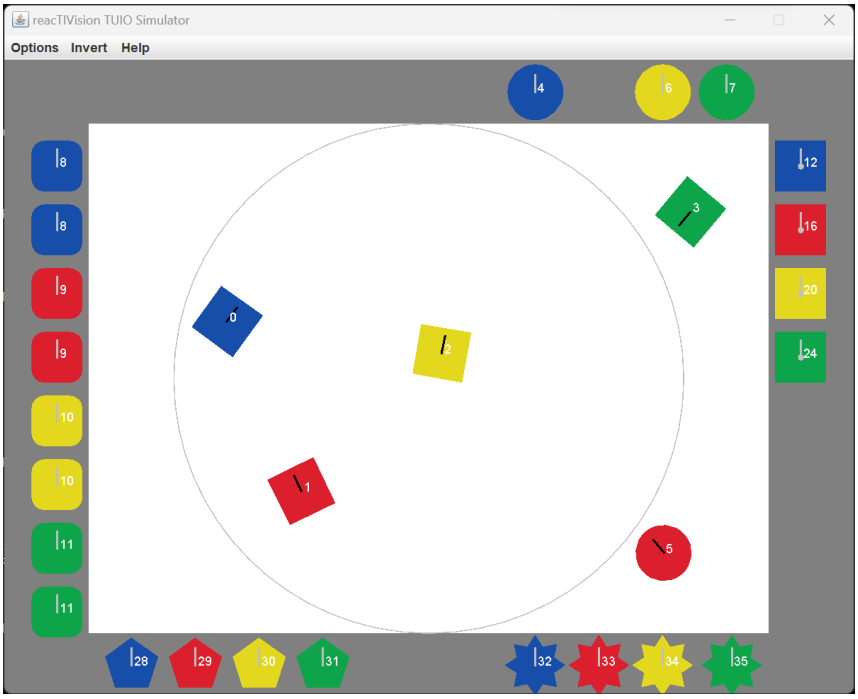


Now you are ready to run the experience. Press the play button in the top and once the game has loaded, you should be able to manipulate the square and circle by placing your tokens on the MT557D and moving them around.

Working Remotely

There are two tools which make it much easier to develop for a tangible tabletop, without having to always be in the same space. These tools are TUIO simulator and Git with Github.

TUIO simulator lets you simulate interaction with tokens through a simple application allowing you to move, rotate, and lift digital representations of tokens. The simulator sends TUIO information across port 3333, which Unity and Uniducial are listening for, and can react to. Of course, the simulator has its limitations, but it is a quick and easy alternative when the tangible tabletop is not available. This tool is the primary reason for enabling Unity to run in the background.



Another very useful tool when developing for a tangible tabletop is Github. Once your Unity project is set up in Github, it allows you to continue working remotely, committing changes as you do. When you are back at the tangible tabletop to test these new changes, you just have to pull those new changes. This becomes increasingly useful when development is collaborative, as multiple designers can be working remotely, simultaneously, using TUIO simulator for initial testing, until everyone is back at the tangible tabletop.

Unity projects contain a lot of files that are both large and unnecessary to push to the Git repository. We recommend setting up a Git repository with a .gitignore file as well as initializing Git LFS (Large File Storage). A variety of guides on setting up Unity with Git are available, but as a simple first setup, we recommend the following youtube video:

https://youtu.be/_ewoEQFEURg

With the information presented in this manual, you should be able to set up and run a fully functional development environment for tangible tabletop-based experiences. We highly encourage you to explore all the features Unity and its scripting system, as it will allow you to design and develop the exact experience for your needs.