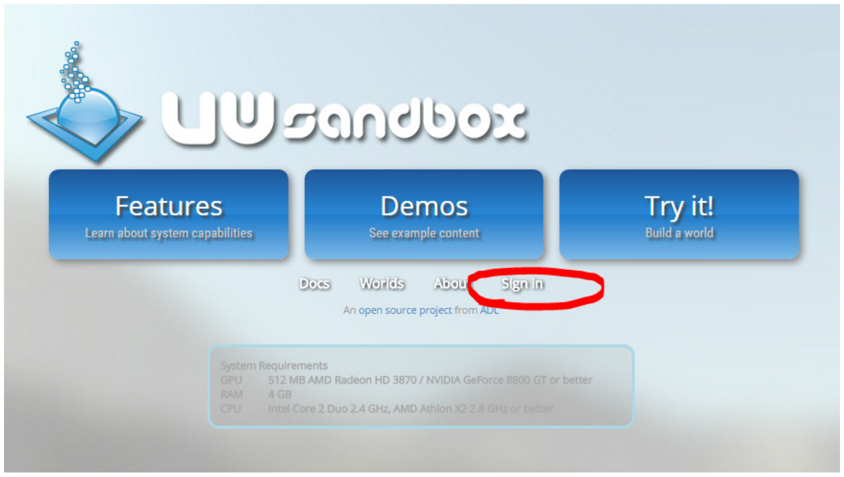
xAPI in a Virtual World Tutorial Instructions

This guide will walk you through adding xAPI performance tracking to an existing virtual world environment. The environment is already created with some simple goals, and we will expand it such that the accomplishment of these goals creates records in the ADL Learning Record Store. We will accomplish this by using the virtual environment's built in xAPI wrapper support, which mirrors almost exactly the xAPI wrapper you've seen in previous sessions. Before we begin, we've got a few steps to take to get set up. You can find a copy of this document at http://bit.ly/1HYkp1j

Test Compatibility

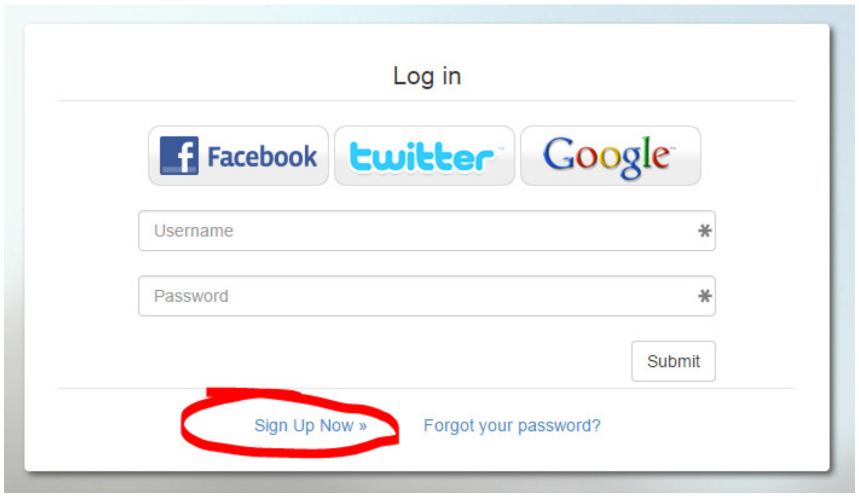
The Sandbox Virtual World is an HTML5 application that requires WebGL support. You can discover if your browser is supported by visiting http://bootcamp.adlnet.gov/100/adl/sandbox/test.

Setup

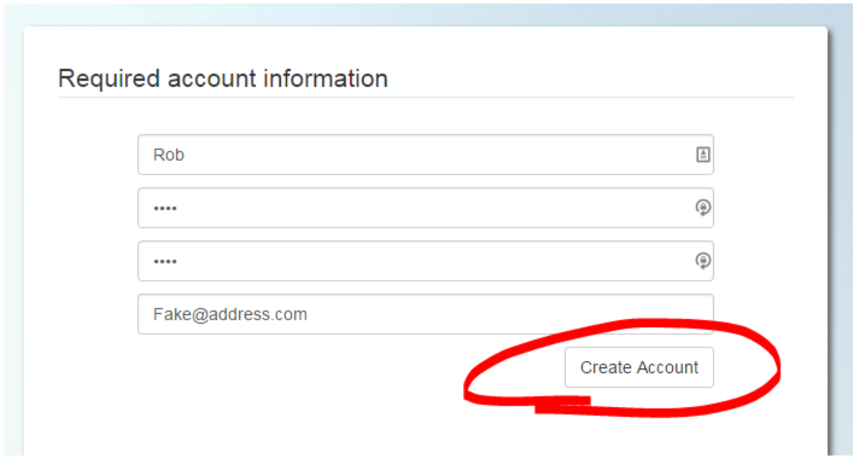
1. You'll need an account on the server in order to author your project, so please sign up. Feel free to use any dummy name and email address you like. We'll be deleting the virtual machine that hosts this virtual world after the session. To sign up, please visit http://bootcamp.adlnet.gov/100/, click "Sign In," then "Sign Up Now," and finally "Create Account."
    a. 

System Requirements

GPU	512 MB AMD Radeon HD 3870 / NVIDIA GeForce 8800 GT or better
RAM	4 GB
CPU	Intel Core 2 Duo 2.4 GHz, AMD Athlon X2 2.8 GHz or better



b.



Rob

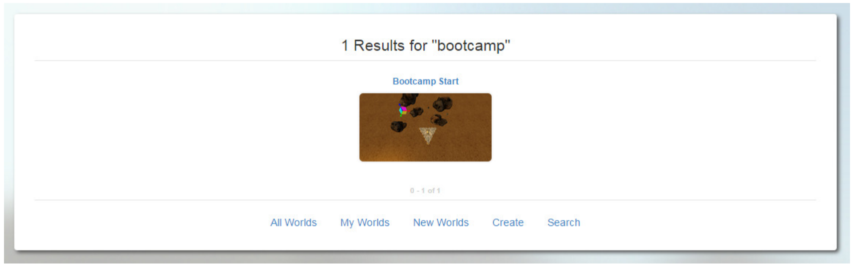
••••

••••

Fake@address.com

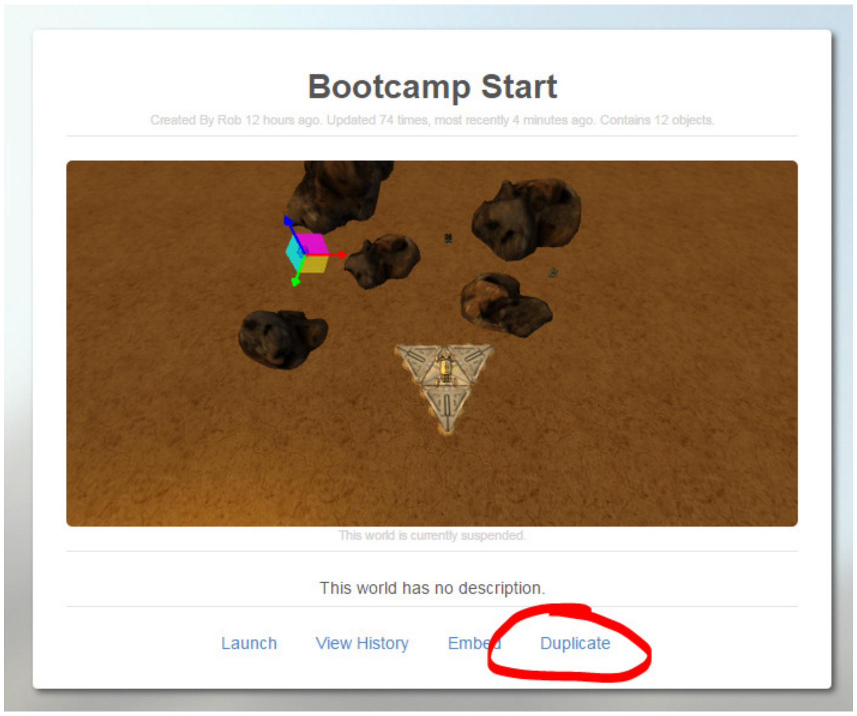
c.

2. Once you have an account, you can create your own copy of the starting world. The starting world is already set up with simple goals and behaviors, to which we will add xAPI tracking. To clone the world, we first need to find it in the worlds list. Click the "Worlds" link from either the top banner or the homepage, and search for "Bootcamp."

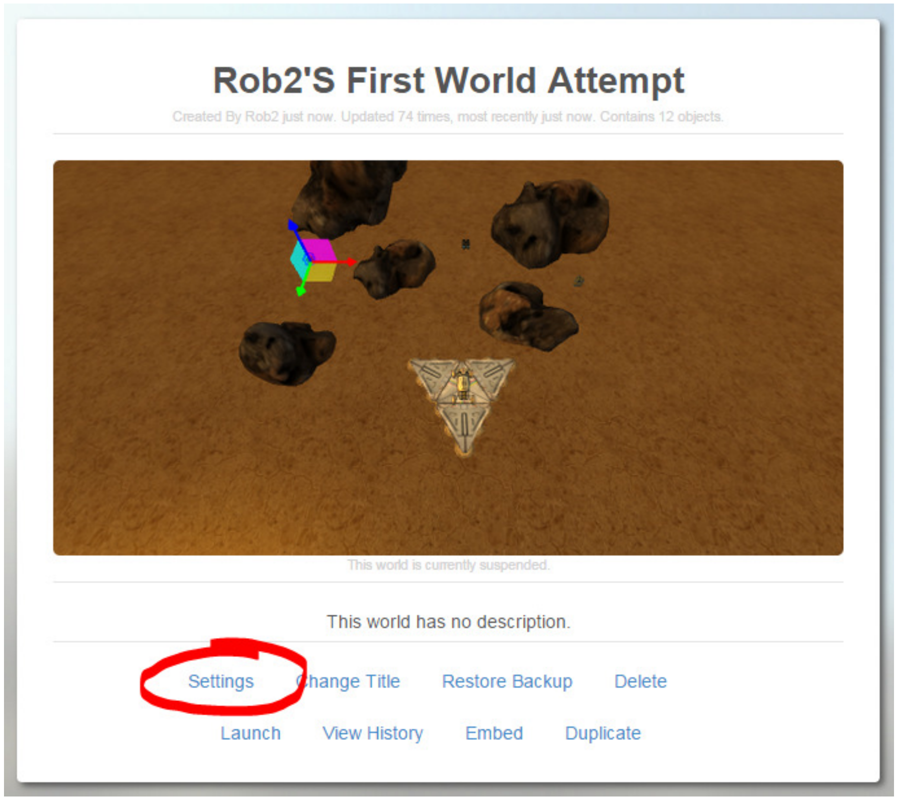


a.

3. Click on the search result titled "Bootcamp Start." On the resulting page, look for the duplicate link. This link will create a copy of the world that you will "own" for this exercise.

a. 
Bootcamp Start
Created By Rob 12 hours ago. Updated 74 times, most recently 4 minutes ago. Contains 12 objects.
This world is currently suspended.
This world has no description.
Launch
View History
Embed
Duplicate

4. After you have clicked the “Duplicate” link, you should be prompted to enter a new title and description. Please select a title you can remember. If you should ever forget which world is yours, look for the “My Worlds” link from the world search page.
5. Click “Create.”
6. Now, we have a copy of the world that belongs to you. We do want to do a tiny bit more setup. Click “Settings.”



Rob2'S First World Attempt

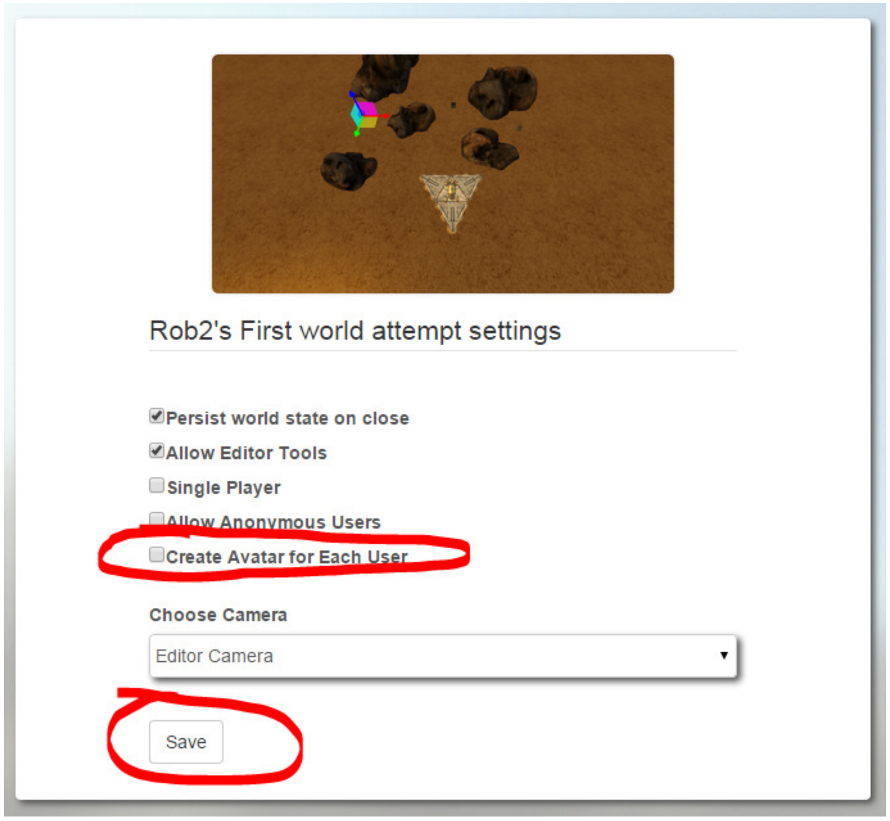
Created By Rob2 just now. Updated 74 times, most recently just now. Contains 12 objects.

This world is currently suspended.

This world has no description.

a.

7. Uncheck “Create Avatar for Each User,” then click “Save.”



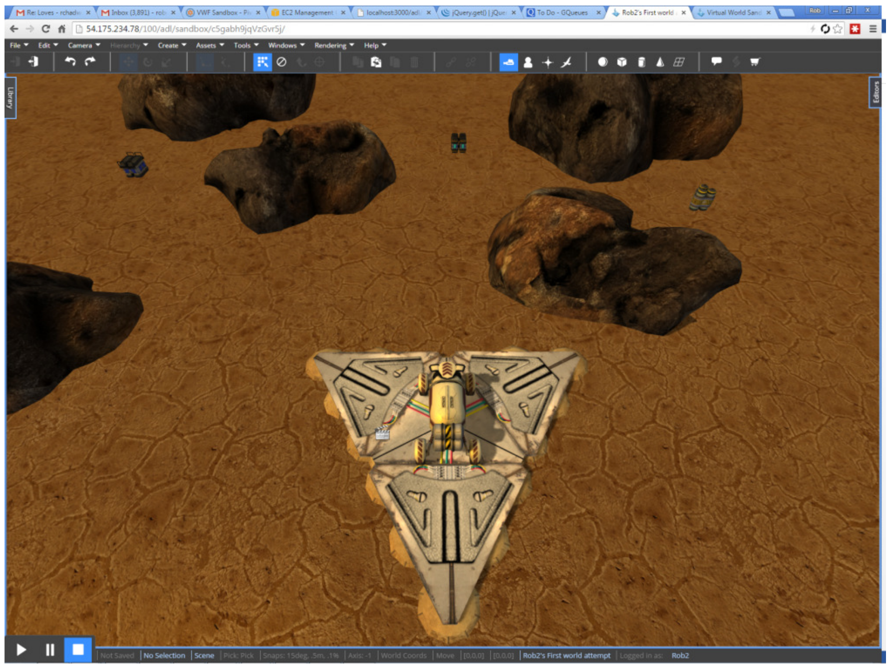
Rob2's First world attempt settings

- [✓] Persist world state on close
- [✓] Allow Editor Tools
- [ ] Single Player
- [ ] Allow Anonymous Users
- [ ] Create Avatar for Each User

Choose Camera: Editor Camera

a.

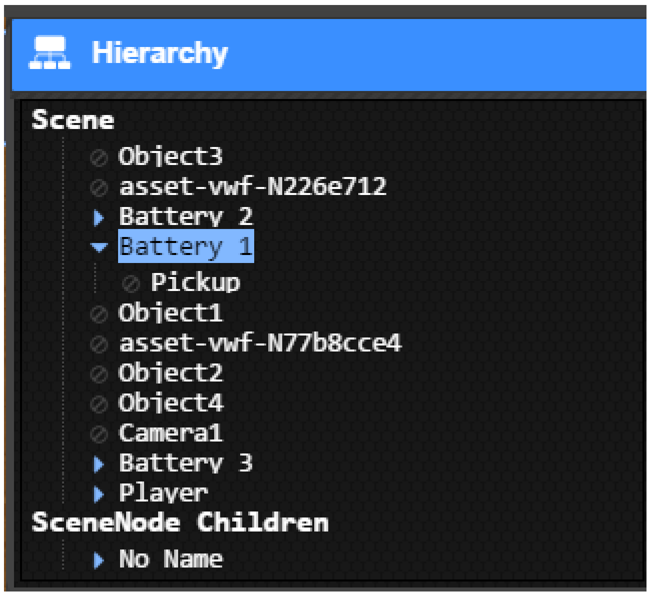
8. You can now launch the world. Click “Launch.” Wait for the world to load….

a. 

Create a Behavior

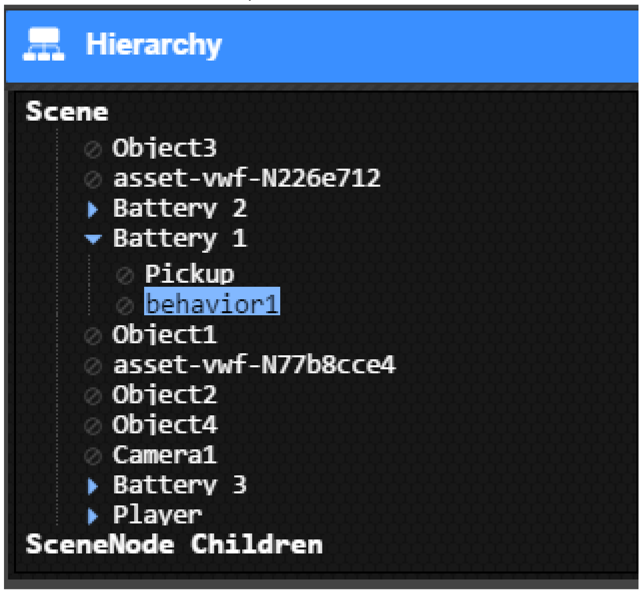
Now that we're all set up, and you have your own copy of the game world, we can begin adding the xAPI code. In this game, the rover must drive to pick up the batteries. We will create a reusable behavior that, when applied to a battery, will post an xAPI statement. Once we've added this behavior to a single battery, we will save it to our personal inventory and then copy it to the additional batteries. Each battery should report via xAPI who was playing and when it was retrieved.

1. Select "Battery 1." There are several ways that you can select this object. The easiest is to open the "Hierarchy" editor and click on the object's name. You should see something like this in the Hierarchy.



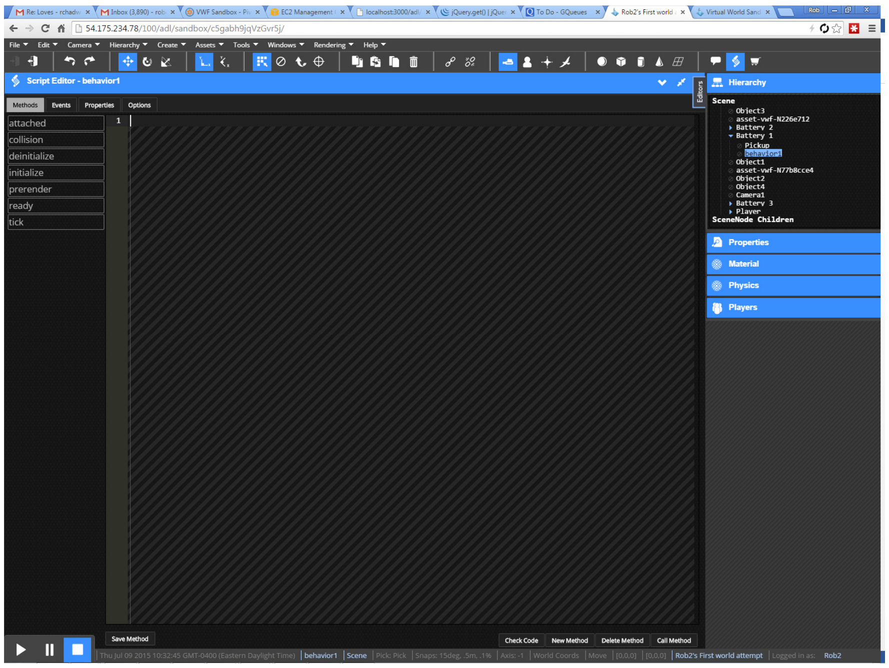
a.

2. Now, we will add a blank behavior. From the drop down menu at the top of the screen, find "Create," then "Behaviors," and then "Blank Behavior." Click this menu item, and you should see a new behavior appear below the "Battery 1" in the Hierarchy. This behavior will contain our xAPI code. Click it to select it, as below.



a.

3. Now that you have the new behavior selected, we can start adding the script. Find the "Script Editor" button on the top toolbar (it's the second from the right). When you click this, a new tool will slide up from the bottom of the page. Please maximize this window by clicking its maximize button. You should not have a full-screen script editor.

a. 

4. Find the button near the bottom of the screen that says “New Method.” Click it, and when prompted, name the method “Goal_Complete.” The system will call this method automatically when the rover picks up the battery. Enter “0” for the number of parameters.

Code the xAPI statement

Now, we’ll need to use JavaScript to create the xAPI statement. The Sandbox includes the ADL xAPI Wrapper, so we’ll use the same structures you’ve learned previously. The code you write here will be triggered by the Sandbox when the rover picks up the battery.

1. First, we need to generate a new “registration” for this run of the game. This will help group the statements into a single attempt. Add this code to the top of your function.

a.
```
    if (!this.Scene.attempt)
    {
        this.Scene.attempt = ADL.ruuid();
    }
```

2. Now, we need to setup the xAPI wrapper. The code below will specify the endpoint and authorization information, and send it to the wrapper.

a.
```
var conf = {
        "endpoint": "https://lrs.adlnet.gov/xapi/",
        "auth": "Basic " + toBase64('adlbootcamp:1234'),
    };
this.Scene.xAPI.configure(conf);
```

3. Next, we need to tell the xAPI who is playing. The Sandbox exposes the information through the “UserManager” tool. Enter the code below

a.
```
var actor = {
        account:
        {
            name: _UserManager.GetCurrentUserName(),
            homePage: "https://sandbox.adlnet.gov"
        },
        name: _UserManager.GetCurrentUserName()
}
```

4. For the xAPI Verb, we will use the standard verb, “Completed.” Enter the code below.

   a.
```
var verb = ADL.verbs.completed;
```

5. Now, we need to inform the xAPI about this experience – what happened and where. The code below will query the Sandbox for the title and description of the world, and the name of the object that you’re picking up. Enter the code below.

   a.
```
var object = {
        id: window.location.toString() + "#" + this.parent.id,
        definition:
        {
            name:
            {
            "en-US": _DataManager.getInstanceData().title + "/" +
            this.parent.DisplayName
            },
            description:
            {
                "en-US": _DataManager.getInstanceData().description
            }
        }
}
```

6. Since we’re in the xAPI Bootcamp, and in the Virtual World session, we have some “context” for our statements. The block of code below will tell the xAPI that these statements are part of the Bootcamp and part of this session. We also note that the world you’re in is a clone of the “Bootcamp Start” world. Enter the code below

a.

```
var contextActivities = {
    "grouping": [
        {
            "id": "http: //adlnet.gov/event/2015/xapibootcamp/dev/vw"
        }
    ],
    "category": [
        {
            "id": "http: //adlnet.gov/event/2015/xapibootcamp"
        }
    ],
    "other": [
        {
            "id": _DataManager.getCurrentApplication()
                +
_DataManager.getInstanceData().clonedFrom.replace(/_/g, '/'),
            definition:
            {
                type: "https://sandbox.adlnet.gov/world"
            }
        }
    ]
};
```

7. The only “result” for each of the goals is the time it took to complete. We’re going to ask for the time that the world has been running, and store that as the duration in the result. Enter the code below

a.

```
var result = {
        duration: "PT" + Math.floor(this.Scene.time) + "S"
}
```

8. Finally, we will hook up all these values into a statement, associate that statement with this particular world, and send it. Enter the code below

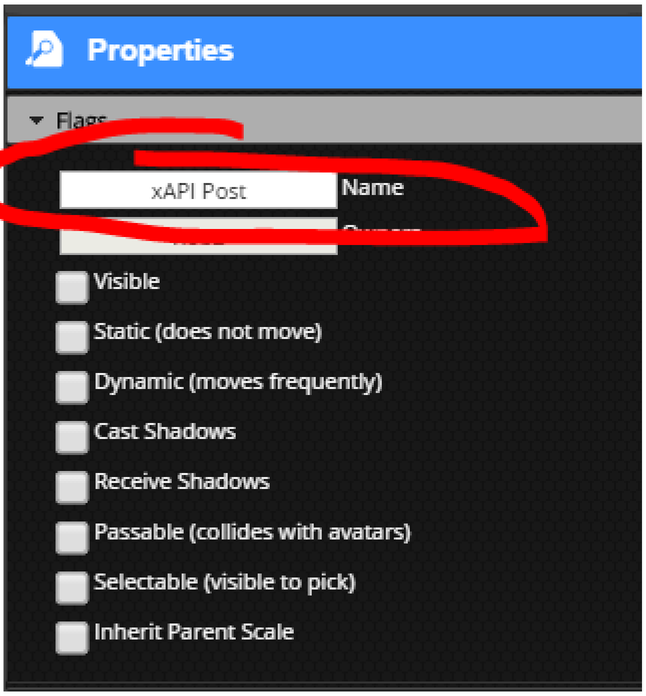
a.

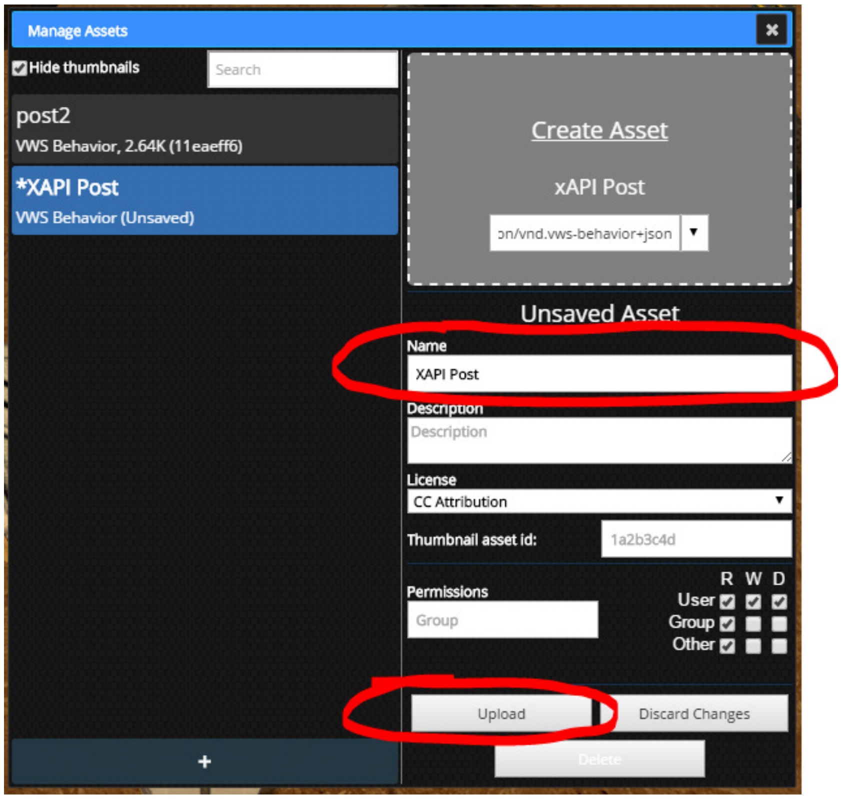
```
var stmt = new ADL.XAPIStatement(actor, verb, object);
stmt.addParentActivity(new
ADL.XAPIStatement.Activity(window.location.toString(),
_DataManager.getInstanceData().title,
_DataManager.getInstanceData().description));

stmt.context.contextActivities = contextActivities;
stmt.context.registration = this.Scene.attempt;
stmt.result = result;

this.Scene.xAPI.sendStatement(null, stmt);
```

9. That’s it! You can test the function by clicking the “Call Method” button. This will actually create a new post to the XAPI.

Apply the behavior to each Battery

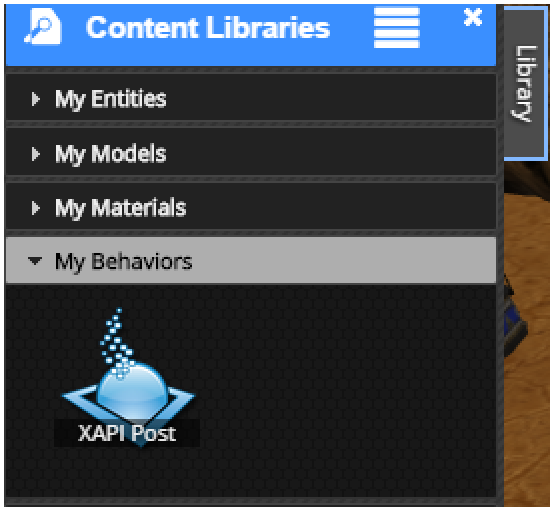
Now that we have the behavior, we are going to copy it to each of the other 3 batteries.

1. Minimize the script editor, but make sure that the behavior is still selected. You can tell by looking at the Hierarchy.
2. Open up the “Properties” tab, and change the name of the behavior to “xAPI Post”
    a. 
    Properties
    Flags
    xAPI Post Name
    Visible
    Static (does not move)
    Dynamic (moves frequently)
    Cast Shadows
    Receive Shadows
    Passable (collides with avatars)
    Selectable (visible to pick)
    Inherit Parent Scale
3. From the top menu, select “Assets,” then “Create New Asset,” and finally “From Sel. Behavior.” In the dialog that appears, again name the behavior “XAPI Post,” and then click “Upload.”



a.

4. Close the “Manage Assets” window, and find the “Library” Tab on the left of the screen. Open this tab, and find the “My Behaviors” section. You should see an icon for “XAPI Post.”

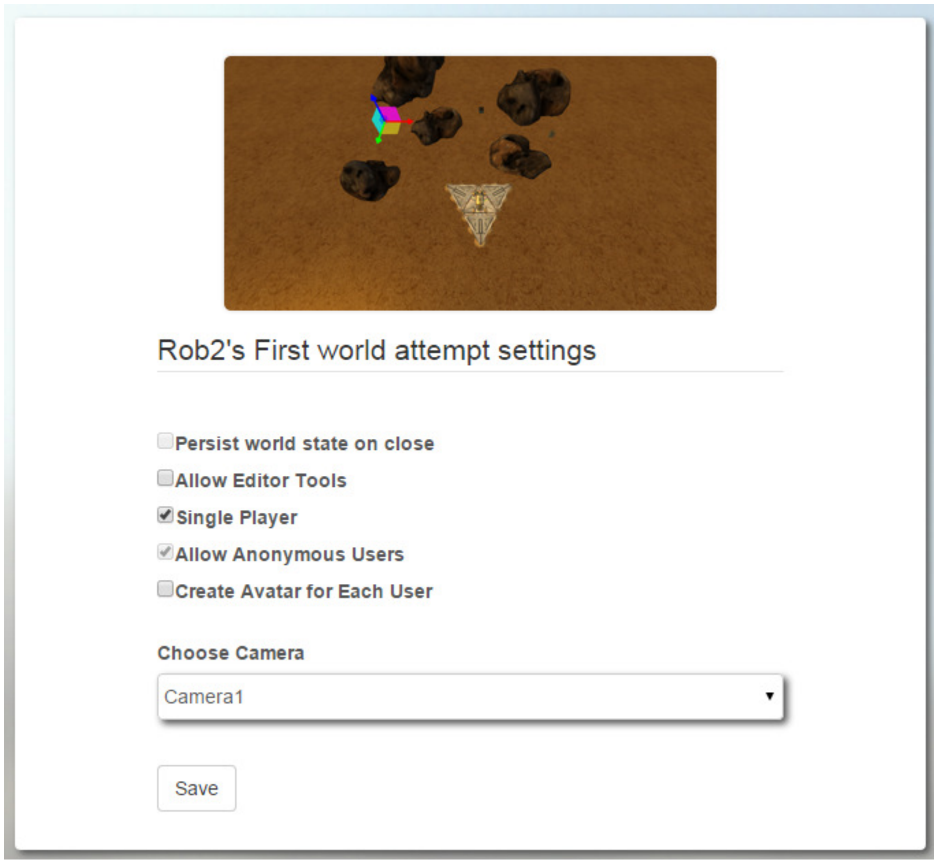


a.

5. Drag this icon and drop it onto the other 2 batteries. They will flash with a green outline to let you know you successfully applied the behavior.
6. That’s it! The world is now set up for XAPI tracking. Find the File menu at the top, and select “Log Out.”

Publish the world

Now that we're finished editing, we can set it up so that visitors simply can play the game, rather than edit it. Return to the settings page for the world, and change the setting per the example below. Once you've saved these settings, other users will be launched directly into the game.



Rob2's First world attempt settings

- [ ] Persist world state on close
- [ ] Allow Editor Tools
- [x] Single Player
- [x] Allow Anonymous Users
- [ ] Create Avatar for Each User

Choose Camera

Camera1

Conclusion

Now that we've enabled this game to track with the xAPI, we can review the data. Let's visit the LRS to review the data that we've created.