Using xAPI to Track Learning Experiences in Unity Projects

Art Werkenthin
RISC, Inc.
Using xAPI to Track Learning Experiences in Unity Projects

ART WERKENTHIN
RISC, INC.
Who am I?

- CEO, RISC, Inc.
- Member ADL cmi5 Working Group
- Author, Learning Solutions Magazine
Agenda

My First Game
Examine libraries to add xAPI to XR
Explore a hybrid solution
How to handle cmi5

Terminology:
1. XR = AR, VR, MR, gaming and simulations
2. xAPI = Experience API
I am not a Unity developer or employee.

This presentation is very technical.
xAPI Libraries

Open Source xAPI Libraries exist for:
- Javascript
- Objective C
- Java
- PHP
- Python
- C#

We will look at C# for Unity
The Leaderboard

Leaders

Your Time: 00:35:17

00:33:48  Mark, Caitlin
00:35:17  Carter, Devlin
00:35:71  Rule, Marble
00:36:67  Werkenthin, Art

Quit
TinCan.Net Library

Open-source library from Rustici Software

My company uses a modified version of this in our LMS/LRS platform.

Let’s see some code...
Issue 1 - .Net Assemblies

- Unity uses a subset of the .Net runtime assemblies
- We need a System.Web to make web requests
- Use link.xml file in the Assets folder in order to include System.Web
private bool sendXAPI;
private int StatementsSent;

private void Start()
{
    winText.SetActive(false);
    leveUpButton.SetActive(false);
    quitButton.SetActive(false);
    rb = GetComponent<Rigidbody>();
    count = 0;

    // Boolean flag indicating whether it
    // is time to send xAPI statements
    sendXAPI = false;
    playerText.text =
        NameTransfer.playerFirstName + ":";
    SetCountText();

    timerIsRunning = true;
}
protected Statement GetStatementTemplate(string verb, string verbDisplay)
{
    // Create a xAPI Statement object
    var s = new Statement
    {
        actor = new Agent
        {
            name = NameTransfer.playerLastName + ", " + 
            NameTransfer.playerFirstName.Trim(),
            mbox = "mailto:" + NameTransfer.playerEmail
        },
        verb = new Verb
        {
            id = new Uri(verb),
            display = new LanguageMap()
        },
        authority = new Agent
        {
            account = new AgentAccount
            {
                name = xAPIConstants.LRSUserId,
                homePage = new Uri(xAPIConstants.agentHomePage)
            },
            timestamp = DateTime.UtcNow
        }
    };

    s.verb.display.Add("en-US", verbDisplay);

    return s;
}
protected void SendxAPILevelComplete(string sceneName, 
            TimeSpan duration_)
{
    // Cannot send statement if we have no actor.
    if (string.IsNullOrEmpty(NameTransfer.playerEmail))
    {
        return;
    }

    var s = GetStatementTemplate(xAPIConstants.verbCompleted, 
                              "completed");

    // Level was completed
    s.result = new Result
    {
        completion = true,
        duration = duration_
    };
}
// The object of our statement is the level completed
var target_ = new Activity
{
  id = xAPIConstants.levelIRI + "/" + NameTransfer.currentLevel,
  definition = new ActivityDefinition
  {
    type = new Uri(xAPIConstants.levelActivityTypeType),
    name = new LanguageMap()
  }
};
target_.definition.name.Add("en-US", sceneName);
s.target = target_;
// Set the game as the parent activity
s.context = new Context
{
  contextActivities = new ContextActivities
  {
    parent = new List<Activity>
    {
      new Activity
      {
        id = xAPIContstants.gameIRI,
        definition = new ActivityDefinition
        {
          name = new LanguageMap()
        }
      }
    }
  }
};

s.context.contextActivities
  .parent[0]
  .definition
  .name.Add("en-US", GameConstants.GameName );
// Call method to send the statement to the LRS.
SendxAPIStatement(s);

if (!lrsResponse.success || !lrsSuccess)
{
    Debug.Log(lrsMessage);
}
protected void SendxAPIStatement(Statement s)
{
    // Create a TinCan.Net object that handles
    // LRS functions
    var lrs = new RemoteLRS
    {
        endpoint = new Uri(xAPIContstants.LRSEndPoint),
        version = TCAPIVersion.V101
    };

    // Provide credentials
    lrs.SetAuth(xAPIContstants.LRSUserId,
                xAPIContstants.LRSPassword);

    lrsResponse = new StatementLRSResponse();
}
// Attempt to send the statement up to 3 times
for (var try_ = 1; try_ < 3; try_++)
{
    lrsMessage = "";
    lrsResponse = lrs.SaveStatement(s);

    if (lrsResponse.success)
    {
        // Display the total number of statements sent
        StatementsSent++;
        countText.text = "Statements Sent: " + StatementsSent;
        lrsSuccess = true;
        break;
    }

    if (lrsResponse.content?.id != null)
    {
        // Store statement ID in case calling routine
        // wants to use it for some reason
        s.id = lrsResponse.content.id;
    }

    lrsMessage = lrsResponse.errMsg;
    Debug.Log("Statement failed: " + lrsResponse.errMsg);
    System.Threading.Thread.Sleep(500);
}
protected void SendxAPISatisfied(TimeSpan duration_
{
    // Cannot send statement if we have no actor.
    if (string.IsNullOrEmpty(NameTransfer.playerEmail))
    {
        return;
    }

    var s = GetStatementTemplate(xAPIConstants.verbSatisfied,
        "satisfied");

    var target_ = new Activity
    {
        id = xAPIConstants.gameIRI,
        definition = new ActivityDefinition
        {
            type = new Uri(xAPIConstants.levelActivityType),
            name = new LanguageMap()
        }
    }

    target_.definition.name.Add("en-US",
        GameConstants.GameName);

    s.target = target_;
OnTriggerEnter

Called with the ball hits a "pickup" object

Initially, I sent the statements from this method, causing UI "glitches"

To fix:

- Created a method to send the statements
- Set a flag in OnTriggerEnter to indicate it was time to send statements
- Called method to send statements from Update()

```
// Is it time to send xAPI statements?
// (SendXAPI is set by OnTriggerEnter())
if (sendXAPI)
{
    SendxAPI();
    sendXAPI = false;
}
```
Gotcha: Code Stripping!

- Code stripping removes parts of libraries that Unity thinks you are not using
- TinCan.Net uses Newtonsoft Json Library
- Uses concept of “reflection”, which is code stripped by Unity
- Solution: Find a “Newtonsoft” library that doesn’t use reflection
Replacing Newtonsoft
Code stripping strikes again!

- I built the game again and it did not send statements.
- Failed when it made a web request.
- Unity was code stripping system.web.
- Fix: csc.rsp file

```
csc.rsp  x  link.xml  x  xAPIConsttants.cs  PlayerController.cs
1 -r:System.Web.dll
```
It Works!

- New build sent statements
- NOW show co-workers
TinCan.Net Pros & Cons

Pros:
• Addresses all “api”
• Flexible

Cons:
• Difficult to integrate with Unity
ADL Unity xAPI Wrapper

- Open-source ADL project
- Send and fetch xAPI Statements from Unity
protected Statement GetStatementTemplate(string verbIri, string verbDisplay, string objectIri, string objectName)
{
    // Create a xAPI Statement object
    var actor = Actor.FromMailbox(NameTransfer.playerEmail, false,
        NameTransfer.playerLastName + "", " +
        NameTransfer.playerFirstName.Trim());

    // Note: Although I set the verbDisplay property,
    // the library does not send it
    var verb = new Verb(verbDisplay, verbIri);

    var activity = new Activity(objectIri, objectName);

    // There is constructor that takes 0 params.
    // So this is the only way to initialize a
    // statement
    var s = new Statement(actor, verb, activity);

    return s;
}
ADL Unity
Wrapper
Pros & Cons

Pros
• Code is simple
• Addresses fetching of statements
• No code-stripping issues

Cons
• Poorly documented
• Less object-oriented than TinCan.Net
• Did not send the “display” value for a verb, even though I set it in code.
• Does not address all properties of a statement
• Sends properties that are NULL
GBLxAPI Library

- Developed under National Science Foundation research grant.
- Open source
- Designed for K-12 (but can be adapted)
- Disclosure: I was asked to do a code review of the latest version and made some small suggestions.

Let’s see some code...
Configuration

In GBLConfig.cs set:

- LrsURL (Your LRS endpoint)
- companyURI (Actor homepage if “account” is used)
- gameURI (The unique identifier for the project)
In GBLInterface.cs set:

- IrsUser (ID to connect to LRS)
- IrsPassword (Password to connect to LRS)
private static Agent playerAsAgent
{
    get
    {
        return GBLXAPI.Agent
            .WithName(NameTransfer.playerName)
            .Build();
    }
}

2 references
private static Activity MainGameActivity
{
    get
    {
        return GBLXAPI.Activity
            .WithID(GBLXAPI.Configuration.gameURI)
            .WithName(GBLXAPI.Configuration.gameName)
            .Build();
    }
    .Build();
}
// Ensure GBLxAPI is initialized.
GBLXAPI.Init(new GBLConfig(GBL_Interface.lrsUser,
    GBL_Interface.lrsPassword));

// Start a timer for this level.
GBLXAPI.Timers
    .ResetSlot((int)GBL_Interface.durationSlots.Level);
if (timerIsRunning && timeElapsed >= 0)
{
    // Get time elapsed from GBLxAPI timer.
    timeElapsed = GBLxAPI.Timers.GetSlot((int)GBL_Interface.durationSlots.Level);

    // Convert the float timeElapsed to minutes, seconds and milliseconds.
    GetMSM(timeElapsed, out var minutes, out var seconds, out var milliseconds);

    // Display the time elapsed to the player.
    timerText.text =
        string.Format("{0:00}:{1:00}:{2:00}", minutes, seconds, milliseconds);
}
var activityDef = new ActivityDefinition
  .WithActor(playerAsAgent)
  .WithVerb("completed")
  .WithTargetActivity(GBLXAPI.Activity
    .WithID(GBLXAPI.Configuration.gameURI + "/level/" + level)
    .WithType("level")
    .WithDefinition(activityDef)
    .Build())
  .WithResult(GBLXAPI.Result
    .WithDuration((float)duration.TotalSeconds)
    .Complete())
  .BuildContext(GBLXAPI.Context
    .WithParents(new List<Activity>
      {
        MainGameActivity
      }
    .Build())
  .Enqueue(GBLxAPICallBackHandler);
}
public void GBLxAPICallBackHandler(bool result, string resultText)
{
    if (result)
    {
        // Statement was successful. Increment count of statements sent and display to player.
        StatementsSent++;
        countText.text = "Statements Sent: " + StatementsSent;
        return;
    }

    Debug.Log("Sending statement failed: " + resultText);
}
Send Satisfied

Same as sending level complete except:

- The game itself is the “object” in the statement
- We do not set a “parent” context activity
# GBLxAPI Pros & Cons

## Pros
- Easy to setup
- Statements are “queued”
- Major overhaul in 2021
- Code clarity

## Cons
- Use of excel files for vocabulary changes does not work
- Only addresses writing of statements
- Issue with newer versions of Unity
I wanted a leaderboard, but GBLxAPI does not fetch statements.

Can I use both GBLxAPI and TinCan.Net?
GBLxAPI is a Wrapper for TinCan.Net
// Build a TinCan.Net query object for
// the LRS to get player results
var query = new StatementsQuery
{
    since = StartDate,
    ascending = true,
    limit = 1000,
    format = StatementsQueryResultFormat.EXACT,
    verbId = new Uri(xAPIConstants.verbSatisfied),
    activityId = new Uri(config.gameURI)
};
```javascript
for (var try_ = 1; try_ < 3; try_++)
{
    lrsResponse = lrs.QueryStatements(query);
    if (lrsResponse.success)
    {
        // The query worked.
        break;
    }
    Debug.Log("Query Statements failed: " + lrsResponse.errMsg);
    System.Threading.Thread.Sleep(500);
}
var statements = lrsResponse.content.statements;
```
What about VR?

- So far, we’ve looked at a desktop game.
- Does this work for VR?
- Short answer: Yes, nothing changes
Summary

Three libraries that can be used with Unity for implanting xAPI

Demonstrated that GBLxAPI with TinCan.Net is best option in most cases.
References

- Faster to competency

- Talent Retention
  https://www.industryweek.com/talent/article/21134021/can-arvr-pull-in-future-talent

- Cost Effective
Libraries

- TinCan.Net
  https://github.com/RusticiSoftware/TinCan.NET
  https://github.com/cawerkenthin/xAPI.Net

- ADL Unity-xAPI-Wrapper
  https://github.com/adlnet/Unity-xAPI-Wrapper

- GBLxAPI
  https://gblxapi.org

- Game example
  https://github.com/cawerkenthin/xAPIAndUnityProject
References

- Improved Retention
  https://trainingindustry.com/articles/learning-technologies/3-ways-virtual-reality-training-is-producing-better-outcomes/#:~:text=Better%20Long%20Term%20Retention&text=Narendra Kini%2C%20CEO%20of%20Miami's%2C%20one%20week%20after%20traditional training

- Serious Games Profile
  https://profiles.adlnet.gov/profile/0dc3dbf4-7ec9-42a2-bb3b-b9487e1b5769

- DoDI 1322.26
  DoDI 1322.26 Fungible References | ADL Initiative (adlnet.gov)
Other links

- GBLxAPI Newtonsoft Issue
  https://github.com/gblxapi/UnityGBLxAPI/issues/2

- Articles
  https://risc-inc.com/sending-xapi-statements-from-a-unity-game
  https://learningsolutionsmag.com/articles/use-the-gblxapi-library-to-send-xapi-statements-from-unity

- cmi5 Specification
  https://aicc.github.io/CMI-5_Spec_Current/

- cmi5 Test Suite
  https://github.com/adlnet/CATAPULT/blob/main/lts/README.md