Scorbit Access on the P^3 System

Nick Baldridge

foramusementonlypodcast@gmail.com

ABSTRACT

Scorbit is a third party platform for tracking scores, game progression, leaderboards, challenges, game status, and more. This document will explain how to use the ScorebitMode and GUI components to access that platform.

Keywords: Scorbit, integration, web service, entry, achievement, leaderboard, heartbeat

INTRODUCTION

Scorbit is an evolving platform outside of the control of Multimorphic. As such, there are certain aspects that are under current or continued development. Over time the needs of the Scorbit Mode and GUI components, and indeed those of third parties may change. Please contact support@scorbit.io or join their Discord for more details at https://discord.gg/FEE7KYbQQY

THEORY OF OPERATION

When the app is bootstrapped, ScorbitMode, if enabled, will start communicating with the service. It provides data about the machine/application including the serial number.

Some unique information is required of each application. These unique pieces of data will be described in each subsection.

Scorbit uses a variety of services to access or provide data. These services are described in the sections below.

The application will send data on a timed basis - if outside of the game, it will let the application know that the game is turned on. If the game is currently active, the application will inform Scorbit of the current game progress, scores, and other data.

Scorbit, as an abstracted series of services, has a few concepts that must be understood to be utilized properly.

0.0.1 Machines

The application/physical machine pairing is known to the Scorbit service as a machine.

0.0.2 VenueMachines

A Machine which is available in a particular location (ex: *Lexy Lightspeed Escape from Earth* at Lookie's Swamp) is known as a VenueMachine. Each VenueMachine must be tied to a Machine in order for the

Scorbit services to function properly.

0.0.3 Providers

A manufacturer is considered a provider. Due to limitations within the Scorbit service for pinball platforms (see Authentication), the manufacturer for released games will be ”multimorphic”. Please contact support@scorbit.io to discuss if an alternative is desired.

0.0.4 Tying a Machine to a VenueMachine

This happens via a special link, used by the Scorbit app on a mobile device. Contact support@scorbit.io to have them tie your game to the venue.

Basic Structure

Scorbit integration is divided into two parts: the Mode-based ScorbitMode and the GUI based Scorbit-GUIOperations. Note that ScorbitGUIOperations are attached to an empty GameObject prefab, but as the functional aspect is just a script, it can be added to the SceneController or any other GameObject.

0.1 Need for Mode and GUI Components

Unity 5.6.x requires an older version of C# that does not include functional modern TLS support on all platforms. Linux and Mac should work using native OpenSSL, however, C# on Windows does not offer TLS support beyond v1.0. Scorbit requires TLS 1.2. Thankfully, Unity does include a handy web service component that allows later versions of TLS on all platforms called UnityWebRequest. This means that each request is actually sent via the GUI side, while responses are typically processed on the Mode side. As you might expect, this can lead to some interesting conditions where a scene may not be loaded and a mode requests that the scene process things. These requests are missed.

0.2 Error Handling

Issues with the various services will log errors that begin with ”Scorbit”.

0.3 Applications

Each application must be registered with the OPDB, then added to Scorbit manually. As each service is maintained by a single user or group, there is a possibility that an application may not be added. If the application is not available in OPDB, there is no way to use Scorbit. Contact support@scorbit.io to ensure your machine is added to the OPDB.

0.4 Adding ScorbitMode

Ideally, ScorbitMode will be added as part of AppBaseGameMode’s mode_started() method.

This allows ScorbitMode to immediately receive commands from other modes.

Once the Attract Scene is loaded, a G2M event is posted:

```
PostGUIEventToModes(”Evt_StartScorbit”, true);
```

0.5 Setting Production Mode

When the application is running on a P3, certain information becomes available that is otherwise unavailable in the Unity simulator. Once you have the integration tested thoroughly in the simulator, and before deployment to your physical P3, you may call the following M2M event from AppBaseGameMode:

```
PostModeEventToModes(”Evt_ScorbitProduction”, 1);
```

This call should NOT be completed if you wish to remain in testing mode. Aside from automatic changes to the Serial (documented below), this will change the endpoint used for Scorbit and will require use of the production mobile application to see your game integration.

0.6 Authentication

The majority of Scorbit endpoints require an authentication token. This token is acquired through a call to the api/stoken endpoint. The ScorbitMode takes care of all the setup for the programmer, however, there are items which must be modified on a per app basis.

0.6.1 UUID

A UUID must be generated and provided to the Scorbit team per app. UUIDs can be generated using:

```
Guid myuuid = Guid.NewGuid();
```

Send this to ScorbitMode using:

```
PostModeEventToModes(”ScorbitUUID”, myuuid);
```

Remember to remove the dashes from the generated UUID before posting.

0.6.2 Serial Number

The Serial Number can only be pulled on a physical machine using:

```
data.GetGameAttributeValue(”Machine Id”).ToString();
```

For testing, ensure that a false serial is entered in ScorbitMode (see ScorbitMode.cs for details). The Scorbit service does not allow for alphanumeric serial numbers as used on the P3, so some mathematical functions are carried out.

0.6.3 Private/Public Keys

The private and public keys need to be generated by the programmer. These are not to be distributed. To generate the keys needed, a computer with OpenSSL is likely easiest.

To generate private key:

```
openssl ecparam -name prime256v1 -genkey -noout -out private.pem
```

Paste the PEM into your app, and Post to ScorbitMode using the method in AppBaseGameMode:

PostModeEventToModes(”Evt_ScorbitPEM”, string as defined below);

Ensure that it includes the header and footer ”BEGIN EC PRIVATE KEY” and ”END EC PRIVATE KEY”. This is used to derive the public key. Also note that the PEM should be pastes as a single line, with '\n' newline characters between the BEGIN and END declaration.

0.6.4 Providers

The provider is hard-coded to ”multimorphic”. Third parties would have to contact Scorbit to be added as additional providers. Contact support@scorbit.io for details and note that the entry in OPDB would also need to reflect this change.

0.6.5 Time

The network time is pulled independently of any other Multimorphic abstraction within ScorbitMode.

0.6.6 Response

Assuming the above are correct, the service will respond with an SToken. This SToken will be used for all other interactions with the service. Services will attempt to pull a new SToken on failure.

The SToken is passed to the ScorbitMode, and is subsequently utilized in all other calls. This allows for the application to change scenes without losing the authentication token.

0.7 Heartbeat

The Heartbeat endpoint sends a small packet of data to let the Scorbit service know that the machine is currently online and functional. It is currently only visible in the Venue Machine listing within the Scorbit application, and shows a green circle if the machine is online, and a red circle if the machine is offline.

No other functionality is available, for example, receiving remote error messages. Other functionality may be planned. Contact support@scorbit.io for details.

Once the system authenticates, the inital heartbeat pulse is sent automatically from ScorbitMode through the event handler for:

Evt_ScorbitSuccessfullyAuthenticated

0.8 Entry

This is the main service used for updating Scorbit with the currently active game session within the app.

The entry service has a variety of required and optional parameters. Each will be defined below, though these are, like anything in the Scorbit ecosystem, subject to change.

0.8.1 Parameters

Session UUID This is a separate UUID that is generated once the first entry is started per game. This is handled automatically for the programmer.

Session Sequence Scorbit uses two pieces of information to track the current state of the game. Session sequence is just an integer that is automatically incremented with each call to entry. This is handled automatically.

Session Time Scorbit uses this as the second piece of information to track the current state of the game. If a user sends over two calls simultaneously, between the Session Time and Session Sequence, the Scorbit backend should be able to determine which is newer/more updated, and present the appropriate information in the mobile application. This is the time since the session started in ms. It is handled by continuous (Update()) posting of:

```
PostGUIEventToModes(”Evt_ScorbitUpdateSessionTime”, Time.deltaTime);
```

Which is then appropriately bundled. The G2M event call is added to each Scenecontroller for any scene in which the game is currently running. This works best either in AppSceneController or within the HomeSceneController contexts as long as that scene remains active, and the programmer can make that determination based on the complexity of their game.

Active This boolean determines whether the Scorbit backend should close the session. ScorbitMode abstracts this away by parsing a boolean sent to the Mode to Mode Event - Evt_ScorbitEntry. A value of true will start or continue an existing session. A value of false will close the existing session.

Current Player This parameter is fed automatically through data.currentPlayerIndex + 1 (indexed starting at 1).

Current Ball Certain game constructs that override the way ball counts and standard game flow may require their own implementations of Evt_ScorbitEntry. As written, this parameter is supplied automatically as data.ball

Current Score For each call to the Entry service, all scores are posted for each active player. This is handled automatically for player counts up to 6. The maximum number of players is 6 in Scorbit. Each call to entry handles removed players by sending -1 to the service for players ¡ 6 but above the total player count. This ensures that additional players are automatically handled, both for additions and subtractions.

Modes Scorbit has a special syntax for dealing with state within the game. Modes are completely optional. This is handled currently through a special key in data.currentPlayer, named ”ScorbitModes”. Syntax and examples follow:

Syntax

1. Mode Category
   - Multiball <MB>
   - Extra Ball <EB>
   - Ball Locked <BL>
   - Wizard Mode <WM>
   - Bonus Multiplier <BX>
   - Not Applicable <NA>(anything that doesn’t fit into a predefined category)
   - Completed <CP>
   - Hide from Session Log <XX>
   - Hide from Clients <ZZ>
2. Colors (optional)

- red
- orange
- yellow
- green
- blue
- purple
- pink

Example

```
data.currentPlayer.SaveData(”ScorbitModes”, ”MByellow:Lights Out;EBpurple:15;WM:Alien
Arbitration”);
```

Usage

Recommended that any time there is a meaningful change in state, the M2M ”Evt_ScorbitEntry” is called, with a parameter of true if the game is still running. Ideally, this would be integrated into the ScoreManager, however, each programmer could add them at each scoring opportunity if desired. The ScorbitModes key can be appended at any time, and will be automatically sent along with the Entry if the key contains data.

An example of ending the game is included in HomeMode’s End() method. A check is made to ensure this is the final ball and final player, then a final call to Entry is made with a parameter of false to close the session. The idea being that the final call to End() will close the session.

After the session is closed, the Scorbit mobile application will present the user with the ability to interact with their session in a variety of ways.

0.9 Achievements

If the programmer would like to add Scorbit achievements to their game, contact support@scorbit.io to receive a CSV layout of the achievements.

ScorbitMode does not currently have an implementation of achievements, though one can be added, easily.

The achievements function using a pull/push system.

First, the application will pull a list of all achievements available for the game. Implementing this when a game has no achievements might cause undesired behavior, so it is left unavailable at the moment.

Next, the application will inform the achievement service as achievements are unlocked. The achievement system keeps track of how many times a person has unlocked an achievement, and some achievements can be hidden or otherwise not exposed in the mobile application.

0.10 Leaderboards

Scorbit has an internal function to allow global leaderboards. As of this writing, to integrate, please contact support@scorbit.io. These leaderboards could easily be added to Attract.

ADDITIONAL CONSIDERATIONS

Cryptography functions are handled by the BouncyCastle plugin, located at https://www.bouncycastle.org/csharp/ , when downloaded, place in the following directory within your Unity project:

Plugins/x86_64/BouncyCastle.Crypto.dll

SUMMARY

Scorbit has several methods for interaction exposed through this integration. There are GUI and Mode-based considerations with the integration and some manual intervention when setting up a new game with Scorbit and the various third party services used. Please contact support@scorbit.io or join their Discord at https://discord.gg/FEE7KYbQQY

SCORBIT GITHUB DOCUMENTATION

https://github.com/scorbit-io/scorbit_api_doc/wiki