README.md

Bitmoji for Games Unity Plugin API

The Bitmoji for Games Unity Plugin provides a convenient set of functions for loading content from the Bitmoji for Games service, including avatars, animations, and props.

NOTE: This plug-in depends on the Snap Kit plug-in for Unity for authentication. Please select Login Kit from the toggles there (Bitmoji Kit and Creative Kit are not necessary for this integration).

If you decide you want to have users' personalized avatars, you need authentication and permission from the Bitmoji group. To obtain permission, please send your non-confidential Snap Kit client ID to games@bitmoji.com. See here for more information.

The Bitmoji for Games Plugin uses a customized fork of GLTFUtility. As such, when creating a new project you **must** make sure that the 4 shaders from GLTFUtility are included in your build, otherwise you will not be able to load most assets. Here is the documentation on how to do so:

To ensure that Unity includes the GLTFUtility shaders in builds, you must add these shaders to the 'Always Included Shaders' list.

1. Open Edit -> Project Settings
2. Open Graphics
3. Scroll to Always Included Shaders
4. Under Size, increase the value by 4 and hit Enter.
5. In the Project panel, navigate to Assets/Bitmoji/BitmojiForGames/Plugins/GLTFUtility/Materials/Built-in.
6. In this directory are 4 .shader files.
7. Drag and drop each of the 4 files into one of the 4 newly created rows in Always Included Shaders.

Many functions have an async variant as well. You can call them with `Async` added to the function name and an additional parameter `Action<GameObject> onFinish` for the callback.

Assets

Level of Detail

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public enum LevelOfDetail : ushort
        {
            LOD0 = 0,
            LOD3 = 3
        };
    }
}
```

Eumeration of the available levels of detail from the Bitmoji for Games service. Possible values are:

- `LOD0` - the high-fidelity version of our avatars. Comes with a fully-articulated rig including individual finger joints, facial blendshapes, and higher-detail geometry. Intended for close-ups and higher-powered devices.
- `LOD3` - the low-fidelity version of our avatars. Comes with a simplified rig, facial texture swaps instead of blendshapes, and low-poly geometry. Intended for smaller views and resource-constrained devices.

Avatars

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static async Task<GameObject> AddAvatarToScene(string avatarId,
                                                               LevelOfDetail levelOfDetail,
                                                               string snapAccessToken,
                                                               GameObject parentObject = null,
                                                               Dictionary<string, string> additionalParameters = null);
    }
}
```

Adds an avatar to the current scene by asynchronously downloading via the Bitmoji for Games service. Returns a GameObject that points to the root node of the instantiated avatar object.

Parameters:

- `string avatarId` - Required - the avatar ID for the request. Retrieved from Snap Kit.
- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `string snapAccessToken` - Required - the authorization token needed for the request. Retrieved from Snap Kit.
- `GameObject parentObject` - Optional - the object in the current scene to parent the avatar to.
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. See **Additional Avatar Parameters** for more information.

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static GameObject AddAvatarToSceneFromFile(string avatarFilePath,
                                                           LevelOfDetail levelOfDetail,
                                                           bool isResourcePath = false,
                                                           GameObject parentObject = null,
                                                           Dictionary<string, string> additionalParameters = null);
    }
}
```

Adds an avatar to the current scene by loading the avatar from a local GLB file. Returns a GameObject that points to the root node of the instantiated avatar object.

Parameters:

- `string avatarFilePath` - Required - the location of the avatar GLB file to load. Typically in `StreamingAssets` if loading at runtime.
- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `bool isResourcePath` - Optional - whether the given file path is a `Resource` or not. If `true`, the loader will attempt to load the resource as a `TextAsset` and read its bytes directly.
- `GameObject parentObject` - Optional - the object in the current scene to parent the avatar to.
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. See **Additional Avatar Parameters** for more information.

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static async Task<GameObject> AddDefaultAvatarToScene(LevelOfDetail levelOfDetail,
                                                                     GameObject parentObject = null,
                                                                     Dictionary<string, string> additionalParameters = null);
    }
}
```

Adds a default avatar to the current scene by asynchronously downloading via the Bitmoji for Games service. A default avatar is a stand-in single color avatar intended for use when users do not have a Bitmoji or their avatar has not loaded yet. This call does not need authentication and can thus be used for prototyping.

Parameters:

- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `GameObject parentObject` - Optional - the object in the current scene to parent the avatar to.
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. See **Additional Avatar Parameters** for more information.

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static async Task<GameObject> AddTestAvatarToScene(LevelOfDetail levelOfDetail,
                                                                  GameObject parentObject = null,
                                                                  Dictionary<string, string> additionalParameters = null);
    }
}
```

Adds a test avatar to the current scene by asynchronously downloading via the Bitmoji for Games service. A test avatar is one that comes from our list of pre-created NPCs. This call does not need authentication and can thus be used for prototyping.

Parameters:

- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `GameObject parentObject` - Optional - the object in the current scene to parent the avatar to.
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. See **Additional Avatar Parameters** for more information.

Additional Avatar Parameters

Most avatar API calls accept a dictionary of additional parameters. This is to allow for optional parameters to be sent to various endpoints without changing the function signature.

Parameters (non-default avatars):

- `usePbr=[true|false(default)]` - Use proper PBR textures for the avatar. LOD0 only.
- `scope=[full(default)|head]` - `full` scope means the entire avatar, head plus body. `head` scope means the head only. Head-only may be useful for face tracker-based applications, for example.

Parameters (test avatars)

- `avatar_id=<avatar ID>` - For test avatars, a random ID is chosen at request time. This ID can be overridden to one of the allowed preset IDs.

Parameters (default avatars)

- `color=<hex color string>` - For default avatars, a random color is chosen at request time. This color can be overridden to one of the allowed preset colors.

Animations

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static async Task<AnimationClip> AddAnimationClipFromLibrary(string animationLibraryId,
                                                                            LevelOfDetail levelOfDetail,
                                                                            string snapAccessToken,
                                                                            string animationBodyType = "default",
                                                                            bool useLegacyClips = true,
                                                                            Dictionary<string, string> additionalParameters = nul
    }
}
```

Adds a Bitmoji-compatible animation clip from the available animation library by asynchronously downloading via the Bitmoji for Games service. Behind the scenes, a GameObject will be temporarily created and then destroyed.

Parameters:

- `string animationLibraryId` - Required - the animation ID from the library for the request.
- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `string snapAccessToken` - Required - the authorization token needed for the request. Retrieved from Snap Kit.
- `string animationBodyType` - Optional - the body type to use for animations, should be either `default` or `heavy`. For LOD0 only. The body type of an avatar can be retrieved from the `AvatarAttributes` component.
- `bool useLegacyClips` - Optional - whether the `AnimationClip` should be loaded as a Legacy animation or not. Typically, animations loaded at runtime function better as Legacy than Mecanim (for example, you cannot currently set the clip to loop in Mecanim).
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. See **Additional Animation Parameters** for more information.

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static AnimationClip AddAnimationClipFromFile(string animationFilePath,
                                                             LevelOfDetail levelOfDetail,
                                                             bool isResourcePath = false,
                                                             bool useLegacyClips = true,
                                                             Dictionary<string, string> additionalParameters = null);
    }
}
```

Adds a Bitmoji-compatible animation clip to the current scene by loading the animation from a local GLB file. Behind the scenes, a GameObject will be temporarily created and then destroyed.

Parameters:

- `string animationFilePath` - Required - the location of the animation GLB file to load. Typically in `StreamingAssets` if loading at runtime.
- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `bool isResourcePath` - Optional - whether the given file path is a `Resource` or not. If `true`, the loader will attempt to load the resource as a `TextAsset` and read its bytes directly.
- `bool useLegacyClips` - Optional - whether the `AnimationClip` should be loaded as a Legacy animation or not. Typically, animations loaded at runtime function better as Legacy than Mecanim (for example, you cannot currently set the clip to loop in Mecanim).
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. See **Additional Animation Parameters** for more information.

Additional Animation Parameters

Most animation API calls accept a dictionary of additional parameters. This is to allow for optional parameters to be sent to various endpoints without changing the function signature.

Currently, no valid additional animation parameters exist.

Props

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static async Task<GameObject> AddPropFromLibraryToScene(string propLibraryId,
                                                                       LevelOfDetail levelOfDetail,
                                                                       string snapAccessToken,
                                                                       GameObject parentObject = null,
                                                                       Dictionary<string, string> additionalParameters = null);
```

```
    }
}
```

Adds a Bitmoji-styled prop from the available prop library to the current scene by asynchronously downloading via the Bitmoji for Games service.

Parameters:

- `string propLibraryId` - Required - the prop ID from the library for the request.
- `LevelOfDetail levelOfDetail` - Required - the level of detail to use for the request. See **Level of Detail** for more information.
- `string snapAccessToken` - Required - the authorization token needed for the request. Retrieved from Snap Kit.
- `GameObject parentObject` - Optional - the object in the current scene to parent the avatar to.
- `Dictionary<string, string> additionalParameters` - Optional - additional parameters to send with the request. Currently, there are no valid parameters to pass into this field.

Stickers

```
namespace Bitmoji.BitmojiForGames {
    public static class Assets {
        public static async Task<Texture2D> GetStickerAsTexture(string avatarId,
                                                                string stickerId,
                                                                bool isFriend = false);
    }
}
```

Add a Bitmoji sticker as a 2D texture by asynchronously downloading via the Bitmoji SDK service.

Parameters:

- `string avatarId` - Required - the avatar ID for the request. Retrieved from Snap Kit.
- `string stickerId` - Required - the sticker ID for the request.
- `bool isFriend` - Optional - indicates whether the sticker being downloaded is for the current user (`false`) or someone else (`true`).

See https://support.canvas.snapchat.com/hc/en-us/articles/360056731112 for more details on getting sticker IDs.

Components

The Bitmoji for Games plug-in provides some special components that will be attached to loaded GameObjects in certain cases.

`FacialExpressionTextures`

The `FacialExpressionTextures` component is a special component that will be attached to the root node of a loaded `LOD3` avatar. Its purpose is to hold the facial swap textures that come with an `LOD3` avatar and use the `FacialSwapAnimationHandler` function to handle face swap events that are triggered by a loaded `LOD3` animation.

`AvatarAttributes`

The `AvatarAttributes` component is a special component that will be attached to the root node of a loaded avatar. Its purpose is to hold information about the avatar that may be useful in certain applications.

Attributes:

- `Assets.CharacterGender Gender` - The gender of the avatar. Currently valid values are `male` and `female` .
- `string AnimationBodyType` - The body type of the avatar to use for animations. Currently valid values are `default` and `heavy` .