Chat UI & Speech Bubbles system

The **Chat UI & Speech Bubbles system** pack provides the base means for easily adding a MMO-style in-game communications system (a scrolling rich-text *chat box* with support for many commands, and a *speech bubbles* system) to your game. The pack, which is designed for the new Unity UI that was released with **Unity 4.6**, focuses on providing the user interface and interaction components, and does so in a thoroughly-documented and extensible way, so that you can easily integrate it with your particular framework.

The behavior and look of the components can be modified easily, most of the time through the unity editor itself. Also, the **full source code** is provided, so that you can check how everything works and modify or extend it as you see fit.

This package does not provide (and it is not intended to) the actual backend system for your chat. That is, it does not provide the chat channels, group channels or private messaging systems. It is designed to provide the user interface for those, which are most often very specific to the game.

Features

General Features

- Implemented with the new Unity UI (UGUI)
- Provided with a conventional, usable look
- Full source code provided
- Well written and commented C# code
- Tries to conform to the current Unity best-practices and to the new Unity UI style

Chat UI

Provides a MMO-style **chat box** - Easy to customize (through the standard new UI tools) - Included (but easy to modify) sample message styles (group chat, spatial chat, private messages) - Emits chat events which can be listened to through the Unity editor (scripting optional)

Speech Bubbles system

Provides a MMO-style **Speech Bubbles** system - Speech Bubbles can be created over any Game Object with a simple call (through a - Singleton-style Manager class) - Bubbles **do not** change size with distance - Bubbles fade-out smoothly when you get too far or when a certain time elapses - Out-of-view bubbles can be automatically hidden. - Size of the Bubbles varies automatically depending on text length (and auto-wraps as needed).

What is included

This package includes several components.

Chat System

The minimum package that you will probably want to add to your project. It contains the scripts, prefabs, assets and other resources that you can use to add the chat components. Within the Chat System folder you can actually find two sub-systems. They will most often be used together, but they do not depend on each other.

Chat UI

Provides the **chat box**. This chat box is implemented through the new Unity UI (UGUI). It is a MMO-style multi-line rich-text chat box. You can type into it, listen for events, and add different types of messages into it. Some assets for the standard look & feel are provided, so that it can be used as-is. It can be customized very easily through the standard new Unity UI approach. For more information, please see the *how to use* section.

Bubbles System

Provides the **Speech Bubbles** system. This system is also implemented through the new Unity UI(UGUI). It is a MMO-style Speech Bubbles system. Generally, in MMO games with a Speech Bubbles system when a player or character speaks a Speech Bubble appears over it. This is a popular way to provide more immersion and to make communication easier, because it conveys spatial information and players can trivially link a message to the player who is saying it, and to "actively listen" to someone.

The main functionality from a user's perspective is to make a Game Object (most often a player or character) say something. The *bubble* will appear over its head, but it is actually implemented through an overlay canvas, and not a world one. This ensures that the system is efficient and that the bubble text is always the same size (that is, does not get smaller as the character's distance increases), which is how most games work.

Chat System Sample

Contains a very simple sample scene, some assets and some scripts to show how the Chat UI and the Speech Bubbles system should be set up and to showcase what they can be used for. Specifically, it includes the sample scene, some sample scripts and some sample resources. It actually relies heavily on some assets provided by the *Standard Assets* package, but those are included.

Some Standard Assets

Some assets from the *Standard Assets* package, which are used for the Sample scene and which do not need to be integrated into your actual project. The character (Ethan) that you can see in the Sample scene, for instance, is from the *Standard Assets* package.

Documentation

The Documentation folder contains this file, which describes the system and components that this

system provides and how to use them.

How to setup from scratch

The included sub-systems (**chat ui** and **speech bubbles**) can be setup independently. This section will describe step-by-step how to setup each of them starting with a blank scene. Even then, you may find it easier to just have a look at the provided *Chat System Sample Scene*.

Setup the Chat UI

1. Add the `GUI_CHAT` prefab (found at `Assets/Chat System/chat/prefabs` to your Scene hierarchy. This object contains the Chat Canvas and the Panel that contains the chat UI. Alternatively, you may add only the Panel to your own Canvas, if you do not want to have more than one in your scene.
2. Create your own chat event handling script. You will probably want to add a `void OnChatCommand(string)` and a `void OnChatSay(string)`. The actual names don't matter, because the functions are referenced from the ChatEventTrigger.
3. Add a ChatEventTrigger script to any object in your scene. You can define the chat event handlers through it in the editor. Add handlers that point to your OnChatCommand and OnChatSay methods.

Setup the Speech Bubbles system

1. Add the `GUI_OVERLAYBUBBLES`prefab (found at `Assets/Chat System/bubble/prefabs`) to your SCene hierarchy. This object
2. contains the Canvas to which the Bubbles will be added, and the scripts needed to control them.

How to use it

Using the Chat UI

- The functionalities that the Chat UI provides can be accessed through the `ChatUIManager` class. `ChatUIManager`is a singleton. It should be placed only once in the Scene, and it can be accessed globally through the `ChatUIManager.Instance`.
- To **add a say-style message** to the chat box, for instance, you can, from anywhere, do: `ChatUIManager.Instance.AddSayText("PlayerName", "Example text!")`
- To **check whether we are in chat-mode** (writing into the chat input field) call: `ChatUIManager.Instance.IsInputFieldFocused()`
- For more *ChatUIManager* features check the `ChatUIManager.cs` file. Every public API method is documented.
- To know when the player has **typed a message** into the chat box you need to **listen** to the *chat events*. There are two types of *chat events*.
  - ChatSayEvent: Standard events emitted when the user types anything that is *not* a command.
  - ChatCommandEvent: Events that are emitted when the user types a command (text that starts with a slash). For instance: "/tell NPC hello".
- You can **listen to events** by defining the handling methods and registering them in a ChatEventTrigger script (as is explained in the *How to setup* section). To learn more about this

style of event handling, which is somewhat typical of the new Unity UI, you may want to read on the new UI's EventTrigger-related topics.

Using the Speech Bubbles system

- To make a Bubble appear over a game object you can do:
  `BubblesManager.Instance.Say(targetGameObject, "Hello!");`
- Optionally, to control **how and where the Bubble is added** you can attach a *BubbleUI* script to your target game object or one of its children. You can change certain parameters on that Script which will affect the behavior of the bubble.
- If you do not manually add a *BubbleUI* to the object then one will be added automatically and transparently when you Say something on that object (which generally won't be a problem).
- If there are problems, it is advisable that the target game object (or one of its children) has a Renderer object to use as a *reference renderer*. An arbitrary *reference renderer* can also be specified through the *BubbleUI* script, if provided. The *reference renderer* is the one that will be used to enable or disable the Bubble dynamically when the object that it is rendered on goes out of view. **All of this is optional and if a Renderer exists the reference renderer can most often be determined automatically**.

Questions & Answers

How do I customize the look of the provided controls for my game?

All of the provided controls are very easy to customize, because they are all implemented through the new Unity UI and sample assets are provided, along with the full source code. Changing the theme of the speech bubbles or the look of the Chat UI itself can be done very easily by simply referencing different 9-slice images, or even modifying the provided ones. Changing the colors of the text messages to add to the *Chat UI* can be done simply by changing the properties in the ChatUIManager instance, which can be found by default into the `GUI_CHAT/Chat Canvas` object. Adding new types of message to the Chat UI can be done by modifying the ChatUIManager script, or simply by relying on the lower-level ChatUIManager.Instance.AddText() method to add raw Unity-UI RichText.

Is your package enough to have a full and working chat system in my game? Does it include networking code?

No. The *Chat UI and Speech Bubbles system* package focuses on providing the components related to the user interface and user interaction. The underlying chat systems vary too much depending on the game, are arbitrarily complex and are very dependent on the specific game architecture.

In a single-player game in which you may want to speak to NPCs only, you would barely need to add anything to the system, other than the NPC speech logic itself, and the responses parsing.

In a fully-fledged MMO, however, you would use this package as a base but you would need to add many things. You would need to add networking code, a chat server to receive and forward the messages, server-side support for the different kinds of messages (spatial chat, private messages, etc.).

Can I add custom commands to the chat?

Custom commands can be added very easily. To add support for custom commands, you need to register a ChatCommandEvent listener in the ChatEventTrigger script, as described in the *how-to* section.

In this handler you will receive every command that the user types, so you would simply parse the command yourself and take whatever action you want to.

How can I ask other questions, get support, or provide feedback?

Please, send an e-mail to luis.rodgi@gmail.com.

Version history

Version 1.0 (02 July 2015) - Initial release version. Includes a Chat UI and a Speech Bubbles system.

Acknowledgements

- To the **Unity developers** for the quite nice long-awaited UI that they have released (which powers this package).
- To the Manapebbles Unity blog for a quite useful post on the new Unity UI on how to simulate a world-space canvas.
- To those of you who have bought this package
- To those of you who have provided very useful feedback.

Contact

This package has been developed by Luis Rodríguez. You can contact him to send your feedback, questions or suggestions at luis.rodgi@gmail.com.