Documentation

Contents

INTRODUCTION

What is Gemini?

Put simply, Gemini is a game service. It is NOT a game server. So what's the difference?

A game server is a physical object that handles receiving, storing, and sending data about a game state. When you play a multiplayer game over the Internet, everything about the current match is being transmitted and relayed to its participants. The server is the reason you can see your teammate obliterate an enemy with a rocket launcher, even though your teammate's game state is in a client computer halfway across the world.

A game service, on the other hand, handles global data about the game. A service runs on a server, but it does not act as a server system would. *Steamworks* and *GameSpy*, for example, are both game services that handle transmission of global data. *Steamworks* is the reason why you can see leaderboards and daily achievements for many popular games. *Halo: Reach*'s game service is the reason why you know how many people are online at any given time.

When playing a game, you can hardly tell the difference between what is handled by a server and by the service. But for those of us in the game development industry, this difference is extremely important. Most developers worry a great deal about "The Server", as if getting a remote match running on a LAN is something amazing. In fact, to get any further than that you'll need a service for your game.

Enter Gemini.

Why Gemini?

Hold on, I know what you're thinking. Why should you entrust the success of your game to Gemini? Why do this when there are a whole host of other options available?

The answer is simple: Gemini is dedicated (not in the way a server is dedicated, though). Gemini was made exclusively for UDK. Its creator has spent the past three years working with UDK. And it was designed specifically for fellow UDK developers.

A lot of the major commercial game services boast a great deal of features and a robust gaming history, but they don't work hand-in-hand with UDK. *Steamworks*, for example, was built by Valve, for Valve. Epic Games recently acquired a partnership with Valve for the service, but this boon comes with a lot of drawbacks – not the least of which is the fact that a *Steamworks* license is VERY demanding.

Gemini is still in an infantile pre-alpha state, but feedback from the gaming community is sure to change that quickly. And not only do I welcome feedback, if there are any questions or problems I am willing to help solve them. Try getting that kind of attention from *GameSpy* ☺.

Oh, and did I mention...Gemini is free?

GETTING STARTED

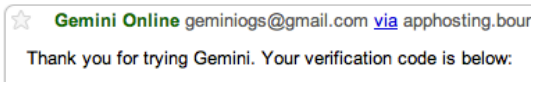
Setting Up Your Account

If you're reading this you probably already have a Gemini account set up. Wait...what's that you say? You don't? No problem, just head on over to http://geminionlinegs.appspot.com/ to sign up.

1)
Enter your name and email. This is just to check that you are a valid user and not a bot – the information will not be stored.

Click "Submit".

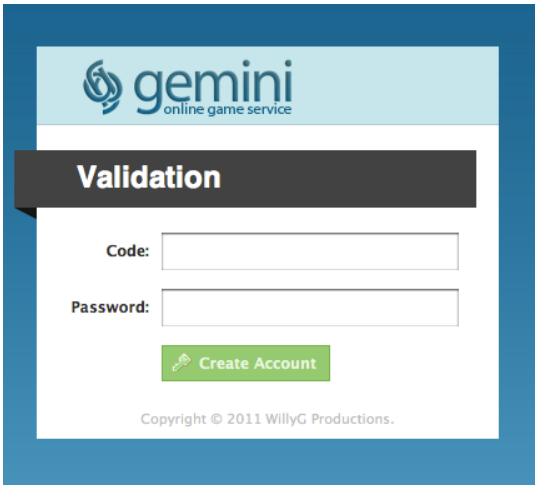
2)
Check your email. Gemini should have sent you a validation code.



3)
Enter the validation code you received in the email in the "Code" box.

Then enter a password for your account that is 3-15 characters long and contains only letters and numbers.

Click "Create Account".



And that's it!

Basic UDK Integration

Now that you have an account, the next step is to get it working with UDK. If you already have experience with UDK's TCPLink system this should be a trivial exercise, you can get right to coding and have a workable system in a few hours.

But wait. Why do that when I've already done it for you? That's right: download the handy little "UDK Starter Kit" from http://willyg302.wordpress.com/gemini/ and you'll have access to the full sources of a basic system using Gemini in UDK!

This system is far from complete, though (it's called "Starter Kit" for a reason). What it does is fetch and parse the messages from your Gemini account given a password and field size and print them to the log. In order to actually do something useful, you WILL have to do a bit of coding. But that's a topic for later on in this documentation.

How Gemini Works

Before diving into UnrealScript, step back for a moment. You'll need to know how Gemini works.

UDK is an amazingly complex beast, but in the Internet department it's rather primitive. It communicates strictly through POST/GET requests, which are the simplest ways that a computer can communicate with a server. A POST request sends data to a site, and a GET request gets data from a site. It's that simple.

Therefore, Gemini has to operate with this in mind. When UDK gives a POST request to Gemini, Gemini takes the data from the request and adds it to your account. Options for receiving data are extremely limited: you can only specify a single string (called "content") to send to Gemini.

When UDK sends a GET request, it actually sends it to a special page on Gemini called "service.jsp". This page is a printout of the visible messages in your account, and it looks something like this:

> <@>Test 2 (again) <@>Test of the better interface <@>JFrame Extended Trial 1 <@>Trial with JFrame <@>cuz i have the pass! :D <@>had to be me <@>aww snap who posted donuts <@>Message 7 <@>Message 6 <@>donuts

UDK receives the contents of this page in a single uninterrupted string, and it then parses the string to retrieve your messages.

You may be thinking, "Well geez, there's not much you can do with string messages!" But you would be wrong. With a little bit of creativity, it's not hard to see how the entire state of a complex FPS can be saved in the space of 10 or so strings. For example, you could just allocate the first string to how many people are online: "14172". Or if you wanted to handle individual game modes, the string could be "14172,423,4332,567,3123" and that would mean "14172 total players, 423 playing CTF, 4332 playing Deathmatch..." and so on.

The system is rather simple, and that is a major benefit. Because UDK is not bogged down with backend SQLs and the like, it can instead focus its power on another thing: your game. All it does is send data to Gemini, and get some data back.

IN-DEPTH GUIDE

The Field

One of the most important concepts to understand is your account's field: the messages (individual strings) that are visible on your "service.jsp" page.

Gemini saves messages in a string list called Messages, and it saves it in reverse order. When you add a new message to your account, it goes into Messages(0) and all the old messages get pushed back one; Messages(4) is now Messages(5), and so on. Regardless of whether the message is visible or not, it still exists in Messages until it is cleared.

However, there is another variable unique to your account called FieldSize. This determines the number of messages that are VISIBLE. Only visible messages get sent to UDK, so it is very important to make sure that your FieldSize is appropriate for your game (we'll get to setting it later).

What Gemini does is set a variable X=FieldSize. Then it displays the first X messages from the list Messages onto "service.jsp" for UDK to grab. For example, with X=7 our previous example would be:

> <@>Test 2 (again) <@>Test of the better interface <@>JFrame Extended Trial 1 <@>Trial with JFrame <@>cuz i have the pass! :D <@>had to be me <@>aww snap who posted donuts

And with X=3 it would be:

> <@>Test 2 (again) <@>Test of the better interface <@>JFrame Extended Trial 1

Using URLs

UDK communicates with Gemini via URL. More specifically, UDK sends a request via this format:

> http://geminionlinegs.appspot.com/service.jsp?content=your+message+here&pass=donuts&code=e34b56c

The question mark allows us to pass data in the form of parameters to the site. So what this does is send three variables, "content", "pass", and "code" to the "service.jsp" page. In this case, assuming your account password was "donuts" and the first 7 characters of your validation code were "e34b56c", you would have just posted a new message (namely "your message here") to the field.

The "pass" parameter is required for you to get the data you need, since it is the only way for Gemini to know what account to access. If you do not provide a "pass" parameter the "service.jsp" page will simply read "ERROR: Account not found." (It would do the same if you provide a password for an account that doesn't exist).

The "code" parameter is also required as a checker to make sure only the owner of the account can access it. If you do not provide this parameter, Gemini will throw an error. It will also do this if you provide the wrong code.

However, the "content" parameter is optional. If you do not provide it, Gemini simply displays the current messages and sends them to UDK – the equivalent of a GET request.

Special Service Functions

The service includes several functions that may be called from the "content" field, and which perform operations other than POST requests. Note that these functions are case-sensitive and exclusive to the service (that is, don't post any message that includes "flush", "replace", etc., or else it will be interpreted as a function).

flush#

Deletes messages that are older than # messages old. To flush all messages, use "flush0" (warning: this would likely cripple the database on the UDK side). Your UDK should automatically handle flushing old messages at every session end, so this function may be redundant if used.
-EX: "flush5" would keep only messages 0-4 in the account, since 0,1,2,3,4 counts as 5 messages (with 0 as base)

replace#,X

Replaces message # in the service with a new message X. # must be an integer less than the field size of your account, where 0 is the newest (top-most) message when viewing the "service.jsp" page. This is extremely useful since it doesn't push old messages down the stack.
-EX: "replace7,Hello World" would replace the 8th message in the field with "Hello World"

delete#

Deletes message # in the service (which pushes all messages below it up in the stack). Currently the message must be visible to be deleted, and so # must be an integer less than the field size, with 0 being the top-most message when viewing your account's "service.jsp" page.
-EX: "delete8" deletes the 9th message in the field and pushes all messages below it up

<@>

Not really a function, but UDK's way of parsing Gemini's string dump into individual messages. Notice that each message you POST is appended to <@> which allows it to be split at that point. So, it is possible to hold more messages than the field size by putting <@> into your POST.
-EX: Posting "hello<@>world<@>!" would cause UDK to receive 3 messages, "hello", "world", and "!"

setfieldsize#

Sets your account's field size to #. # must be an integer between 1 and 50; the maximum is 50 to avoid slowing down UDK unnecessarily when doing multiple GET requests. The default is 10, lightweight games can generally go 3-5, MMORPG-type games may need 30+.
-EX: "setfieldsize15" would set the field size of your account to 15 – "service.jsp" would display 15 messages

perform#,op$,num&

Performs a basic (+, -, *, /) operation on message # using operation $, where ($=1) is add, ($=2) is subtract, ($=3) is multiply, and ($=4) is divide. The other input is an integer &. The message # must be an integer, and the resulting value is always an integer.
-EX: "perform2,op3,num5" would multiply the 3rd message in the field by 5 and store the result to the same message

append#,X

Appends string X to the end of message #. # must be an integer less than the field size of your account, where 0 is the newest (top-most) message when viewing the "service.jsp" page. This is a faster way to update messages when the entire string does not have to be deleted.
-EX: If message 4 is "Hello World", then "append3,!" would replace it with "Hello World!"

terminate

Deletes your account from the system. If you decide to quit using Gemini, please send this message to your account to terminate it, or email me your password so that I can do it. Warning: once you terminate your account, you can never recover its data. Use with extreme caution.
-EX: Do you really need one?

If you think of a function that should be included in Gemini, email me an explanation of the function and its purpose. If I deem it useful enough I will add it!

UDK System Setups

Although I won't code any of these examples for wide release, I will give you an idea on how to set up some basic systems using Gemini via a UDK backend.

[Number of Players Online]
Let one of your messages # simply be the number of players online as an integer. Then use perform#op1num1 to add a player to the count when he logs in, and perform#op2num1 to remove a player when he logs out. There is some concern with a double overwrite (that is, two players logging in at the same time may update the value only once if the system doesn't update fast enough), so note the value may not be accurate.

[Daily Achievement]
Let one of your messages always be the daily achievement. Store it in such a way that your UDK system can easily parse the achievement and interpret it. For example, "Hot Potato3,250Get a triple kill with a grenade" can be interpreted as "Title: Hot Potato, Iteration: 3, Points Awarded: 250, Description: Get a triple kill with a grenade".

[Leaderboards]
Let one of your messages always be the leaderboard. For example, if you wanted to store the top five scores in a match you could do: "JimBob,24000,Ezekiel123,21000,BoyardeeKid,19000,Donuts, 7000,Johann,5000". You can see that every odd value is a username, and the next value is its corresponding score. Upon match end, have UDK grab the scores and compare them, and add when appropriate using "replace#X".

[Total Plays & Other Data]
For example, how many times has your game been played by anyone ever? This is an easier version of the "number of players online" example – you simply don't subtract the player upon logoff. Again, may be inaccurate.

If you have a system you would like to implement but have no idea how to do it, just email me for help and we'll sort things out. Also, if I like the idea enough I'll add it to this guide.

RANDOM OTHER STUFF

Contact Me

I have two emails, but for Gemini-related stuff you should use geminiogs@gmail.com. Please use a descriptive subject line and assume that I'm incredibly stupid when explaining your problem. Allow several days for me to respond.

Boring Legal Stuff

By using Gemini, you agree to the Gemini Terms of Service, available here: http://willyg302.wordpress.com/gemini/.

The Terms of Service applies to and covers Gemini's website, http://geminionlinegs.appspot.com/ and all its sub-pages, regardless of whether the Terms of Service appears at these locations.

The End

Happy gaming!