Server-side function

TriggerLatentClientEvent

This function is the same as triggerClientEvent except the transmission rate of the data contained in the arguments can be limited and other network traffic is not blocked while the data is being transferred.

> **Note:** You should avoid triggering events on the root element unless you really need to. Doing this triggers the event on every element in the element tree, which is potentially very CPU intensive. Use as specific (i.e. low down the tree) element as you can.

Syntax

```
bool triggerLatentClientEvent ( [table/element sendTo=getRootElement(),] string name, [int bandwidth=50000,] [bool persist=fa
lse,] element theElement, [arguments...] )
```

Required Arguments

- **name:** The name of the event to trigger client side. You should register this event with addEvent and add at least one event handler using addEventHandler.
- **theElement:** The element that is the source of the event. This could be another player, or if this isn't relevant, use the root element.

Optional Arguments

- **sendTo:** The event will be sent to all players that are children of the specified element. By default this is the root element, and hence the event is sent to all players. If you specify a single player it will just be sent to that player. This argument can also be a table of player elements.
- **bandwidth:** The bytes per second rate to send the data contained in the arguments.
- **persist:** A bool indicating whether the transmission should be allowed to continue even after the resource that triggered it has since stopped.
- **arguments...:** A list of arguments to trigger with the event. You can pass any Lua data type (except functions). You can also pass elements. The total amount of data should not exceed 100MB.

Returns

Returns *true* if the event trigger has been sent, *false* if invalid arguments were specified.

triggerClientEvent vs triggerLatentClientEvent test by DreTaX

I used a simple outputChatBox test, and devtools to see the performance results. Using triggerClientEvent will immediately launch all the data to the client, and will end faster however when using triggerLatentClientEvent, the server needs an unnoticeable (really small) amount of time to launch up the event in to a thread. This gives benefits to your server, because as stated above, using latent event the server resource uses 0% CPU (Doesn't hold the main thread back), but when using the normal trigger It gave me a 15% CPU usage. Putting both of these in a loop will provide you sufficient understanding.

```
for i = 1, 10000 do
   --trigger the event to client side using one of the functions.
end
```

I suggest using latent event instead to ensure your server won't hold back to any sync struggles.