Task 16 Spike: Messaging Extended

OPTIONAL

Context

Message systems can support a number of very useful features, such as broadcasting, filtering messages based on game-world properties, repetition of messages, and scheduling message release.

Knowledge/Skill Gap:

The developer has implemented or is using an existing messaging system and needs to include additional useful features.

Goals

“Extend” the previous spike to include one or more of the following additional message system features. Update your design documents to reflect the changes of your new system.

Possible extension features/support:

- Broadcast messages (specified by the sender)
- Filtering of messages before delivery/pickup (by the blackboard/dispatch system, not the sender),
- Scheduling of messages for the future,

You may want (or need) to include the ability for the sender to “delete” or cancel messages in support of the above features.

Examples for filtering of message could be based on game entity values/types, or locations

You need to produce:

1. Updated design documents (UML class, module, sequence etc) as applicable, clearly showing what you have had to add to support your additional features, and an
2. Updated working code demonstration within the Zorkish game example.

Expected Output

Repository

1. Code
2. Spike Report

Canvas

1. Spike Report

Notes

- Only take on this extension if you are comfortable with your other core work so far. You can always come back to this when you are better prepared.
- You'll need to give some thought to what form your messages will take. Some options are:
    - Create a Message class that contains all the message information
    - Create a Messaging class tree with different Message subclasses containing different information
        - The subclasses in such a tree can be created based on any of: message types, message components, recipient types, sender types, or combinations of these - if you go down this path you'll need to put thought into how each Message subtype will be used.
    - Have a message be a pointer to a chunk of data

        - dangerous for a great many reasons, but has some advantages - if you choose this method, you should demonstrate awareness of the advantages and disadvantages in your specification and spike report
    - A string
        - Sometime simple is best - a string can often contain all the necessary information for a simple message, and is guaranteed to be human readable when things break
    - Other message data formats
        - There are a lot of other message data formats, each with their own advantages and disadvantages, such as XML, JSON, CSV, binary data, etc.
- You'll need to give some thought to how your addressing scheme will handle multiple recipients, and to what will differentiate an announcement-style message from a normal one:
    - will they be different objects?
    - will they use a different system?
    - will they be transparently handled by your messaging system without the sender and receiver knowing the difference?
- You'll also need to consider how your Blackboard will handle messages:
    - will it make repeated attempts to send a message?
    - will objects repeatedly check the blackboard for messages?
    - what will your Blackboard do when a destination is unavailable or invalid?
    - what will your Blackboard do after a message has been received? will it keep or discard it? how long will it be kept?
- It's OK for the answer to the above questions to be "not implemented" or "the system breaks" - you just need to have demonstrated you've given some thought to these and similar problems