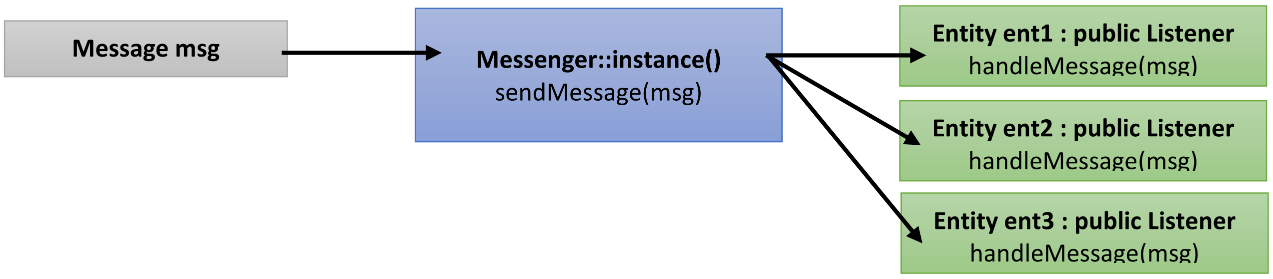
Messaging Specification

FORD KILLEEN 9731822

Below is my messaging system specification as designed for and used in my Zorkish game implementation.

The Flow

Here is a basic diagram showing the flow of a message, as it gets sent to the Messenger for forwarding to all listeners.



Sending a Message

Messages can be sent from anywhere within the project by creating a Message object and passing it as a pointer to the `sendMessage(Message* msg)` method of the Messenger singleton.

```
Messenger::instance().sendMessage(new Message(Tag::DAMAGE, "10"));
```

Receiving a Message

Messages are received by the `handleMessage(Message* msg)` method in any class that inherits from the `Listener` class. The messenger instance sends down messages to all registered listeners to be handled independently by each listener.

```
void Entity::handleMessage(Message* _msg)
```

Message Addressing

This particular messaging system does not use an addressing protocol, rather it gives a message a `Tag` that can then be used by the listeners to determine whether they care about the message or not.

```
enum Tag { ALL, HEALTH, DAMAGE };
```

Message Contents

A message consists of a unique id (an integer) which is assigned to each message by the `Messenger`, a Tag which identifies who or what should care about this message and what the message will contain and finally the contents of the message itself in string format.

```
private:
    int id;
    Tag tag;
    string content;
```

Registering as a Listener

To register as a listener the item just needs to be added to the Messenger instance through the `addListener(Listener* listener)` method. Once added as a listener that object will get every message sent out by the Messenger and be able to determine whether or not it should care about it.

```
Messenger::instance().addListener(player);
```

Sender Details

Currently my messaging system does not pass through any data regarding who sent the message or where it came from.