Spike: 12
Title: Announcements & Blackboards

Author: Ford Killeen, 9731822

Goals / deliverables:
Goals this spike aims to achieve:

- Extend upon Zorkish by adding to the messaging system to allow for announcements and blackboards
- To put some thought into how announcements and blackboards are implemented. Blackboards letting entities read messages at any time and announcements being sent to all entities.

Deliverables required:

- Code showing off the updated messaging system
- Spike report
- Messaging specification

Technologies, Tools, and Resources used:
The following is required to complete this spike:

- Visual Studio 2015
- Zorkish game specification
- Online C++ references

Tasks undertaken:
The list below details the steps taken to complete this spike.

- The first thing I did was add a `map<int, Message*>` to the `Messenger` class to act as the blackboard to which messages will be written.

```
map<int, Message*> blackboard;
```

- Next the `addBlackboardMessage(Message* msg)`, `removeBlackboardMessage(int msg)` and `viewBlackboard()` methods were added to Messenger class. This allows messages to be written to the blackboard and removed when they are no longer needed, as well as be read by entities at any time.

```
void addBlackboardMessage(Message* _message);
void removeBlackboardMessage(int _mID);
map<int, Message*> viewBlackboard();
```

- I then added the `sendAnnouncement(Message* msg)` method which would send the message to all registered listeners.

```
void sendAnnouncement(Message* _message);
```

- The `Listener` class had no way for entities to check the blackboard when they needed to, this was fixed by adding a `checkBlackboard()` method to the `Listener` class.

```
virtual void checkBlackboard() = 0;
```

- This was then tested by letting the player leave a note within the game, which would be put on the blackboard and allowed to be read back at a later time. The current `sendMessage(Message* msg)` calls were changed to `sendAnnouncement(Message* msg)` so that all entities received the messages.

```
cout << "You leave a message on a note." << endl;
Messenger::instance().addBlackboardMessage(new Message(Tag::ALL, cmds[2]));
```

```
cout << "You look at the paper note, it says:" << endl;
map<int, Message*> msgs = Messenger::instance().viewBlackboard();
for (map<int, Message*>::iterator i = msgs.begin(); i != msgs.end(); i++)
    cout << " - " << i->second->getContents() << endl;
```

```
 -> use paper hello
You leave a message on a note.
Entity health: 20

 -> use eyes
You look at the paper note, it says:
 - HELLO
Entity health: 20
```

What we found out:

By completing this spike we found out how useful different messaging methods are and how they can be used. Now we are able to send announcements that will be received by all registered listeners, for cases where multiple entities need to know about an event, and how a blackboard can be used to leave messages for other entities which may want to check them some time in the future.