Sprint 2 Feedback – Feb 6, 2014

# Client

* You seem to be using function pointers rather than an abstract class with methods on it that can have subclasses derived from it to take the place of function pointers. This is a more object-oriented approach to the solution. For example, you could have   
  class OnMouseHoverListener{  
   void handleEvent(SDL\_MouseEvent \*evt) = 0;  
   };  
   which could have subclasses created to define the handleEvent method. Then, you would put a pointer to this object instead of a pointer to a function.
* You only allow one handler for each event. It is often useful to have several handlers for each event and to create a vector of the handlers. When the event occurs, each of the handlers on the chain is called. This allows subclasses of GUI elements to inherit the behavior of their parent class and then augment it by adding new handlers to the chain.
* The GuiContainer contains lists of buttons, text elements etc. These are all derived from GuiElement so why not have just one list of GUIElements? Is this not simpler than maintaining separate lists? Then to draw the elements in the container you just go through the list of elements and tell each of them to draw themselves.
* GuiContainer does not inherit from GUIElement. Is a container not just another GuiElement which draws th4e elements within it? If it becomes a GuiElement, then you can place a container anywhere on the screen and have elements within it. This will require that you implement a pointer to the parent window for every GuiElement and offsets will become relative to the parent container.

# Server

* It says that each game will have 6 clients. Surely this is a maximum and less than that can be handled efficiently as a vector would do? (I see this is resolved in your data structure)
* It is not obvious how ClientLiaison will access the data necessary to update the client.
* It looks like part of DBLiaison is to maintain a local cache of objects from the database. This is actually a good idea, but how it is implemented needs to be thought out more carefully. The simplest idea is that the entire game is read into memory at once and then written to the database at once. A more compelx idea is that each game maintain a list of modified objects which need to be written to the database because they have been modified. This will be more efficient, but will be more work. It might also be useful in determining what needs to be sent to the clients.
* If you do decide to use XML as the interchange format for use with the level editor, it should be converted to the same internal data structures as used by the database people.

# Network

* In Serializable the concept of Data being used to specify what needs to be serialized seems to specific to this problem. I think the idea needs to be generalized to where you can specify individual fields which need to be serialized or deserialized. At the least, it needs a higher level resolution that will be more applicable to a wider range of data structures.
* I expected that the Packet would have an easy way to turn it into a stream of bytes. It might even implement Serializable. At the moment, this is noticeably missing.
* I never did find the UML people referred to in the SCRUM report. Did it fail to get checked in?

# Database

* Skeleton code is missing the .sln file to open the project
* One thing that I do not see in the data structures is the concept of a game. It looks like the player is the equivalent of the game and uses a teamID to represent the game. It would be simpler to use a Game object to represent a game and then it would have a list of players and the structure of the game. This would yield one copy of the structure of the game rather than one copy per player which is what it looks like you have now.
* Network is using a namespace for their code. Should you adopt the same top-level namespace?