Cluedo Logic – How to

Overview on the Logic subcomponent

The logic interacts with the other subcomponents using the Observer Pattern. Observer is used by the logic to notify other components of events happening within it. Interaction and input from other components to the logic is done by calling methods exposed by the logic class itself.

Notifications details

The sender of an Observer Notification is always the CU_GAME class that is set as Observable (subject). To receive notifications sent by the logic, you need to subscribe to the Observable using the “add_observer” method. The controller classes already manage this mechanism.

In order to make this mechanism work, a classe of the another component must to inherit from CU_OBSERVER class and implement its *update*() method.

These are the notifications sent from the logic. The notifications are represented by a CU_MESSAGE object which contains two attributes: a string ID and an array of ANY data.

ID	*Array of data*	*Description*
new_turn	<<>>	It is sent at the end of the turn.
new_player	<<a_player: CU_PLAYER>>	It is sent when a new player is created. a_player is the new CU_PLAYER object
player_removed	<<a_player: CU_PLAYER>>	It is sent when a player is removed from the game. a_player is the removed player.
player_loses	<<a_player: CU_PLAYER>>	It is sent when a player loses the match. a_player is the player who loses.
player_moved	<<a_player: CU_PLAYER>>	It is sent when a player moves his pawn. a_player is the player who moved.
moved_pawn	<<?>>	It is sent when a pawn is moved as a consequence of a suggestion/accusation.

moved_weapon	<<?>>	It is sent when a weapon is moved. It happens when the weapon is teleported due to a suggestion/accusation.
suggestion_is_correct	<<>>	It is sent when the suggestion is correct.
suggestion_can_be_rejected	<<rejecter: CU_PLAYER, suspect: CU_SUSPECT_CARD, weapon: CU_WEAPON_CARD, room: CU_ROOM_CARD>>	It is sent when the suggestion can be rejected. The rejecter is the player who is able to reject the suggestion. The other attributes are the cards in the suggestion.
accusation_ok	<<>>	It is sent when the accusation is correct. This implies the end of the match
accusation_wrong	<<>>	It is sent when the accusation is wrong. It does not imply that the player loses, since there is a separate notification for it
new_game_state	<<status: INTEGER>>	It is sent when we pass from one phase of the game to the next one. status is the new integer value of the game_state attribute, as defined in the CU_ENUM_GAME_STATES class

Input to the logic

The correct way to give inputs to the logic is calling a method in one of the classes inheriting from CU_LOGIC. These classes implement some common methods for the flow of the game, and some individual methods for actions happening in only one game mode.

This part of the interface if still not fully implemented as there is a need to understand what other components really the logic to provide them.

In general, the rcv_message() method can be used for sending general messages and commands to the logic. It receives as parameter a CU_MESSAGE object which should be used with the same logic described above for the Observer notifications.

While the development goes on, we will integrate more methods and more messages in the rcv_message() method, but the key to this is that Team Milano1 should receive requests and details from Team Rio Cuarto1 in order to understand what the other components need and what message formats they will use to send things to the logic. It is easy for the logic to change from the representation of a room as a CU_ROOM or as an INTEGER represented in it respective ENUM class, but other components should give us hints about what is easier to use and pass for them.