Model View Controller (MVC) draft

Problematiche: azioni multiple all’interno di uno stesso turno.

Casi:

1)Quando ho possibilità di scelta sul metodo da pagare.

2)Casistiche di trade: possono essere proposte più opzioni di trade da alcune carte ed,inoltre, il giocatore deve avere anche la possibilità di rifiutare il trade.

3)Esistenza di una carta scomunica che chiede di guadagnare un legno oppure una pietra in meno. Quindi è necessario chiedere al player se vuole sacrificare un legno o una pietra.

4)Secondary action: “Prendi una carta”,”Attua produzione/raccolto” con successiva richiesta del numero di schiavi da utilizzare e scelta di quale carta.

-----------------------------------------------------------------------------------------------------------------------------------------------

Come gestirlo in MVC?

1)Utilizzo di un Controller centralizzato.  
Viene creata una classe Controller legata al Player, con al suo interno diverse variabili che verranno poi utilizzate come “flag”(es: -boolean EnableSecondaryAction, takeCard ).

Il Controller, una volta eseguita l’azione primaria, verifica i flag e attiva le parti di codice (secondary actions) adatte alla situazione, delegando il compito di interagire con l’utente alla view.  
(pro: non necessità di deep-cloning delle carte perché le risorse non si metterebbero in pausa)

2)Quando il gioco deve interrompersi per interfacciarsi con l’utente viene chiamato il metodo di notifica più adatto, tramite la view.

In quale modo?

Come si comunica alla view questa richiesta?  
  
Come fa il controller a sapere che deve aspettarsi una determinata azione?

Richiede deepcloning per evitare deadlock!?

* C è un round controller: creato all ‘ inizio del turno , associato a un giocatore e che resta in funzione per tutto il turno.
* All’ inizio del round il round-controller si aspetta un azione e vedrà se è valida creando l action-controller adeguato, e semmai eseguendo il “run” che modificherà il model
* Se una carta ha necessità di interagire con utente:

1. Setta un flag di una classe associata al player che setta ad esempio che il controller dovrà aspettarsi una “ secondary action”
2. Setta l ‘ observer come changed
3. Si chiude

* L esecuzione ritorna al round controller che controlla il flag e quindi sa se deve aspettarsi un comando dalla view(e questo permette anche di fargli partire il timeout d attesa)

Questo esempio vale per la secondary action. a livello della scelta su quale metodo del pagamento sarà il towerActionController a controllare il flag e a sapere che deve aspettarsi un azione