Sprint 1 Backlog **[06/07/2017 - Cesena]**

Durante questo lungo sprint ci siamo anche incontrati a gruppi o tutti insieme per decidere diversi dettagli implementativi prima di partire con la programmazione.
In totale ci siamo incontrati circa 5 volte.

Revisione delle cose fatte:

Branch 'feature-prolog'

Abbiamo scritto la teoria che gestisce il comportamento del giocatore (controllo se posso fare un certo movimento, se ho mangiato qualcosa o se sono stato mangiato, ecc. ecc.) e creato dei test per verificarne il funzionamento, testando separatamente le funzionalità di pacman e le funzionalità dei ghost.

Siamo riusciti ad arrivare ad una versione funzionante, possibilmente migliorabile, di questo comportamento.

Fare i test non è stato molto divertente, ma è stato interessante fare parti diverse. È stato molto importante fare per bene la parte di progettazione e uml subito in modo da avere una base comune.

Branch 'feature-gui'

È andata abbastanza bene, abbiamo quasi completato la versione locale del gioco per quanto riguarda la sua parte prettamente visuale.

Restano da fare l'integrazione con la parte di model e controller per ottenere il funzionamento del gioco. Ora infatti il gioco funziona solo in modo visuale senza prendere valori dal model o rispondere al comportamento specificato da prolog.

L'unica parte di controller già implementata è una prima versione del keylistener per poter muovere il personaggio all'interno della mappa creata usando la tastiera.

Branch 'feature-characterModel'

Abbiamo iniziato a creare le interfacce degli oggetti del model che riguardano i personaggi e le loro implementazioni. Abbiamo lasciato aperte alcune questioni legate al comportamento perchè andranno implementate usando il comportamento dato dalle teoria in prolog.

Branch 'feature-GameModel'

Abbiamo creato una prima versione delle interfacce e delle relative implementazioni degli oggetti che andranno a comporre il model del gioco stesso. Abbiamo creato una serie di

classi che rappresentano gli elementi della partita (i frutti, i punti, …) e altre che rappresentano elementi della mappa (p.e. I blocchi che compongono i muri).
Infine abbiamo creato una classe (Playground) che rappresenta un'intera mappa di gioco, e che verrà usata dalla view per disegnare i campo per la partita corrente.
Sicuramente sarà possibile ottimizzare diversi aspetti di queste implementazioni.