


Third Sprint

Third Sprint - Development

Sprint Backlog (foglio *User Stories 3rd Sprint Planning*):  Tabella user stories

Tasks assegnati ad ogni membro del team

Al seguente link, nel foglio *Tasks 3rd Sprint* è possibile osservare i tasks assegnati ad ogni membro del team e il loro stato al termine della sprint (completati, assegnati ma non completati, non assegnati).

 Tabella user stories

User Stories aggiunte o modificate dal Product Owner:

- Nessuna User Story da aggiungere per il Product Owner

Third Sprint - Review

User Stories complete alla fine dello Sprint

Sono state completate tutte le User Stories pianificate per questa terza Sprint; in più, avendo concluso queste in un tempo minore rispetto a quello stimato inizialmente, è stata aggiunta anche l'ultima User Story mancante, ovvero la 5.4. Di seguito, l'elenco di tutte le User Stories portate a termine:

- 3.1) Zoom
- 4.1) Inserimento di poligono arbitrari
- 4.2) Inserimento di testo come forma con dimensione personalizzabile
- 4.3) Rotazione di una forma ad angolo arbitrario
- 4.4) Specchiatura orizzontale e verticale di una forma
- 4.5) Allungamento orizzontale o verticale di una forma
- 5.1) Selezione multipla e raggruppamento di forme
- 5.2) Separazione di un gruppo di forme
- 5.3) Salvataggio forma personalizzata creata
- 5.4) Persistenza e riutilizzo delle forme personalizzate

Tasks per la User Story 5.4

5.4.1 Aggiunta componenti grafici

Aggiunta di componenti grafici all'interno della toolbar, nella sezione dedicata alle forme personalizzate salvate, che fungeranno da punti di accesso per l'importazione e l'esportazione di pacchetti di figure personalizzate da parte dell'utente.

5.4.2 Implementazione logica bottoni

Implementare la logica di importazione ed esportazione di librerie di forme personalizzate. In particolare questi due metodi devono essere scatenati al click dei rispettivi bottoni. Implementare inoltre il FileChooser e i rispettivi metodi nella classe model.

5.4.3 Test funzionali importa ed esporta

Verificare il seguente caso d'uso, l'utente salva un certo numero di figure personalizzate, tramite il contextMenu e assegnandogli un nome univoco per la forma. Successivamente può chiudere e riaprire l'applicazione, e attraverso l'apposito bottone selezionare il file da cui desidera importare le figure che aveva precedentemente salvato. Le forme personalizzate salvate riportano tutti i nomi che gli avevano assegnato e disegnandole i loro attributi coincidono con quelli in fase di salvataggio. Inoltre sarà possibile per l'utente ricreare quelle forme.

User Stories rifiutate dal Product Owner: nessuna

Project Velocity misurata: 45 story points.

Durante la Sprint abbiamo ritenuto opportuno, dopo una consultazione con il Product Owner, effettuare un backlog refinement in cui abbiamo unito le User Stories 5.4 e 5.5; la nuova User Story prodotta è: **5.4 Persistenza e riutilizzo delle forme personalizzate**, a cui sono stati assegnati 8 story points e 8 value points, generando un BFTB unitario.

Design aggiornato

Cliccando sul link seguente è possibile osservare il Class Diagram aggiornato (non sono state inserite tutte le classi dei package state e command per motivi di leggibilità, ma ciò non porta alcuna perdita di contenuto informativo):

■ [ClassDiagram_TerzaSprint.svg](#)

La Sprint è partita con l'obiettivo di introdurre nuove forme, come testo e poligoni arbitrari, caratterizzate da esigenze diverse rispetto a quelle gestite dal Factory. L'adozione di stati specializzati ha permesso di gestirle efficacemente, nonostante ciò il Factory si è confermato una scelta valida per l'estendibilità con nuove forme regolari.

Inoltre, durante questa Sprint abbiamo deciso di utilizzare il pattern **Composite** per la gestione di gruppi di forme eterogenee. Questo prevede la necessità di un'interfaccia comune che deve essere implementata sia dalla classe Container (**Gruppo**) sia dalle foglie, ovvero le singole forme (che nel nostro caso sono sia le forme regolari, ovvero **Rettangolo**, **Linea** ed **Ellisse**, sia **Testo** e **Poligono**). Per fare ciò l'interfaccia **FormaPersonalizzabile** è stata modificata, inserendo la firma di tutti i metodi che rappresentano operazioni comuni sia agli elementi semplici che a quelli complessi per la modifica dello stile. In questo modo, sia la classe Gruppo che le classi delle forme possono implementare la stessa interfaccia comune, che rappresenta l'interfaccia Component del pattern e che rende possibile manipolare insiemi di forme eterogenee attraverso un'unica interfaccia (la classe Gruppo nella pratica implementa **FormaBidimensionale** che a sua volta estende **FormaPersonalizzabile**).

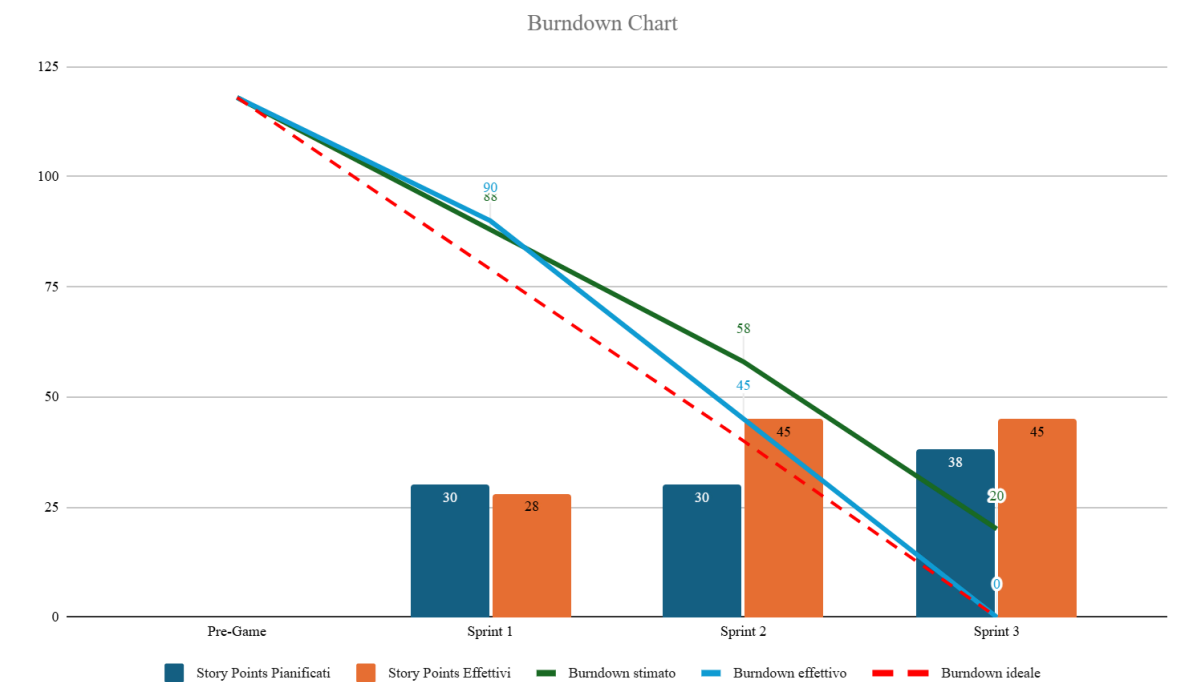
La possibilità di riutilizzare i comandi sul gruppo composito è stata garantita dall'implementazione di un'interfaccia comune da parte dei componenti, ovvero **FormaPersonalizzabile**, che ha permesso di applicare facilmente gli stessi comandi anche ai nuovi oggetti, semplificando l'integrazione. Per operazioni complesse, come il cambio di colore o lo spostamento in avanti di ogni singolo elemento del gruppo (dove ogni "foglia" mantiene una propria identità) è stato introdotto un comando specifico. Questo itera sugli elementi del gruppo, applicando a ciascuno il comando esistente pensato per le singole figure, già testato e affidabile. Sebbene questa soluzione si discosti dal tipico uso del pattern Command in ottica undo/redo, meno naturale da integrare in una struttura gerarchica come quella del Composite, si è comunque ottenuta una soluzione solida, combinando i vantaggi di entrambi i pattern: modularità, riusabilità e correttezza del comportamento.

Infine, per la gestione delle forme personalizzate riutilizzabili dall'utente, è stato adottato il pattern **Prototype**, che consente di clonare figure precedentemente create anziché ricrearle ex novo. È stata implementata una mappa che associa il nome della forma all'istanza corrispondente di **FormaPersonalizzabile**. Quando

l'utente richiede l'inserimento di una figura salvata, il sistema ne genera una copia tramite clonazione, preservando tutte le proprietà originali.

Burndown Chart

Di seguito è possibile osservare il burndown chart aggiornato in cui vengono messi a confronto: il burndown ideale, ovvero il numero di story points da effettuare idealmente in ogni sprint in modo tale da raggiungere il completamento del progetto nel tempo messo a disposizione per le tre Sprint; il burndown stimato, che tiene traccia degli story points ancora da completare in base alla stima della velocità per ogni sprint, e il burndown effettivo, che invece rappresenta il lavoro realmente completato per ogni Sprint. Come è possibile notare dal grafico, il progetto è stato portato a termine entro le tre Sprint grazie ad un incremento della velocità rispetto alle stime nelle Sprint due e tre.



#Sprint	Story Points Pianificati	Story Points Effettivi	Burndown stimato	Burndown effettivo	Burndown ideale
Pre-Game			118	118	118
Sprint 1	30	28	88	90	79
Sprint 2	30	45	58	45	40
Sprint 3	38	45	20	0	0

Third Sprint - Retrospective

Le decisioni prese durante il retrospective meeting di questa Sprint, che potranno essere un valido punto di partenza per progetti futuri, sono:

Stop (things to stop doing):

- Rimandare il refactoring di codice problematico.
- Affrontare modifiche sostanziali senza una discussione tecnica preventiva.

Less of:

- Rimandare l'esecuzione dei test senza completare la User Story, iniziandone però una nuova.

Keep doing:

- Supporto tra i membri del team.
- Attenzione alle Definition of Done e ai Criteri di Accettazione delineati.
- Delegare le task relative a testing ai membri del team che non si occupano dell'implementazione della funzionalità in esame.
- Condivisione trasparente degli ostacoli riscontrati.

More of (more things to do):

- Fare più attenzione nel mantenere la dashboard di Trello aggiornata e consistente(ad esempio archiviando e spostando le schede concluse).
- Aggiungere i commenti JavaDoc subito dopo aver completato la logica relativa alla User Story considerata.
- Incontri informali di brainstorming su problemi tecnici ricorrenti.

Start (things to start doing):

- Svolgere sessioni di code review in coppie a rotazione per facilitare l'individuazione di errori logici e migliorare la qualità del codice.
- Pair programming in momenti chiave.