Third Sprint Planning

Velocità stimata del team per la Sprint: 38 story points

User Stories scelte per la Sprint

Al seguente link nel foglio *User Stories 3rd Sprint Planning* è allegata la tabella contenente tutte le User Stories scelte per la terza sprint.

https://docs.google.com/spreadsheets/d/1jYiJdclKfv4hlbiJVQQ3ZSFj0A611gO-KJ7naodZIP0/edit?usp=sharing

Task per ciascuna User Story pianificata

User story 3.1 Zoom

3.1.1 Aggiunta componente grafico

Aggiunta del componente grafico nella toolbar per la gestione dei 4 livelli di zoom.

3.1.2 Implementazione logica zoom

Implementazione della logica di funzionamento dei 4 livelli di zoom.

3.1.3 Test funzionale zoom

Esecuzione dei functional test cases per il corretto funzionamento dei 4 livelli di zoom.

User story 4.1 Inserimento di poligono arbitrari

4.1.1 Inserimento bottone creazione poligono

Utilizzare Scene Builder ed aggiungere un bottone per la creazione di un poligono nella toolBar, inoltre impostare un evento legato all'onClick nel controller.

4.1.2 Creazione classe Poligono

Creazione della classe Poligono nel package delle forme, in particolare questa deve estendere Polygon di javaFX e implementare FormeBidimensionali. Quindi scrivere e fare l'override di tutti i metodi necessari andandosi a uniformare con le altre forme già implementate.

4.1.3 Creazione ComandoCreazionePoligono

Creare una nuova classe ComandoCreaPoligono la quale implementa l'interfaccia CommandInterface, l'execute e l'undo devono permettere la creazione di un nuovo poligono con la relativa eliminazione.

4.1.4 Creazione StatoCreaPoligono

Creare una nuova classe StatoCreazionePoligono nel package degli stati. In particolare questa classe deve permettere l'attivazione dei tipici eventi del controller andando a implementare l'interfaccia StateInterface.

4.1.5 Aggiunta metodo alla classe Model

Creare un nuovo metodo ***creaPoligono*** nella classe Model il quale si occuperà della creazione effettiva e dell'istanziazione del Poligono.

4.1.6 Scrittura UnitTest per Poligono
Creare gli unit test per la classe Poligono.
4.1.7 Aggiornamento Carica e Salva File
Aggiornare i metodi “caricaFoglio” e “salvaFoglio” per includere il caricamento e il salvataggio di un poligono.
4.1.8 Test funzionali Poligono
Eseguire test funzionali per verificare che il passaggio allo StatoCreazionePoligono avviene correttamente, così come la creazione della figura. Inoltre eseguire test funzionali per il caricamento e salvataggio di un file con dei poligoni diversi e con diversi attributi.

User story 4.2 Inserimento di testo come forma con dimensione personalizzabile
4.2.1 Aggiunta componenti grafici
Aggiunta dei componenti grafici nella toolbar per l’inserimento del testo e la scelta della relativa dimensione.
4.2.2 Aggiunta classe Testo
Aggiunta della classe ***Testo*** nel package delle forme che deve estendere Text di JAvaFX implementare ***FormeBidimensionali***. Quindi, scrivere e fare l’override di tutti i metodi necessari seguendo ciò che è stato fatto con le altre forme già implementate.
4.2.3 Aggiunta classe ComandoCreazioneTesto
Aggiunta della classe ***ComandoCreazioneTesto*** per l’implementazione della logica di funzionamento della creazione del testo con la dimensione scelta e l’annullamento di tale operazione (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).
4.2.4 Aggiunta classe StatoCreazioneTesto
Aggiunta della classe ***StatoCreazioneTesto*** nel package degli Stati; essa deve imlementare l’interfaccia ***StateInterface*** ed è necessaria per permettere l’attivazione degli eventi di click sinistro e destro (dei relativi metodi presenti in ***StateInterface*** viene effettuato l’override) .
4.2.5 Integrazione logica testo nel DashboardController
Integrazione della logica di funzionamento per la creazione del testo nella classe ***DashboardController***.
4.2.6 Aggiunta metodi classe Model
Aggiunta alla classe ***Model*** dei metodi necessari alla creazione e all’eliminazione di un testo.
4.2.7 Aggiornamento metodi salva e carica file
Aggiornamento dei metodi per salvare e caricare un file di disegno con le operazioni necessarie al salvataggio e al caricamento di un testo.
4.2.8 Scrittura test unitari testo
Scrivere i test unitari relativi alla funzionalità di creazione del testo.
4.2.9 Esecuzione test funzionali
Eseguire i functional test cases relativi alla funzionalità di creazione del testo.

User story 4.3 Rotazione di una forma ad angolo arbitrario

4.3.1 Aggiunta bottone per la rotazione nel ContextMenu

Inserimento di un campo di testo all'interno del ContexMenu per la gestione dell'angolo di rotazione della forma selezionata. In particolare deve essere consentito l'inserimento di un qualsiasi grado.

4.3.2 Creazione comando cambia angolo

Creazione di una nuova classe ComandoCambiaAngolo all'interno del package command, in particolare questa classe deve implementare l'interfaccia CommandInterface, l'execute deve garantire la rotazione di una forma mentre l'undo permettere di tornare all'angolo antecedente all'esecuzione del comando.

4.3.3 Aggiunta metodo nella classe Model

Aggiungere il metodo "modificaAngolo" nella classe model. Questo si occuperà dell'impostazione della rotazione scelta alla figura selezionata.

4.3.4 Aggiornamento Carica e Salva File

Aggiornare i metodi "caricaFoglio" e "salvaFoglio" per includere il caricamento e il salvataggio di forme ruotate.

4.3.5 Test funzionali Forme Ruotate

Eseguire i test funzionali per verificare la corretta integrazione dei moduli implementati che vanno a eseguire la rotazione della figura selezionata con l'angolo fornito. Inoltre eseguire test funzionali per il caricamento e salvataggio di un file con forme diverse ruotate.

User story 4.4 Specchiatura orizzontale e verticale di una forma

4.4.1 Inserimento componente grafico per lo specchiamento

Aggiornare il contexMenu della forma con due bottoni, per effettuare rispettivamente uno specchiamento della forma orizzontale o verticale.

4.4.2 Creazione delle classi per il comando

Creazione delle classi comandoMirroringVerticale e comandoMirroringOrizzontalele, nel package command, quali implementano l'interfaccia CommandInterface per le due nuove funzionalità da aggiungere al sistema.

4.4.3 Implementazione della logica di funzionamento

All'interno della classe model, andare a definire i due metodi per l'implementazione della logica di funzionamento per quanto riguarda lo specchiamento orizzontale e verticale.

4.4.4 Integrazione delle funzionalità

Integrare la funzionalità prevedendo che, alla pressione dei due bottoni, venga creato il comando responsabile della corretta esecuzione di entrambe le operazioni, con supporto alla loro reversibilità.

4.4.5 Aggiornamento gestione file

Aggiornare i metodi "caricaFoglio" e "salvaFoglio" per includere il caricamento e il salvataggio di forme specchiate.

4.4.6 Scrittura test unitari
Scrivere i test unitari per verificare il corretto funzionamento dell'operazione di specchiamento.
4.4.7 Test funzionali Specchiamento
Eseguire i test funzionali rispetto allo specchiamento verticale e orizzontale di una forma. Inoltre testare il caricamento e il salvataggio di figure specchiate.

User story 4.5 Allungamento orizzontale o verticale di una forma
4.5.1 Aggiornamento menu contestuale
Aggiornamento dei campi di testo all'interno del ContexMenu per la gestione dell'allungamento arbitrario delle forme Poligono e Testo lungo le loro dimensioni.
4.5.2 Implementazione dell'allungamento delle forme Testo e Poligono
Implementazione dei metodi relativi all'allungamento per le forme Testo e Poligono. Questi devono permettere, di allungare le dimensioni sull'asse scelto. In particolare facendo l'override dei metodi ereditati da FormaBidimensionale.
4.5.3 Aggiornamento Carica e Salva File
Aggiornare i metodi "caricaFoglio" e "salvaFoglio" per includere il caricamento e il salvataggio di forme allungate.
4.5.4 Scrittura test unitari
Scrivere i test unitari per verificare il corretto funzionamento dell'operazione di allungamento delle forme lungo i loro assi.
4.5.5 Test funzionali Allungamento Testo e Poligono
Eseguire i test funzionali per il corretto funzionamento dell'operazione di allungamento di un poligono o di un testo, verificando in particolare: il corretto salvataggio di un file contenente una forma "allungata" e successivamente il caricamento di tale file, e l'operazione di annullamento dell'allungamento.

User story 5.1 Selezione multipla e raggruppamento di forme
5.1.1 Aggiornamento design
Aggiornamento del design con l'inserimento delle classi necessarie all'implementazione del pattern Composite.
5.1.2 Aggiunta componente grafico
Aggiunta del componente grafico nella toolbar per la selezione multipla e il raggruppamento di forme.
5.1.3 Modifica dell'interfaccia FormaPersonalizzabile
Modifica dell'interfaccia ***FormaPersonalizzabile*** aggiungendo le firme di tutti i metodi utilizzati per modificare lo stile e la posizione di una forma. In questo modo, tali metodi potranno essere sovrascritti (override), garantendo coerenza nelle firme sia quando si opera su una singola forma sia quando si lavora su un gruppo di forme, così da rispettare il pattern Composite scelto.

5.1.4 Aggiunta classe Gruppo

Aggiunta della classe ***Gruppo*** nel package delle forme che deve implementare ***FormaPersonalizzabile***. Quindi, scrivere e fare l'override di tutti i metodi necessari seguendo ciò che è stato fatto con le altre forme già implementate.

5.1.5 Aggiunta classe ComandoCreazioneGruppo

Aggiunta della classe ***ComandoCreazioneGruppo*** per l'implementazione della logica di funzionamento della creazione di gruppo con le forme selezionate (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).

5.1.6 Aggiunta classe StatoCreazioneGruppo

Aggiunta della classe ***StatoCreazioneGruppo*** nel package degli Stati; essa deve implementare l'interfaccia ***StateInterface*** ed è necessaria per permettere l'attivazione degli eventi di click sinistro e destro (viene effettuato l'override dei relativi metodi presenti in ***StateInterface***) .

5.1.7 Integrazione logica creazione gruppo nel DashboardController

Integrazione della logica di funzionamento per la creazione di un gruppo di forme selezionate nella classe ***DashboardController***.

5.1.8 Aggiunta metodi classe Model

Aggiunta alla classe ***Model*** del metodo necessario alla creazione di un gruppo.

5.1.9 Aggiornamento metodi salva e carica file

Aggiornamento dei metodi per salvare e caricare un file di disegno con le operazioni necessarie al salvataggio e al caricamento di un gruppo di forme.

5.1.10 Modifica classe ComandoModificaColoreRiempimento

Aggiunta della classe ***ComandoModificaColoreRiempimento*** per includere la gestione della modifica del colore di riempimento di un gruppo di forme e del relativo annullamento dell'operazione (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).

5.1.11 Modifica classe ComandoModificaColoreBordo

Modifica della classe ***ComandoModificaColoreBordo*** per includere la gestione della modifica del colore di bordo di un gruppo di forme e del relativo annullamento dell'operazione (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).

5.1.12 Creazione classe ComandoModificaColoreBordoSingolo

Aggiunta della classe ***ComandoModificaColoreBordoSingolo*** per l'implementazione della modifica del colore di bordo di una singola forma e del relativo annullamento dell'operazione (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).

5.1.13 Creazione classe ComandoModificaColoreRiempimentoSingolo

Aggiunta della classe ***ComandoModificaColoreRiempimentoSingolo*** per l'implementazione della modifica del colore di riempimento di una singola forma e del relativo annullamento dell'operazione (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).

5.1.14 Scrittura test unitari gruppo

Scrivere i test unitari relativi alla funzionalità di creazione del gruppo di forme selezionate.

5.1.15 Esecuzione test funzionali
Eseguire i functional test cases relativi alla funzionalità di creazione del gruppo di forme selezionate.

User Story 5.2 Separazione di un gruppo di forme
5.2.1 Aggiunta componente grafico
Aggiunta del componente grafico nella toolbar per la separazione di un gruppo di forme.
5.2.2 Aggiunta classe ComandoSeparazioneGruppo
Aggiunta della classe ***ComandoSeparazioneGruppo*** per l'implementazione della logica di funzionamento della separazione di gruppo di forme (riscrittura dei metodi ***execute()*** e ***undo()*** di ***CommandInterface***).
5.2.3 Aggiunta classe StatoSeparazioneGruppo
Aggiunta della classe ***StatoSeparazioneGruppo*** nel package degli Stati; essa deve implementare l'interfaccia ***StateInterface*** ed è necessaria per permettere l'attivazione degli eventi di click sinistro e destro (viene effettuato l'override dei relativi metodi presenti in ***StateInterface***) .
5.2.4 Integrazione logica separazione gruppo nel DashboardController
Integrazione della logica di funzionamento per la separazione di un gruppo di forme nella classe ***DashboardController***.
5.2.5 Aggiunta metodi classe Model
Aggiunta alla classe ***Model*** del metodo necessario alla separazione di un gruppo.
5.2.6 Scrittura test unitari gruppo
Scrivere i test unitari relativi alla funzionalità di separazione di un gruppo di forme selezionate.
5.2.7 Esecuzione test funzionali
Eseguire i functional test cases relativi alla funzionalità di separazione di un gruppo di forme.

User story 5.3 Salvataggio forma personalizzata creata
5.3.1 Inserimento componenti grafici
Dedicare una sezione specifica della toolbar alla gestione della lista di figure personalizzate che l'utente desidera salvare.
Inoltre, aggiungere un nuovo elemento al menu contestuale che consenta di salvare una figura all'interno di tale lista.
5.3.2 Implementazione della logica della gestione della lista
Associare al componente grafico la possibilità di mostrare all'utente il nome assegnato alla figura salvata, mantenendo un collegamento tramite una mappa tra il nome visualizzato nella GUI e l'oggetto figura corrispondente.

5.3.3 Implementazione della logica di salvataggio della figura

Associare al componente grafico l'interazione che fornisce all'utente la possibilità di assegnare un nome specifico(unico) alla figura che desidera salvata, andando a mantenere questo binding tra il nome e la figura tramite una mappa.

5.3.4 Creazione dello stato figura personalizzata

Introdurre la nuova classe StatoFiguraPersonalizzata, responsabile della creazione della figura personalizzata selezionata all'interno della finestra di lavoro in seguito a un clic del mouse. La creazione della figura nella finestra di lavoro deve essere gestita attraverso il comando di incolla già implementato, efficace anche per questa casistica.

5.3.5 Integrazione funzionalità di creazione di una forma personalizzata salvata

Stabilire l'integrazione tra i diversi moduli in modo che, alla selezione di una figura personalizzata precedentemente salvata, venga attivato lo stato appropriato, permettendo così l'inserimento della figura nell'area di lavoro tramite un clic del mouse.

5.3.6 Esecuzione test funzionale figura personalizzata

Verificare il seguente caso d'uso, l'utente può salvare una figura selezionata tramite il contexMenu e dopo aver scelto un nome univoco per la forma. Successivamente può ricreare la forma salvata, la quale compare all'utente in una lista. Per ricrearla egli deve cliccare sul nome della forma da ricreare e poi sulla lavagna. A questo punto deve comparire la forma salvata nelle coordinate del click. L'utente può inoltre creare infinite volte la stessa figura personalizzata salvata.

Sprint Goal

Un primo obiettivo di questa Sprint è introdurre quattro livelli di zoom, per migliorare l'usabilità dell'interfaccia e facilitare la creazione dei disegni con maggiore precisione. Un altro obiettivo centrale è estendere le funzionalità di disegno: oltre alle forme regolari, l'utente potrà inserire testi con dimensione personalizzabile e poligoni di forma arbitraria. Verranno inoltre introdotte nuove trasformazioni grafiche, come allungamento, rotazione e specchiatura, per offrire una maggiore libertà nella modifica delle forme.

Inoltre, sarà possibile creare gruppi di forme selezionando più elementi e gestirli come un unico oggetto, con la possibilità di separarli in qualsiasi momento. Infine, sarà implementata una funzionalità per salvare forme personalizzate, utilizzabili all'interno del foglio di disegno e esportabili su file per un facile riutilizzo in progetti futuri.