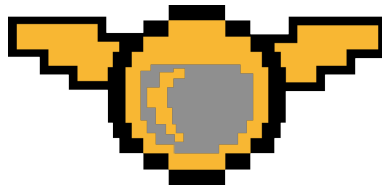


UNIVERSITA' DEGLI STUDI DI MESSINA

ANNO ACCADEMICO 2018/2019

Relazione di Ingegneria del Software

Flappy Bird



Prof. S. Di Stefano

Federica Zuccarello e Aldo Martino

Indice

Indice	1
1. Introduzione	2
2. Processi Software	3
2.1 Cos'è SCRUM?	3
Membri attivi	4
Eventi SCRUM	5
2.2 Specifiche dei Requisiti del Software	7
Analisi dei requisiti	7
2.3 Sviluppo del Software	11
Tecnologie Utilizzate	11
Progettazione dell'architettura	13
Progettazione	13
Operazioni di ottimizzazione	27
2.4 Component Diagram	27
2.5 Class Diagram	28
3 Conclusione	29

1. Introduzione

Ciò che si vuole realizzare è un videogioco in 2D semplice ma d'impatto.

Il suo nome è Flappy bird e sarà realizzato da un gruppo di due persone che utilizzano la metodologia SCRUM.

Lo scopo del gioco è quello di fare punteggio fluttuando orizzontalmente sulla mappa evitando di scontrarsi con gli ostacoli presenti all'interno di essa o di cadere al suolo; ciò causerebbe la fine della partita e la visualizzazione del punteggio così ottenuto, che sarà proporzionale alla lunghezza della tratta percorsa dal giocatore.

L'avanzamento del cursore, ovvero del giocatore, lungo la mappa, avverrà in maniera automatica; non sarà possibile fermarsi o mettere in pausa il gioco se non a causa del termine della partita.

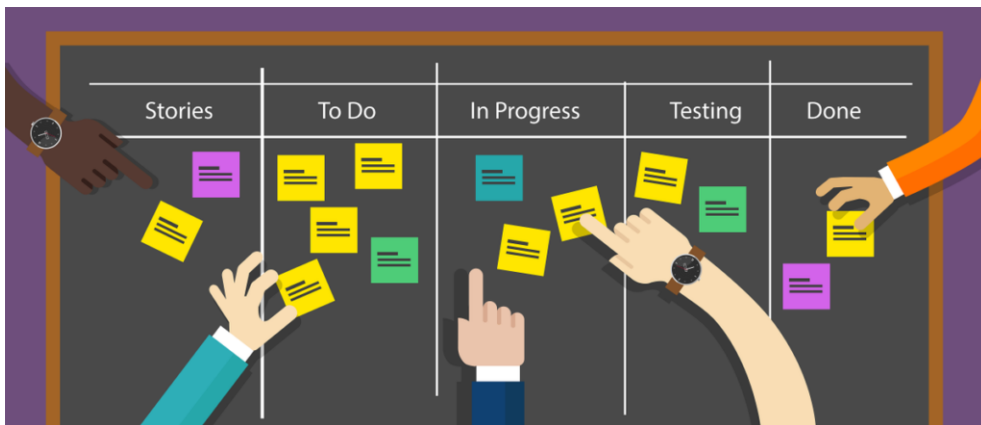
L'utente per evitare gli ostacoli lungo il cammino deve utilizzare il tasto sinistro del mouse, inoltre con lo stesso cercherà di non cadere al suolo.

La schermata di gioco iniziale presenta un menù dal quale è possibile iniziare una partita ed accedere al tutorial.

2. Processi Software

La creazione del prodotto software avviene grazie all'esistenza di attività contenute ed organizzate da diversi modelli di processo, tra cui il modello a cascata, l'ingegneria del software basata su componenti e quello da noi scelto: lo *sviluppo evolutivo*, in particolare agile con metodologia SCRUM.

I requisiti sono sviluppati in modo incrementale secondo le priorità dell'utente che individua i requisiti poiché è parte integrante del team.



2.1 Cos'è SCRUM?

Scrum è un framework agile per lo sviluppo del software: è iterativo e incrementale, ed è stato messo a punto per gestire progetti e prodotti software. Un buon punto di partenza per comprendere il funzionamento del framework Scrum è iniziare col vedere come si procede verso la realizzazione del prodotto finito e delle parti che lo compongono:

- Stakeholder;

l'idea di prodotto nasce analizzando le specifiche degli stakeholders ovvero tutte quelle persone interessate alla realizzazione del prodotto stesso (clienti, utilizzatori finali, venditori).

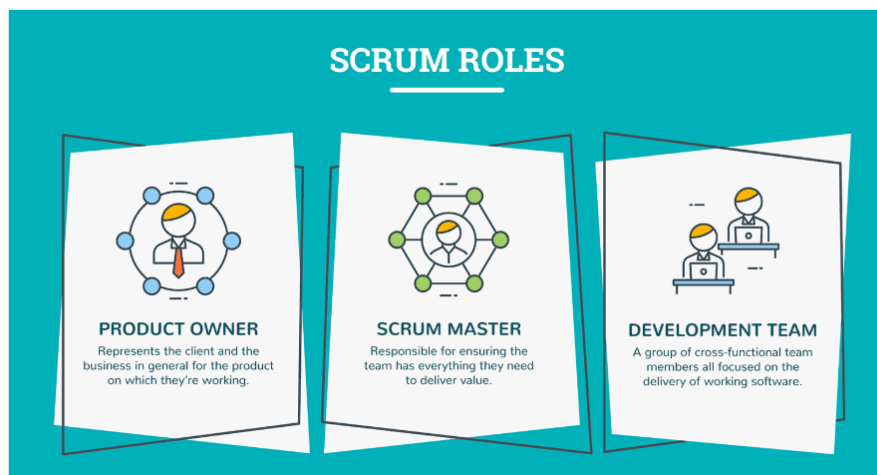
- User story;
sono delle brevi descrizioni o caratteristiche dette e scritte dagli stakeholders che le richiedono. Queste specifiche saranno poi implementate dal team di sviluppo.
- Product Backlog;
tutte le storie create vengono raggruppate nel Product Backlog, una struttura a pila i cui elementi che stanno in alto sono quelli che verranno prelevati e implementati per primi, e sono quindi caratterizzati dall'avere una dimensione opportuna, in modo che il team possa implementarli prima della fine dello sprint. In basso nella pila invece sono presenti elementi che sono più grossi e per i quali quindi è necessario un ulteriore processo di affinamento prima che siano messi in lavorazione.

Membri attivi

- Product Owner;
rappresenta gli Stakeholder ed è la voce del cliente, si assicura che il prodotto dia valore al business, scrive le user story, assegna loro la priorità e le aggiunge al product backlog.
- Scrum Master;
è responsabile della rimozione degli ostacoli che limitano la capacità del team di raggiungere l'obiettivo dello Sprint e gli incrementi previsti. E' una figura manageriale, ma non è un team leader, ma colui che facilita una corretta esecuzione del processo. Detiene l'autorità relativa all'applicazione delle norme, presiede le riunioni importanti e pone sfide alla squadra per migliorarla. Tiene concentrato il Team e verifica i compiti.

- Team;

il team di sviluppo è responsabile della consegna del prodotto cioè svolge il lavoro effettivo: analisi, progettazione, sviluppo, comunicazione tecnica, documentazione ecc... In ultima analisi è il team stesso che deve portare a termine gli sprint.



Eventi SCRUM

1. Backlog refinement;

il product owner incontra il team per discutere delle storie presenti nel Product Backlog e per assegnargli le relative priorità aiutandolo a valutarne costi e rischi. Il product owner deciderà così quali storie faranno parte del release backlog. Questa operazione di raffinamento viene chiamata anche grooming.

2. Sprint planning;

all'inizio di ogni sprint si tiene uno sprint planning meeting dov'è presente anche il product owner e lo scrum master e nel quale viene pianificato il lavoro da svolgere durante lo sprint. Il product owner si presenta con delle storie e il team dovrà scegliere quale implementare durante lo sprint odierno suddividendo ognuna di esse in task.

3. Sprint;

è l'unità base dello sviluppo scrum, dura generalmente da uno a quattro settimane. Durante questo lasso di tempo il team sviluppa le user stories che si è impegnato a consegnare alla fine dello sprint(daily sprint). Il prodotto artefatto finale viene definito incremento.

4. Daily SCRUM;

riunione giornaliera di breve durata nella quale il team è portato a rispondere a tre semplici domande:

Cosa ho fatto ieri?

Cosa farò oggi?

Quali sono gli impedimenti che hanno ostacolato ieri il mio lavoro?

Analizzando queste domande la squadra comprende il lavoro svolto e quanto ne rimane portandola ad acquisire maggiore affinità.

5. Release burndown chart;

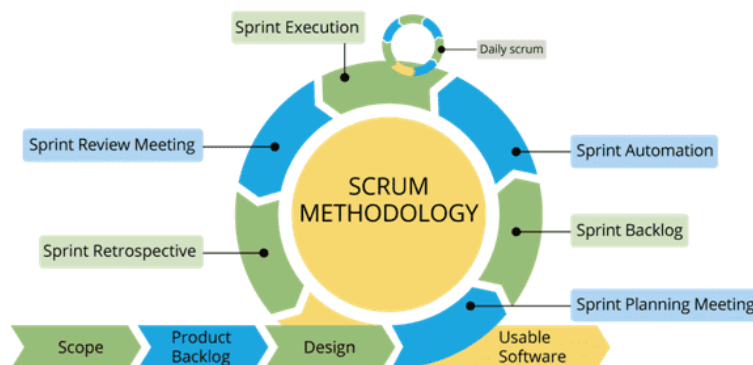
grafico che serve a tenere sotto controllo la velocità del team.

6. Sprint review;

Incremento di prodotto potenzialmente rilasciabile. Si tiene un incontro fra il team, gli stakeholder, il product owner e lo scrum master nel quale vengono mostrati ai clienti gli incrementi di prodotto realizzati.

7. Sprint retrospective;

riunione fra team e scrum master focalizzata sull'analisi del team e del suo lavoro.



2.2 Specifiche dei Requisiti del Software

Questo processo è fondamentale per capire e definire quali servizi sono richiesti dal sistema, e per identificare i vincoli all'operatività e allo sviluppo. Si valuta se le necessità degli stakeholders possono essere soddisfatte con le tecnologie hardware e software correnti.

Analisi dei requisiti

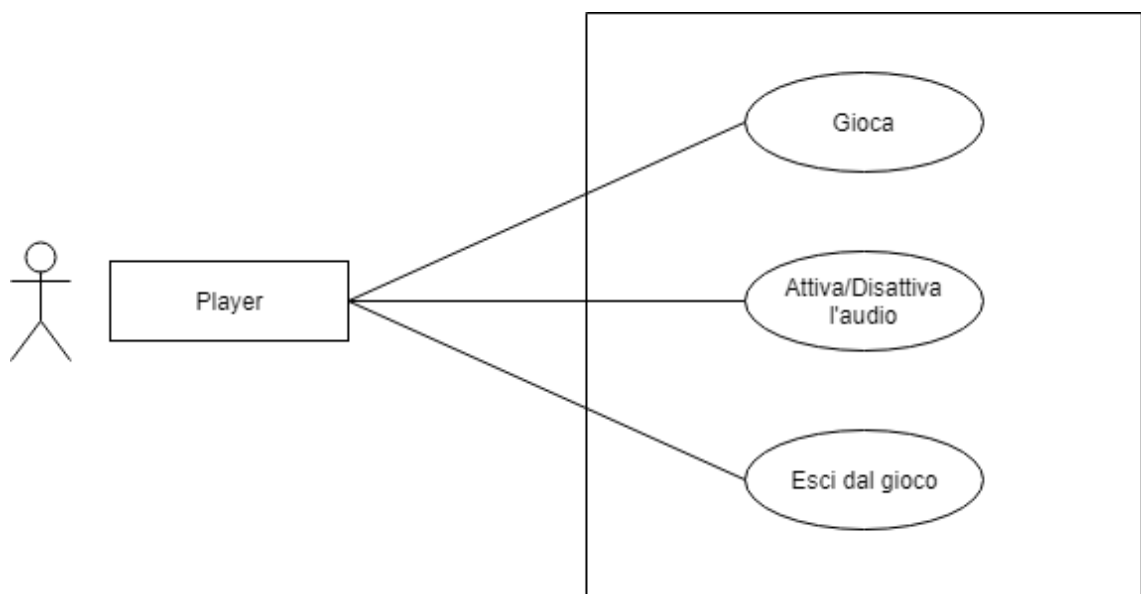
Nei metodi agili come l'extreme programming, i requisiti sono sviluppati in modo incrementale secondo le priorità dell'utente che individua i requisiti, poiché parte del Team.

Questi saranno la descrizione dei servizi forniti e dei suoi vincoli operativi. Si hanno due livelli di descrizione dei requisiti:

- **REQUISITI UTENTE**, descrivono i requisiti funzionali e non funzionali in modo comprensibile per gli utenti del sistema, senza un'approfondita conoscenza tecnica del sistema:
 - *Tutorial*,
il gioco deve fornire un tutorial per spiegare il funzionamento del gioco all'utente che ha appena iniziato a giocare;
 - *Disattivazione Audio*,
nell'interfaccia principale l'utente dovrebbe avere la possibilità di disattivare o attivare l'audio del gioco a suo piacimento;
 - *Punteggio*,
il giocatore dovrebbe visualizzare alla fine della sua partita il punteggio effettuato;
 - *Medaglie*,
alla fine della partita insieme al punteggio ottenuto nella partita corrente, l'utente dovrebbe visualizzare il suo record ottenuto;
 - *Difficoltà*,
il giocatore dovrebbe subire un aumento di difficoltà con l'aumentare del punteggio così da rendere più entusiasmante il gioco;

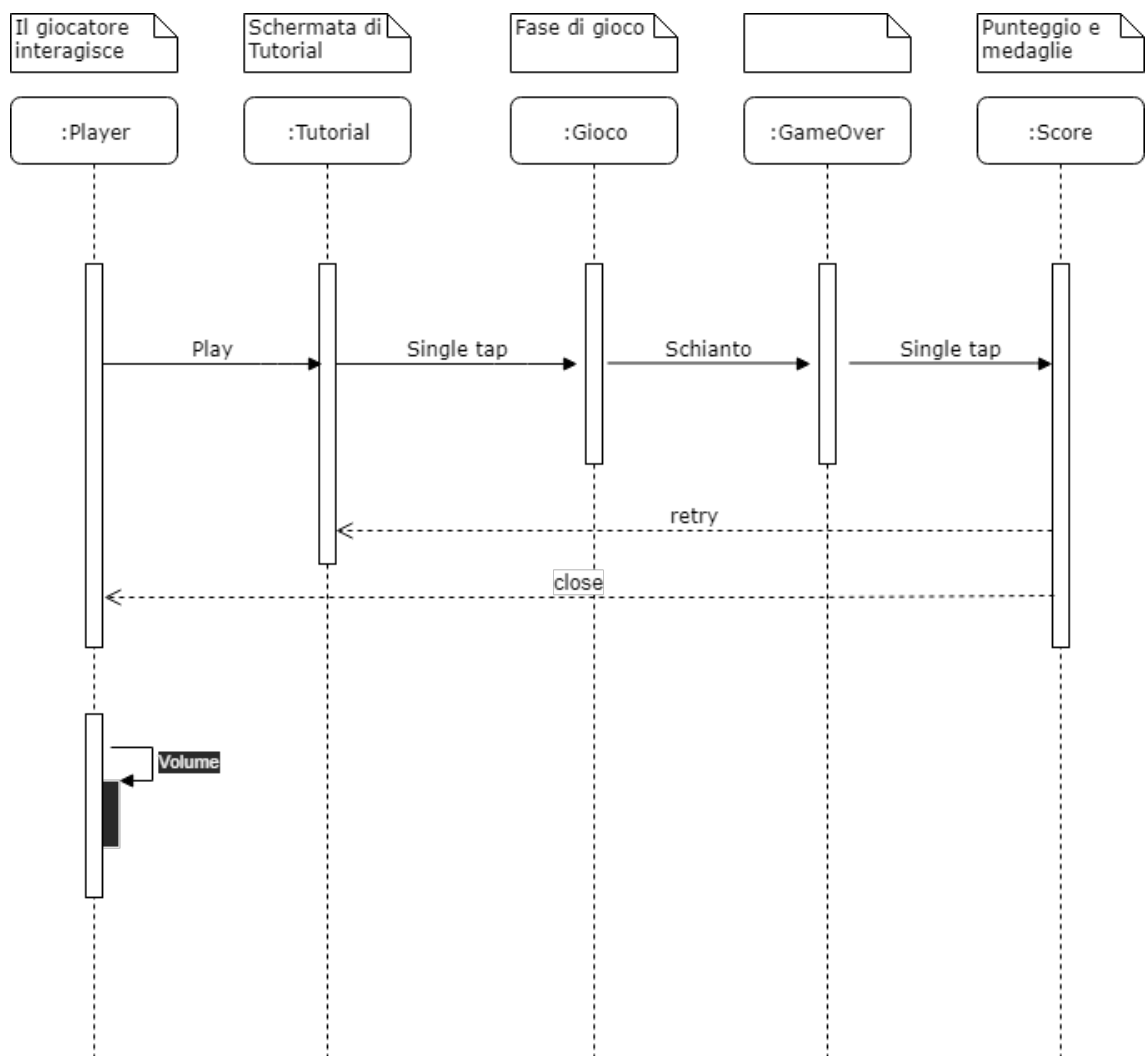
Use Case Diagram

Sono diagrammi dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Sono impiegati soprattutto nel contesto della Use Case View (Vista dei Casi D'uso).



Sequence Diagram

E' un tipo di Iteraction Diagram proposto da UML(Unified Modeling Language) e rappresenta i requisiti del Software dal punto di vista dell'utente, evidenzia il flusso temporale di elaborazione.



- **REQUISITI DI SISTEMA**, definiscono le funzioni, i servizi e i vincoli operativi del sistema in modo dettagliato, si dividono in:
 - **requisiti funzionali**, descrivono quello che il sistema dovrebbe fare:
 - * quando si accederà al gioco si potrà avviare la partita o attivare/disattivare il suono;
 - * iniziata la partita si avvierà il Tutoria esplicativo dei comandi di gioco;
 - * quando ci si troverà nella situazione di GameOver apparirà il punteggio ottenuto;
 - * insieme al punteggio ottenuto dovrà esserci il punteggio migliore effettuato;
 - * si ritornerà al menù principale con la possibilità di giocare nuovamente;
 - **requisiti non funzionali**, sono requisiti che non riguardano direttamente le specifiche funzioni fornite dal sistema, ma:
 - * **requisiti del prodotto**, specificano il comportamento del prodotto:
 - l'interfaccia utente deve essere realizzata con una semplice pagina in HTML5;
 - richiede solo memoria cache sul browser da cui è avviato;
 - deve essere adatto a tutti i tipi di architetture;
 - deve essere responsive;
 - deve richiedere almeno un motore grafico 2D;
 - * **requisiti organizzativi**, politiche e procedure di organizzazione del cliente e dello sviluppatore:
 - il linguaggio di programmazione da utilizzare deve essere JavaScript;
 - il metodo utilizzato deve essere SCRUM;
 - la consegna deve avvenire il 15 Gennaio 2019 con documentazione completa in \LaTeX ;
 - * **requisiti esterni**:
 - deve essere SinglePlayer;
 - freeware;
 - non richiedi requisiti di sicurezza;
 - non richiedi requisiti di riservatezza;

- **requisiti di dominio**, derivano dal dominio dell'applicazione del sistema, ne riflettono i limiti:
 - * deve essere richiesta una connessione Internet;
 - * deve essere richiesto un Browser;

2.3 Sviluppo del Software

Spesso quando si adottano i metodi di sviluppo agile, gli output del processo di progettazione non sono documenti di specifica separati, ma sono rappresentati dal codice del programma. Proprio per questo motivo si è deciso di utilizzare dei metodi strutturati, che si basano sulla produzione di modelli grafici UML (Unified Modeling Language). E' importante che gli stadi iniziali e finali della progettazione si intreccino e che tutto sia rappresentato in maniera estesa e completa. Gli strumenti CASE (Computer-Aided Software Engineering) possono essere utilizzati per generare uno scheletro del programma a partire dalla progettazione.

Tecnologie Utilizzate

- Azure DevOps
Fra i molteplici software proposti dal mercato abbiamo scelto di utilizzare Azure DevOps come ambiente di sviluppo SCRUM. La scelta è ricaduta su quest'ultimo perché è un ambiente gratuito di semplice utilizzo che ci ha permesso di utilizzare una dashboard come product backlog facilitando la creazione delle user stories e la loro suddivisione in task. E' possibile visualizzare chi ha creato le stories ed assegnarle ai vari componenti del team. Il framework ci ha permesso inoltre di collegare la repository di Github e di utilizzare i comandi di git tramite l'interfaccia di Azure stesso.
- Construct 2
Per sviluppare il videogioco abbiamo utilizzato Construct 2. Questo IDE è progettato per creare giochi in 2D, e viene fornito con un sacco di risorse che rendono questo compito molto semplice, come, ad esempio, un motore fisico che fa sì che gli elementi in gioco si muovano in base alla legge di gravità. Sono anche presenti bit grafici e sonori, come sprite, sfondi ed effetti sonori. Inoltre, è molto semplice aggiungere qualsiasi file multimediale dal di fuori dell'applicazione.

- Git

Git è un software di controllo versione utilizzabile da interfaccia a riga di comando. Come git repository abbiamo scelto Github ed è possibile visualizzarne il contenuto accedendo al seguente link:

<https://github.com/Redherz/FlappyBird>

- Draw.io

E' una applicazione di diagrammi completamente gratuita e abilitata per Google Drive(TM) che ci ha permesso di disegnare: lo Use Case Diagram e il Sequence Diagram

- L^AT_EX


















La relazione è stata scritta in LateX, linguaggio di markup per la preparazione di testi, basato sul programma di composizione tipografica TEX;

Progettazione dell'architettura

Questa fase consiste nell'identificazione dei sottoinsiemi che formano il software. Per prima cosa il Product Owner si interfaccia con gli Stakeholders con l'idea di progetto. L'unione di tutte le specifiche raccolte si rielaborano come storie da raggruppare nel Product Backlog.

Backlog Refinement

Il Product Owner decide le priorità relative alle storie in ordine dal basso verso l'alto:

 	Order	Work Item Type	Title	State
	1	User Story	>  BestScore	● New
	2	User Story	>  CorrezioneTubi	● New
	3	User Story	>  Audio	● New
	4	User Story	>  Effect Player	● New
	5	User Story	>  Mode Touch	● New
	6	User Story	>  Riprovare	● New
	7	User Story	>  Collisioni	● New
	8	User Story	>  Score	● New
	9	User Story	>  Event Tubi	● New
	10	User Story	>  Event Player	● New
	11	User Story	>  First Event Sheets	● New
	12	User Story	>  Creazione Layout: Gioco	● New
	13	User Story	>  Creazione Layout: Menù Principale	● New
	14	User Story	>  Creazione Assets	... ● New

Progettazione

La progettazione nelle metodologie SCRUM avviene mediante degli Sprint di durata prestabilita in base al carico di lavoro che il Team può svolgere.

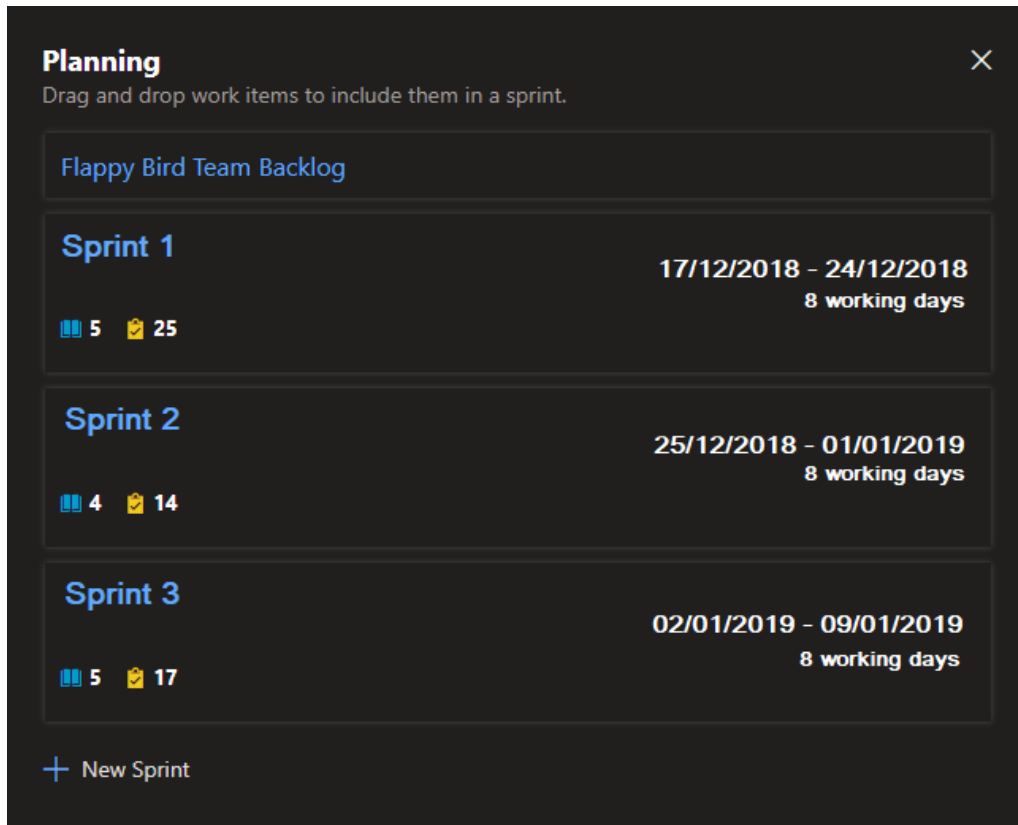
Sprint Planning

Nella prima fase dello Sprint Planning vengono suddivise le storie in task e in base al peso si stabiliscono i gruppi di storie per formare gli sprint.

+	Order	Work Item Type	Title	State	Story ...
+	1	User Story	BestScore	...	New
		Task	Aggiunta BestScore nel GameOverScreen	New	
		Task	Salvataggio in Locale BestScore	New	
	2	User Story	Correzione Tubi	New	
		Task	Sovraccarico Tubi	New	
		Task	Run Tubi e Terreno con GameOver	New	
		Task	Altezza tubi	New	
		Task	Sovrapposizioni layer oggetti	New	
		Task	Rendere Responsive	New	
	3	User Story	Audio	New	
		Task	Aggiungere file audio	New	
		Task	Aggiungere icona Audio	New	
		Task	Associare ogni azione ad un Audio	New	
		Task	Attivare/Disattivare Audio	New	
	4	User Story	Effect Player	New	
		Task	Ondulazione al salto	New	
	5	User Story	Mode Touch	New	
		Task	Implementare funzionalità touch per smartphone o tablet	New	
	6	User Story	Riprovare	New	
		Task	Aggiunta tasto OK	New	
		Task	Aggiunta tasto Menù	New	
		Task	Ancoraggio bottoni al GameOverScreen	New	
		Task	Funzione che cambia il layout	New	
		Task	Reset variabili	New	
	7	User Story	Collisioni	New	
		Task	Ground	New	
		Task	Tubi	New	
		Task	Over sky	New	
		Task	Compare GameOverScreen	New	
	8	User Story	Score	New	
		Task	Pin on GameOverScreen	New	
		Task	Incremento Score	New	
		Task	Inserimento FrameMedaglie	New	
		Task	Funzione gestione medaglie in base al punteggio	New	
	9	User Story	Event Tubi	New	
		Task	Behaviors Tubi	New	
		Task	Spawn Tubi	New	
		Task	Function random distance TuboSu e TuboGiu	New	

10	User Story	▼ 📄 Event Player	● New
	Task	📄 Jump player	● New
	Task	📄 Gravity player	● New
	Task	📄 Sine player	● New
11	User Story	▼ 📄 First Event Sheets	● New
	Task	📄 Switch tra Layouts	● New
	Task	📄 Behaviors Player	● New
	Task	📄 Behaviors Title	● New
	Task	📄 Behaviors Play	● New
	Task	📄 Behaviors Tutorial	● New
	Task	📄 Ground Collision	● New
	Task	📄 Ground speed	● New
12	User Story	▼ 📄 Creazione Layout: Gioco	● New
	Task	📄 Insert Background	● New
	Task	📄 Insert Ground	● New
	Task	📄 Insert Player	● New
	Task	📄 Insert Tutorial	● New
13	User Story	▼ 📄 Creazione Layout: Menù Principale	● New
	Task	📄 Insert Background	● New
	Task	📄 Insert ground	● New
	Task	📄 Insert Player	● New
	Task	📄 Insert Title	● New
14	User Story	▼ 📄 Creazione Assets	● New
	Task	📄 Player	● New
	Task	📄 Background	● New
	Task	📄 Play Button	● New
	Task	📄 Title	● New
	Task	📄 Medals	● New
	Task	📄 ScoreBoard	● New
	Task	📄 Tutorial	● New
	Task	📄 Obstacle	● New

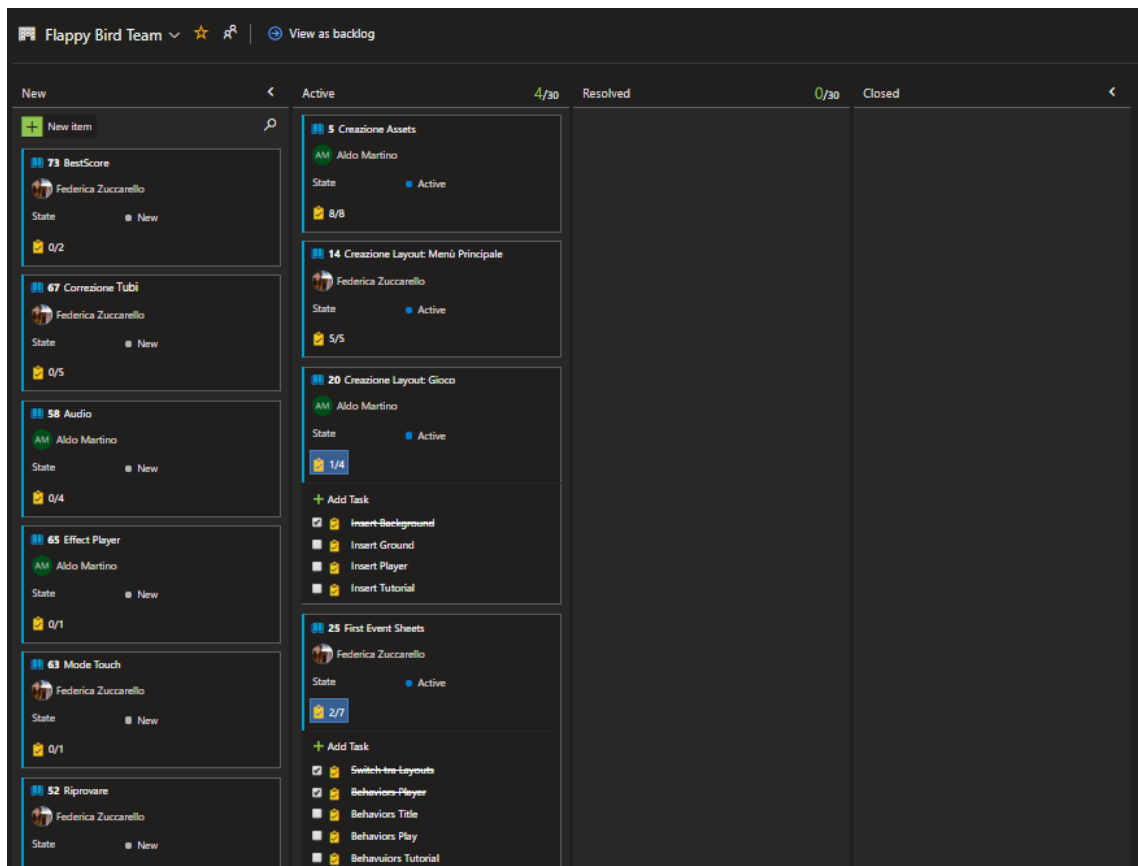
Nella seconda fase si assegnano i task ai membri del team.
Nell'ultima fase si stabiliscono le date di ogni sprint suddividendo il carico di lavoro in tre range come riporta l'immagine sottostante.



- Il primo sprint va dal 17/12/2018 al 24/12/2018 (8 giorni di lavoro).
Le storie da completare sono 5 contenenti in totale 25 task.
- Il secondo sprint va dal 25/12/2019 al 01/01/2019 (8 giorni di lavoro).
Le storie da completare sono 4 contenenti in totale 14 task.
- Il terzo sprint va dal 02/01/2019 al 09/01/2019 (8 giorni di lavoro).
Le storie da completare sono 5 contenenti in totale 17 task.

Sprint 1 - 17/12/2018 24/12/2018

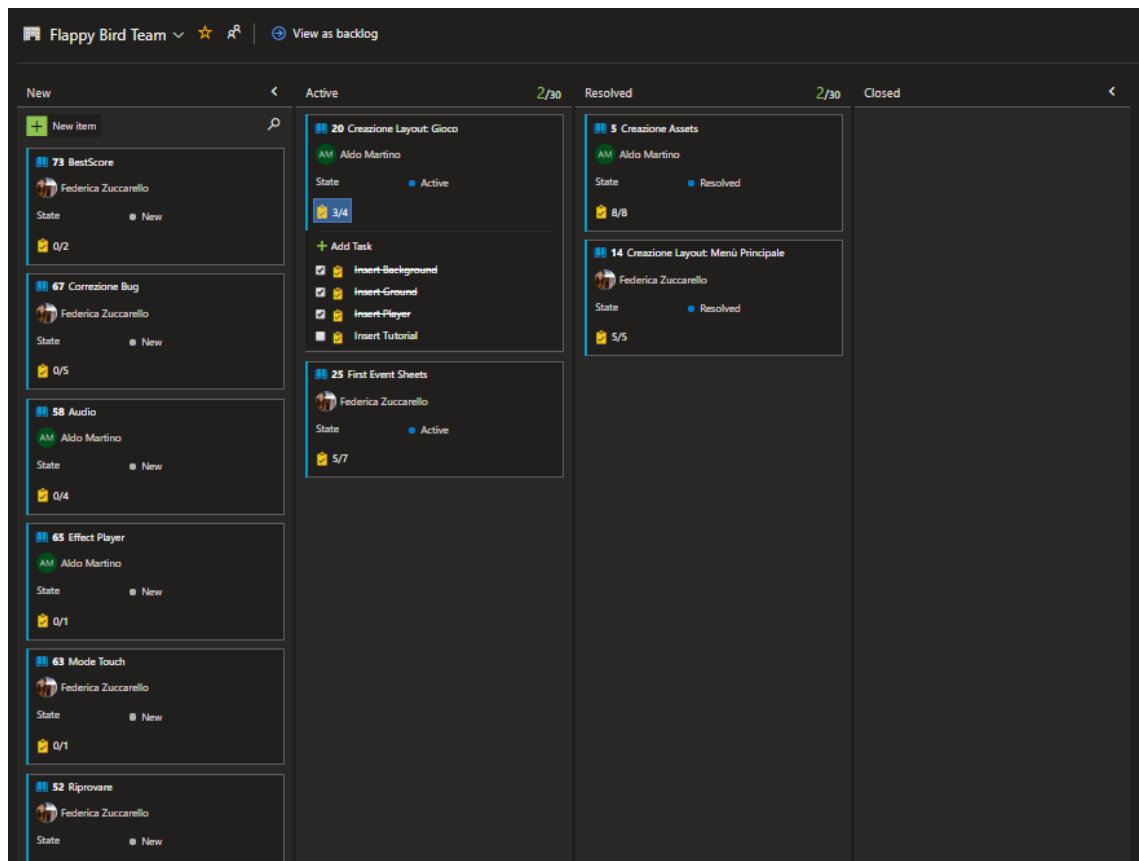
Si può vedere dallo screen del primo sprint che tutte le storie da completare si trovano nello stato *Active*, mentre quelle dei prossimi sprint sono in stato *New*.



Come si evince dalla figura per ogni storia sono presenti i rispettivi task che la compongono; a ognuna è stato assegnato un membro del Team che avrà il compito di portarla al termine.

Sprint 1

Durante il primo Sprint alcune storie non sono ancora completate e si trovano nella sezione *Active*, mentre quelle già completate si trovano nello stato *Resolved*.



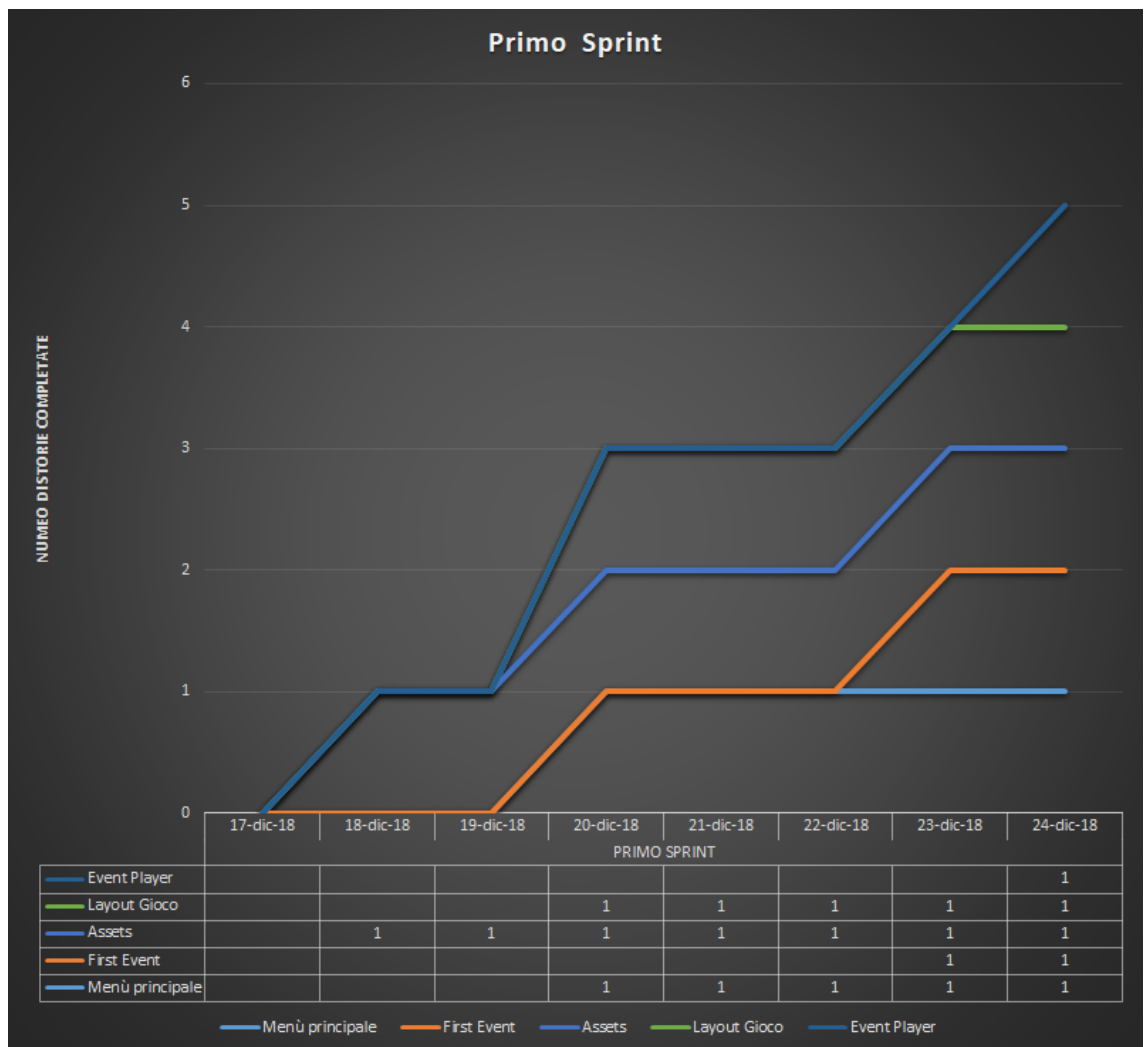
Sprint Retrospective

In questa fase importante di Scrum dopo aver completato il primo Sprint, si mettono insieme tutte le parti realizzate dai vari membri del Team e si analizzano e testano per ricercare eventuali Bug o falle nel videogioco. In questo momento non si è trovato nulla di rilevante, poiché non si era in grado di fare una release da far testare agli stakeholders in quanto le storie erano improntate sulla realizzazione grafica del videogioco.

Release Burdown Chart

Il grafico mostra la velocità del Team nello svolgere le storie. Rappresenta un andamento crescente in maniera costante, infatti si può vedere che:

- le linee colorate rappresentano le Storie;
- l'asse X i giorni dello Sprint;
- l'asse delle Y il numero di storie completate;
- nel grafico sottostante i numeri 1 rappresentano la storie completata.

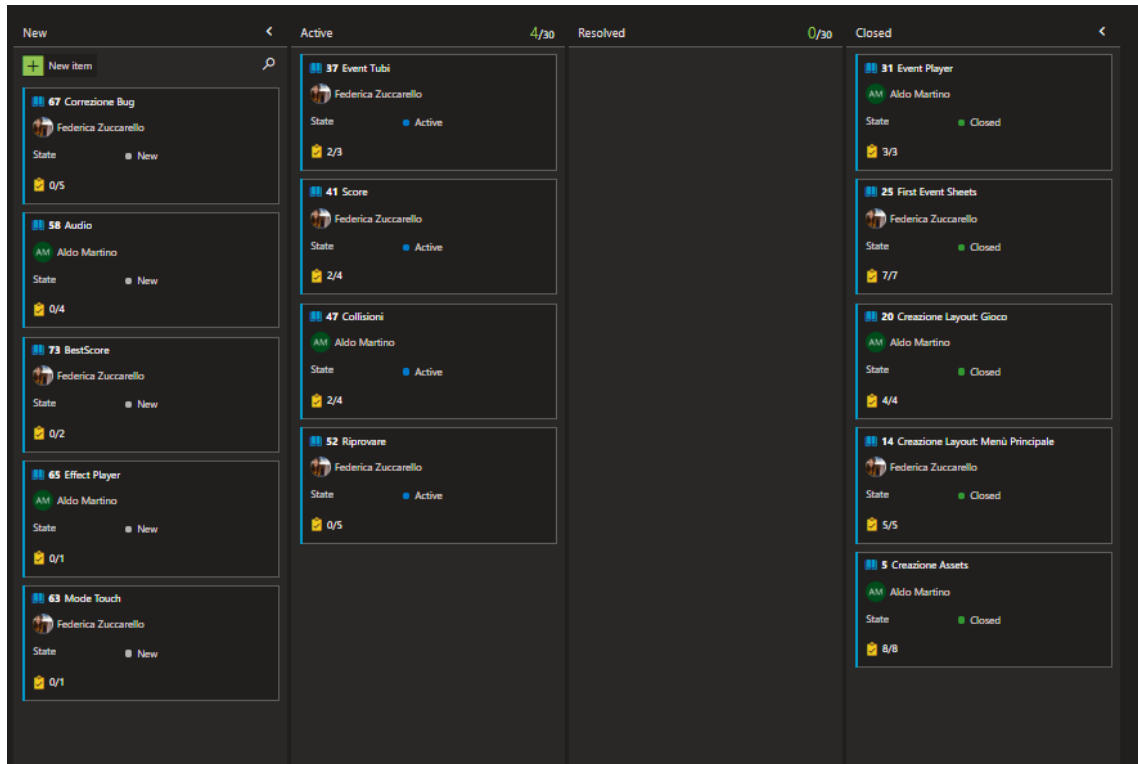


1. Nel primo giorno non si è completata ancora nessuna storia infatti sono ancora a 0;
2. Nel secondo giorno la storia Asserts è stata completata quindi il valore sull'asse Y sale a 1;
3. Il terzo giorno rimane costante poichè non sono state completate storie;
4. Il quarto giorno ne sono state completate 2 quindi il valore del grafico sale a 3;
5. Il quinto e il sesto giorno è costante;
6. Il settimo giorno si completa 1 storia quindi sale a 4;
7. L'ultima storia viene portata al termine l'ottavo giorno così da completare le 5 storie del primo Sprint;

In ultima analisi si può notare che il lavoro è stato portato al termine regolarmente e nel tempo previsto.

Sprint 2 - 25/12/2018 01/01/2019

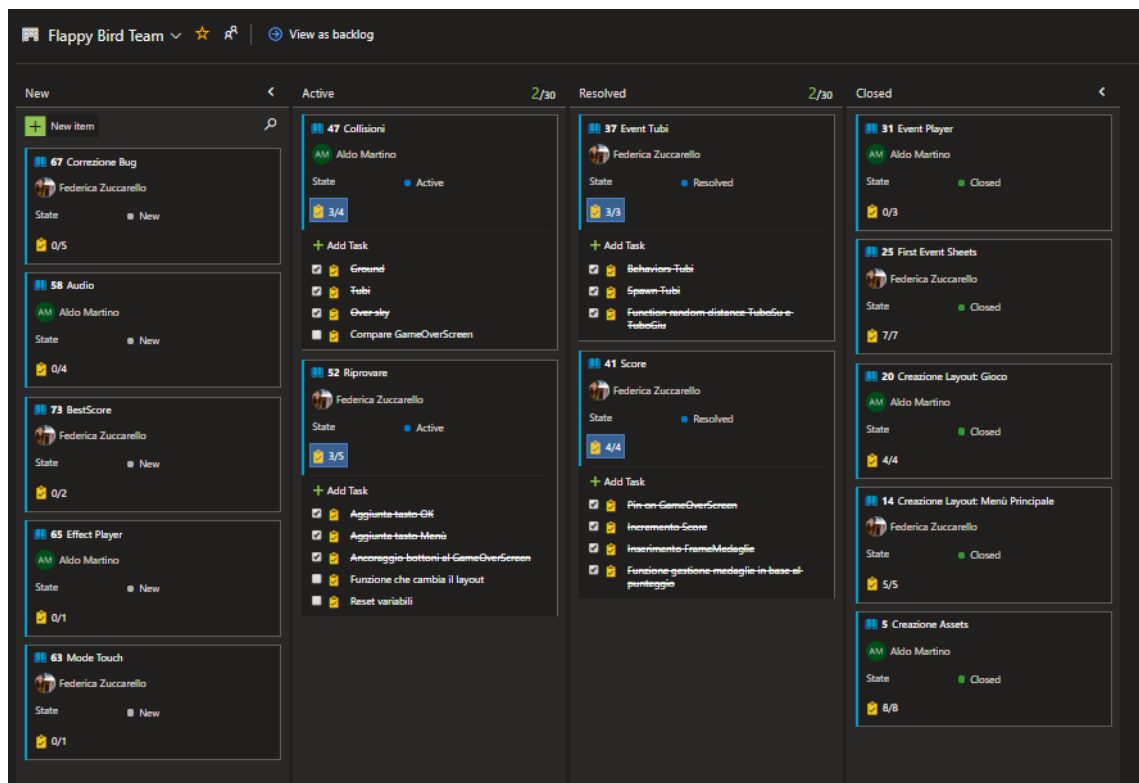
Si può vedere dallo screen del secondo sprint, che tutte le storie da completare si trovano nello stato *Active*, mentre quelle dei prossimi sprint sono in stato *New*.



Per ogni storia si noti che ci saranno i task contrassegnati e a chi spetta il compito di svolgerla, inoltre le storie completate del primo sprint si trovano tutte nello stato *Closed*.

Sprint 2

Si può vedere dallo screen effettuato durante il secondo sprint che tutte le storie ancora incomplete, ovvero che mancano ancora alcuni task da completare, si trovano nello stato *Active*, mentre quelle già completate si trovano nello stato *Resolved*.



Sprint Retrospective

Dopo aver completato il Secondo Sprint e aver messo insieme le parti relative alle storie risolte, si sono effettuati i test ed è stata rilevata la presenza di Bug nel videogioco; quest'ultimi sono stati risolti nel seguente modo:

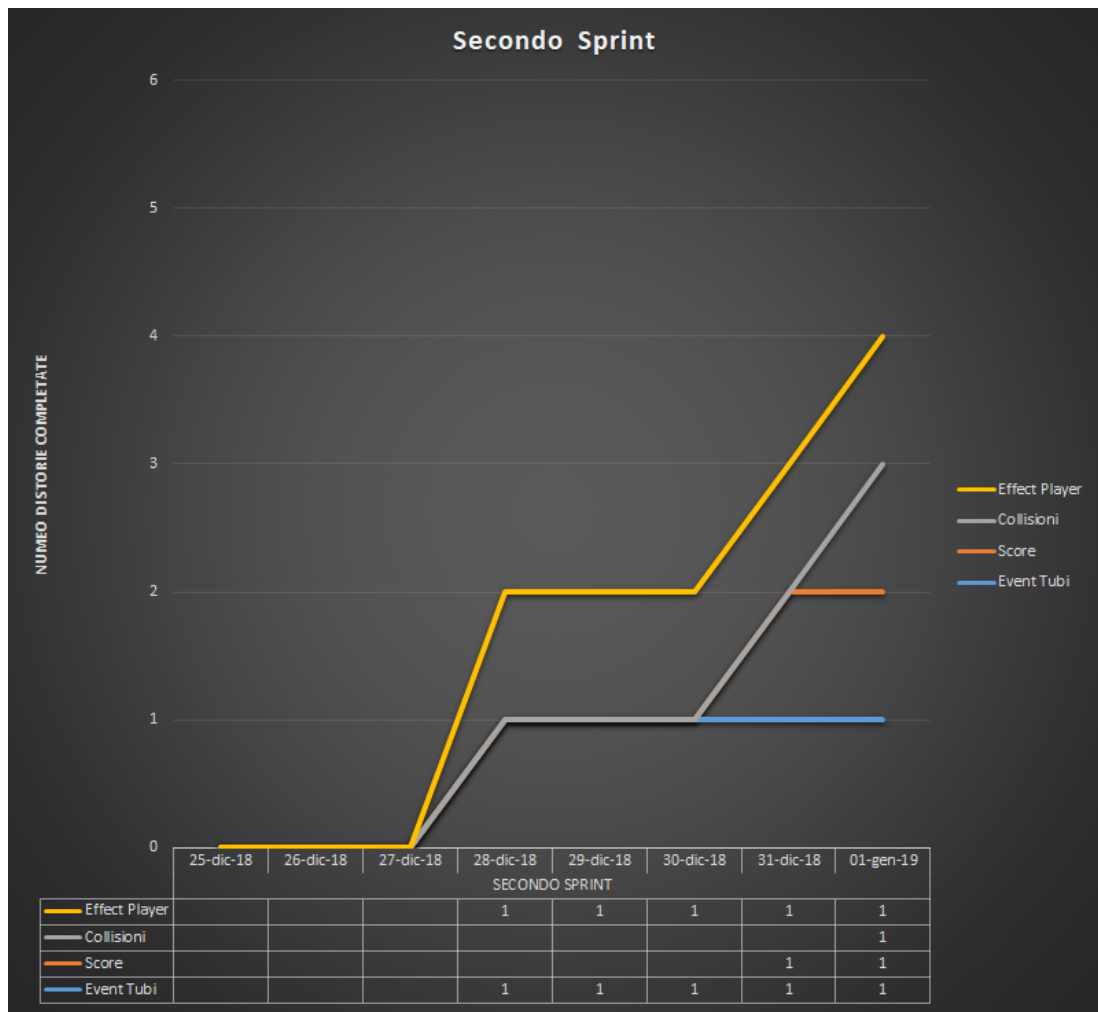
- Mettendo insieme le funzioni relative agli oggetti Terreno, Tubi e GameOver, i primi due continuano a muoversi mentre compare il GameOver.

La soluzione da noi adottata è stata quella di disabilitare la variabile CustomMovement dei Tubi e del Terreno quando è attiva la variabile di GameOver;

- Avendo generato in maniera casuale la comparsa dei Tubi, si nota che alcune volte il valore dei Tubi supera il campo visivo e fluttuano nell'aria.
Si restringe quindi il valore del range sull'asse Y;
- Nel GameOverScreen non compariva il simbolo della medaglia, nonostante fosse stata utilizzata la funzione Pin() tra i due.
Si è risolto inserendo la funzione MoveToTopOfLayer() alle medaglie.

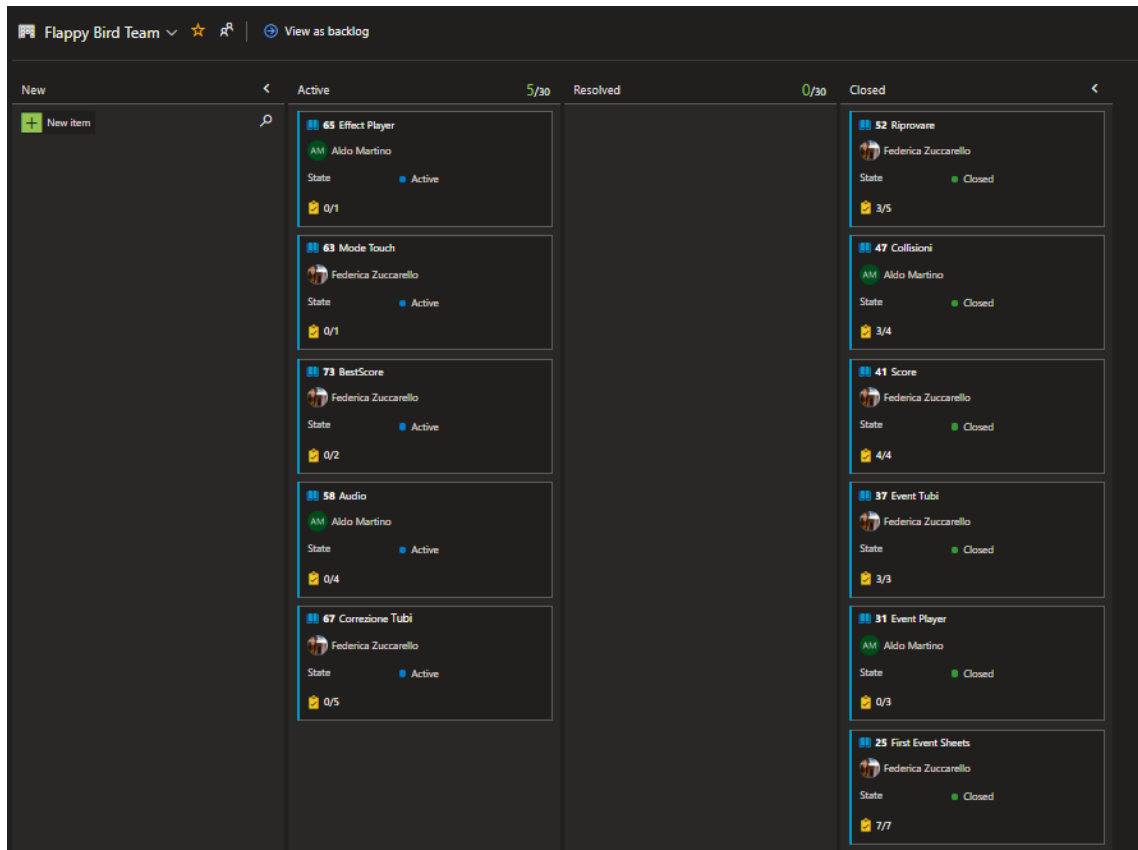
Release Burdown Chart

Il grafico mostra un andamento più lento rispetto a quello del Primo Sprint in quanto, nonostante il numero di task minore, il loro svolgimento è risultato più complesso;



Sprint 3 - 02/01/2019 09/01/2019

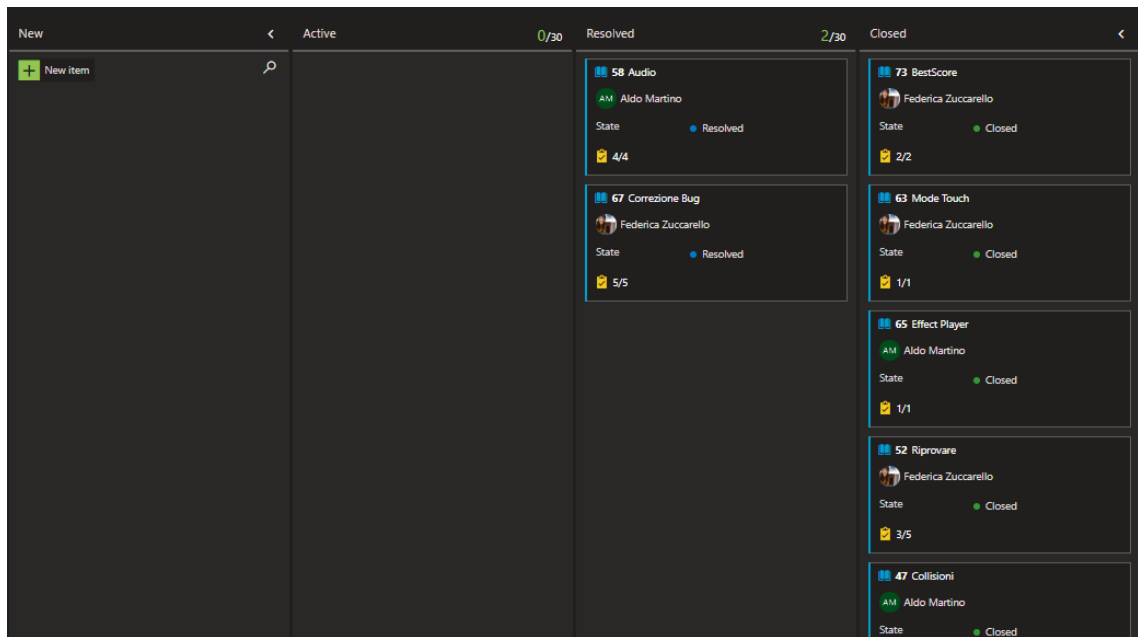
Si può vedere dallo screen del Terzo Sprint, che tutte le storie da completare si trovano nello stato *Active*, mentre nello stato *New* non è presente nessuna storia poiché è l'ultimo Sprint.



Le storie completate del Primo e del Secondo Sprint si trovano tutte nello stato *Closed*.

Sprint 3

Si può vedere dallo screen effettuato durante il Terzo Sprint che tutte le storie già completate si trovano nello stato *Resolved*.



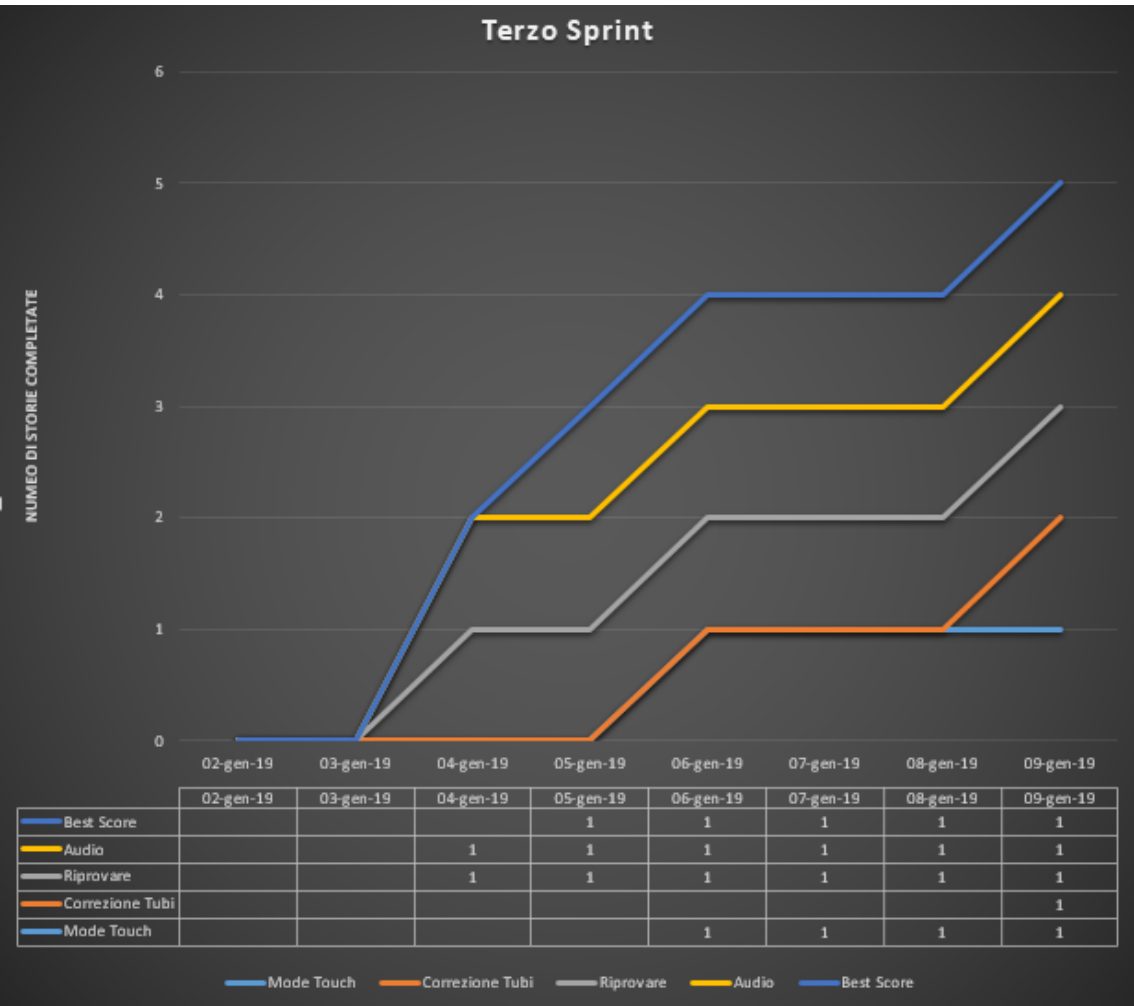
Sprint Retrospective

Dopo aver completato il Terzo Sprint e aver messo insieme le parti relative alle storie risolte, si sono effettuati i test ed è stata rilevata la presenza di un Bug nel videogioco; quest'ultimo è stato risolto nel seguente modo:

- Il Record non viene salvato in locale, quindi si aggiunge una variabile globale `bestScore` salvata nella cache del Browser attraverso un'altra variabile `topScore` che si aggiornerà ogni qualvolta il valore di `bestScore` sia maggiore di `topScore`.

Release Burdown Chart

Il grafico dimostra con il suo andamento crescente che tutti i task sono stati portati al termine in tempo



Operazioni di ottimizzazione

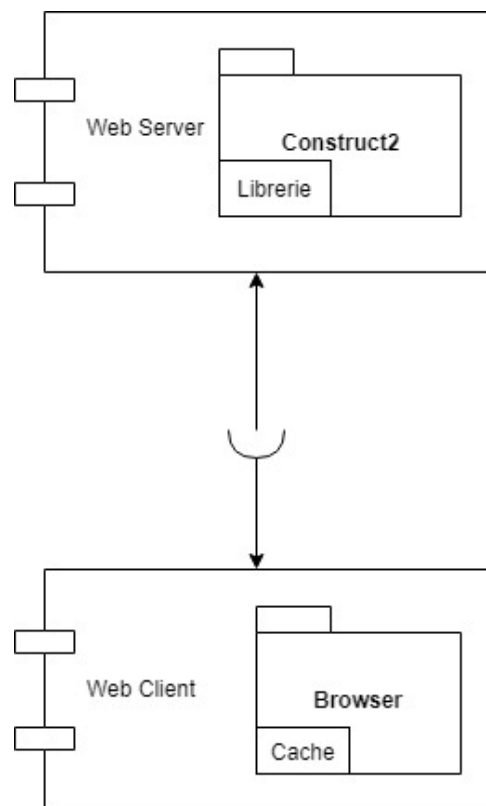
- In modalità debug si nota che il numero dei tubi aumenta andando avanti nel gioco. Può causare appesantimento del gioco soprattutto su mobile poichè ogni tubo creato è un oggetto in più da memorizzare. Ciò è stato risolto distruggendo il tubo stesso quando arriva a fine scena. Abbiamo testato ciò in modalità Debug e visto il miglioramento del 70 per cento.

2.4 Component Diagram

Il Component Diagram è un tipo di diagramma UML che ha lo scopo di descrivere i componenti utilizzati per realizzare le funzionalità del software.

Si usa per visualizzare i componenti fisici di un sistema e descrivere l'organizzazione e le relazioni tra di stessi.

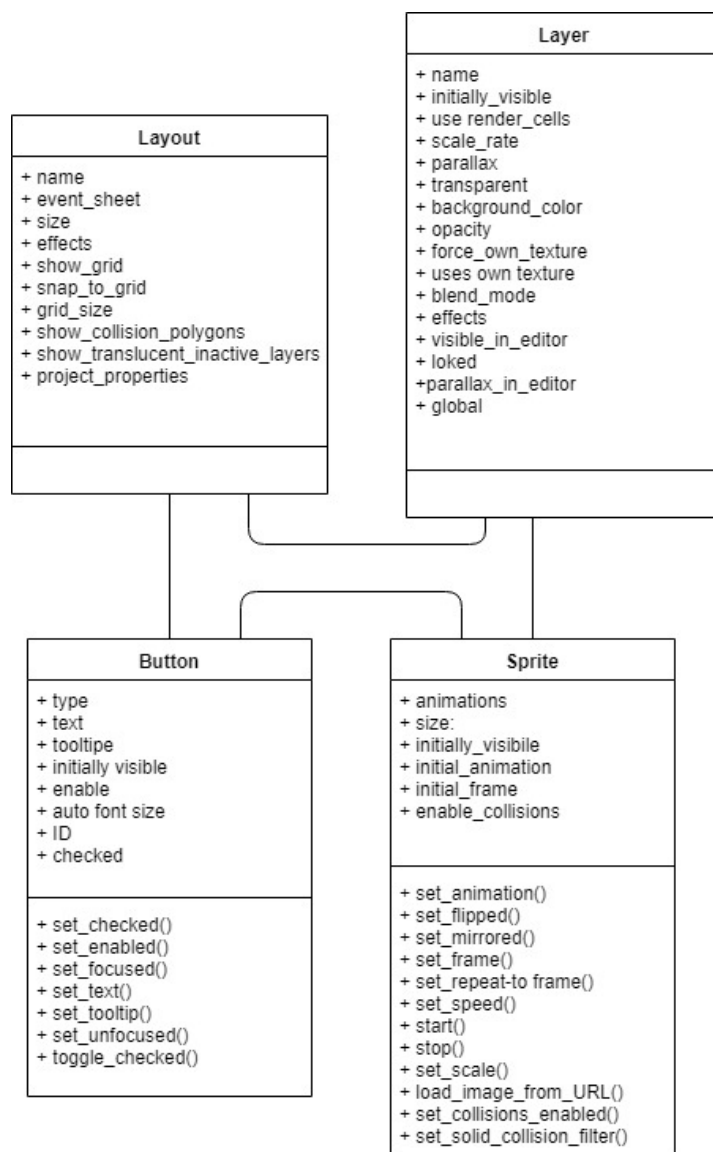
In riferimento al software realizzato, il diagramma è molto semplice: una delle componenti è il web server dove sono contenuti i file relativi all'applicazione, mentre l'altra è il browser usato dall'utente per accedere al gioco.



2.5 Class Diagram

Il Class Diagram è un diagramma UML utilizzato per la costruzione di applicazioni software. E' una rappresentazione grafica della vista statica del sistema e rappresenta diversi aspetti dell'applicazione. Una raccolta di diagrammi di classe rappresenta l'intero sistema.

Il diagramma delle classi descrive gli attributi e le operazioni di una classe e anche i vincoli imposti al sistema. Sono ampiamente utilizzati nella modellazione di sistemi object oriented perché sono gli unici diagrammi UML che possono essere mappati direttamente con linguaggi orientati agli oggetti.



3 Conclusione

Di seguito è riportato il link del videogioco disponibile online:

<http://zuccarellofederica.altervista.org/FlappyBird/index.html>

