Progetto di

MODELS AND ALGORITHMS FOR DATA VISUALIZATION

CORSO DI LAUREA IN INGEGNERIA INFORMATICA E ROBOTICA – A.A. 2024-2025

Dipartimento di Ingegneria

DOCENTE

Prof. Giuseppe Liotta

BOARD GAMES RECOMMENDATION

STUDENTI

Federico Ombelli

Valentina Milighetti

GitHub: https://github.com/FedeOmb/boardgamesInfoVis

Sommario

1. Introduzione

Questo lavoro ha l'obiettivo di progettare un sistema di visualizzazione per esplorare la top 100 dei giochi da tavolo del 2023 secondo il sito web BoardGameGeek.com.

BoardGameGeek è una piattaforma molto diffusa tra gli appassionati di giochi da tavolo, raccoglie un grande numero di informazioni e dati sui giochi ed è basata sui contributi degli utenti che possono aggiungere nuovi giochi e recensirli. Sulla base dei feedback degli utenti il sito realizza una classifica dei migliori giochi.

L'interfaccia sviluppata consente di analizzare e confrontare i giochi sulla base di diverse caratteristiche, con un focus particolare sull'individuazione di relazioni tra giochi simili e sulle preferenze condivise dagli utenti.

Il sistema è pensato per essere intuitivo e orientato sia ad appassionati di giochi da tavolo che ai neofiti che vogliono scoprire nuovi giochi, anche sulla base di quelli già apprezzati, ma anche a esperti nel campo o organizzatori di eventi che intendono approfondire dei pattern o caratteristiche dei giochi più alti nel ranking.

Nelle sezioni seguenti vengono descritte le fasi del progetto: nel **capitolo 2** viene trattata la *raccolta dei requisiti* che consiste nell'individuazione degli utenti finali, l'estrazione e trasformazione dei dati e la definizione delle operazioni previste per gli utenti. Segue, nel **capitolo 3**, la fase di *design* in cui vengono definiti gli idiomi visuali, le interazioni, le scelte architetturali e tecnologiche e gli algoritmi utilizzati.

Infine, nel **capitolo 4** è descritta la fase di *realizzazione*, in cui vengono descritte le varie funzionalità implementate anche facendo uso di immagini dell'interfaccia.

2. Raccolta dei requisiti

2.1 Utenti finali

Il sistema di visualizzazione è progettato per essere utilizzato da alcune categorie di utenti che rientrano nella definizione di *utenti casuali* e *utenti esperti nel settore*.

Tra i primi sono inquadrati gli appassionati di giochi da tavolo ma anche neofiti del mondo dei giochi da tavolo o organizzatori di eventi. Gli interessi di questi utenti sono ad esempio:

- Trovare nuovi giochi da tavolo raccomandati sulla base di quelli già apprezzati;
- Scoprire i giochi con cui approcciarsi al mondo dei giochi da tavolo, individuandoli ad esempio per semplicità (bassa età consigliata) o tempo di gioco breve;
- Confrontare i giochi in base al numero di giocatori, tempo di gioco e categoria.

Tra gli *utenti esperti* rientrano i produttori di giochi da tavolo (come designer e editori) e rivenditori che sono interessati ad analizzare la classifica per individuare le esigenze di mercato e quindi i giochi che vendono maggiormente.

2.2 Data Modeling

2.2.1 Estrazione dei dati

Il dataset proviene dal *Graph Drawing Contest* del 2023[1] e contiene la top-100 dei giochi da tavolo secondo il sito BoardGameGeek.

Il dataset elenca i 100 giochi come record JSON in un array: ogni gioco è dotato di una serie di attributi quali ID, anno di uscita, posizione nella classifica top 100, età minima o massima dei giocatori.

La voce “recommendations” / “fans_liked” contiene un elenco degli ID degli altri giochi che sono piaciuti ai giocatori di questo, la voce “types” contiene due liste: “categories” con le categorie a cui appartiene il gioco e “mechanics” con le meccaniche che utilizza il gioco.

La voce “credit” contiene la lista dei designer del gioco mentre la voce “type” contiene le tipologie principali del gioco chiamate anche “subdomains” [2]. L’attributo “type” non era originariamente presente nel dataset scaricabile, ma è stato estratto dalle API di BoardGameGeek per una completezza di informazioni, in quanto ritenuto utile per vari task di interazione. Le operazioni di *data extraction* sono state eseguite usando un server *node.js* e sono specificate nel file *data_extraction.js*, maggiori dettagli nel **capitolo 3.3** *Scelte architetturali e tecnologiche*.

Di seguito viene mostrato un esempio di istanza JSON del gioco "*7 Wonders Duel*".

---

1. Il dataset è disponibile sul sito del Graph Drawing Contest 2023 al link mozart.diei.unipg.it/gdcontest/2023/creative/
2. In BoardGameGeek i “subdomains” o “types” sono delle macrocategorie in cui il sito suddivide i giochi, sono in totale 8. Per maggiori dettagli: boardgamegeek.com/wiki/page/Subdomain

```
{
    "id": 173346,
    "title": "7 Wonders Duel",
    "year": 2015,
    "rank": 19,
    "minplayers": 2,
    "maxplayers": 2,
    "minplaytime": 30,
    "maxplaytime": 30,
    "minage": 10,
    "rating": {
        "rating": 8.0984,
        "num_of_reviews": 80420
    },
    "recommendations": {
        "fans_liked": [
            148228,
            70919,
            233867,
            84876,
            ...
        ]
    },
    "types": {
        "categories": [
            {
                "id": 1050,
                "name": "Ancient"
            },
            {
                "id": 1002,
                "name": "Card Game"
            }
        ],
        "mechanics": [
            {
                "id": 3001,
                "name": "Layering"
            },

            ...
        ]
    },
    "credit": {
        "designer": [
            {
                "id": 9714,
                "name": "Antoine Bauza"
            },
            {
                "id": 1727,
                "name": "Bruno Cathala"
            }
      ]
    },
    "type": [
             "Strategy Games"
    ]
}
```

2.2.2 Pulizia dei dati

Il dataset ha un formato ben strutturato tuttavia, dall'analisi dei dati, è emerso che gli id presenti nei "*fans_liked*" di un gioco non erano tutti relativi a giochi della top 100. Di conseguenza, è stato necessario rimuovere gli id dei giochi non presenti in top 100.

2.2.3 Trasformazione dei dati

Ai fini realizzativi, che prevedono l'utilizzo della libreria JavaScript D3.js, è stata utile un'operazione di *data mapping* che ha comportato la modifica della struttura del file JSON includendo i dati in due proprietà: "*nodes*" e "*links*", che contengono rispettivamente i nodi e gli archi del grafo.

In particolare, invece che avere una lista di campioni, è riportata la lista dei giochi da tavolo e i loro metadati nella proprietà "nodes", mentre la proprietà "links" contiene le coppie di giochi legate da una relazione di gradimento da parte della stessa comunità di giocatori. La proprietà "links" è stata popolata a partire dalla proprietà "fans_liked", contenuta in "recommendations", di ciascun gioco. Dato che la relazione fans_liked implica un ordinamento, è possibile che ci siano dei link che collegano gli stessi giochi ma con direzione opposta.

La struttura risultante è del tipo:

```
{
    "nodes": [
       {
          "id":174430,
          ...
       },
       ....
    ],
    "links": [
       {
          "source":174430,
          "target":291457
       },
       ...
    ]
 }
```

Inoltre, per semplificare le operazioni sul file JSON, sono state riorganizzate alcune proprietà in particolare:

- il campo "rating", che contiene a sua volta un oggetto con due campi "rating" e "num_of_reviews", è stato semplificato realizzando due campi distinti nel nodo;
- il campo "types", che contiene un oggetto con due campi "mechanics" e "categories" ciascuno contenente a sua volta un array, è stato semplificato creando due campi distinti nel nodo;
- infine il campo "credits", che contiene un array "designers", è stato semplificato eliminando il campo credits e includendo l'array nel nodo.

Per semplificare eventuali operazioni di esplorazione delle collaborazioni tra i designer o combinazioni di categorie/meccaniche in comune, si è deciso di creare per ognuno di questi attributi un grafo in cui mantenere per ogni categoria, meccanica e designer, un nodo con il suo

nome, il suo id e la lista dei giochi associati. È presente un arco tra due nodi se due designer, categorie o meccaniche sono presenti in uno stesso gioco. Ciascuna rete è descritta in un file JSON con struttura analoga a quella del dataset originario.

Ad esempio la struttura della rete dei designer è la seguente:

```
{
    "nodes": [
       {
          "id": 69802,
          "name":"Isaac Childres",
          "games": [
             174430,
             291457
          ]
       },
       ....
    ],
    "links": [
       {
          "source":69802,
          "target":32887,
          "weight":1
       },
       ...
    ]
 }
```

2.2.4 Dataset types

Il processo di *data modeling* ha portato alle seguenti strutture dati:

Network: top 100 dei giochi da tavolo

Dataset type: **Network (oriented)**

- *nodi* (items) rappresentano i giochi
- *archi* (links) rappresentano la relazione tra due giochi che piacciono agli stessi utenti
- *frecce* rappresentano la direzione dei link

Gli **attribute types** dei nodi sono i seguenti:

- **Id**: attributo *categorico*, identificativo del gioco su BoardGameGeek
- **Title**: attributo *categorico*, nome del gioco
- **Year**: attributo *ordinale*, anno di pubblicazione
- **Rank**: attributo *ordinale*, posizione nella top 100
- **Min players**: attributo *quantitativo*, minimo numero di giocatori
- **Max players**: attributo *quantitativo*, massimo numero di giocatori
- **Min play time**: attributo *quantitativo*, tempo minimo di gioco
- **Max play time**: attributo *quantitativo*, tempo massimo di gioco
- **Min age**: attributo *quantitativo*, età minima per giocare
- **Rating**: attributo *quantitativo*, voto da 1 a 10 assegnato dagli utenti
- **Num of reviews**: attributo *quantitativo*, numero di votazioni
- **Categories**: lista di attributi *categorici*:
  - **Id**: identificativo della categoria su BoardGameGeek
  - **Name**: nome della categoria
- **Mechanics**: lista di attributi *categorici*:
  - **Id**: identificativo della meccanica su BoardGameGeek
  - **Name**: nome della meccanica di gioco
- **Designer**: lista di attributi *categorici*:
  - **Id**: identificativo del designer su BoardGameGeek
  - **Name**: nome del creatore del gioco
- **Type**: lista di attributi *categorici* rappresentanti il nome del tipo di gioco

I *link* rappresentano la relazione tra due giochi attraverso due campi *source* e *target* di identificativi dei giochi: se è presente un link tra due giochi significa che i fan del primo (source) hanno gradito anche il secondo (target). I link non presentano metadati e quindi sono privi di *attribute types*.

Network: designers

Dataset type: Network

- *nodi* (items) rappresentano i designer
- *archi* (links) rappresentano la relazione tra due designer che hanno collaborato per uno stesso gioco

Gli **attribute types** dei nodi sono i seguenti:

- **Id**: attributo *categorico*, identificativo del designer su BoardGameGeek
- **Name**: attributo *categorico*, nome del designer
- **Games**: lista degli id dei giochi nella top 100 creati dal designer

Gli **attribute types** dei link sono i seguenti:

- **Source:** attributo *categorico*, id del designer della collaborazione
- **Target**: attributo *categorico*, id del designer della collaborazione
- **Weight:** attributo *quantitativo,* numero di giochi della top 100 in cui i designer source e target hanno collaborato

Network: categories

Dataset type: **Network**

- *nodi* (items) rappresentano le categorie
- *archi* (links) rappresentano la combinazione di due categorie che occorre per uno stesso gioco

Gli **attribute types** dei nodi sono i seguenti:

- **Id**: attributo *categorico*, identificativo della categoria su BoardGameGeek
- **Name**: attributo *categorico*, nome della categoria
- **Games**: lista degli id dei giochi nella top 100 che rientrano in quella categoria

Gli **attribute types** dei link sono i seguenti:

- **Source:** attributo *categorico*, id della categoria della combinazione
- **Target**: attributo *categorico,* id della categoria della combinazione
- **Weight:** attributo *quantitativo*, numero di giochi della top 100 che presentano la combinazione di categorie source e target

Network: mechanics

Dataset type: **Network**

- *nodi* (items) rappresentano le meccaniche
- *archi* (links) rappresentano la combinazione di due meccaniche che occorre per uno stesso gioco

Gli **attribute types** dei nodi sono i seguenti:

- **Id**: attributo *categorico*, identificativo della meccanica su BoardGameGeek
- **Name**: attributo *categorico*, nome della meccanica
- **Games**: lista degli id dei giochi nella top 100 che rientrano in quella meccanica

Gli **attribute types** dei link sono i seguenti:

- **Source:** attributo *categorico,* id della meccanica della combinazione
- **Target**: attributo *categorico*, id della meccanica della combinazione
- **Weight:** attributo *quantitativo*, numero di giochi della top 100 che presentano la combinazione di meccaniche source e target

2.3 Task Modeling

Di seguito sono elencati i task definiti e la loro formulazione (traduzione) in coppie <*action*, *target*>.

2.3.1 Elenco dei task

1. Trovare il gioco con nome <*Title*>
2. Identificare i giochi che hanno la stessa tipologia
3. Identificare i giochi che piacciono ai fan di un gioco con nome <*Title*>
4. Per ogni nodo, confrontare le statistiche (numero di giocatori, tempo di gioco, voto, età minima) con quelle dei giochi raccomandati
5. Confrontare le statistiche (numero di giocatori, tempo di gioco, voto, età minima) per ogni categoria/meccanica/designer/anno di uscita
6. Confrontare la posizione dei giochi nella top 100
7. Identificare le meccaniche più diffuse tra i giochi nella top 100
8. Identificare le categorie più diffuse tra i giochi nella top 100
9. Identificare i designer che hanno creato più giochi nella top 100
10. Dato un designer/categoria/meccanica <*Name*>, visualizzare i giochi che sono nella top 100
11. Confrontare gli anni di uscita dei giochi
12. Dato un anno <*year*> visualizzare i giochi rilasciati in quell'anno
13. Determina se il gioco <*Title1*> è correlato al gioco <*Title2*>
14. Dati due designer, visualizzare i giochi in cui hanno collaborato
15. Date due categorie, visualizzare i giochi che rientrano in entrambe
16. Date due meccaniche, visualizzare i giochi che rientrano in entrambe

2.3.1 Traduzione dei task

Per ogni task sono individuate le coppie <*action, target*>

1. Trovare il gioco con nome <*Title*>:
    - **Locate** il nodo <*Title*>
2. Identificare i giochi che hanno la stessa tipologia:
    - **Browse** i nodi con attributo categorico <*type*>
3. Identificare i giochi che piacciono ai fan di un gioco con nome <*Title*>:
    - **Locate** il nodo <*Title*>
    - **Explore** i nodi adiacenti al nodo <*Title*>
4. Per ogni nodo, confrontare le statistiche (numero di giocatori, tempo di gioco, voto, età minima) con quelle dei giochi raccomandati:
    - **Locate** il nodo <*Title*>
    - **Explore** i nodi adiacenti al nodo <*Title*>
    - **Compare** gli attributi categorici e quantitativi del gruppo di nodi
5. Confrontare le statistiche (numero di giocatori, tempo di gioco, voto, età minima) per ogni categoria/meccanica/designer/anno di uscita:
    - **Locate** il nodo <*CatName*> nella rete delle categorie/meccaniche/designer
    - **Lookup** gli <*Id*> dei giochi del nodo <*CatName*>
    - **Locate** i nodi <*Id*> nella rete dei giochi
    - **Compare** gli attributi categorici e quantitativi del gruppo di giochi
6. Confrontare la posizione dei giochi nella top 100:
    - **Compare** l'attributo ordinale dei nodi
7. Identificare le meccaniche più diffuse tra i giochi nella top 100:
    - **Derive** l'attributo quantitativo "numero di giochi" per ogni nodo nella rete delle meccaniche
    - **Compare** l'attributo quantitativo
8. Identificare le categorie più diffuse tra i giochi nella top 100:
    - **Derive** l'attributo quantitativo "numero di giochi" per ogni nodo nella rete delle categorie
    - **Compare** l'attributo quantitativo
9. Identificare i designer che hanno creato più giochi nella top 100:
    - **Derive** l'attributo quantitativo "numero di giochi" per ogni nodo nella rete dei designer
    - **Compare** l'attributo quantitativo
10. Dato un designer/categoria/meccanica <*Name*>, visualizzare i giochi che sono nella top 100:
    - **Browse** i nodi che hanno attributo categorico <*Name*>
11. Confrontare gli anni di uscita dei giochi:
    - **Compare** l'attributo ordinale <*year*>
12. Dato un anno <*year*> visualizzare i giochi rilasciati in quell'anno che sono presenti in top 100:
    - **Browse** i nodi con attributo ordinale <*year*>
13. Determina se il gioco <*Title1*> è correlato al gioco <*Title2*>:
    - **Locate** i nodi <*Title1*> e <*Title2*>

- **Identify** il link tra i nodi <*Title1*> e <*Title2*>

14. Dati due designer, visualizzare il numero dei giochi in cui hanno collaborato:
    - **Locate** i nodi <*Name1*> e <*Name2*>
    - **Identify** il link tra i nodi <*Name1*> e <*Name2*>
    - **Lookup** l'attributo quantitativo del link
15. Date due categorie, visualizzare il numero dei giochi che rientrano in entrambe:
    - **Locate** i nodi <*Name1*> e <*Name2*>
    - **Identify** il link tra i nodi <*Name1*> e <*Name2*>
    - **Lookup** l'attributo quantitativo del link
16. Date due meccaniche, visualizzare il numero di giochi che rientrano in entrambe:
    - **Locate** i nodi <*Name1*> e <*Name2*>
    - **Identify** il link tra i nodi <*Name1*> e <*Name2*>
    - **Lookup** l'attributo quantitativo del link

3. Design

3.1 Design della visualizzazione

3.1.1 *Visualizzazione 1*: Unconstrained Straight-line Layout, Bar Chart e Dumbbell Chart

Unconstrained Straight-line Layout

Per rappresentare la rete delle raccomandazioni della top 100 dei giochi da tavolo si è scelto di utilizzare una rappresentazione di tipo *Node-Link*, in cui ogni nodo rappresenta un gioco e ciascun link rappresenta la relazione di raccomandazione tra due giochi. In particolare, è stato scelto un idioma *uncostrained straight-line layout* perché ritenuto più adatto rispetto a una rappresentazione layered per i task di esplorazione topologica che abbiamo definito, quali l'esplorazione dei nodi adiacenti ad un altro e la visualizzazione di eventuali cluster nella rete. Infatti, il focus della visualizzazione è sul raggruppamento per tipo e sulle relazioni *fans_liked*.

Mark:

- *Glyph* (cerchio suddiviso in settori circolari) per i nodi
- *Straight-lines* per i link
- *Arrows* per la direzione dei link

Channel:

- Nodi:
  - *Posizione*: codifica la correlazione tra i nodi vicini
  - *Dimensione:* rappresenta l'attributo quantitativo "rank" dei nodi
  - *Colore (hue):* rappresenta i tipi del gioco

Semantic:

- Nodi:
  - *Posizione:* la prossimità tra due nodi indica l'appartenenza alla stessa tipologia
  - *Dimensione:* inversamente proporzionale alla posizione dei giochi nella top 100
  - *Colore (hue):* per identificare i tipi del gioco, se è presente un solo tipo il *glyph* è un cerchio con il colore corrispondente, se ne sono presenti più di uno il *glyph* si divide in più settori circolari, ciascuno colorato del colore corrispondente al tipo rappresentato
- Link:
  - La direzione della freccia indica la direzione del link, possono essere presenti anche dei link bidirezionali

Chart addizionali - Bar Chart

Questo idioma permette di confrontare l'età minima, il voto e la distribuzione delle categorie per il gruppo di giochi raccomandati dai fan di un dato gioco, come descritto nel task 4. Di seguito sono elencati i *mark* e *channel* distinti per questi tre grafici, rispettivamente.

Mark:

- *Bars*

Channel:

- *Length:* per l'età minima, il voto o per il numero di giochi da tavolo
- *Spatial region:* per ciascun gioco o per ciascuna istanza dell'attributo categorico Category

Semantic:

- La lunghezza della barra indica l'età minima, il voto per un gioco o il numero di giochi che appartengono alla stessa categoria

Chart addizionali - Dumbbell Chart

Questo idioma permette di confrontare il numero di giocatori minimo e massimo e il tempo di gioco minimo e massimo per il gruppo di giochi raccomandati dai fan di un dato gioco, come descritto nel task 4. Di seguito sono elencati i *mark* e i *channel* distinti per questi due grafici, rispettivamente.

Mark:

- *Dot*
- *Line*

Channel:

- *Position*: rappresenta il tempo di gioco massimo/minimo o il numero di giocatori massimo/minimo del gioco
- *Spatial region:* per ciascun gioco
- *Color (hue):* rappresenta i punti con il valore minimo e il valore massimo
- *Lenght:* rappresenta il range di valori tra il minimo e il massimo

Semantic:

- Per ogni gioco sono specificati i due punti per i valori minimo e massimo: i due punti sono collegati da una linea la cui lunghezza rappresenta il range di valori tra il minimo e il massimo (range di possibili giocatori, range di tempo di gioco)

3.1.2 *Visualizzazione 2*: Bar Chart e Dumbbell Chart

Bar Chart

Mark:

- *Bars*

Channel:

- *Length:* per il numero di giochi da tavolo
- *Spatial region*: per ciascuna istanza dell'attributo categorico Category, Mechanic o Designer

Semantic:

- La lunghezza della barra indica il numero di giochi che appartengono alla stessa categoria, meccanica o che hanno lo stesso designer

Chart addizionali

Analogamente alla **sezione 3.1.1**, è possibile confrontare il voto, l'età minima, il tempo di gioco, il numero di giocatori e la distribuzione delle categorie per un gruppo ristretto di giochi, cioè i giochi appartenenti alla stessa categoria, meccanica o designer. Gli idiomi scelti sono quindi analoghi: **Bar Chart** e **Dumbbell Chart**, ma contestualizzati a questi dati.

3.1.3 *Visualizzazione 3*: Grouped Bar Chart, Bar Chart e Dumbbell Chart

Grouped Bar Chart

Mark:

- *Bars*

Channel:

- *Length:* per il numero di giochi da tavolo
- *Colore (hue):* per il tipo di gioco
- *Spatial region:* per ciascuna istanza dell'attributo ordinale "year" (anno di uscita)
- *Raggruppamento* delle barre dei tipi di giochi per uno specifico anno di uscita

Semantic:

- La lunghezza della barra indica il numero di giochi che sono dello stesso tipo, usciti nello stesso anno
- Il colore identifica il tipo dei giochi
- L'appartenenza delle barre alla regione spaziale di un anno rappresenta il numero di giochi usciti nello stesso anno e che sono dello stesso tipo

Bar Chart

Mark:

- *Bars*

Channel:

- *Length:* per il numero di giochi da tavolo
- *Spatial region:* per ciascuna istanza dell'attributo ordinale "year" (anno di uscita)

Semantic:

- La lunghezza della barra indica il numero di giochi che sono usciti nello stesso anno

Chart addizionali

Analogamente alla **sezione 3.1.1**, è possibile confrontare il voto, l'età minima, il tempo di gioco, il numero di giocatori e la distribuzione delle categorie per l'insieme di giochi usciti in uno stesso anno. Gli idiomi scelti sono quindi analoghi: **Bar Chart** e **Dumbbell Chart**, ma contestualizzati a questi dati.

3.1.4 *Visualizzazione 4*: Circular Layout - Chord Diagram

Viene utilizzato questo idioma per tre visualizzazioni: per rappresentare la rete delle collaborazioni tra designer e per rappresentare le combinazioni di coppie di categorie o meccaniche che sono in comune in un gioco.

Per mantenere una visualizzazione chiara e facile da capire per l'utente, si è scelto di visualizzare un massimo di 10 elementi nel chord diagram, selezionando i nodi della componente connessa più grande presente nelle reti di collaborazione tra designer, categorie e meccaniche e selezionando tra questi i top 10 in base al numero di giochi.

Mark:

- *Circular arc*: per i nodi
- *Curve:* per i link

Channel:

- Nodi:
  - *Colore (hue):* per identificare il nodo
  - *Radial position:* per ciascuna istanza dell'attributo categorico Category, Mechanic o Designer
- Link:
  - *Curve width:* proporzionale al numero di collaborazioni
  - *Curve hue*

Semantic:

- Due nodi sono collegati se è presente una collaborazione tra due designer/categorie/meccaniche, l'arco ha un peso pari al numero di collaborazioni. Per collaborazione

si intende che esiste un gioco che comprende come attributi entrambi i nodi, per cui si individuano collaborazioni tra designer e combinazioni di categorie e meccaniche
- I nodi sono ordinati circolarmente per avere meno incroci possibili
- La tinta codifica ciascuna istanza dell'attributo categorico name
- La tinta dei link è unica per ciascun link e corrisponde alla tinta del nodo source

3.2 Design dell'interazione

Di seguito sono descritte le operazioni previste per l'interazione degli utenti con le visualizzazioni.

Paradigma di interazione

Per le varie visualizzazioni è previsto un paradigma di interazione di tipo "*Overview first, zoom and filter, then details-on-demand*" noto come *Visualization Mantra* di *Ben Shneiderman* secondo il quale è utile iniziare con una overview, una visualizzazione di insieme, per esplorare i dati e selezionare quelli di interesse tramite operazioni di filtraggio o zoom nel caso di una rete. Una volta individuati i dati da analizzare si può procedere a visualizzarli in dettaglio.

I task di interazione previsti sono descritti di seguito insieme ai relativi paradigmi per ciascuna visualizzazione.

Visualizzazione 1: Top 100 Network

- **Exploration:** attraverso il paradigma panning è possibile muoversi con il mouse per mostrare le varie parti della rete.
- **Zooming:** è possibile controllare il livello di zoom della visualizzazione interagendo con il mouse o usando tre tasti appositi (zoom in, zoom out, reset).
- **Abstract Zooming (details-on-demand):** il passaggio del mouse sopra a un link o a un nodo mostra una tooltip contenente più informazioni, inoltre, il click su un nodo mostra un pannello riepilogativo del gioco che contiene i titoli dei giochi della stessa fan base e i chart aggiuntivi selezionabili con dei tasti.
- **Selection:** il click su un nodo lo evidenzia, colorandolo di nero. Viene utilizzato un altro paradigma, cioè generalized selection, quando viene costruito il convex hull dei nodi di un tipo specificato. Inoltre, il passaggio del mouse sopra a un tipo nella legenda evidenzia i nodi di quel tipo.
- **Connecting:** il click o il passaggio del mouse evidenzia il nodo e le connessioni con i suoi vicini (paradigma linking).
- **Filtering:** i dati dei chart addizionali sono quelli del gruppo di nodi selezionato (nodo cliccato e suoi vicini).
  Inoltre, una barra di ricerca permette di digitare il nome di un gioco così da individuarlo, evidenziarlo e mostrare il pannello informativo.
- **Reconfiguring:** il click su un nodo permette di trascinarlo, modificando il layout della rete.

Visualizzazione 2: Bar Chart

- **Selection:** il click su una barra la evidenzia
- **Reconfiguring:** tramite un tasto di selezione è possibile scegliere l'attributo da visualizzare sull'asse y del grafico tra Designers, Categories e Mechanics.
- **Filtering:** è possibile interagire con degli slider che limitano il numero di elementi da visualizzare.
- **Abstract Zooming (details on demand):** il passaggio del mouse su una barra mostra una tooltip contenente il numero di giochi per quella selezione. Inoltre, il click su una barra mostra i chart aggiuntivi, selezionabili con dei tasti.

Visualizzazione 3: Grouped Bar Chart + Bar Chart

- **Reconfiguring:** un tasto permette di passare dal grouped bar chart al bar chart classico, cambiando quindi l'idioma visuale.
- **Selection:** il click su una barra per il bar chart o sull'area del grafico relativa a un anno per il grouped bar chart la evidenzia.
- **Filtering:** è possibile interagire con due slider che selezionano il range degli anni da visualizzare nell'asse x del grafico, inoltre nel caso del grouped bar chart, è presente una legenda che permette di selezionare i tipi di giochi da visualizzare nel grafico.
- **Abstract Zooming (details on demand):** il passaggio del mouse su una barra mostra una tooltip contenente il numero di giochi per quella selezione. Inoltre, il click su una barra o sull'area del grafico relativa a un anno mostra i chart aggiuntivi, selezionabili con dei tasti.

Visualizzazioni 1, 2 e 3: Chart addizionali (bar chart, dumbbell chart)

Per quanto riguarda gli idiomi che formano gli *additional charts*, quali bar chart e dumbbell chart, i task di interazione sono gli stessi in tutte le visualizzazioni in cui sono presenti e sono i seguenti:

- **Reconfiguring** ed **Encoding:** il click sul tasto visualizza lo specifico grafico, cambiando quindi l'idioma visuale (bar chart o dumbbell chart) oppure modifica l'attributo da visualizzare sull'asse y, a seconda del grafico selezionato dall'utente.
- **Filtering:** l'interazione con un tasto permette di limitare il numero di elementi da visualizzare (questo task di interazione è presente solo per le Visualizzazioni 2 e 3).
- **Abstract Zooming:** il passaggio del mouse su una barra, nel caso dei bar chart mostra una tooltip con i titoli dei giochi relativi a quell'attributo o informazioni sul gioco, dipendentemente al valore sull'asse y. Il passaggio del mouse sui dot del dumbbell chart mostra una tooltip con il valore di quel punto, il passaggio sopra alla linea mostra il range di valori tra minimo e massimo.

Visualizzazione 4: Chord Diagram

- **Abstract Zooming:** il passaggio del mouse sopra a un link mostra una tooltip con i dettagli della collaborazione/combinazione, ovvero la lista dei giochi coinvolti.
- **Reconfiguring:** tramite un tasto di selezione è possibile scegliere la rete da visualizzare tra quella dei Designers, Categories e Mechanics.
- **Selection:** il passaggio del mouse sopra a un nodo evidenzia tutti i link (*chords*) che coinvolgono quel nodo. Il click sul nodo rende la condizione permanente, mentre cliccando sull'area esterna al diagramma si ripristina la colorazione originale.

3.3 Scelte architetturali e tecnologiche

Il sistema progettato consiste in un'applicazione web interattiva che fa uso delle seguenti tecnologie:

- *HTML*
- *CSS*
- *JavaScript*
- *D3.js*: una libreria javascript open source con un approccio low level che permette di interagire con gli elementi grafici per realizzare visualizzazioni di dati anche complesse. Per questo progetto è stata utilizzata la versione 6 di D3.
- Node.js: per le fasi di *data extraction* e *data modeling*

L'applicazione prevede la presenza di una dashboard principale da cui è possibile selezionare la visualizzazione aprendo la relativa pagina. Ciascuna pagina è realizzata con tecnologie web standard quali HTML per la struttura delle sezioni e CSS per lo stile. Il codice Javascript che gestisce il caricamento dei dataset e la realizzazione delle visualizzazioni è incluso in file javascript separati per ogni pagina. Per le sezioni in comune tra le pagine come i chart addizionali e le funzioni di supporto, il codice in comune è stato inserito in due file javascript separati "*additional_charts.js*" e "*utils.js*" che sono caricati da tutte le pagine HTML che ne fanno uso.

Gli utenti possono interagire in vari modi: è possibile utilizzare il mouse per puntare, selezionare, fare panning, scrolling, zooming. Sono presenti anche dei tasti appositi per un'interazione ancora più intuitiva.

Node.js è utilizzato in fase di *data extraction* per fare varie richieste alle API di BoardGameGeek (BGG XML API)[3] tramite le quali sono stati ottenuti gli attributi *type* dei giochi, come specificato nel **capitolo 2.2.1**, e le immagini dei 100 giochi utilizzate per arricchire la visualizzazione finale. I package specifici di Node.js utilizzati sono:

- *"axios"*, per le richieste http
- *"xml2js",* per il parsing da XML a oggetto JavaScript

3. La documentazione delle XML API di BoardGameGeek è disponibile su boardgamegeek.com/xmlapi

3.4 Algorithm engineering

Di seguito sono descritti gli algoritmi utilizzati per creare le visualizzazioni, in particolare: il modello *force directed* per la rete top 100 orientata con il calcolo degli inviluppi convessi e l'algoritmo per il calcolo delle componenti connesse per le reti di designer, categorie e meccaniche per la creazione del *Chord Diagram*.

3.4.1 Force directed (Spring embedder)

Il modello *Spring Embedder* è impiegato per i layout node-link di reti e associa ai nodi delle cariche elettrostatiche repulsive e ai link delle forze elastiche attrattive. In questo modo i nodi tendono ad allontanarsi tra loro, mentre i link tendono ad avvicinarli: una volta arrivata all'equilibrio, la rete tende a evidenziare gli eventuali cluster.

La libreria D3.js utilizza *d3.force* per creare una simulazione: attraverso un modello iterativo che aggiorna le posizioni dei nodi in base alle forze è possibile disegnare la rete.

Per facilitare la convergenza, viene utilizzato il parametro *alpha,* un numero da 0 a 1 che rappresenta la "temperatura" nel modello *simulated annealing*. All'inizio della simulazione viene assegnato ad alpha un valore di 1, per poi diminuire ad ogni iterazione fino ad annullarsi quando viene raggiunta la convergenza.

Per applicare le forze alla simulazione sono state utilizzate le seguenti funzioni, delle forze predefinite che *d3.forceSimulation()* prende in ingresso:

- *d3.forceLink()*: modella le forze elastiche dei link
- *d3.forceManyBody()*: regola le forze elettrostatiche repulsive tra i nodi
- *d3.forceCollide():* evita che i nodi collidano tra loro
- *d3.forceCenter():* posiziona la visualizzazione al centro dell'interfaccia
- *d3.forceX():* spinge gli elementi verso una posizione desiderata lungo l'asse X
- *d3.forceY():* spinge gli elementi verso una posizione desiderata lungo l'asse Y

È stata utilizzata un'altra forza aggiuntiva di tipo *d3.forceCollide()* per evitare che le label dei nodi si sovrappongano.

In aggiunta a queste forze, ne è stata creata una personalizzata, *forceCluster()*, che avvicina ulteriormente i nodi con stesso attributo *type*, di seguito è indicato il codice implementato per questa forza:

```
function forceCluster(alpha) {
  const strength = 0.15;
  const padding = 250; //distanza tra i cluster
  let nodes;
  let centers = {};

  function force(alpha) {
    nodes.forEach(d => {
      d.type.forEach(nodeType => {
        if (centers[nodeType]) {
          // Attrai il nodo verso il centro del suo gruppo
          d.vx += (centers[nodeType].x - d.x) * strength * alpha;
          d.vy += (centers[nodeType].y - d.y) * strength * alpha;
        }
```

```
      });
    });
  }
  force.initialize = function(_nodes) {
    nodes = _nodes;
    // Calcola i centri per ogni tipo
    const types = Array.from(new Set(nodes.flatMap(d => d.type)));
    types.forEach((type, i) => {
      const angle = (i / types.length) * 2 * Math.PI;
      centers[type] = {
        x: (width/2) + Math.cos(angle) * padding,
        y: (height/2) + Math.sin(angle) * padding
      };
    });
  }
  return force;
}
```

3.4.2 Convex Hull

Gli inviluppi convessi (*Convex Hull*) vengono calcolati per i nodi dello stesso tipo nella rappresentazione *Unconstrained Straight-Line* per poter colorare l'area delimitata dal poligono ottenuto. L'inviluppo convesso di un insieme di punti è il poligono convesso di superficie minima che racchiude l'insieme.

D3.js permette di calcolare l'inviluppo convesso di un insieme di punti (*points)* attraverso la funzione *d3.polygonhull(points)*. Questa funzione secondo la documentazione utilizza l'algoritmo *Andrew's monotone chain* che ha una complessità di $O(n \log n)$. [4]

3.4.3 Componenti connesse

Per selezionare la top 10 degli elementi da visualizzare con l'idioma Chord Diagram, si è scelto di implementare il calcolo delle componenti connesse all'interno delle reti di collaborazione tra designers, categorie e meccaniche.

Questo approccio permette di risolvere il problema per cui gli elementi selezionati possono non avere giochi in comune tra loro ed essere quindi isolati all'interno del Chord Diagram.

Questo problema è stato riscontrato soprattutto nella visualizzazione dei designers, infatti, molti designers hanno creato numerosi giochi ma hanno collaborato con altri soltanto in pochi di essi ed inoltre molti di questi hanno collaborato al massimo con uno o due altri designers.

Per ottenere quindi i 10 elementi da visualizzare si è filtrato inizialmente dalle reti di collaborazioni solo i nodi che hanno almeno una connessione.

Tra questi si calcolano le componenti connesse usando una visita in profondità (DFS), prendendo la componente massima trovata e selezionando all'interno di questa i primi 10 nodi ordinati in base alla loro diffusione nei giochi.

Nel caso in cui la componente avesse meno di 10 elementi, vengono aggiunti ulteriori nodi tra quelli più diffusi all'interno dei giochi e non ancora selezionati.

[4] Dettagli sulla funzione d3.polygonhull sono presenti al link https://d3js.org/d3-polygon

Di seguito si riporta lo pseudocodice che segue la logica appena descritta e mostra i passaggi fatti nel codice JavaScript:

```
Funzione prepareData(dataset):
  Inizializza:
    - Una mappa gioco → individui (designer, categorie, meccaniche)
    - Un set di individui che collaborano
    - La rete (nodi e link) a partire dal dataset

  Per ogni individuo nella rete:
    Per ogni gioco associato a quell'individuo:
      Aggiungilo alla lista degli individui per quel gioco nella mappa

  Per ogni link nella rete:
    Aggiungi i due estremi (source e target) al set dei collaboratori

  Trova la componente connessa più grande:
    Per ogni individuo:
      Calcola la componente connessa a partire dal suo ID (con DFS)
      Se la componente è più grande della più grande trovata finora:
        Salvala

  Filtra i nodi della componente più grande

  Se i nodi della componente sono meno di 10:
    Ordina gli altri collaboratori (non inclusi) per numero di giochi
    Aggiungine fino ad arrivare a 10 nodi totali

  Ordina i 10 nodi selezionati per numero di giochi in ordine decrescente

  Crea un set con gli ID dei 10 individui selezionati

  Filtra i link esistenti per mantenere solo quelli tra gli individui selezionati

  Per ogni link filtrato:
    Calcola i giochi in comune tra i due nodi collegati

  Crea la matrice per il chord diagram:
    - Assegna un indice a ogni nodo selezionato
    - Popola la matrice usando i pesi dei link rilevanti

  Restituisci:
    - I 10 nodi principali
    - I link tra di loro, con giochi in comune
    - La matrice per la visualizzazione

Funzione findConnectedComponentDFS(startId, network):
  Inizializza un set di nodi visitati

  Funzione ricorsiva dfs(currentId):
    Se il nodo è già stato visitato, ritorna
    Altrimenti:
      Aggiungi il nodo ai visitati
      Trova tutti i link che lo connettono ad altri nodi
      Per ogni nodo adiacente non ancora visitato:
        Chiama dfs(nextId)

  Avvia la visita DFS dal nodo startId

  Restituisci l'insieme dei nodi visitati (la componente connessa)
```

4. Realizzazione

In questa sezione viene descritto lo sviluppo del sistema e dell'interfaccia utente, facendo uso di immagini e di esempi di utilizzo.

Il sistema si compone di una pagina iniziale, index.html, che permette di aprire le varie visualizzazioni, descrivendole brevemente:

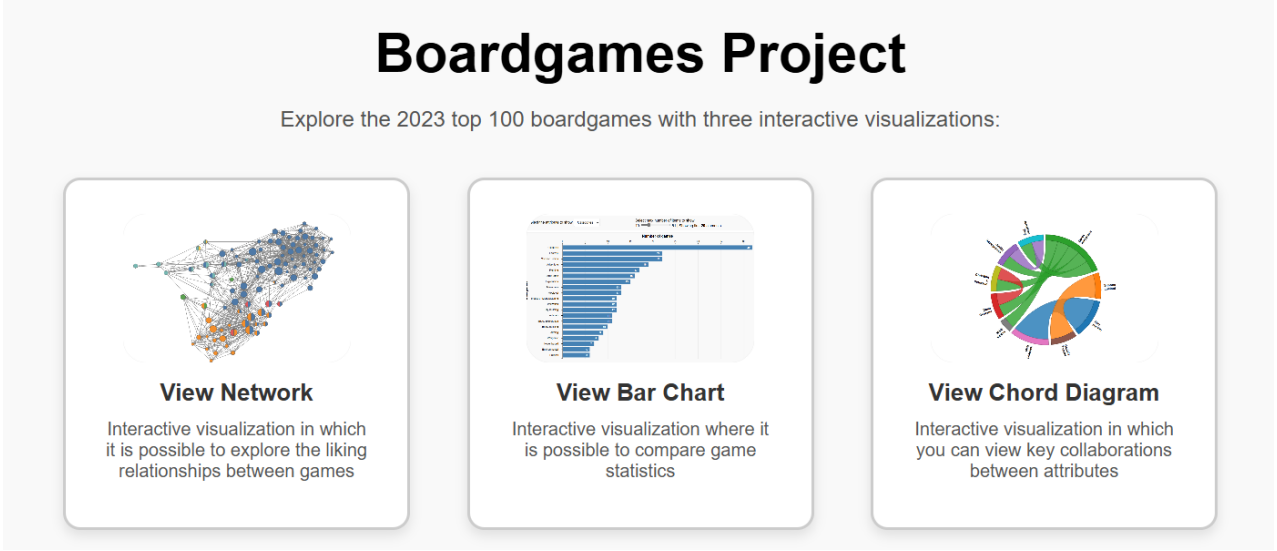


***Figura 1**: pagina iniziale*

I 3 bottoni reindirizzano rispettivamente alle pagine:

- **index_forcedirected.html**, per la *Visualizzazione 1*
- **index_barchart.html** per le *Visualizzazioni 2 e 3*
- **index_chorddiagram.html** per la *Visualizzazione 4*

Segue la lista dei 100 giochi da tavolo ordinati secondo la loro posizione in classifica, in modo da avere un riferimento in dashboard. Per ciascun gioco è possibile cliccare sul bottone "*View in Network*", che apre la pagina della *Visualizzazione 4*, evidenziando il nodo relativo al gioco cliccato e mostrando le sue informazioni (come in *Figura 5*).

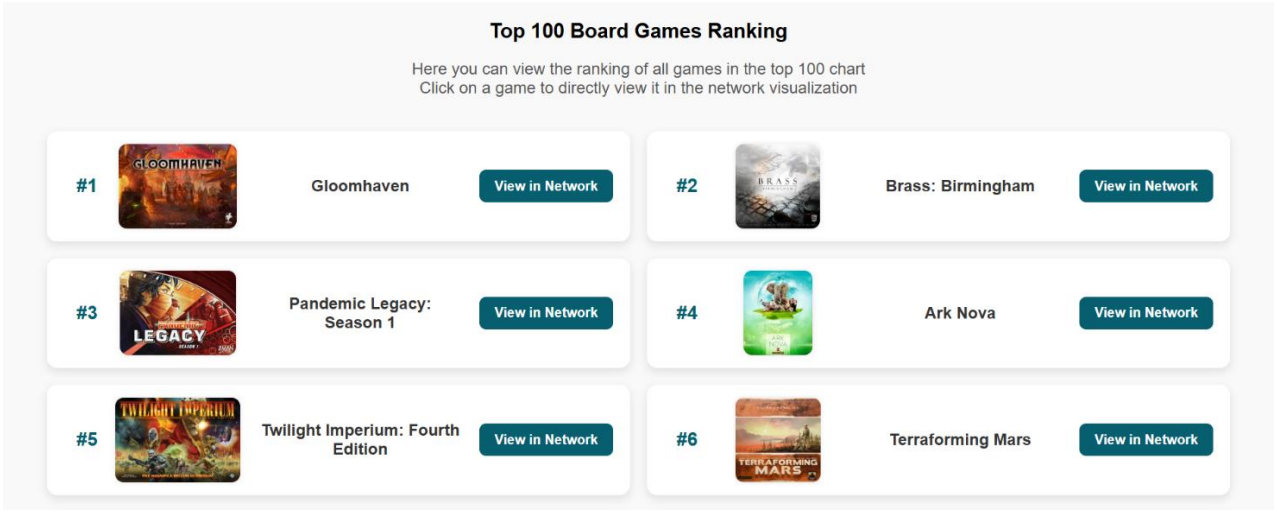


***Figura 2**: elenco dei giochi da tavolo in top 100 nella pagina iniziale*

4.1 index_forcedirected.html

La pagina è dotata dello script *forcedirected.js* che gestisce il caricamento del dataset della rete dei top 100 e genera la visualizzazione con d3.js.

La rete si presenta con una dimensione adatta allo schermo e in modo tale che si veda nella sua interezza al centro della pagina. In alto a sinistra è presente un bottone che permette di tornare alla schermata iniziale.

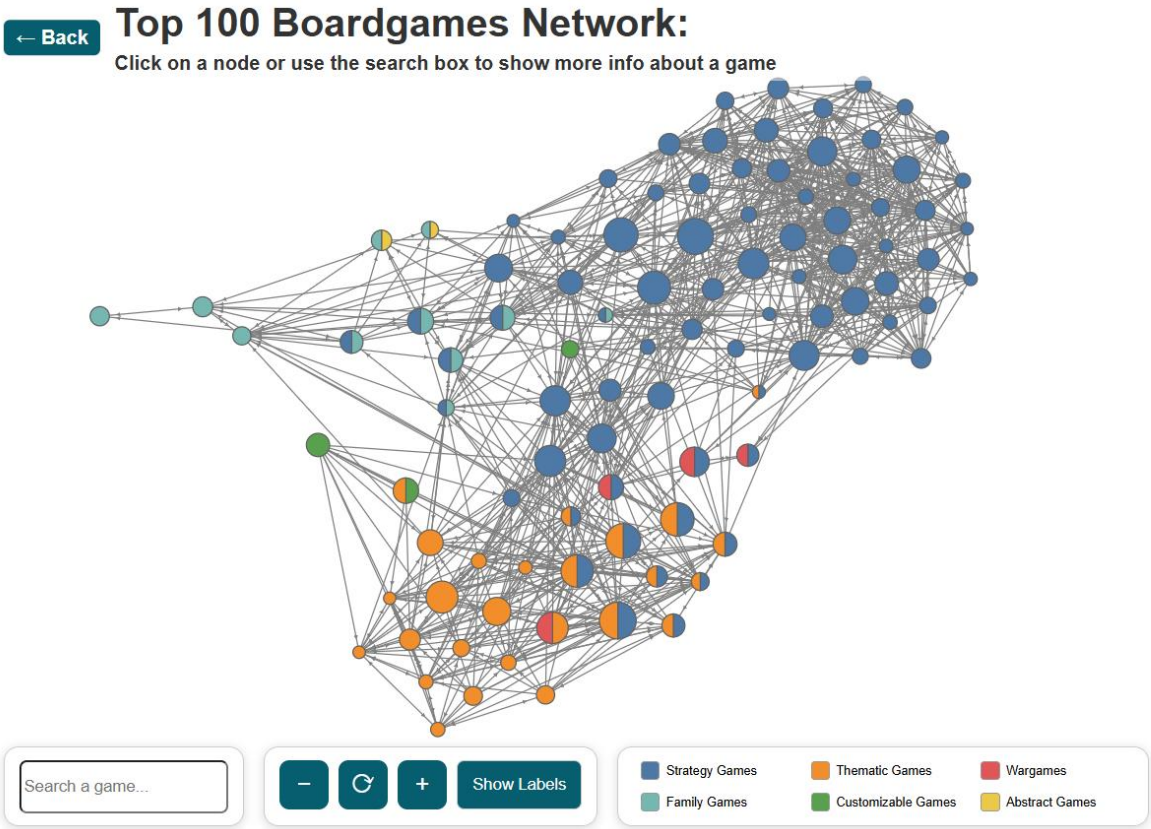
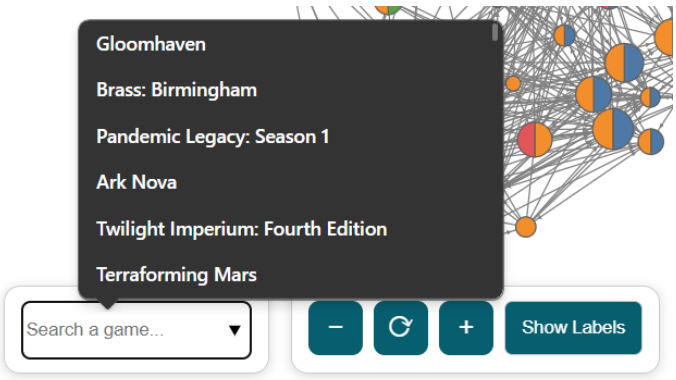


***Figura 3**: schermata iniziale della visualizzazione della rete*

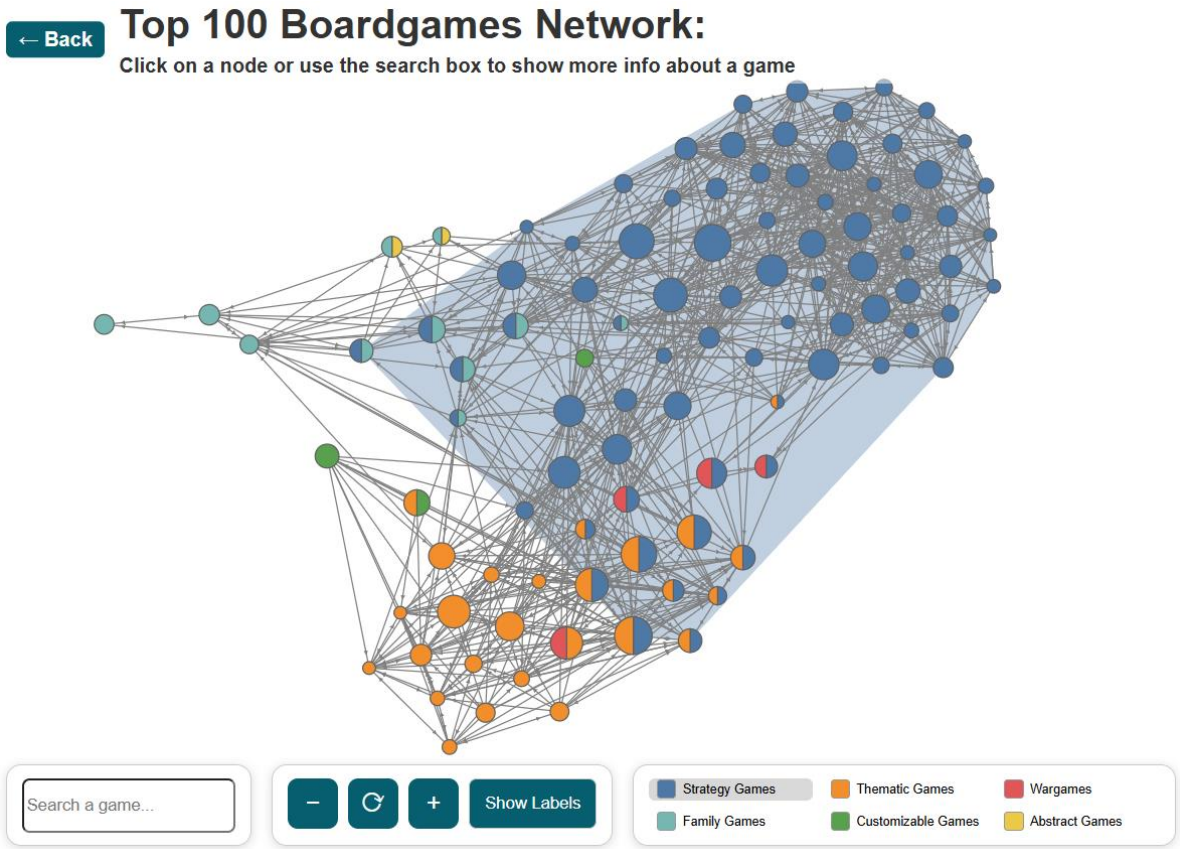
In basso sono presenti delle sezioni di interazione, in ordine:

- una barra di ricerca dei giochi da tavolo dal titolo, con auto completamento e suggerimenti di titoli:

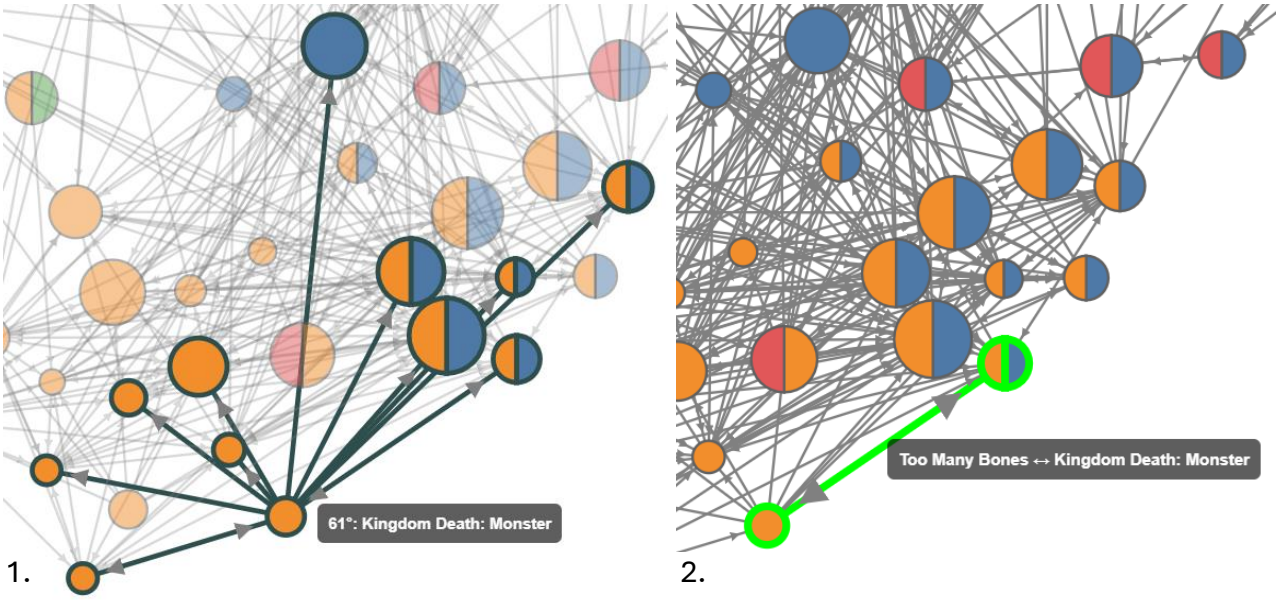


- lo zoom-in/zoom-out e reset dello zoom
- un bottone per visualizzare o nascondere le etichette dei titoli dei giochi sui nodi

- legenda dei tipi di gioco con dei bottoni che mostrano i convex hull. Ad esempio, il convex hull dei giochi da tavolo di tipo Strategy appare come segue:



Il passaggio del mouse su un nodo mostra una *tooltip* con il rank del gioco e il titolo, mentre l'interazione con un link mostra i titoli dei due giochi separati da una freccia che indica la direzione del link (può essere bidirezionale). Un esempio di queste interazioni è riportato di seguito:



1. 2.

***Figura 4**: esempio di passaggio del mouse sopra a un nodo (**1.**) e sopra a un link (**2.**)*

Il click su un nodo evidenzia i nodi adiacenti e fa comparire un pannello sulla destra che mostra il titolo del gioco cliccato, l'immagine e altre informazioni, tra cui i titoli dei giochi piaciuti ai fan del gioco evidenziato:

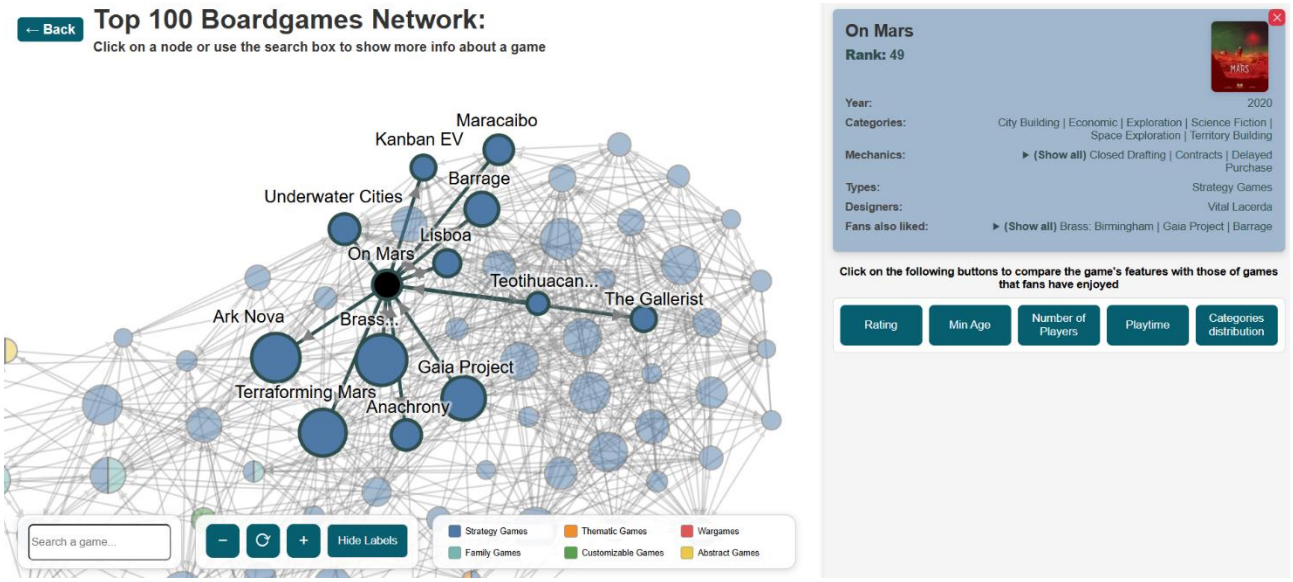
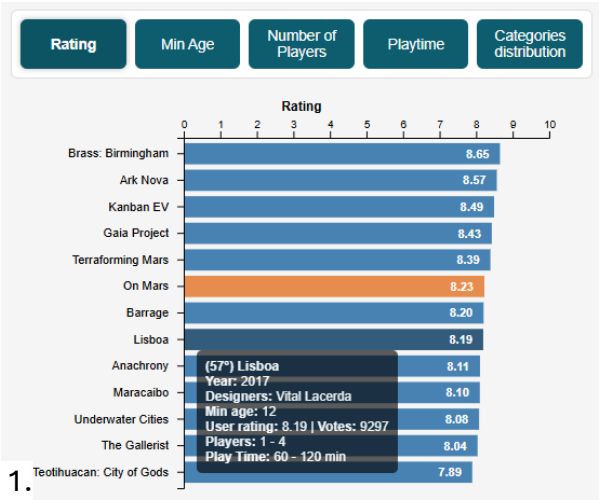
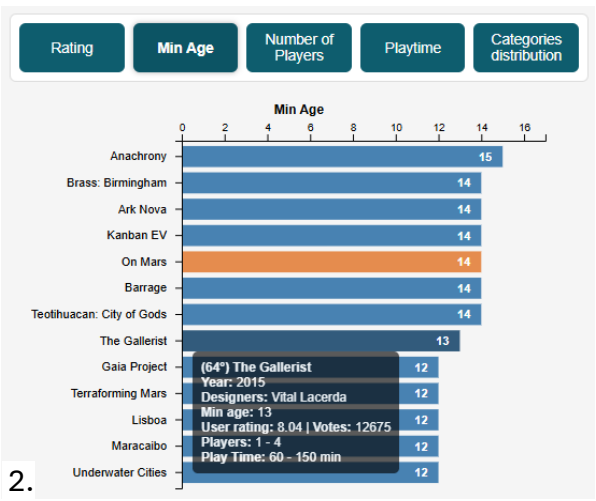


***Figura 5**: click sul gioco "On Mars", colorato di nero*

È possibile interagire con i 5 bottoni del pannello, che permettono di visualizzare i grafici che confrontano il voto, età minima, numero di giocatori, tempo di gioco e distribuzione delle categorie del cluster di giochi formato dal gioco cliccato e da quelli piaciuti ai fan di questo.

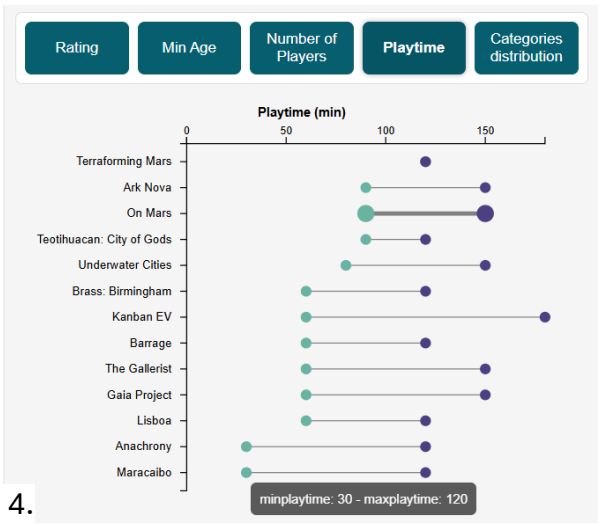


1.

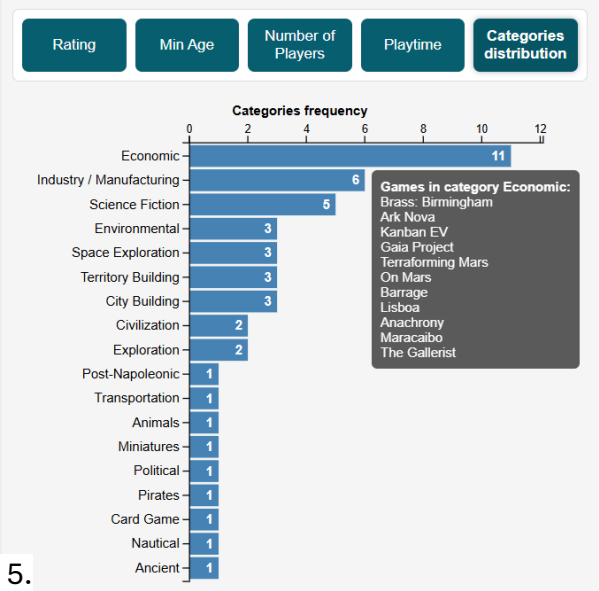


2.

3.



4.



5.

***Figura 6**: grafici dei fans_liked del gioco cliccato, che confrontano voto (**1.**), età minima (**2.**), numero di giocatori (**3.**), tempo di gioco (**4.**) e distribuzione delle categorie (**5.**). Per i primi quattro grafici i valori del gioco cliccato sono evidenziati per colore nei bar chart e per dimensione nei dumbbell chart. Sono inoltre visibili le tooltip che compaiono con il passaggio del mouse.*

Per i bar chart di *Rating* e *Min Age*, che presentano sull'asse y i titoli dei giochi, le barre sono cliccabili e innescano l'evento di click del nodo corrispondente.

A questo livello della visualizzazione è possibile cliccare altri nodi oppure chiudere il pannello per avere una vista più ampia della rete, utilizzando il tasto in alto a destra oppure cliccando in qualsiasi punto vuoto dell'area contenente la rete.

4.2 index_barchart.html

La pagina è gestita dallo script *barchart.js* che gestisce il caricamento dei dataset e genera le visualizzazioni con d3.js.

La pagina è divisa in due schede selezionabili tramite due pulsanti in alto, una scheda apre la *visualizzazione 2* con l'idioma Bar Chart di distribuzione di categorie/meccaniche/designers mentre l'altra apre la *visualizzazione 3* con gli idiomi Grouped Bar Chart e Bar Chart per la distribuzione dei giochi per anno di uscita.

Di seguito sono riportati in dettaglio i tre Bar Chart selezionabili all'interno della prima scheda per la *visualizzazione 2*.

La tipologia di dati da rappresentare, tra categorie, meccaniche e designers, è selezionabile da un menu a tendina mentre uno slider permette di scegliere il numero di istanze da visualizzare nell'asse y.

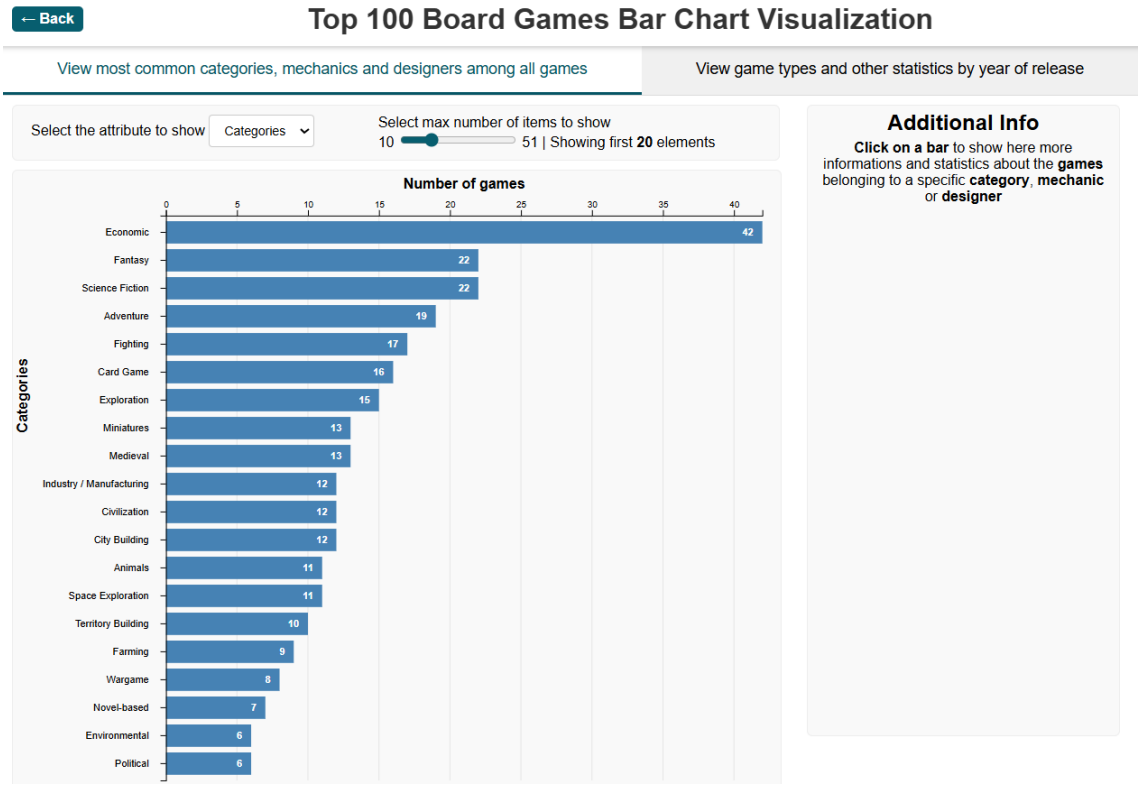


***Figura 7**: Bar Chart per la distribuzione delle categorie nei giochi*

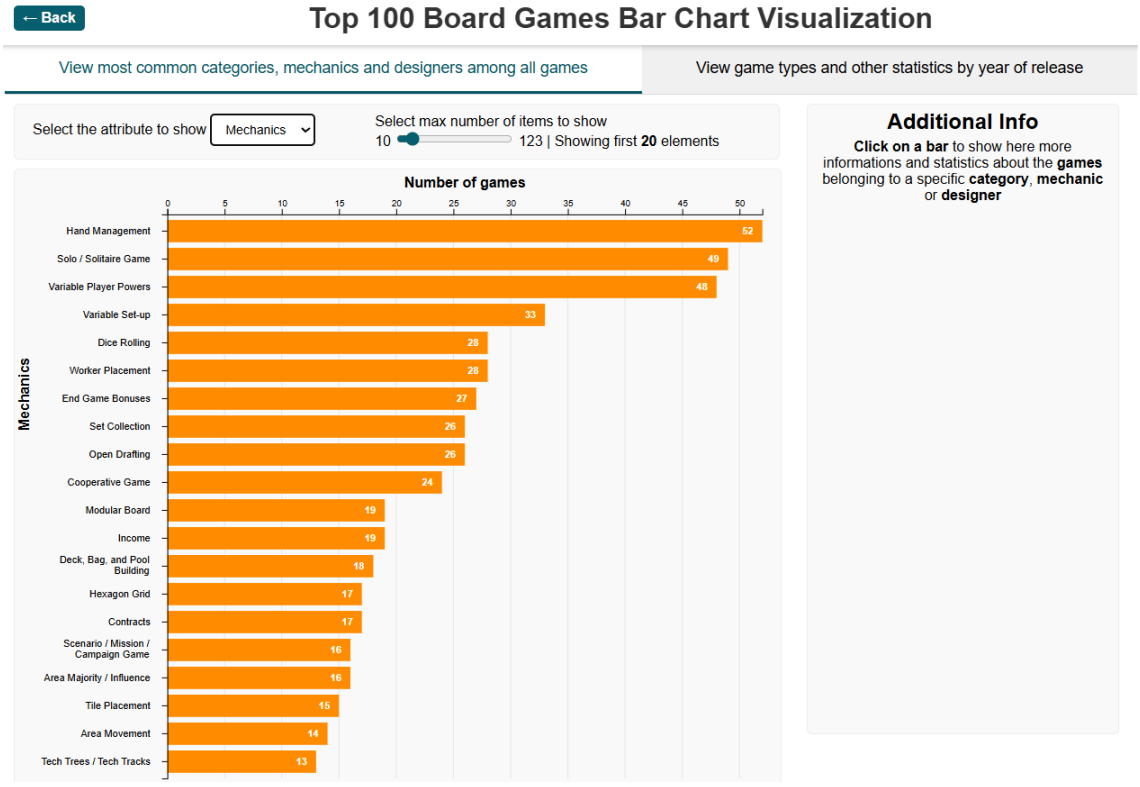


***Figura 8**: Bar Chart per la distribuzione delle meccaniche nei giochi*

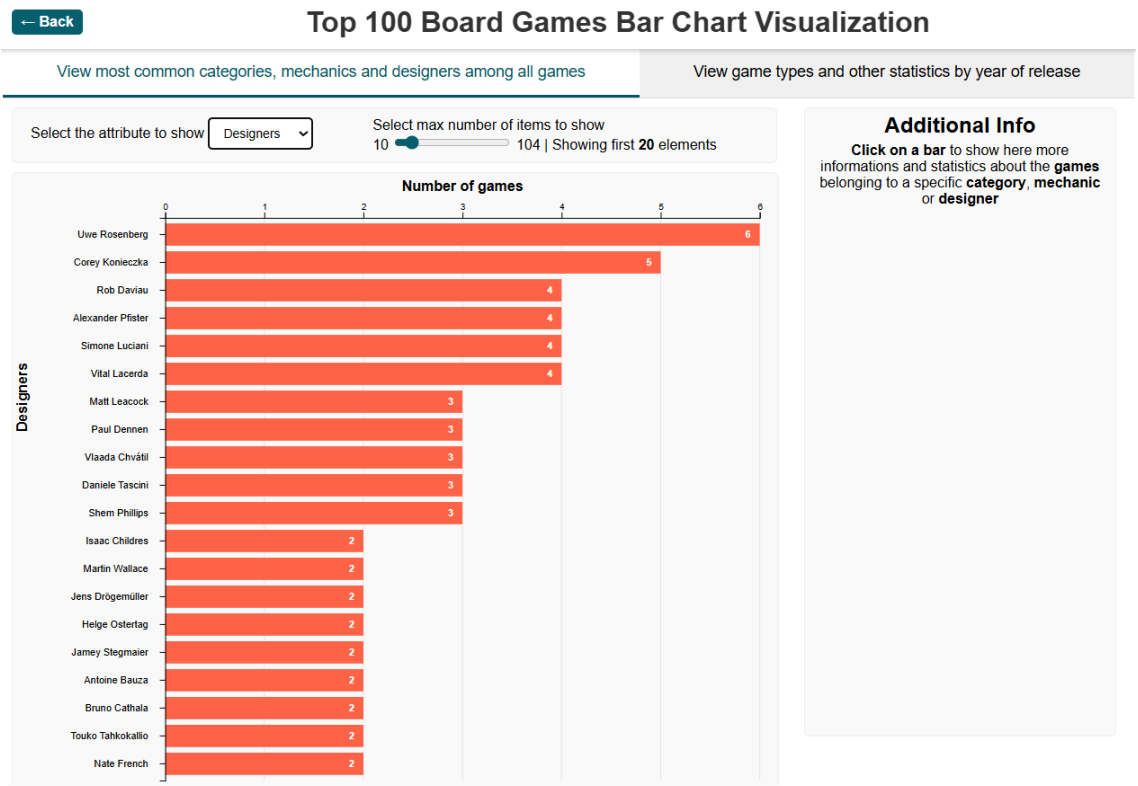


***Figura 9**: Bar Chart per la distribuzione dei designers nei giochi*

Il click su una barra popola l'area a destra del grafico con i *Chart addizionali* per l'insieme dei giochi da tavolo che presentano la categoria, meccanica o designer selezionato. In particolare, sono presenti i Bar Chart per la distribuzione di voto e per l'età minima e i Dumbbell Chart per il range di tempo di gioco e per il numero di giocatori.

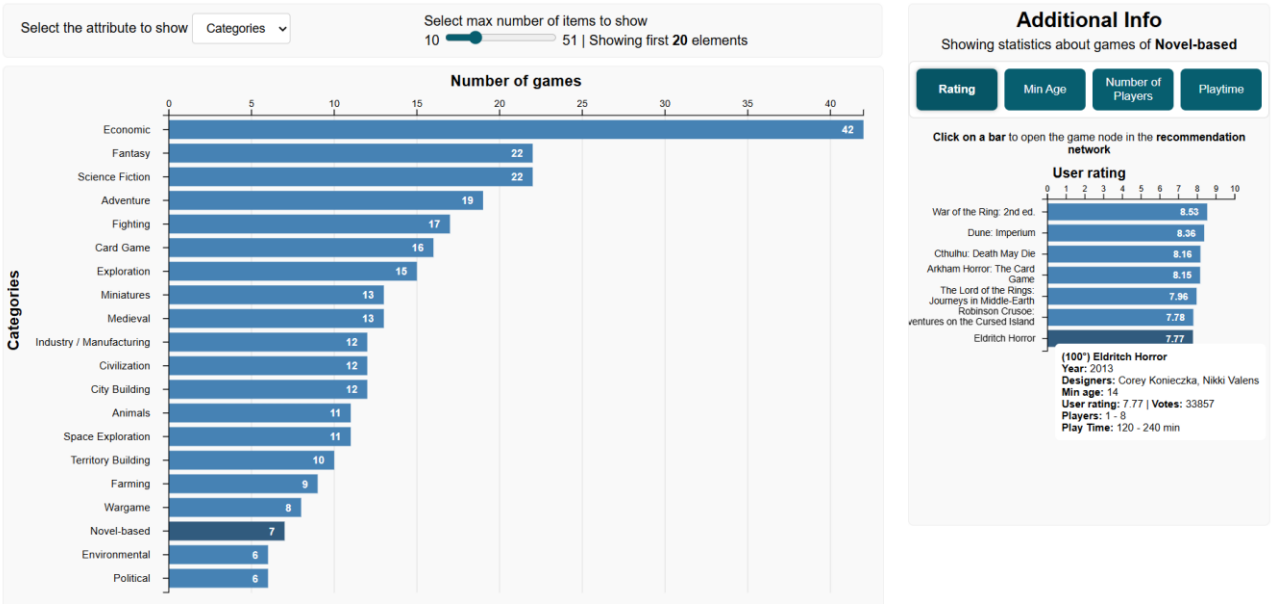


***Figura 10:** evento di click su una barra*

Anche le barre dei Bar Chart addizionali sono cliccabili e permettono di aprire, in una nuova pagina, il nodo relativo al gioco cliccato all'interno della rete della top 100 (*Visualizzazione 1*), come in ***Figura 5***.

Selezionando la seconda scheda della pagina si apre la *visualizzazione 3*. Nella parte alta sono presenti due bottoni che permettono di cambiare idioma tra *Bar Chart*, che visualizza la distribuzione dei giochi per anno di uscita, e *Grouped Bar Chart*, che visualizza i tipi di giochi usciti per ogni anno. Il Grouped Bar Chart presenta anche una legenda che associa un colore ad ogni tipo di gioco e permette di mostrare/nascondere le barre relative a specifici tipi.

È possibile inoltre limitare il range di anni di uscita da visualizzare sull'asse x tramite due slider per la selezione dell'anno minimo e massimo da visualizzare nel grafico.

***Figura 11**: Grouped Bar Chart (**1.**) e Bar Chart (**2.**) della distribuzione dei giochi per anno di uscita*

Anche in questo caso è previsto l'evento di click del mouse per mostrare i grafici addizionali, come quelli descritti per la ***Figura 6***, oltre al Bar Chart della distribuzione delle categorie. Il click è previsto per l'area del grafico relativa a un anno nel caso del Grouped Bar Chart e per la barra nel caso di Bar Chart. I dati rappresentati in questi grafici sono quelli dei giochi che hanno come anno di uscita quello dell'anno selezionato.

Segue un esempio dell'interazione appena descritta:

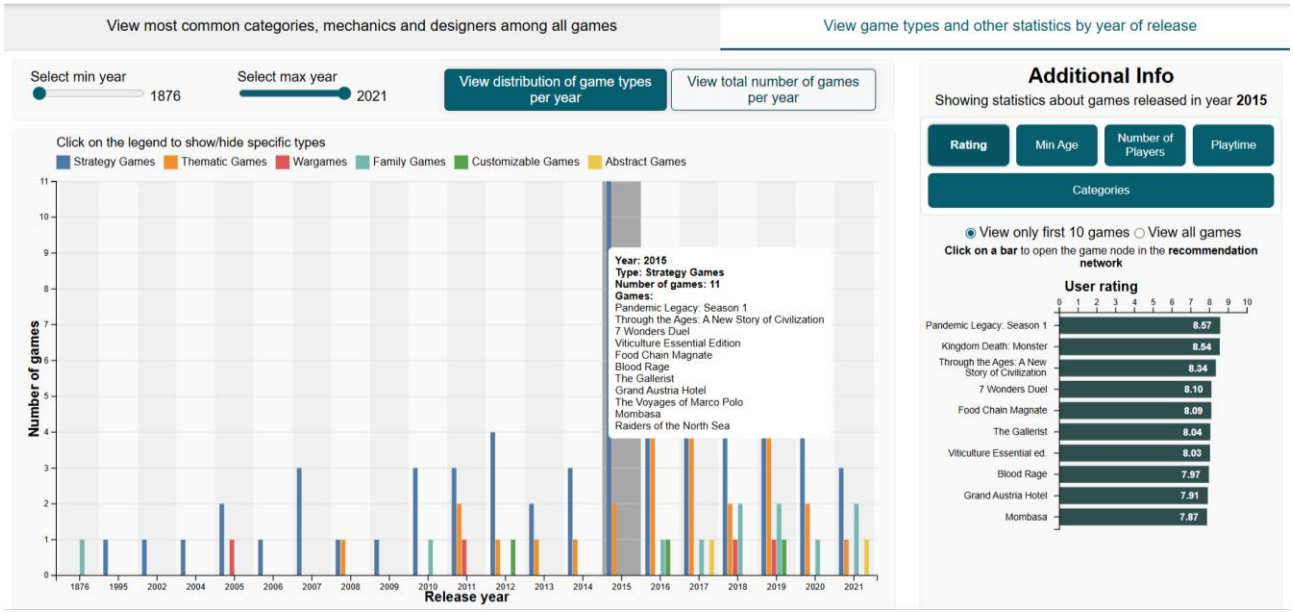


***Figura 12**: evento di click dell'anno 2015 sul Grouped Bar Chart*

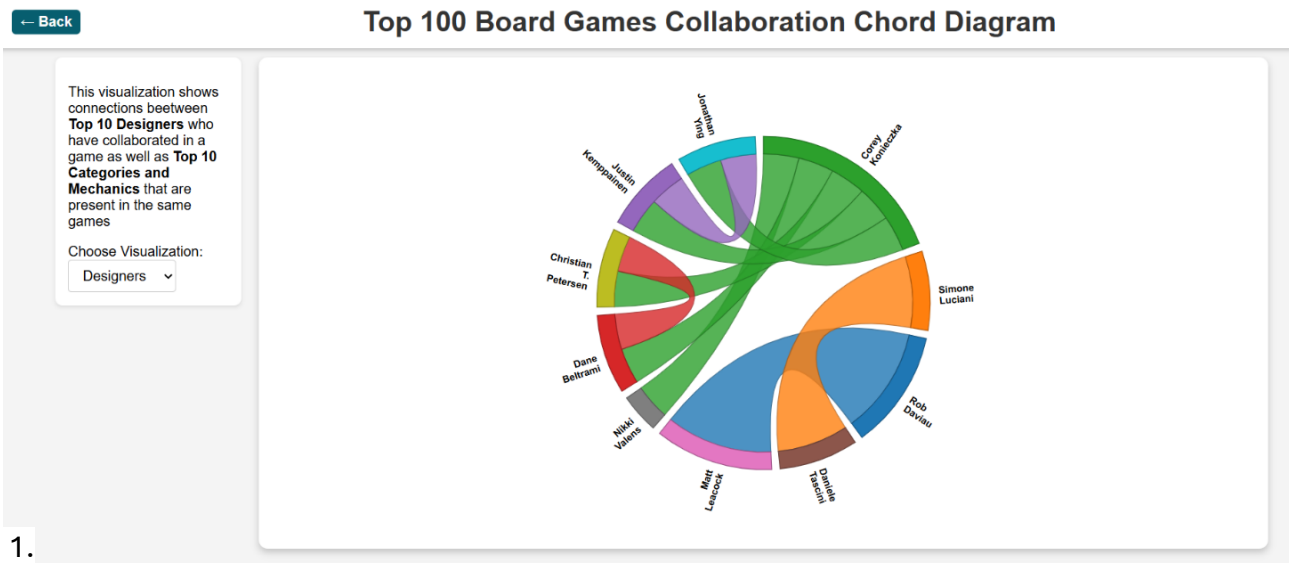
Anche in questo caso, il click sulle barre dei giochi dei Bar Chart addizionali per il voto e l'età minima riporta alla visualizzazione del gioco nella rete della top 100.

4.3 index_chorddiagram.html

La pagina è gestita dallo script *chorddiagram.js* che gestisce il caricamento delle tre reti di collaborazione di designer, categorie e meccaniche e genera le visualizzazioni con d3.js.

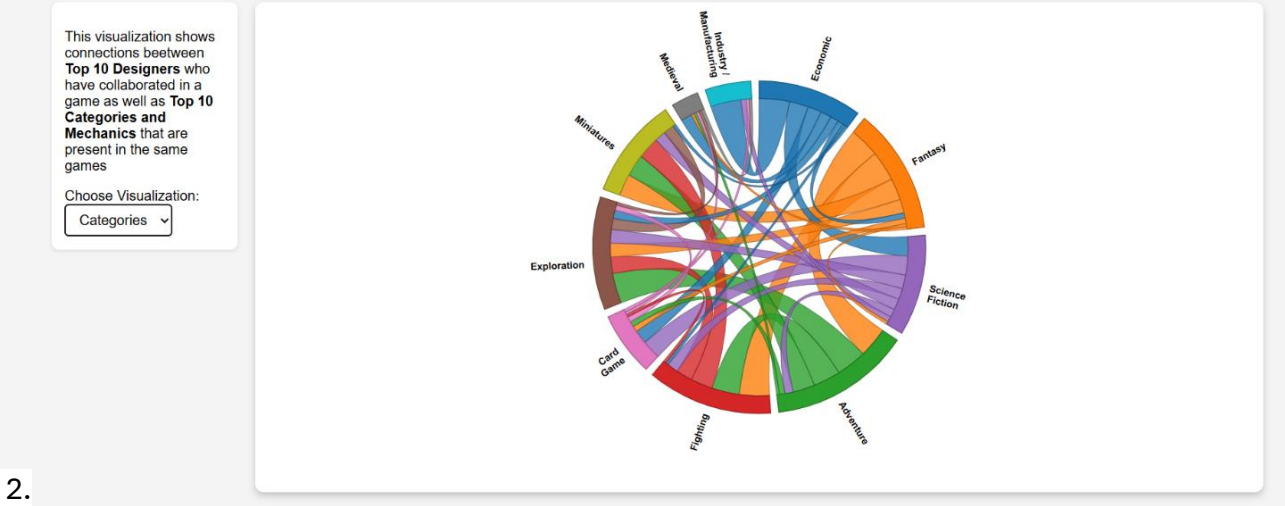
Inizialmente viene visualizzato il *Chord Diagram* relativo alla rete dei designers; la presenza di un dropdown menu permette di selezionare le altre reti delle categorie o meccaniche.

Seguono le immagini del diagramma per le tre reti:



1.

2.



3.

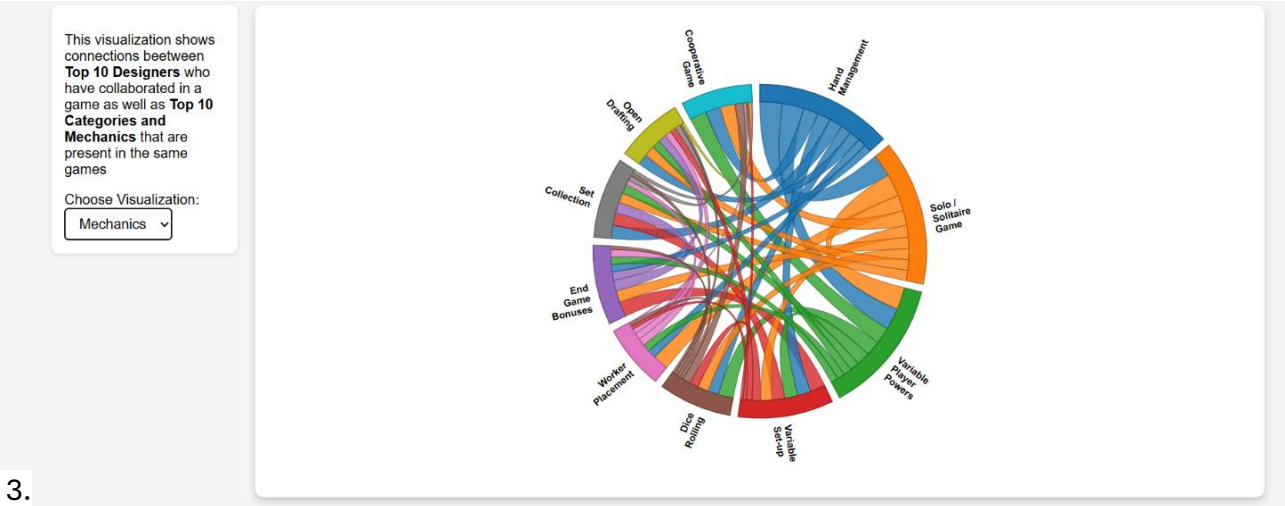


***Figura 13**: Chord diagram delle collaborazioni per la rete dei designers (**1.**), delle categorie (**2.**) e delle meccaniche (**3.**)*

I nodi (arc) e i link (ribbon) della visualizzazione rispondono agli eventi del mouse:

- il passaggio del mouse sopra a un link evidenzia il link e mostra una tooltip contenente i giochi della collaborazione/combinazione (***Figura 14***)
- Il passaggio del mouse sopra a un nodo evidenzia i link incidenti al nodo e nasconde gli altri. Il click sul nodo rende permanente la condizione, mentre il click fuori dal diagramma ristabilisce la colorazione originaria (***Figura 15***)

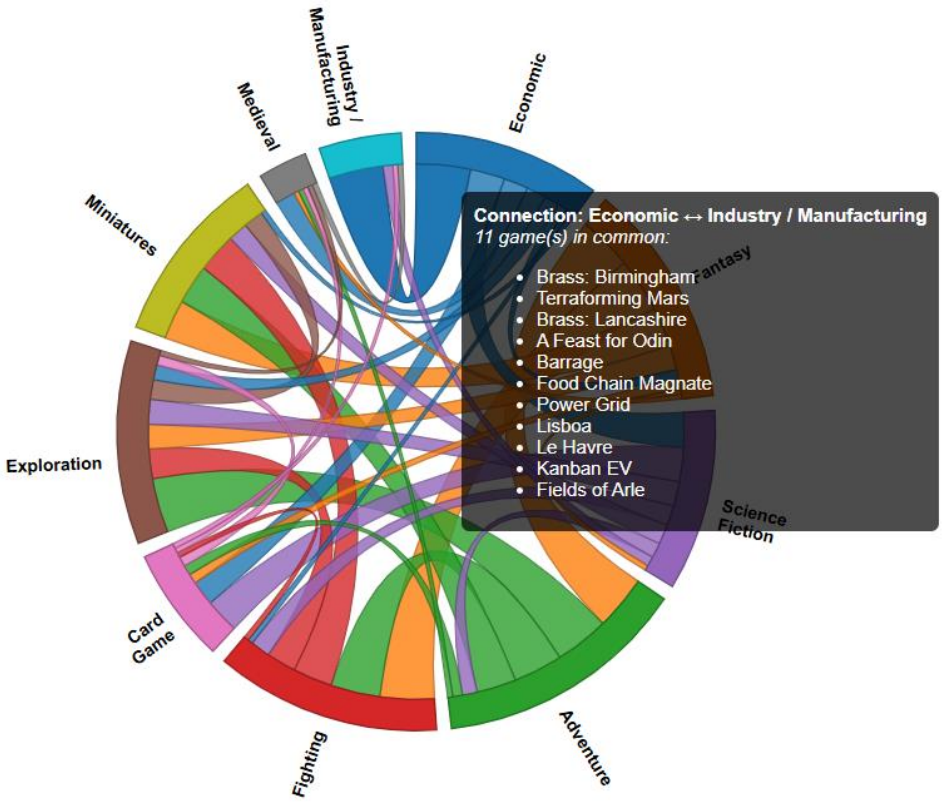


Figura 14: *dettaglio del passaggio del mouse sopra un link (ribbon)*

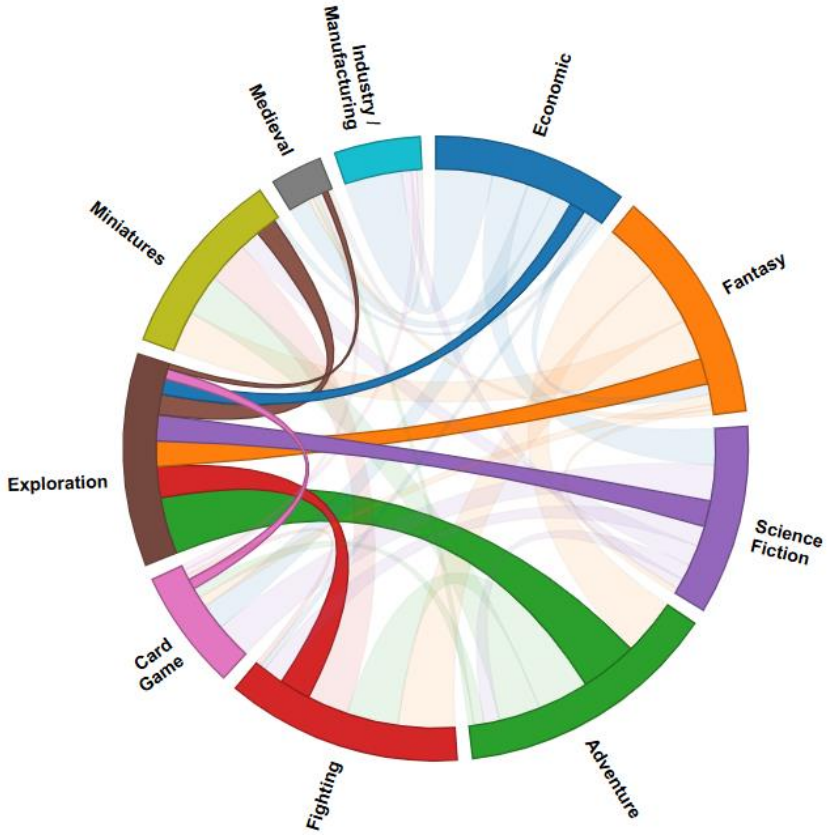


Figura 15: *dettaglio del click su un nodo (arc)*