nintendo_gw_viz

December 12, 2023

1 Nintendo Game & Watch: Data Visualizations

The goal of this project is to explore the Nintendo Game & Watch dataset through visualizations. These visualizations provide an easy and intuitive way for retro game collectors to gain insights and a deeper understanding of the evolution and impact of Nintendo Game & Watch over time.

1.1 Exploratory Data Analysis (EDA)

This notebook follows the six practices of EDA by Google: discovering, structuring, cleaning, joining, validating, and presenting. Note that the six practices of EDA are iterative and non-sequential, so the order used in this notebook may differ from the above.

1.1.1 1 - Discovering

Imports

[1]:
```
# Standard library modules
import datetime as dt
import math
import pathlib
import re
from typing import Union

# Third-party modules
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib.lines import Line2D

# First-party modules
from utils import (
    generate_timeline_levels,
    get_timeline_levels,
    visualize_games_produced,
    visualize_games_released,
    visualize_outliers,
    visualize_series_released,
    visualize_timeline,
)
```

Datasets

[2]:

```
# Nintendo Game & Watch dataset
dataset_filepath = pathlib.Path("data/nintendo_game_and_watch.csv")
df = pd.read_csv(dataset_filepath)
```

First look at the data

[3]:

```
df.head(10)
```

[3]:

```
        game       series  model                    acronyms  \
0       Ball       Silver  AC-01             ACrobat, ACtion   
1    Flagman       Silver  FL-02              FLag, FLagman   
2     Vermin       Silver  MT-03  Mogura Tataki, Mole Tunnel   
3       Fire       Silver  RC-04                      ResCue   
4      Judge       Silver  IP-05           ImPact, InterPret   
5    Manhole         Gold  MH-06                     ManHole   
6     Helmet         Gold  CN-07                CoNstruction   
7       Lion         Gold  LN-08                        LioN   
8  Parachute  Wide Screen  PR-21                   PaRachute   
9    Octopus  Wide Screen  OC-22                     OCtopus   

     date of release  release order   produced rarity               battery  \
0     April 28, 1980              1    250 000     VR  2 x LR-43 (or SR-43)   
1       June 5, 1980              2    250 000     XR  2 x LR-43 (or SR-43)   
2      July 10, 1980              3    500 000      R  2 x LR-43 (or SR-43)   
3      July 31, 1980              4  1 000 000      R  2 x LR-43 (or SR-43)   
4    October 4, 1980              5    250 000     XR  2 x LR-43 (or SR-43)   
5   January 29, 1981              6    500 000      R  2 x LR-43 (or SR-43)   
6  February 21, 1981              7    500 000      R  2 x LR-43 (or SR-43)   
7     April 29, 1981              8    250 000      R  2 x LR-43 (or SR-43)   
8      June 19, 1981              9  1 200 000     VC  2 x LR-43 (or SR-43)   
9      July 16, 1981             10  1 200 000     VC  2 x LR-43 (or SR-43)   

                                         description  
0  Your task is to control the hands of a juggler…  
1  Simon type of game. You have to repeat the seq…  
2    Hit the moles as they surface from the tunnels.  
3  You control two guys with a stretcher (which a…  
4  Hit your opponent with a hammer or avoid his h…  
5  Your task is to save passersby from falling in…  
6     Reach the shed while avoiding falling objects.  
7  You control two tamers who try to stop the lio…  
8  Save as many paratroopers as you can with your…  
9     Get the treasure and avoid octopus' tentacles.  
```

Summary information

[4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63 entries, 0 to 62
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   game             63 non-null     object
 1   series           63 non-null     object
 2   model            63 non-null     object
 3   acronyms         63 non-null     object
 4   date of release  63 non-null     object
 5   release order    63 non-null     int64 
 6   produced         63 non-null     object
 7   rarity           63 non-null     object
 8   battery          63 non-null     object
 9   description      63 non-null     object
dtypes: int64(1), object(9)
memory usage: 5.0+ KB
```

Descriptive statistics

[5]:

```
df.describe()
```

[5]:

```
       release order
count      63.000000
mean       32.000000
std        18.330303
min         1.000000
25%        16.500000
50%        32.000000
75%        47.500000
max        63.000000
```

1.1.2 2 - Cleaning

Duplicates

[6]:

```
# Check for duplicates
n_duplicate_rows = df.duplicated().sum()
n_rows = len(df)
print(f"Duplicate rows: {n_duplicate_rows} ({(n_duplicate_rows / n_rows):.2%})")
```

```
Duplicate rows: 0 (0.00%)
```

The dataset does not have any duplicates.

Missing values

[7]:

```
# Check for missing values
df.isna().sum()
```

[7]:

```
game                0
series              0
model               0
acronyms            0
date of release     0
release order       0
produced            0
rarity              0
battery             0
description         0
dtype: int64
```

The dataset does not have any missing values.

Data types

[8]:

```
# Convert "produced" values to numeric
df["produced"] = df["produced"].replace("Unknown", np.nan)  # replace "Unknown"␣
 ↪values with NaN
df["produced"] = df["produced"].str.replace(" ", "").astype("Int64")  # remove␣
 ↪all whitespaces and convert the type
```

[9]:

```
# Convert "rarity" values to numeric (ordinal)
rarity_dict = {
    "XC": 1,
    "VC": 2,
    "C": 3,
    "UC": 4,
    "R": 5,
    "VR": 6,
    "XR": 7,
}
df["rarity"] = df["rarity"].map(rarity_dict)
```

[10]:

```
# Convert "date of release" values to datetime
df["date of release"] = pd.to_datetime(df["date of release"])
```

[11]:

```
# Check data types
df.dtypes
```

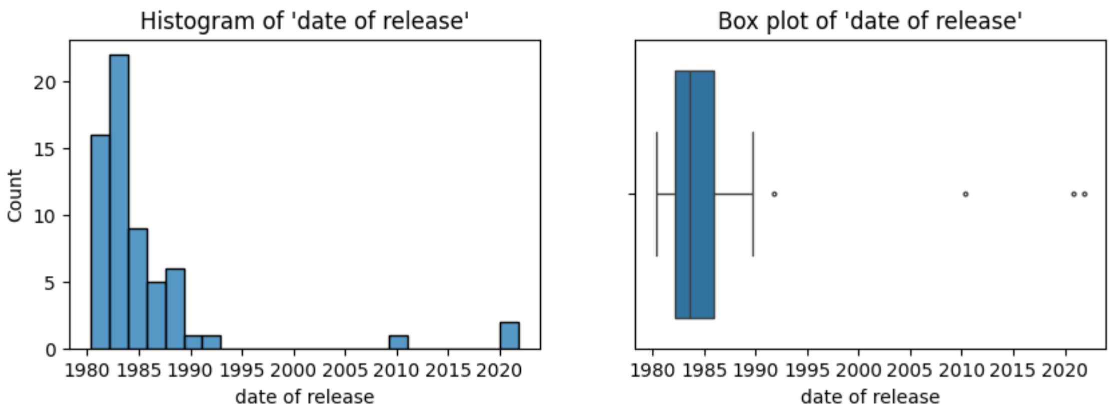
[11]:

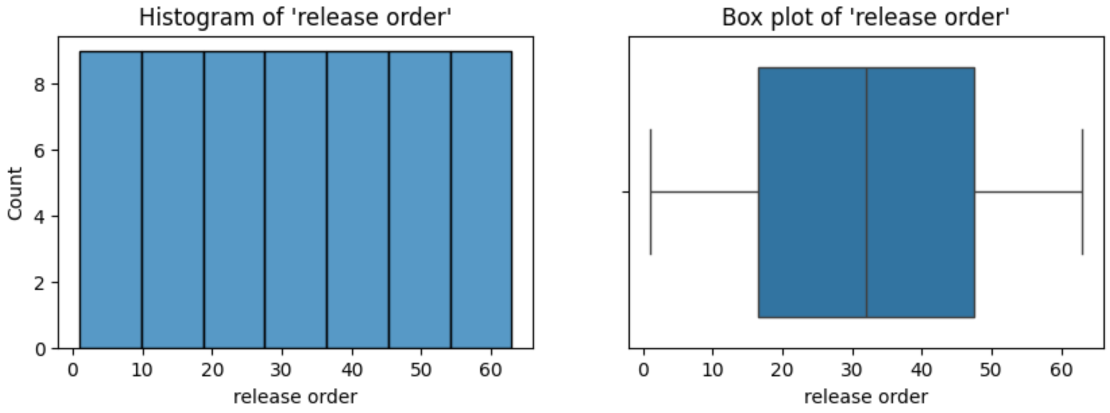
```
game                       object
series                     object
model                      object
acronyms                   object
date of release    datetime64[ns]
release order               int64
produced                    Int64
rarity                      int64
```

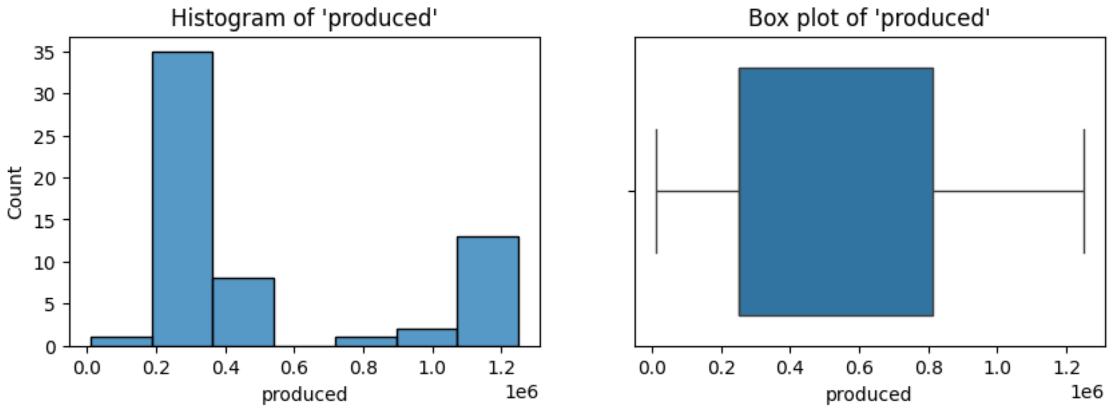
```
battery                         object
description                     object
dtype: object
```

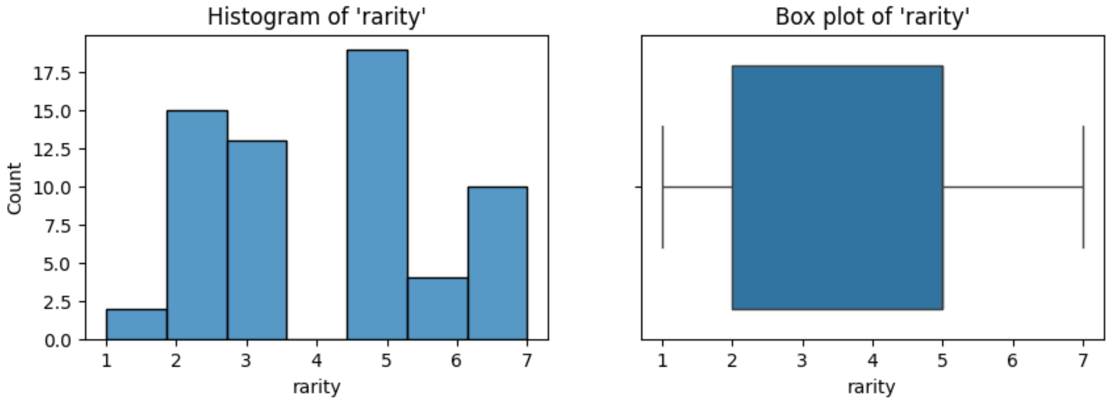
Outliers

[12]:
```python
# Create histogram and box plot visualizations for each column to check for outliers
for column_name in ["date of release", "release order", "produced", "rarity"]:
    visualize_outliers(df, column_name)
```









The `date of release` column has some outliers in the 2000s. However, these values are correct.

1.1.3 3 - Structuring

Series

[13]:

```
# Create a series data frame
series_df = df.copy()
series_df = series_df.groupby(["series"]).agg(
    from_date=pd.NamedAgg(column="date of release", aggfunc="min"),
    to_date=pd.NamedAgg(column="date of release", aggfunc="max"),
    games=pd.NamedAgg(column="series", aggfunc="count"),
    produced=pd.NamedAgg(column="produced", aggfunc="sum")
).sort_values(by="from_date").reset_index()
series_df["from"] = series_df["from_date"].dt.year
series_df["to"] = series_df["to_date"].dt.year
series_df = series_df.drop(["from_date", "to_date"], axis=1)
series_df
```

[13]:
```
               series  games  produced  from    to
0              Silver      5   2250000  1980  1980
1                Gold      3   1250000  1981  1981
2         Wide Screen     10   8950000  1981  1982
3        Multi Screen     15  10250000  1982  1989
4     New Wide Screen      8   4200000  1982  1991
5           Table Top      4   1000000  1983  1983
6     Panorama Screen      6   1500000  1983  1984
7         Super Color      2    500000  1984  1984
8    Micro VS. System      3    750000  1984  1984
9      Crystal Screen      3    750000  1986  1986
10    Special Edition      1     10000  1987  1987
11            Reissue      1         0  2010  2010
12      Colour Screen      2         0  2020  2021
```

Games

[14]:
```
# Create a games data frame
games_df = df.copy()
games_df = games_df.drop(["acronyms", "rarity", "battery", "description"],␣
 ↪axis=1)
games_df = games_df.sort_values(by="release order", ascending=True).
 ↪reset_index(drop=True)
games_df
```

[14]:
```
                    game           series    model date of release  \
0                   Ball           Silver    AC-01      1980-04-28
1                Flagman           Silver    FL-02      1980-06-05
2                 Vermin           Silver    MT-03      1980-07-10
3                   Fire           Silver    RC-04      1980-07-31
4                  Judge           Silver    IP-05      1980-10-04
..                   …                …        …             …
58                 Zelda     Multi Screen    ZL-65      1989-08-26
59     Mario the Juggler  New Wide Screen   MB-108      1991-10-14
60                  Ball          Reissue  RGW-001      2010-04-28
61      Super Mario Bros.    Colour Screen  HXA-001      2020-11-13
62   The Legend of Zelda    Colour Screen  HXB-001      2021-11-12

    release order  produced
0               1    250000
1               2    250000
2               3    500000
3               4   1000000
4               5    250000
..            …        …
58             59    250000
59             60    250000
60             61      <NA>
```

```
61              62      <NA>
62              63      <NA>

[63 rows x 6 columns]
```

Games released

[15]:

```
# Create a games released data frame
games_released_df = df.copy()
games_released_df = games_released_df.groupby(games_released_df["date of␣
 ↪release"].dt.year).agg(
    games=pd.NamedAgg(column="game", aggfunc="count"),
).sort_values(by="date of release").reset_index()
games_released_df = games_released_df.rename(columns={"date of release":␣
 ↪"year"})
games_released_df
```

[15]:

```
    year  games
0   1980      5
1   1981     10
2   1982      8
3   1983     15
4   1984      7
5   1985      2
6   1986      4
7   1987      2
8   1988      5
9   1989      1
10  1991      1
11  2010      1
12  2020      1
13  2021      1
```

1.1.4 4 - Validating

Series

[16]:

```
# Check for duplicates
series_df.duplicated().any()
```

[16]:

```
False
```

[17]:

```
# Check for missing values
series_df.isna().any()
```

[17]:

```
series      False
games       False
produced    False
from        False
to          False
```

```
dtype: bool
```

[18]:
```
# Check data types
series_df.dtypes
```

[18]:
```
series      object
games        int64
produced     Int64
from         int32
to           int32
dtype: object
```

Games

[19]:
```
# Check for duplicates
games_df.duplicated().any()
```

[19]:
```
False
```

[20]:
```
# Check for missing values
games_df.isna().any()
```

[20]:
```
game               False
series             False
model              False
date of release    False
release order      False
produced            True
dtype: bool
```

[21]:
```
# Check data types
games_df.dtypes
```

[21]:
```
game                       object
series                     object
model                      object
date of release    datetime64[ns]
release order               int64
produced                    Int64
dtype: object
```

Games released

[22]:
```
# Check for duplicates
games_released_df.duplicated().any()
```

[22]:
```
False
```

[23]:
```
# Check for missing values
games_released_df.isna().any()
```

```
[23]: year     False
      games    False
      dtype: bool
```

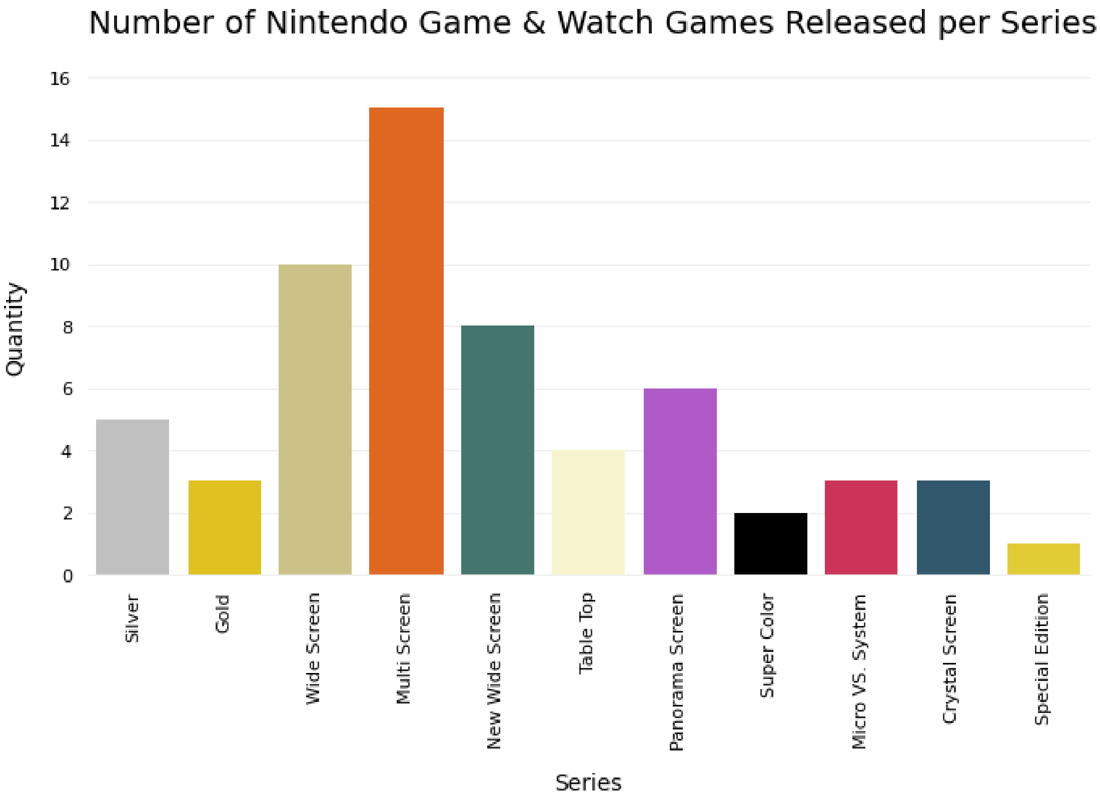
```
[24]: # Check data types
      games_released_df.dtypes
```

```
[24]: year     int32
      games    int64
      dtype: object
```

1.1.5 5 - Presenting

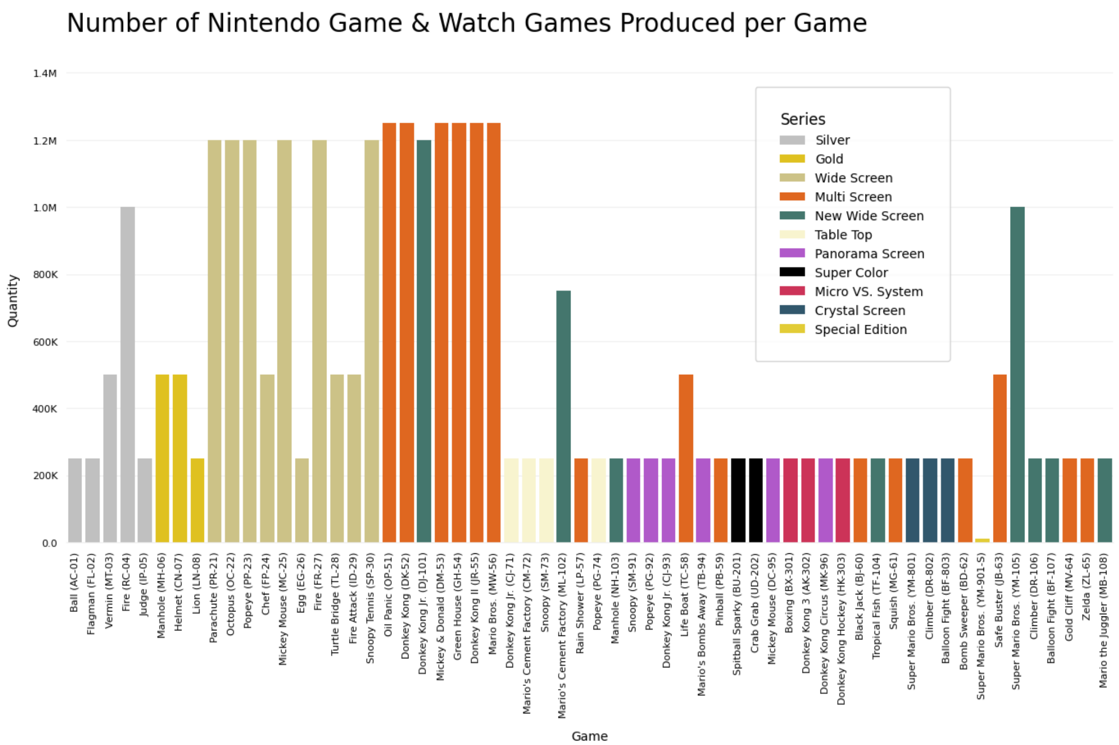
Series released

```
[25]: # Create a bar plot visualization showing the number of games released per␣
        ↪series
      visualize_series_released(series_df, max_release_year=1991)
```



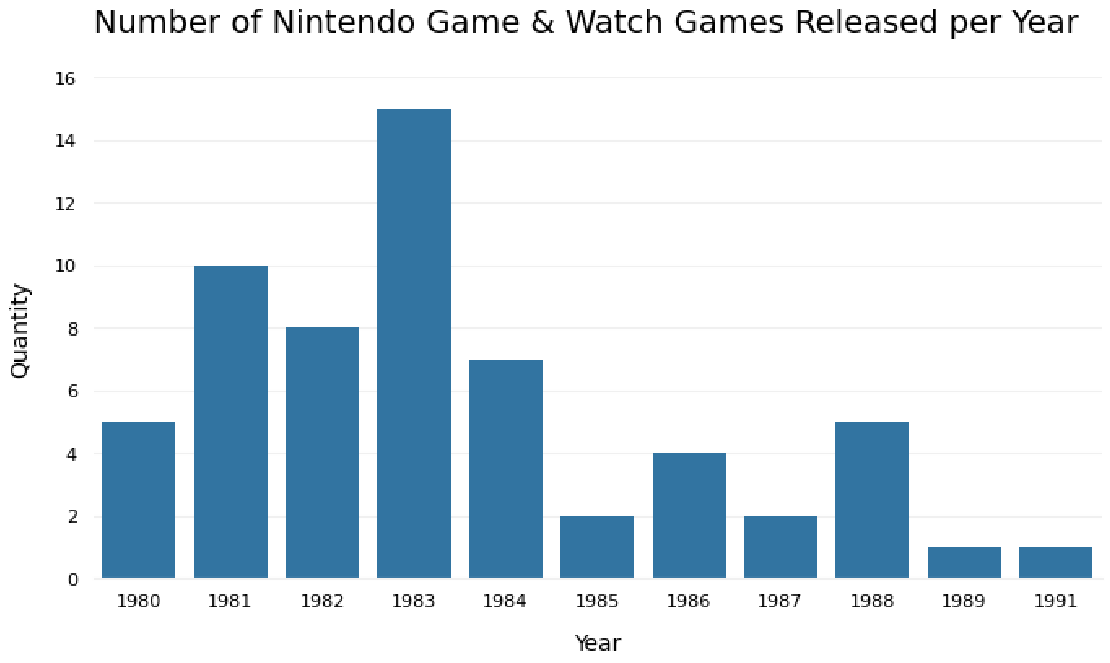
Games produced

```
[26]: # Create a bar plot visualization showing the number of games produced per game
      visualize_games_produced(games_df, max_release_year=1991)
```



Games released

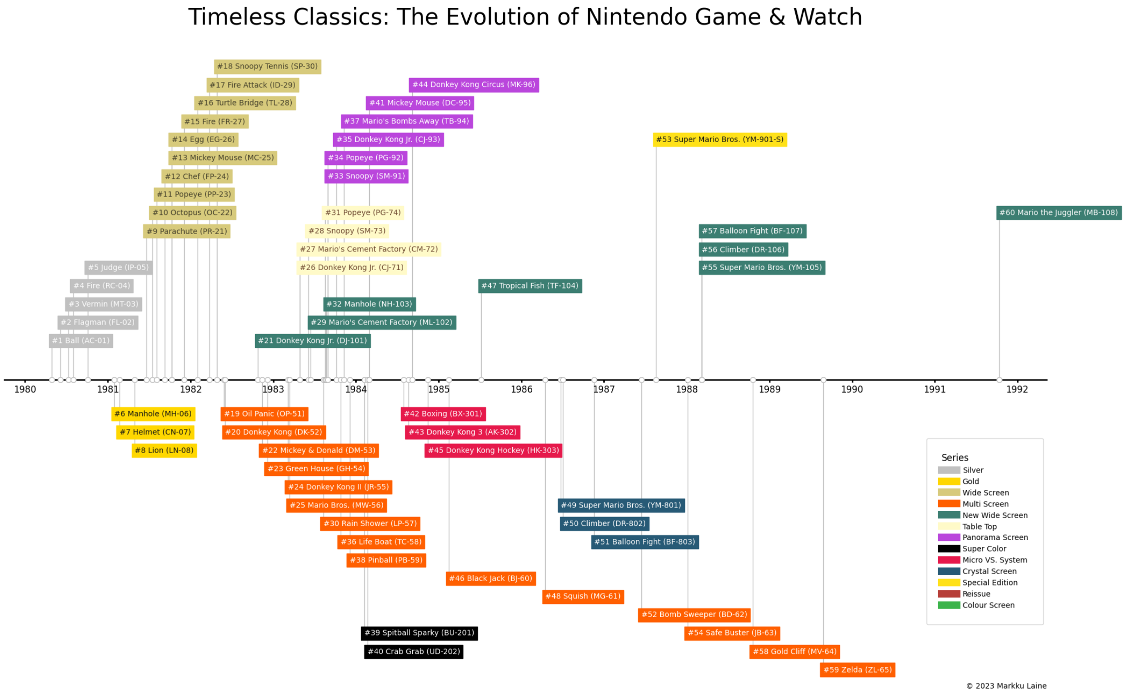
[27]:
```python
# Create a bar plot visualization showing the number of games released per year
visualize_games_released(games_released_df, max_release_year=1991)
```



Timeline

[28]:

```python
# Create a timeline visualization to display the evolution of Nintendo Game & Watch
visualize_timeline(games_df, max_release_year=1991, auto_levels=False)
```


Timeless Classics: The Evolution of Nintendo Game & Watch
#18 Snoopy Tennis (SP-30)
#17 Fire Attack (ID-29)
#16 Turtle Bridge (TL-28)
#15 Fire (FR-27)
#14 Egg (EG-26)
#13 Mickey Mouse (MC-25)
#12 Chef (FP-24)
#11 Popeye (PP-23)
#10 Octopus (OC-22)
#9 Parachute (PR-21)
#5 Judge (IP-05)
#4 Fire (RC-04)
#3 Vermin (MT-03)
#2 Flagman (FL-02)
#1 Ball (AC-01)
#44 Donkey Kong Circus (MK-96)
#41 Mickey Mouse (DC-95)
#37 Mario's Bombs Away (TB-94)
#35 Donkey Kong Jr. (CJ-93)
#34 Popeye (PG-92)
#33 Snoopy (SM-91)
#31 Popeye (PG-74)
#28 Snoopy (SM-73)
#27 Mario's Cement Factory (CM-72)
#26 Donkey Kong Jr. (CJ-71)
#32 Manhole (NH-103)
#29 Mario's Cement Factory (ML-102)
#21 Donkey Kong Jr. (DJ-101)
#47 Tropical Fish (TF-104)
#53 Super Mario Bros. (YM-901-S)
#57 Balloon Fight (BF-107)
#56 Climber (DR-106)
#55 Super Mario Bros. (YM-105)
#60 Mario the Juggler (MB-108)
#6 Manhole (MH-06)
#7 Helmet (CN-07)
#8 Lion (LN-08)
#19 Oil Panic (OP-51)
#20 Donkey Kong (DK-52)
#22 Mickey & Donald (DM-53)
#23 Green House (GH-54)
#24 Donkey Kong II (JR-55)
#25 Mario Bros. (MW-56)
#30 Rain Shower (LP-57)
#36 Life Boat (TC-58)
#38 Pinball (PB-59)
#39 Spitball Sparky (BU-201)
#40 Crab Grab (UD-202)
#42 Boxing (BX-301)
#43 Donkey Kong 3 (AK-302)
#45 Donkey Kong Hockey (HK-303)
#46 Black Jack (BJ-60)
#48 Squish (MG-61)
#49 Super Mario Bros. (YM-801)
#50 Climber (DR-802)
#51 Balloon Fight (BF-803)
#52 Bomb Sweeper (BD-62)
#54 Safe Buster (JB-63)
#58 Gold Cliff (MV-64)
#59 Zelda (ZL-65)
Series
Silver
Gold
Wide Screen
Multi Screen
New Wide Screen
Table Top
Panorama Screen
Super Color
Micro VS. System
Crystal Screen
Special Edition
Reissue
Colour Screen
© 2023 Markku Laine