Lab 1 - GameEye Product Description

Angelos Angelopoulos

Old Dominion University

CS 411W: Professional Workforce Development II

Professor Janet Brunelle

September 13, 2020

Version 1

Table of Contents

List of Figures

List of Tables

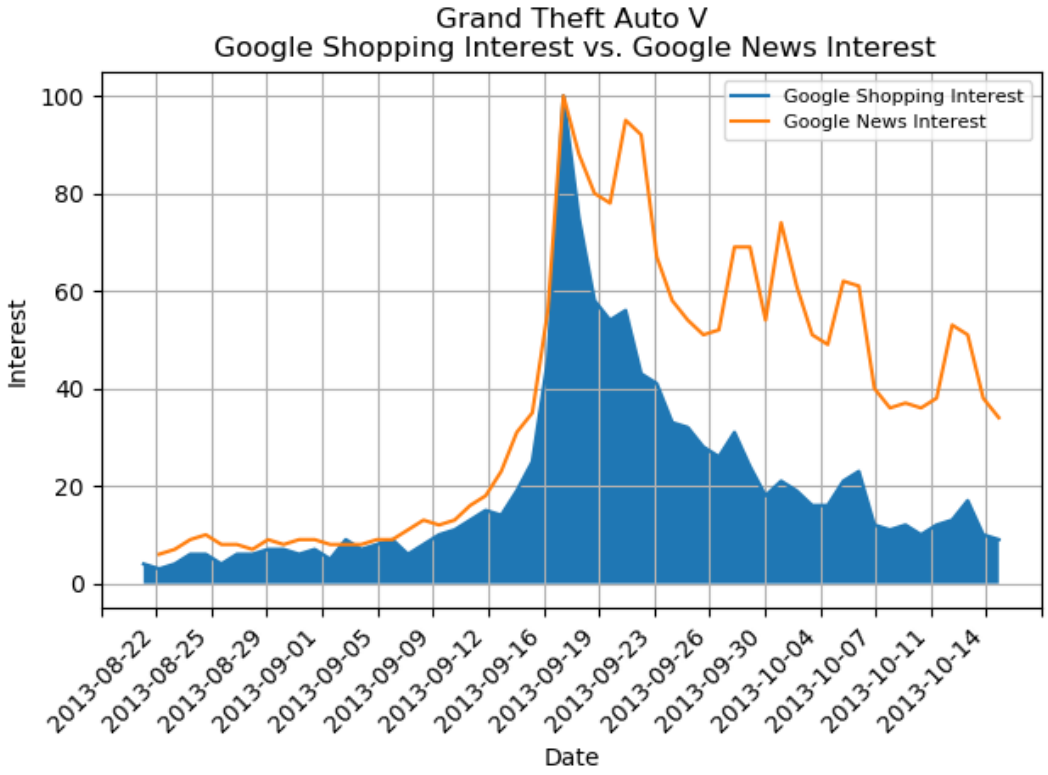
1. Introduction

The video game industry has seen incredible growth in past decades. Today, there are more than 2.47 billion gamers worldwide (Gough, 2019), and the video game industry is projected to reach a value of $196 billion in 2022, surpassing the film industry. Thousands of video games are released every year (Anderton, 2019). Just in 2018, more than 9000 games were released for the PC on the Steam platform, and this number has been increasing in the past years (Gough, 2019). By also considering console and mobile games, this number increases to tens of thousands (Gough, 2019). With so many games coming out annually, a problem starts to surface. Gamers may desire to play many upcoming games but staying up to date with all of them can be time-consuming.

Video games are costly to produce both in money and time. A typical video game takes one to seven years to make, and some games take even longer than that. For example, the popular game *Team Fortress 2*, was in development for nine years (Dietz, 2011). Because video games often have long development times, it gets tiresome to look for news online. Part of a game's development budget includes marketing, as getting the word out for a video game is crucial for driving sales, as shown in Figure 1. Large development studios spend millions on marketing. This gives rise to another problem: indie video games, games from small development studios, may lose the public spotlight and may even be forgotten with time. There are several reasons for this: (a) indie studios having smaller marketing budgets, (b) games staying in development for too long, and (c) large development studios attracting more players and away from other games, hurting indie developers in the process.

Figure 1

Shows Google Shopping interest vs Google News interest for the game Grand Theft Auto V shortly before and after its release. The data is sourced from Google Trends.



Another problem is that information and news about video games is very decentralized and spread over many different sources. There are many independent video games news websites that report on the industry. Game developers officially report on Twitter and their official game websites, developers and gamers upload videos on YouTube, and developers and gamers post on Reddit. Even the news websites cannot keep track of the sheer number of games. The result is an increased burden on gamers, who must spend more time looking for information online.

To solve these problems, a way to aggregate, categorize, and rank game information is needed. The solution is GameEye, a platform where users can search for and follow video

games. Users will have a personal watchlist of video games. GameEye will keep track of followed games and notify users when new content is released online about them. GameEye will collect and categorize video game data from the internet, including news articles, tweets, Reddit posts, images, and YouTube videos. Using machine learning, news articles and tweets will be classified depending on their content, and a multi-factor impact score will be computed and assigned based on how impactful the news is for a game and its users. GameEye aims to be the gamer's eye in the video game industry.

2. GameEye Product Description

GameEye has five core functionalities. The first is the aggregation of online resources about video games, including news articles, tweets, Reddit posts, images, and videos. The second is the organization of online resources by game and type (e.g. grouping news articles, grouping tweets). The third functionality uses machine learning to classify and rank news articles and tweets based on their subject and their impact on the game and the user. The fourth is the notification of users about new content. The fifth is the ability for users to search for games to add to their watchlists, with support for autocompleting game titles.

The goals of the application are to make the lives of video game players easier, as well as to help game developers maintain more attention to their games. GameEye aims to be the middleman between the gamer and the video game industry, performing tasks and analysis that the gamer would otherwise have to do manually. The application seeks to automate the discovery of new video game information and the categorization and ranking of it to deliver relevant information to users.

2.1. Key Product Features and Capabilities

GameEye is superior to other video game aggregators, like N4G (2020), because it aggregates a larger variety of resources and displays information with finer granularity by default: per-game instead of per-genre or per-platform. Another main aspect in which GameEye is superior to other aggregation services, especially N4G, is that it uses automatic aggregation. This means that the content is not user-curated but is collected and processed automatically. Furthermore, the application uses notifications to inform users about new content. Other gaming aggregation services do not offer notifications for new content. GameEye also uses machine learning to classify information based on the subject and to rank it based on its impact on games and the users. These features are unique selling points (USPs) of the application. The innovation of GameEye lies in its finer-level automated aggregation and machine learning capabilities. Features are described in more detail in the next few paragraphs.

General: The application will be designed and built as a progressive web application. Progressive web applications are a new generation of web applications. Unlike traditional web applications, progressive ones can run offline and provide features while the user is offline. GameEye will be cross-platform, targeting desktops and mobile devices, and will include offline support, local caching, and connection interruption resiliency features.

Authentication: Using a secure third-party authentication provider, the application will provide secure login and registration for users and will support login and registration using external provider accounts, such as social media, further increasing accessibility to users. Persistent sessions so that users do not have to login every time, as well as two-factor authentication and password recovery mechanisms, are to be implemented.

Account Management: Standard account management functionality including the ability for users to change their passwords, modify their profile information, as well as the ability to delete their accounts is planned.

Game Tracking: Every user will have a personal watchlist of video games they want to stay updated about. Aggregation of game thumbnails will be implemented to also add a visual cue about games next to their titles. Aggregated information will be organized by news articles, tweets, Reddit posts, images, and videos. Thumbnail images will be included for news articles and tweets. Finally, a list of the most-watched games on the platform will be implemented so that users can discover promising games.

Searching: Using data from the IGDB (Twitch, 2020) video game database, the application will provide the ability to search for video games based on their title. Support for autocompletion of game titles will also be implemented to enhance user experience.

Web Scraping: The application will automatically aggregate new information about video games watched by users at regular time intervals. Aggregated information includes news articles from video game news websites, tweets from official game Twitter feeds, Reddit posts from official game subreddits, videos from official game YouTube channels, and images. New game information is kept for 90 days and information older than 90 days will be archived but will still be accessible to users.

Machine Learning: The application will use machine learning models and natural language processing to classify news articles and tweets based on what they are about (e.g. release date announcement, major game update, minor game update). An impact score will also be computed. This score is based on the impact of the news that an article reports on its reference

game or its players. Lastly, the application will extract important information, such as version numbers and dates, from news articles and tweets based on their classification.

Notifications: The application will send out cross-platform push-notifications to users for new game updates. A UI count of the notifications for each game and each resource category will also be displayed. Notifications about suggested video games based on the most-watched games list are also planned.

Settings: The application will be customizable in that users will be able to choose if they want to see archived resources, if they want to see impact scores, and if they want to receive notifications. Users will also be able to select which resource categories to be notified about.

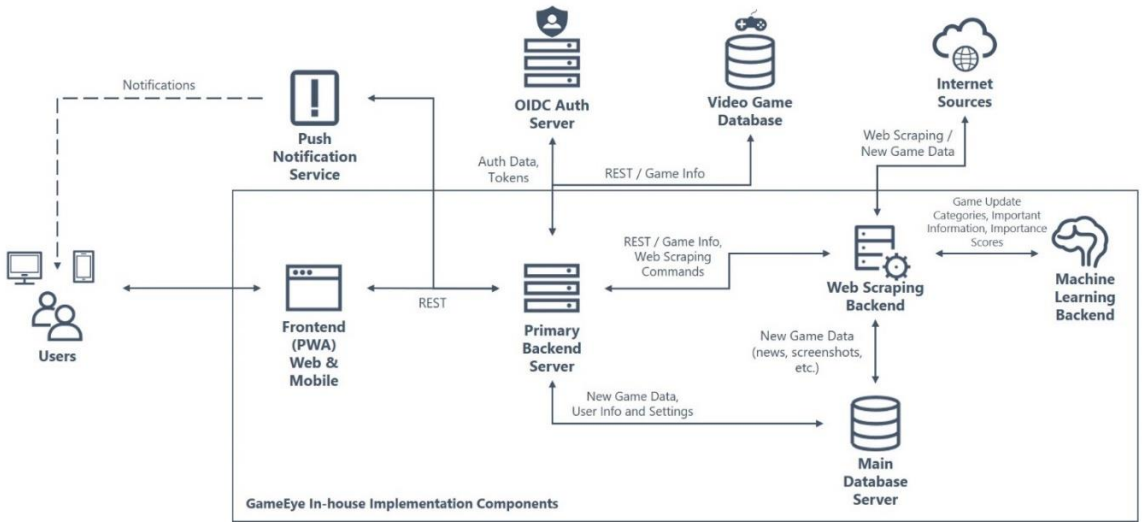
GameEye seeks to make it easy for gamers to stay up to date about their favorite games. The automated web scraping of the application, as well as its machine learning capabilities, seek to minimize human intervention and save time. GameEye will scrape internet sources at regular time intervals, analyze the scraped data using machine learning, and notify users about what it retrieved. A direct result is that users will get all the information delivered directly to them.

2.2. Major Components (Hardware/Software)

GameEye consists of several in-house and third-party components. The in-house components are: (**a**) frontend, (**b**) primary backend, (**c**) web scraping backend, (**d**) machine learning backend, and (**e**) main database server. The third-party components are: (**a**) IGDB, a video game database (Twitch, 2020), (**b**) Firebase Authentication, a Google authentication service (Google Inc., 2020), and (**c**) Firebase Cloud Messaging, a push notification service (Google Inc., 2020). These components are shown in Figure 2.

Figure 2

Shows the major functional components of GameEye and their interaction.



The first component is the frontend**.** The frontend is a progressive web application (Google Inc., 2020) built with Angular (Google Inc., 2020), Workbox (Google Inc., 2020), HTML, CSS, and TypeScript. It will serve as the UI for GameEye. As a progressive web application, it will be cross-platform and will target desktop and mobile devices. The frontend will be hosted on an external-facing web server.

The second component is the primary backend. This is the backend that communicates with the frontend, powers its dynamic functionality, and coordinates the web scraping backend. It is built with the Spring framework (VMware Inc., 2020) and Java. It is an external-facing web server and provides a REST API for communication with the frontend, web scraping backend, databases, and other third-party services.

The third component is the web scraping backend. This is the backend that performs web scraping. It communicates with the machine learning backend to classify scraped video game data and extract important information before putting it in the database. It is built with Spring

framework, jsoup (Hedley, 2020) for web scraping, and Java. It is an external-facing web server exposing a REST API for launching web scraping operations.

The fourth component is the machine learning backend. It communicates with the web scraping backend to classify video game data and extract important information. It is built with Python, Keras (Chollet et al., 2020), scikit-learn (Cournapeau et al., 2020), and spaCy (Explosion AI, 2020). It is an internal GPU-powered TensorFlow Serving (Google Inc., 2020) server exposing a REST API for machine learning inference.

The fifth component is the main database server. This includes an internal MongoDB (MongoDB Inc., 2020) database that stores all the data related to GameEye, including data retrieved from web scraping. Notably, it stores data retrieved from IGDB for increased performance and redundancy. It also includes an ElasticSearch (Elastic NV, 2020) database that stores game titles and allows for very fast and accurate autocompletion. The main database server is accessed by the main and web scraping backends.

The sixth component of the application is IGDB, a large video game database. IGDB is owned by Twitch (Humphries, 2019) and by extension Amazon. It contains a lot of information about video games, such as important URLs, platforms, genres, and descriptions. IGDB offers a REST API to access the content of the database. GameEye will use IGDB to receive video game information necessary for web scraping, such as game titles and URLs. Notably, IGDB permits creating a competing product, which essentially allows the replication of their database and the storage of the information in GameEye's main database, bypassing the need to use their API in the future.

The seventh component is Firebase Authentication, an authentication service offered through Google's Firebase platform. Firebase Authentication is a free cross-platform authentication service that offers security and relatively simple integration with client applications. Creating an authentication system from scratch is difficult and laborious, thus the incentive to utilize a tried and tested solution. Using Firebase Authentication, GameEye will allow users to login with an email and password, login with external providers like their Google accounts, as well as register for new accounts.

The eighth component is Firebase Cloud Messaging. Firebase Cloud Messaging is a cross-platform messaging solution developed by Google that can be used to send notifications. Like Firebase Authentication, Cloud Messaging is part of Google's Firebase platform. GameEye will use the service to send push-notifications to users. Specifically, the primary backend will be connecting to Cloud Messaging to create and send notifications.

The frontend, backends, and main database server ideally require dedicated hardware for optimal performance. Thus, the required hardware for the application are servers to host the in-house components. An alternative to dedicated hardware is Docker (Docker Inc., 2020), for running multiple components as containers on a single or a couple of high-performance servers.

3. Identification of Case Study

GameEye is being developed for casual and highly active gamers alike. Both types of end-users benefit from the application. Casual gamers who play a few games less frequently would value a plug-and-play solution where they can enter a few games that they are interested in and save time by getting notified about important news. Highly active gamers who frequently play and are interested in many games will also benefit from the application as they will not have to

manually keep track of so many games. The future users of the application are not expected to change without significant updates to the feature set of the application.

The case study for the application is the ODU Gaming Club. This user group is composed of ODU students who are interested in video games. These users may either be casual or highly active gamers and fit GameEye's target groups. ODU Gaming Club members make for a great group of testers who can provide feedback to improve GameEye. It is important that their insight is obtained during development, and thus are a valuable case study.

4. GameEye Product Prototype Description

The prototype will contain a small subset of features of the real-world prototype. It will be focused on functionality related to news articles. The biggest difference between the prototype and the real-world product will be the number of resources that are supported. The real-world product will support news articles, tweets, Reddit posts, images, and videos. The prototype, however, will only support news articles. Additionally, machine learning features in the prototype will be limited to impact scoring. Table 1 lists the features of the real-world product and the prototype in more detail.

Table 1

Lists the features that will be available in the real-world product and the prototype.

Category	Feature	RWP	Prototype
General			
	Cross-Platform Support (Desktop, Mobile)	Full Functionality	Full Functionality
	Offline Support	Full Functionality	No Functionality
	Local Caching	Full Functionality	No Functionality
	Connection Interruption Resiliency	Full Functionality	No Functionality
Authentication			

	User Login	Full Functionality	Full Functionality
	User Registration	Full Functionality	Full Functionality
	External Provider Login & Registration	Full Functionality	Full Functionality
	Persistent Sessions	Full Functionality	Full Functionality
	Password Recovery	Full Functionality	Full Functionality
Account Management			
	Change Password	Full Functionality	Full Functionality
	Delete Account	Full Functionality	No Functionality
Searching			
	Search for Games	Full Functionality	Partial Functionality
	Search Autocompletion	Full Functionality	Partial Functionality
Game Tracking			
	Add Games to Watchlist	Full Functionality	Partial Functionality
	News Articles (Web Scraping)	Full Functionality	Partial Functionality
	Tweets (Web Scraping)	Full Functionality	No Functionality
	Reddit Posts (Web Scraping)	Full Functionality	No Functionality
	Images (Web Scraping)	Full Functionality	No Functionality
	Videos (Web Scraping)	Full Functionality	No Functionality
	Resource Thumbnails	Full Functionality	Partial Functionality
	Game Thumbnails	Full Functionality	Partial Functionality
	Source Website Redirection	Full Functionality	Partial Functionality
	Resource Organization	Full Functionality	Full Functionality
	Archived Resources	Full Functionality	No Functionality
	Most-Watched Games List	Full Functionality	Full Functionality
Settings			
	Show Archived Resources Option	Full Functionality	Full Functionality
	Show Impact Scores Option	Full Functionality	Full Functionality
	Receive Notifications Option	Full Functionality	Full Functionality
	Receive Notifications Per Category Option	Full Functionality	Partial Functionality
	Submit Feedback	Full Functionality	No Functionality
Machine Learning			
	Impact Scoring	Full Functionality	Partial Functionality
	Resource Classification	Full Functionality	No Functionality
	Important Information Extraction	Full Functionality	No Functionality
Notifications			
	Push-Notifications for New Game Updates	Full Functionality	Full Functionality

	UI Count of Notifications for Each Game	Full Functionality	Full Functionality
	UI Count of Notifications for Each Resource Category	Full Functionality	Full Functionality
	Cross-Platform Notifications	Full Functionality	Full Functionality
	Suggested Video Game Notifications	Full Functionality	No Functionality

4.1. Prototype Architecture

The prototype will be using the same architecture as the real-world product. The primary difference is that all the in-house components will be hosted on one physical server, the virtual machine provided by the CS department, instead of dedicated ones. Furthermore, there will be no replicated servers and no load balancing, which would be present in the real-world product. The software components are: (a) frontend, (b) backend, (c) MongoDB database, and (d) ElasticSearch database. The backend also includes the web scraping and machine learning backends. The two databases are part of the main database server in the MFCD. The prototype MFCD is the same as the one in Figure 2.

4.2. Prototype Features and Capabilities

The prototype aims to demonstrate the real-world applications of GameEye. It primarily demonstrates how a user: (a) can access the platform, (b) add games to their watchlist, (c) see recent news articles for each game and, (d) see impact scores for each article, and, (d) receive notifications for news articles. Furthermore, the prototype demonstrates how GameEye functions internally, and how web scraping, machine learning, game title autocompletion, and user authentication and authorization are performed. Overall, the prototype seeks to show how users can save time by using the application to keep track of video game news.

From a functional standpoint, our goals are to: (a) have a functional authentication system, (b) have functional news article web scraping, (c) have functional watchlists and game tracking,

(d) have functional machine learning impact scoring, and (e) have functional news article notifications. Optimally, we aim for the prototype to be a usable product, for a limited number of games and for tracking news articles only.

We aim to demonstrate success by providing a high-quality prototype that demonstrates how GameEye works and how it can be extended to support new features. On the frontend side, we aim to deliver an intuitive and modern user interface using elements of Material design (Google Inc., 2020), with a simple user flow and a straightforward user experience. On the backend side, we aim to deliver high-performance logic, endpoint security, and reliability through robust exception handling and test-driven design.

We will take appropriate steps to mitigate the several identified risks. News article web scraping will be performed on RSS feeds, which rarely change. We will also replicate a portion of IGDB into our database, to increase redundancy, performance, and ensure that GameEye can work even if IGDB goes offline. To ensure user security, we will be using Firebase Authentication and several third-party authentication providers, like Google and Microsoft, for authenticating users.

4.3. Prototype Development Challenges

GameEye is a complex application and involves a lot of data. There are several challenges that we must overcome during prototype development. First, the team needs to learn new technologies and frameworks which are required for implementation. Second, the team needs to collect and label around 12,000 news article titles for training the machine learning model for impact scoring. Third, we need to implement robust web scraping that is easy to modify in case source websites change. Fourth, as we deal with a lot of game data, we need to devise a database schema that minimizes data duplication. Fifth, we need to replicate a portion of the IGDB database, which is

challenging as the REST API is extensive and the database contains a large amount of data. Sixth, because GameEye is a cross-platform application, we need to ensure that it works in a variety of platforms and screen sizes. Seventh, we are dealing with a multiplex machine learning problem that involves analyzing the text of news article titles to deduce the impact of the news they report. The combinatoric space of possible words is large and a great amount of training data is needed to achieve high accuracy. It is not possible to collect and label such a large amount of data for the prototype, so the accuracy of the machine learning model cannot be guaranteed.

5. Glossary

Aggregator: A platform or service that collects and centralizes data.

Angular Framework: Platform for building mobile and desktop applications.

API: Application Programming Interface; a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other services.

Amazon Web Services (AWS): Amazon subsidiary that provides on-demand cloud computing services and APIs.

CSS: Cascading Style Sheets; used to stylize webpages.

Guest: Initial role for users who have not created an account on GameEye.

Hitlist: List of highly watched video games by users.

HTML: HyperText Markup Language; used as markup for documents meant to be displayed in a web browser.

Impact Score: A score from 1 to 3 on the impact some news has on a game and its players. It is computed using machine learning.

Internet Games Database (IGDB): Database of known video games, accessed through a REST API to populate GameEye's database.

Indie Games: Games developed by individuals or smaller teams of people without the financial support of larger game publishers.

IDE: Integrated Development Environment. A software application that includes a set of programming tools, notably a code editor and a debugger interface, to facilitate programming.

IntelliJ IDEA: IDE developed by JetBrains to write Java applications and will be used in the backend development of GameEye.

JavaScript: Object-oriented programming language used to create dynamic behavior on webpages.

JSoup Library: Java library for working with real-world HTML.

JUnit Java Framework: A testing framework for Java.

Keras: An open-source neural-network library written in Python.

MongoDB: A cross-platform document-oriented database program.

Noise Filtering: Removal of news articles and other content that is irrelevant or unimportant to the user.

OIDC Authentication: Open-ID Connect (OIDC) is an authentication protocol based on the OAuth 2.0 family of specifications.

Progressive Web Application (PWA): A type of application software delivered through the web which is built using common web technologies including HTML, CSS, and JavaScript. PWAs are an evolution of traditional web applications and can be used to a certain degree while offline.

Python: An interpreted, high-level, general-purpose programming language.

Representational State Transfer (REST): A software architectural style used in creating web services.

RSS Feed: Really Simple Syndication (RSS) is a web feed that allows users and applications to access updates to websites in a standardized, computer-readable format.

Scikit-learn: Open-source machine learning library for the Python programming language.

SpaCy: Open-source library for advanced natural language processing.

Spring Framework: Application framework and inversion of control container for the Java platform.

Tester: GameEye prototype users who will provide feedback on their experience with the application.

Web Scraping: Automated extraction of data from websites.

WebStorm: IDE developed by JetBrains for writing JavaScript and web-related code.

6. References

Anderton, K. (2019, June 26). *The Business of Video Games: Market Share for Gaming Platforms in 2019 [Infographic]*. Retrieved from: https://www.forbes.com/sites/kevinanderton/2019/06/26/the-business-of-video-games-market-share-for-gaming-platforms-in-2019-infographic/#c1793427b254

Dietz, J. (2011, June 23). *30 Games That Emerged from Development Hell*. Retrieved from: https://www.metacritic.com/feature/games-that-shed-vaporware-status

Gough, C. (2019, August 9*). Number of games released on Steam 2018*. Retrieved from: https://www.statista.com/statistics/552623/number-games-released-steam/

Gough, C. (2019, August 9). *Number of gamers worldwide 2021*. Retrieved from: https://www.statista.com/statistics/748044/number-video-gamers-world/

Gough, C. (2019, October 9). *Google Play: Number of available games by quarter 2019*. Retrieved from: https://www.statista.com/statistics/780229/number-of-available-gaming-apps-in-the-google-play-store-quarter/

Humphries, M. (2019, September 18). *Twitch Acquires Gaming Database Website IGDB.* Retrieved from: https://www.pcmag.com/news/twitch-acquires-gaming-database-website-igdb

Rose, M. (2014, May 15). *How the surge of Steam releases will affect game developers*. Retrieved from: https://www.gamasutra.com/view/news/217583/How_the_surge_of_Steam_releases_will_affect_game_developers.php