HW Submission

David Greene

10/06/2023

Memory Management for Draw It or Lose It

To effectively manage memory in the software application Draw It or Lose It, there are several considerations and specific approaches you can take:

* Optimize image loading: Since the game requires rendering and displaying high-definition image files, it's important to optimize the loading process. Consider implementing techniques such as lazy loading, where images are loaded only when they are needed, to reduce memory usage. Some leading vendors for lazy loading include:
  + Google Picasso: A popular image loading library for Android that provides features such as lazy loading, caching, and image transformations.
  + Glide: Another popular image loading library for Android that offers similar features to Picasso.
  + ImageLoader: A powerful image loading library for Android that supports loading images from various sources, including local storage, remote URLs, and content providers.
* Efficient memory allocation: Allocate memory resources dynamically and release them when they are no longer needed. This can be achieved by using appropriate data structures and algorithms to manage memory efficiently. Some leading vendors for memory management tools include:
  + JetBrains Memory Profiler: A powerful memory profiling tool that can help you identify memory leaks and other memory usage inefficiencies.
  + YourKit Java Profiler: Another popular memory profiling tool that offers a variety of features to help you optimize memory usage in your Java applications.
  + MAT (Memory Analyzer Tool): A free and open-source memory profiling tool from Eclipse that can be used to analyze memory dumps from Java applications.
* Caching: Implement a caching mechanism to store frequently used images in memory. This can help reduce the need for repeated loading and improve the overall performance of the game. Some leading vendors for caching solutions include:
  + Google Guava Cache: A popular caching library for Java that provides features such as expiration policies, eviction strategies, and concurrency control.
  + Caffeine: A high-performance caching library for Java that offers features such as asynchronous caching, eviction policies, and tiered caching.
  + EhCache: Another popular caching library for Java that provides a variety of features to help you improve the performance of your applications.
* Memory profiling and optimization: Regularly profile the memory usage of the application to identify any memory leaks or inefficient memory usage. Optimize the code and data structures to minimize memory footprint and improve performance.

**Storage Management for Draw It or Lose It**

To determine how much storage is needed and manage storage effectively for the game application, we need to consider the following approaches:

* Estimate file size: Calculate the total size of the image library by multiplying the number of images (200) with the average file size (8 megabytes). This will give you an estimate of the storage requirements for the image files.
* Compression techniques: Explore image compression techniques to reduce the file size without significant loss in quality. This can help optimize storage usage and reduce the overall storage requirements for the game. Some leading vendors for image compression tools include:
  + TinyPNG: A free and easy-to-use online tool for compressing PNG images.
  + ImageOptim: A free and open-source image optimization tool for macOS.
  + OptiPNG: A free and open-source image optimization tool for Linux and Windows.
* Database management: Consider using a database to store and manage the image files. This can provide efficient storage and retrieval mechanisms, as well as support for indexing and searching capabilities. Some leading vendors for database management systems include:
  + MySQL: A popular open-source relational database management system (RDBMS).
  + PostgreSQL: Another popular open-source RDBMS.
  + Microsoft SQL Server: A commercial RDBMS from Microsoft.
* Cloud storage: Evaluate the option of using cloud storage services to store the image library. This can provide scalability, reliability, and easy access to the files across different platforms. Some leading vendors for cloud storage services include:
  + Amazon Web Services (AWS) S3: A scalable and reliable cloud storage service from AWS.
  + Microsoft Azure Blob Storage: A cloud storage service from Microsoft that offers similar features to AWS S3.
  + Google Cloud Storage: A cloud storage service from Google that offers a variety of features, including object storage, file storage, and archival storage.

Difference Between Memory and Storage Management in Terms of Game Application Functionality

Memory management primarily focuses on the efficient utilization of resources required for transferring files and ensuring the speed and performance of the system. It involves techniques such as memory allocation, caching, and optimization to enhance the user experience and ensure rapid and effective execution of the game.

On the other hand, storage management deals with the storage of files and permanent discs. It involves determining the amount of storage needed, implementing compression techniques, choosing appropriate storage solutions (such as databases or cloud storage), and managing the storage resources efficiently.

References :

For memory management:

Google Picasso. (n.d.). Retrieved from https://square.github.io/picasso/

Glide. (n.d.). Retrieved from https://bumptech.github.io/glide/

ImageLoader. (n.d.). Retrieved from https://github.com/nostra13/Android-Universal-Image-Loader

JetBrains Memory Profiler. (n.d.). Retrieved from https://www.jetbrains.com/profiler/

YourKit Java Profiler. (n.d.). Retrieved from https://www.yourkit.com/java/profiler/

MAT (Memory Analyzer Tool). (n.d.). Retrieved from <https://www.eclipse.org/mat/>

For storage management:

TinyPNG. (n.d.). Retrieved from https://tinypng.com/

ImageOptim. (n.d.). Retrieved from https://github.com/ImageOptim/ImageOptim

OptiPNG. (n.d.). Retrieved from http://optipng.sourceforge.net/

MySQL. (n.d.). Retrieved from https://www.mysql.com/

PostgreSQL. (n.d.). Retrieved from https://www.postgresql.org/

Microsoft SQL Server. (n.d.). Retrieved from https://www.microsoft.com/en-us/sql-server/

Amazon Web Services (AWS) S3. (n.d.). Retrieved from https://aws.amazon.com/s3/

Microsoft Azure Blob Storage. (n.d.). Retrieved from https://azure.microsoft.com/en-us/services/storage/blobs/

Google Cloud Storage. (n.d.). Retrieved from https://cloud.google.com/storage/