# 3D scene resource management algorithm

2018/7/19

# Problem

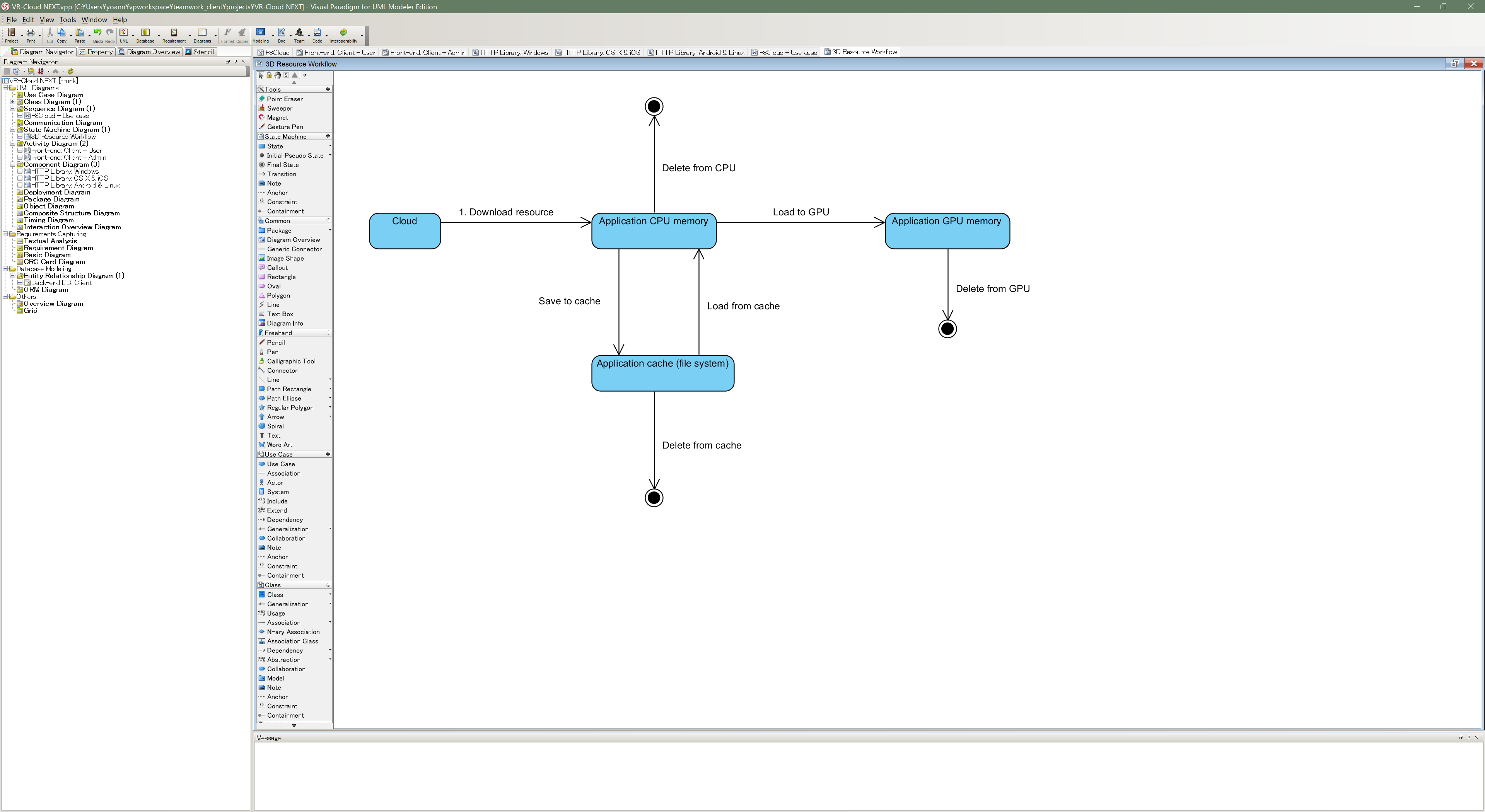
Develop an algorithm that optimizes the number of times an object of a 3D scene must be downloaded, cached and actually loaded in the CPU memory and GPU memory in the following system architecture.

# System architecture

The system includes:

1. A cloud where the original data is stored.
2. The application memory space (Application CPU memory) where the data must be loaded in order to carry out further actions.
3. The application GPU memory where the data must be loaded before being displayed on the screen
4. The application cache storage where the data can be temporary saved in order to release some CPU memory.

In the current problem we consider that the data is never deleted from the cloud but that is must be deleted from other areas in order to not overload the device physical resources.



# Data architecture

The data structure includes two components:

1. Object model: in reality it can be one or several files representing 1 object that must be treated by the application. In the following we just consider that one object model is any binary resource of any size. Each object model is independent from other object models.
2. Object instance: The object instance component defines the relationship between a 3D space and one object model. One object instance stores 3D coordinates that defines a position and a reference to one object model. Several object instances can refer to the same object model and several object instances could have the same 3D coordinates even if unlikely in a real situation.

The algorithm should be abstract enough to not be dependent on any data format but for debug and test purposes we suggest to use something similar the following format:

**Object model:**

ID: Unique integer number

Size: Size in byte

Radius: Radius of the circumscribed sphere of the model

Version: Integer representing the version of the model

Name: String

**Object instance:**

ID: Unique integer number

Model Ref: Integer that is the unique ID representing the object model

Position: array of 3 doubles representing the X,Y,Z coordinates in any Cartesian coordinate system.

Model radius: Radius of the circumscribed sphere of the model

Model version: Version number of the referred model

Name: String

# What the algorithm must do

The application is displaying a 3D representation of the object models where they are referred by the model instances. Without any optimization the application would load all the instances from the cloud then all the referred model and load every model data to the GPU in order to display the 3D space.

However the download time of every model from the start can be important, unnecessary downloads can occur when the application is restarted. The CPU and/or GPU memory capacity may not be enough to load all the data.

To solve these problems we want to develop and algorithm that take in account the current view point that is displayed by the application and decides when to download, save/load/delete cache, load to/delete from GPU or CPU in an optimal manner.

During the development of the algorithm we can consider that all the object instances are already loaded in the CPU memory and can stay in the CPU memory. The algorithm should just work on downloading/loading/unloading objects models.

Principle to take in account:

1. Download data from cloud only when strictly necessary
2. Use cache when no more CPU memory is available. (the available memory amount should be a parameter of the algorithm)
3. Don’t load more data to the GPU than what the GPU can store (the available memory amount should be a parameter of the algorithm), prioritize object that take an important part in the scene from the current application point of view.
4. Take in account save to cache times, load from cache time, load to GPU time and try to avoid unnecessary operations.

The algorithm should not know what type of 3D scene it is dealing with and always respects the principles described above. If depending on the type of 3D scene some strategies are clearly better than others, we can in the future consider having different implementations. For the first version the algorithm should be as general as possible and be tested with the two test cases described below.

# Main test cases

1. The application navigates at street level (simulation of a vehicle or pedestrian) in a 3D representation of a city where one model represent (A) a building used by one or only few object instances (B) road equipment/furniture (signals, signs, bench, bus stops etc..) each of them referred in many object instances in the scene.
2. The application navigates in a building or facility in all 3D directions in a random manner. (A) One model represent the facility and is used by only one instance (B) other models can be equipment/furniture each of them referred in many object instances.