Software Requirements Specification

for

MindFloat

(A mobile AR tool to create and organize UML designs in a shared virtual/AR space)

Version 1.0.0

Prepared by Shubhadeep Sarkar

Heritage Institute of Technology, Kolkata

23/01/2023

Software Requirements Specification

for

MindFloat

(A mobile AR tool to create and organize UML designs in a shared virtual/AR space)

Version 1.0.0

Prepared by Shubhadeep Sarkar
2051268

Heritage Institute of Technology, Kolkata

23/01/2023

Revision History			
Date	Revision Description	Approval	Version
23/01/2023	First Draft		1.0.0

1. Introduction

1.1 Purpose

During brainstorming, a person quickly comes up with many ideas, mostly loosely related. Keeping track of all ideas and having a global perspective is very important. This is the purpose of MindFloat, by organizing ideas as text-blocks, their interconnections (by making a graph) and providing a complete-coverage view using augmented reality.

A lot of design can't be cramped up in a scene (or user screen); MindFloat uses the spatial awareness of the user, about its immediate environment, and maps/arranges the diagrams/text fields into it. Thus, the global view required during planning, is achieved by spatially organizing the 'ideas' as simple text fields, displayed in an augmented-reality space around the user. Arrangement is dynamic and user should have high fidelity control. This is the prime purpose of MindFloat, to generate a 'walk-around' AR UML diagram/graph.

The graph thus created will be easy to manipulate and move around. Ease of constructing and changing the mind-map will be there. Also, some algorithms to generate optimized view of the user-provided graph should be implemented. This will help when building large designs and projects.

Any planning or brainstorming is a teamwork. MindFloat should be able to share the same design/idea-graph to multiple users. This should be dynamic (AR) and in Realtime.

Overall, MindFloat will sever as a platform to help create large designs, provide a compact but global view of the design, collaborate with other users and help optimize the design actively.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement SpecificationDocuments.
SRS - Software Requirement Specifications
MindFloat will also be referred to as 'app' in the document.

1.3 Intended Audience and Reading Suggestions

Intended Audience:

- Engineers: MindFloat can be used by engineers in fields such as software development, mechanical engineering, and electrical engineering to organize and visualize their ideas for technical projects.
- Business professionals: MindFloat can be used by individuals in roles such as project managers, marketing strategists, and business analysts to organize and visualize their ideas for business projects and campaigns.
- Designers: MindFloat can be used by designers in fields such as architecture, graphic design, and product design to brainstorm, plan, and organize their designs.
- Educators: MindFloat can be used by educators and students to organize and visualize their ideas for research projects, class projects, and study materials.

- Entrepreneurs: MindFloat can be used by entrepreneurs and startup founders to organize and visualize their ideas for new business ventures.
- Researchers: MindFloat can be used by researchers in fields such as science and social science to organize and visualize their ideas for research projects.
- Personal users: MindFloat can be used by individuals to organize and visualize their personal ideas, such as home renovation plans, travel itineraries, and personal goals.

Reading Suggestions:

- General users should read the introduction to better understand the goals of the app and the intended audience use cases. This way proper use and feedback can be achieved. Also skimming through the flow models will be beneficial for practical use.
- Developers should read beginning with the overview sections and proceeding through the overall description, system features, nonfunctional features, with spatial stress on the System Features.

1.4 Product Scope

MindFloat is meant for highly mobile device with handheld screen and decent AR supporting sensors. Thus, general smartphone users fall in the scope of the app. The data held can be stored and factored from the web, thus diagrams/graphs displayed can be huge in size with almost no limitations. The AR space can be shared with at most 7 dynamic users (explained further in Design and Implementation Constraints), thus large teams are not focused use cases.

2. Overall Description

Product Perspective

The application is to used primarily by designs, both technical and other wise. The user is mainly knowledgeable of design paradigms. The application should feel more like a tool, though immersive.

User Classes and Characteristics

The User Classes can be broadly divided into creator, manager and guest viewers. The grapher is the user class to design a diagram from scratch. The architect designs the dynamic constraints and the designer sets the visual aspects of the graph.
Guests and desktop-viewer can be another class which can have access to a desktop interface.

3. External Interface Requirements

User Interfaces

The focus group of the application is android mobile devices. Touch screen and friendly UI is to be designed upon. More to it gesture control and basic button such as volume buttons will help.

Hardware Interfaces

A hardware interface requirement would include the necessary specifications for the device it will run on. This may include minimum processing power, RAM, storage, and screen resolution. Additionally, the app may require specific hardware components such as a gyroscope, accelerometer, or a compass to accurately track the device's orientation and movements. The device may also need a camera to display

the AR content over the real world. The app's performance may also be impacted by the quality and capabilities of the camera, such as image resolution and image stabilization features. In summary, the hardware interface requirements for an AR app are essential to ensure the app runs smoothly and provides a high-quality AR experience for users.

Software Interfaces

A software interface requires a graphics library, AR engine, or AR development platform to support AR content creation and rendering. The app may also need to integrate with APIs from other software and hardware components, such as the device's camera or motion sensors. Additionally, the app may need to support features such as user accounts, data storage, and user preferences. The software interface requirements for an AR app are important to ensure the app integrates seamlessly with the device and provides a reliable, seamless AR experience for users.

Communications Interfaces

Simple cloud-based database and state save feature. Chat capability and voice over API.

4. System Features

1. Generate View

4.1.1 Description and Priority

The View is generated and made ready for the system to display. Either a previous view is unpacked or parameters for a new view is determined from scratch. A new view may have a dummy diagram. The main purpose is to present a simple view generation for the user and at the backend unpack the view-data properly into the memory. This also entails fetching and creation from/into database.

Priority MEDIUM (4,3,6,4): While loading large views, feature should be efficient enough on limited system requirements.

4.1.2 Stimulus/Response Sequence

4.1.2.1 Stimulus: User chooses to generate new view or load a previous one.
Response: Redirects to Create View (4.2) or Load View (4.3).

4.1.2.2 Stimulus: User local position changes (wrt. its environment)
Response: AR displayed View is updated.

4.1.2.3 Stimulus: User turns the mobile horizontal.
Response: Redirect to View Snapshot.

4.1.2.4 Stimulus: Touch gestures.
Response: Feedforward to AR engine and update View AR display.

4.1.2.5 Stimulus: Click on “change view perspective” button (various type of views/users)
Response: Load respective on-display buttons for appropriate user group perspective (architect/grapher/designer).

4.1.2.6 Stimulus: Click on “Floating Settings” button.
Response: View the transparent-layout settings list; change and implement display settings.

4.1.2.7 Stimulus: Click on “freeze” button (turn ON).
Response: Freeze the AR display, save AR system and user environment dynamic parameters.

4.1.2.8 Stimulus: Click on “freeze” button (turn OFF).
Response: fetch new AR display from engine. Unload parameters of AR system and

user environment and resume AR display. Show render-wait (blur display) while system may be unresponsive or calibrating and erratic.

4.1.2.9 Stimulus: Interaction with the AR-diagram.
Response: If required, “floating-button” and their windows can be directed to be passed to this activity to display.

4.2 Create View

4.1.1 Description and Priority

A new View’s parameters are determined. Various details are user provided; some default/estimation to parameters is provided. Space-Augmentation is estimated either by user input or Space Aug Estimator (4.4). The view is compiled into the View object and ready for deployment. This is mainly a passive feature.

Priority LOW (2,2,3,1): View compilation and data structure used should be efficient.

4.1.2 Stimulus/Response Sequences

4.2.2.1 Stimulus: Focuses on a parameter.
Response: show a brief on the parameter and show more button. Show default/computed value for the parameter in the input UI field.

4.2.2.2 Stimulus: Clicks on “Compile” to compile the view.
Response: In backend, process and test the view requested. Show the resulting parameters (size etc. functional and non-functional specs of the view to be generated).

4.2.2.3 Stimulus: Click on “Save and continue”
Response: Save the view data-structure to file and forward the activity control.

4.2.2.4 Stimulus: Click on “Share”.
Response: launch Intent on the view data-structure.

4.3 Load View

4.3.1 Description and Priority

Shows options to load a view from. Local file system, web-database, file-intent or from MindFloat share space. Also checks if Space Augmentation should be regenerated or fetched from Space Aug Estimator (4.4), redirects respectively.

Priority LOW (2,3,2,2).

4.3.2 Stimulus/Response Sequences

4.3.2.1 Stimulus: Enters the option for view source.
Response: Launch appropriate intent and fetch file.

4.3.2.2 Stimulus: Enters MindFloat share space view code.
Response: Fetches MindFloat web database for requested view. If view exists load it or show failure acknowledge.

4.3.2.3 Stimulus: Clicks on confirm source.
Response: Show details and user-specific permissions.

4.3.2.4 Stimulus: Clicks done.
Response: load view data-structure into memory and forward activity control.

4.4 Space-Aug Estimator

4.1.1 Description and Priority

Generates AR-mapping for later use. Scans the immediate environment of user. Estimates plane/wall anchors and tracking points, light estimation. Sets bounds for the view or the AR-diagram. If a view is loaded with different environment, it tries to match the environments for easy deployment. The spatial mapping is shown to the user and confirmed.

The main goal is to provide a well-defined AR-space to the view and help in mapping static (load view of different env) or dynamic mapping over AR-sharing.

Priority HIGH (7,6,9,6): CPU intensive work and data-sharing over the internet/cloud must be efficient.

4.4.2 Stimulus/Response Sequences

4.4.2.1 Stimulus: The User moves around the environment.
Response: AR-engine registers and shows the plane tracked and anchor points being recognized.

4.4.2.2 Stimulus: The User selects a plane and marks it as bound types (wall/ceiling/vertical/horizontal).
Response: A floating button comes up registering the user input.

4.4.2.3 Stimulus: The User finalizes the scoping.
Response: AR-engine produces the final bounds and renders its estimate, given metric data of the bounds.

4.4.2.4 Stimulus: The final bounds are tweaked. Lighting is adjusted.
Response: AR-engine re-compiles the change and shows the result.

4.4.2.5 Stimulus: If any active mapping is required, diagram orientation is given by prompt.
Response: Data is registered.

4.4.2.6 Stimulus: The user finalizes the env model.
Response: The AR-env is packed into a preferred data-structure and published in the local memory/cloud. Activity control is forwarded.

4.5 View Snapshot

4.5.1 Description and Priority

This feature provides user with a quick 3D summary of the diagram in the generated view. It is visually minimal (as in color space, object geometry). The Space-Aug provides with the env minimal structure model. The 3D model is to be bird's eye view of the room. This is rather static representation. It is imperative that the model is smooth to pan/zoom/navigate.

This also readies any data required for quick export.

Priority MEDIUM (6,4,5,3).

4.5.2 Stimulus/Response Sequences

4.5.2.1 Stimulus: The user rotates the device horizontal.
Response: Activity is started with bird's eye view then slow animation to a 3D model.

4.5.2.2 Stimulus: The user navigated around the 3D model (simple pan zoom etc.)
Response: The model is rendered appropriately.

4.5.2.3 Stimulus: The user clicks on settings.
Response: The activity is forwarded to general settings/snapshots.

4.5.2.4 Stimulus: The user clicks on export.
Response: Provides choice for quick export or export. Redirects accordingly.

4.5.2.5 Stimulus: The user rotates back the device vertically.
Response: Activity is subsumed. The model is destroyed if efficient.

4.6 Export Snapshot

4.6.1 Description and Priority

Exports the snapshot as image or as a 3D model. The current camera pose (orientation and location) is stored in the export. Images are generated by occluding. Small amount of settings/parameter tweaks can be presented to the user.

Priority LOW (3,2,6,3): image or 3d object generation should be decoupled and activity should be as intermittent and low on memory as possible.

4.6.2 Stimulus/Response Sequences

4.6.2.1 Stimulus: The user confirms the snapshot.
Response: Generate the image (default export type) and save to file. Forward control.

4.6.2.2 Stimulus: The user crops or resizes the diagram. Changes verbosity. Changes color space.
Response: Register parameter change. Update changes.

4.6.2.3 Stimulus: The user clicks setting.
Response: redirect to settings activity.

4.6.2.4 Stimulus: The user selects export type.
Response: Drop down list used to register.

4.7 Create T

4.7.1 Description and Priority

Create an instance of diagram basic building block in MindFloat, called T or “thought bubble”. Feature should be clutter free and easy UX. Generates the code for diagram. Updates the diagram display. After generation the T is on focused mode but with minimal floating attribute settings.

For minimal UI trigger will be hand gestures. Control will be forwarded from generate view activity. High fidelity of AR diagram surfaces required.

4.7.2 Stimulus/Response Sequences

4.7.2.1 Stimulus: User double clicks in hand gesture pointing into the AR view
Response: The background diagrams blur a little and a new bubble is generated in the diagram with text field selected for input. The year display will freeze and keypad will be enabled for input.

4.7.2.2 Stimulus: User confirms the basic AR text field.
Response: The bubble is generated the background diagrams are resume to be rendered the transition from typing to AR display will be blurred and smooth. Small floating attribute buttons will be enabled. Tea object will be on Focus mode.

4.7.2.3 Stimulus: User swipes out T creation. Swipe will be hand gesture also.
Response: The intermittent t block created will fade away and vanish.

4.8 Move T

4.8.1 Description and Priority

This feature helps to move a tea block in the diagram with easy UX. Movement is hand gesture as well as device pose controlled. Various bound markings and adjacent block associations can be highlighted. In case of conflicting spatial obstruction with a block or edge generation Huawei is updated to shown error and revert option is also provided. To keep the functionalities mode and highly reactive no optimization or heavy computation is done.

4.8.2 Stimulus/Response Sequences

4.8.2.1 Stimulus: From focus T mode the user selects to move the block.
Response: All the floating attributes are hidden and movement guidelines and margins are displayed. If the user is new to move a block, hand gestures proxies are shown

4.8.2.2 Stimulus: The user moves a block using hand gestures.
Response: The view diagram is updated. Respective margins are highlighted / shown on to the diagram.

4.8.2.3 Stimulus: The user focuses out of the block.
Response: The block position is fixed and control is forwarded to generate view.

4.9 Focus T

4.9.1 Description and Priority

This is a basic function same as UI elements are focused. Focusing a tea should enable user to add attributes to it and manipulate small spatial movement of itself and edges. More attribute settings can be accessed by floating buttons. This feature should also contain the logic to transfer button generation to system UI and not just AR.

4.9.2 Stimulus/Response Sequences

4.9.2.1 Stimulus: The user focuses a T by hand gesture.
Response: The T block is highlighted the diagrams behind are a bit blurred. All dynamic attributes are compactly shown and floating attributes buttons are shown. The edges from an into the block are highlighted as well.

4.9.2.2 Stimulus: The user clicks/swipes on static attributes.
Response: A list of attributes is called scrolled and shown dynamic parameters are also shown minimally.

4.9.2.3 Stimulus: The user clicks on dynamic attributes.
Response: The dynamic attributes are displayed minimally and the whole background diagram is overlayed with the dynamic relation. The user can lock onto a specific dynamic attribute using a floating button

4.9.2.4 Stimulus: The user focuses / locks on a specific dynamic attribute.
Response: The whole diagram is said to a pseudo freeze mode and only the blocks and edges interrelated with the dynamic attribute are shown. A separate floating window accommodated somewhere over or beside the whole diagram is also shown / generated.

4.9.2.5 Stimulus: The user clicks on move floating button
Response: The T block is set to move mode.

4.10 Floating Attributes

4.10.1 Description and Priority

This is a general feature of displaying attributes, buttons, small markers and simple text fields in the AR diagram. Generation of such floating window should be cohesive with the associated block or adjacent blocks. The UI is simple and non-blocking in nature.

Importantly the display of the floating window should be dynamic to the camera / user pose and some degree of static orientation with respect to the user environment.

4.10.2 Stimulus/Response Sequences

Stimulus and responses are specific to the context of generation. Behavior is similar to List UI element.

4.11 AUTO Spatial Optimizer

4.11.1 Description and Priority

One of the most important features of mind float is to organize large URL diagrams into the most logical and reader friendly format. This optimizer uses in algorithm to rearrange spatial the blocks and its edges.

This will be different to semantic rearrangement and can be applied by the user directly after graph generation. The use of user environmental model and the size attributes of the block is essential to this feature.

The rearrangement will be incremental in nature and can be applied to a linear degree by the user. The user can anchor any element to an arrangement and apply further the optimization.

Priority HIGH (9, 7, 10, 7): optimization algorithm is the crux. As well as dealing with manifold dynamic diagram generation must be very robust.

4.11.2 Stimulus/Response Sequences

4.11.2.1 Stimulus: The user clicks for auto spatial optimization from quick settings in the activity generate view.
Response: The AR diagram lightly freezes and environment models is highlighted. Intermittent grouping an optimization models can be shown but minimally on do the diagram. The user controls of focusing of selecting anything in the AR diagram is frozen.

4.11.2.2 Stimulus: Optimizer engine completes/generates rearrangement manifold.
Response: The final rearrangement is directly shown and a linear slider is used to input the degree of rearrangement.

4.11.2.3 Stimulus: The user changes the rearrangement slider.
Response: The diagram is smoothly shifted along the rearrangement manifold all attribute generation is frozen. Small details / properties of the current rearrangement is briefly shown.

4.11.2.4 Stimulus: The user clicks on a diagram element to anchor it.
Response: The anchor button is shown on the diagram block / element. Orthogonal spatial bound matching is briefly shown. For a small time, the block anchored can be moved slightly in the diagram. The manifold is recalibrated in the backend.

4.11.2.5 Stimulus: The user clicks on finalize button.
Response: The spatial optimizer displays a summary in an intermittent floating window. Generate view activity is resumed.

4.12 Generate Dynamic Snapshot

4.12.1 Description and Priority

This feature allows you to make dynamic 3D models of the diagram. This is meant to be a heavy task does should also be supported by a cloud engine. The basics snapshot of a diagram is to be exported as a 3D model. Also, different groupings and attribute highlights can be adjusted to be shown when focused in the 3D model.

This feature will highly help any guest users to understand a specific development in the diagram. This basically is it summarization of dynamic attributes.

4.12.2 Stimulus/Response Sequences

4.12.2.1 Stimulus: The user clicks on generate dynamics snapshot.
Response: The dynamics snapshot page opens the diagram changes from Air view to 3D model with same camera pose perspective. The user can change the camera pose.

4.12.2.2 Stimulus: The user locks on the camera pose.
Response: The snapshot view is fixed on visible dynamic attributes and block grouping are listed with visibility toggles

4.12.2.3 Stimulus: The user toggles visibility of an attribute.
Response: Snapshot is shown to dynamically change.

4.12.2.4 Stimulus: The user exports and shares the snapshot
Response: A list of eligible user IDs to share to our shown which can be taken and marked okay. User can also download it locally on the machine.

4.12.2.5 Stimulus: The user of process confirms
Response: The snapshot is created. The dynamics snapshot page exits.

4.13 Static Optimizer

4.13.1 Description and Priority

Optimizer used for visual display and graphical parameters. This feature is primary for desk view use case. The user is provided with a comprehensive interface for changing the visual attributes such as size thickness shape adjacency behaviors. This feature is mainly important to rectify edge conflicts and group edges efficient.

The static changes created over web can be approved by the grapher or the underline super user of the graph.

4.13.2 Stimulus/Response Sequences

4.13.2.1 Stimulus: The desk view user selects optimizer static optimizer from the options
Response: The desk view changes into static optimizer view, mainly highlighting graphical attributes of all blocks and edges and providing quick tweak, floating windows beside each block

4.13.2.2 Stimulus: The user selects a block or clicks on weaking floating button beside a block
Response: A property viewer / editor pops up beside the screen. All effecting blocks and diagram elements a highlighted and optimization changes can be seen dynamically.

4.13.2.3 Stimulus: The user commits to a change in the property editor.
Response: The changes are updated in the diagram and the grapher and other users are intimidated with the change. If all superusers approve the change a message is shown.

4.14 User Orchestration

4.14.1 Description and Priority

This is a unified platform to manage all users to be involved in a MindFloat project. Each user is a signed a unique mind float user ID; each diagram in MindFloat has an associated dependency tree and unique ID. The association of a user towards project and its diagram can be made from here. Also, the different roles in the project can be assigned from here. The control of in app communication such as live text or life voice can be set.

4.14.2 Stimulus/Response Sequences

4.14.2.1 Stimulus: The user opens the user orchestration tab.
Response: The user orchestration page loads with components such as- list of user currently in the project search tab to enter user ID and associate other people with the project. The project dependency tree. The currently focused diagram user/security attributes.

4.14.2.2 Stimulus: The user searches for a user ID
Response: The page returns the user ID details and provides choice to associate with the project/diagram.

4.14.2.3 Stimulus: The user associates the users ID with the project.
Response: The database is updated the security details are re checked for classes and the project dependency tree is updated and displayed.

4.14.2.4 Stimulus: The user of clicks on a diagram in the project dependency tree.
Response: The attributes specific to the diagram are shown briefly and all the security / user details / dependencies are shown

4.15 User Rostering

4.15.1 Description and Priority

This is a project management feature to assign and visualize different privileges and clearances and associations related to users and their specific project diagrams. This management tool is important when a project is associated to more than a dozen users. Simple but effective representation of a job rostering is provided with specific emphasis on security privileges of users.

4.15.2 Stimulus/Response Sequences

4.15.2.1 Stimulus: The user clicks on user rostering from user orchestration view or roster users from user attributes
Response: The user orchestration page opens with focus on the specific user.

4.15.2.2 Stimulus: The user drags n drops user ID into the rostering list
Response: The rostering list updates and its details assume if there is a discrepancy the error message is shown and the user ID is shifted to the appropriate place.

4.15.2.3 Stimulus: The user clicks on the timing diagram of the project.
Response: Timing diagram for the whole project is generated with estimated time to completion of the project with already spent time on each diagram is displayed.

4.16 User Comm

4.16.1 Description and Priority

This feature registers in the communication to be accessible to various user of a common project/view diagram. In the back end it generates request to have live communication channels between users. This will directly affect in the network overhead of a diagram.

4.16.2 Stimulus/Response Sequences

4.16.2.1 Stimulus: The user clicks on user communication from user orchestration.
Response: A list of users are shown and the project dependency diagram tree is shown

4.16.2.2 Stimulus: The user clicks on a diagram from the dependency tree
Response: The user communication details associated with the diagram is shown the user can dragon draw user IDs into different communication buckets.

4.16.2.3 Stimulus: The user of clicks on a user ID
Response: The user communication details associated with the user ID is shown and privileges can be revoked or changed.

4.17 User Login

4.17.1 Description and Priority

Login feature for all users. General login mostly forwarded to Google, Google collab, GitHub, and other credential verification third party API.

The landing page of a user should contain a proper summary of the projects and diagrams currently authored by the user

4.17.2 Stimulus/Response Sequences

4.17.2.1 Stimulus: The user choses a third-party verification API to sign up or login
Response: The third-party API is called and redirected hence.

4.17.2.2 Stimulus: The third-party login API sends back acknowledgement.
Response: If the login fails repeat or load the landing page of the user

4.17.2.3 Stimulus: The user creates a new project from the login
Response: The cloud database is updated and a new project ID is provided to the user

4.17.2.4 Stimulus: The user clicks on the project ID
Response: Redirects to generate view

4.18 Billing

4.18.1 Description and Priority

A unified platform for all chargeable services with complete and final bill amount is shown. The user can directly select a third-party payment gateway to clear its payments.

4.18.2 Stimulus/Response Sequences

4.18.2.1 Stimulus: The user clicks billing tab from its landing page.
Response: The billing pages shown with all previous charges paid and to be paid. Due charges are highlighted and the total amount payable is displayed.

4.18.2.2 Stimulus: User clicks on payment history
Response: A list of previous payment details are shown.

4.18.2.3 Stimulus: User clicks on pay amount
Response: Redirects to third party payment gateway

4.18.2.4 Stimulus: Acknowledgement from payment gateway
Response: Payment is validated and database is updated.

4.19 Set Credentials

4.19.1 Description and Priority

All dependencies and privileges associated with the current user is displayed if any of these are self-manageable controls are adjacently displayed

4.19.2 Stimulus/Response Sequences

4.19.2.1 Stimulus: The user clicks on set credentials tab.
Response: The set credentials page is opened and all dependencies and privileges associated with the current user is displayed.

4.19.2.2 Stimulus: The user clicks on a specific privilege.
Response: The page redirects to teams page

4.19.2.3 Stimulus: The user of clicks on personal details. Updates fields.
Response: If the input is valid verify message is shown.

4.19.2.4 Stimulus: User it clicks on confirm.
Response: The database is updated

4.20 Teams

4.20.1 Description and Priority

The user rostering for various teams and projects are shown specific to the user.

4.20.2 Stimulus/Response Sequences

Stimulus and responses are specific to the context of generation. Behavior is similar to List UI element.

4.21 Set T Attributes

4.21.1 Description and Priority

Every tea block will have attributes associated with it. Attributes are specifically a characteristic vector which is automatically created and maintained by the system in a special attribute vectors space. Attributes are meant to embed semantic cohesiveness between t blocks. The user can set the attributes in a generalized way and this user input should be simple and natural to ingest. Converting such input into the higher dimensional attribute vector should be the task of the systems.

For now attributes can be set as a general text description of a tea block and slash or setting colours and uni grams for a tea block. The general text description is not stored face value rather a few keywords are selected by the system and attributed as such with the T block along with other adjectives that are built in the system to present textual description. Stimulus/Response Sequences

4.21.1.1 Stimulus: The user selects a tea and clicks on set attributes.
Response: The isolated g a f for the block subsumes and only hard coded attributes for the block are shown in the GAF. A textual input box is shown a scroll down unigram list is shown and the meta color palette is shown. All these can be clicked and changed thus.

4.21.1.2 Stimulus: The user scrolls and selects a unique Ram unigram
Response: The unigram gets and listed with the attributes any changes in the g a f is updated and shown

4.21.1.3 Stimulus: The user selects general textual input and speaks out or types in additional description.

Response: The text is process and keywords are shown in the same text box. The keywords are clickable and if the use of clicks them those keywords are removed from the general description. This generation of keyword is done by the system AI

4.21.1.4 Stimulus: The users clicks on a keyword in the generated textual attribute. Further clicks confirm
Response: The specific keyword is removed from the attribute box and the rest are added on to the existing list of general textual attribute. The the aggregate textual attributes are shown briefly. Any changes to the gas are color-coded respective to the changes in the textual attributes. This is shown briefly for revision of the changes

4.21.1.5 Stimulus: The user clicks on the colour palette
Response: The different colours in the colour palette describing different gas structures are shown in the minimal GAF. Each colour can be selected and a colour wheel pops up that can be rotated to change the colour and AI perceived gas changes are also shown.

4.21.1.6 Stimulus: The user clicks on a colour and changes it with the colour wheel and confirms by going back.
Response: The changes in the gas are shown and confirmed.

4.22 T Select

4.22.1 Description and Priority

Selecting t shows a subgraph of of the gas that has or is directly connected to the T block and its attributes. This is different from select the attributes as it may contain disconnected subgraphs. Select t should be able to show the architectural class of similar to the T block selected.

This is a prime feature of mind float and is realized in large interconnected diagrams where meaning of different subsystems in the diagram can be easily trays doubt and general structures and correlations between attributes can be visualized. The system AI behind this is one of the crux business logics for MindFloat.

4.22.2 Stimulus/Response Sequences

4.22.2.1 Stimulus: The user hovers upon a T block and locks on it (while in architect context)
Response: The block structure of all the blocks in the diagram reduces and diminishes. The different attributes of the selected T block are shown globally and also the attribute vector is partially shown in the GAF.

4.22.2.2 Stimulus: The user hovers on another tea block and holds the lock on button
Response: The g a f resulting from the T block is union with the previous showing GF

4.22.2.3 Stimulus: The user clicks on a specific attribute flow
Response: The attribute specifics are shown and the GAF highlights the attribute

4.22.2.4 Stimulus: The user selects two different attributes and presses the grouping button
Response: The attribute flows are union and specifics are automatically union and shown to the uses if the user confirms the attributes are made equivalent.

4.23 Show GAF

4.23.1 Description and Priority

This feature shows the GF in its entity with no attribute unsun or minimali subdued. This is different from the global attribute flow landing page in a way that no attribute takes on a focus.

4.23.2 Stimulus/Response Sequences

4.23.2.1 Stimulus: The user select show GF
Response: The camera pants out to a wide angle view and all the t blocks are mini militian minimally shown. Each and every attribute is shown perfectly.

4.24 Manipulate GAF Flow

4.24.1 Description and Priority

This feature is used to control the core relation between attribute vectors. Attributes a derived from each T block each T block and and those attributes are co related to generate classifications of attributes which are them shown as flows. Sometimes it may happen and attribute is too far stressed to a remote T block which does not confirm to the specific attribute. To clip or limit this attribute inheritance we can directly cut the flow of a attribute.

This feature basically reduces the attribute representation as discrete values and directly takes input from the user by manipulating manipulating the extent of flow in the in the graph. The attributes vector representation is changed minimally.

4.25 Grouping T

4.25.1 Description and Priority

This feature is used to group together attributes or t blocks. While grouping attributes the attribute vectors are added, the feature interference summary is shown. The user has some control over the interference. While grouping more than two t blocks the features are added any high interference constructive interference is prompt it to be made into a single feature and a special set of attributes are generated to represent the group. Each of these changes will have the discretion of the uses to be implemented into the GF.

This feature is meant to make creation of attribute floors easiest easier and add on to the auto semantic GF generated by the system.

4.25.2 Stimulus/Response Sequences

4.25.2.1 Stimulus: The user selects two different attributes and presses the grouping button
Response: The attribute flows are union and specifics are automatically union and shown to the uses if the user confirms the attributes are made equivalent.

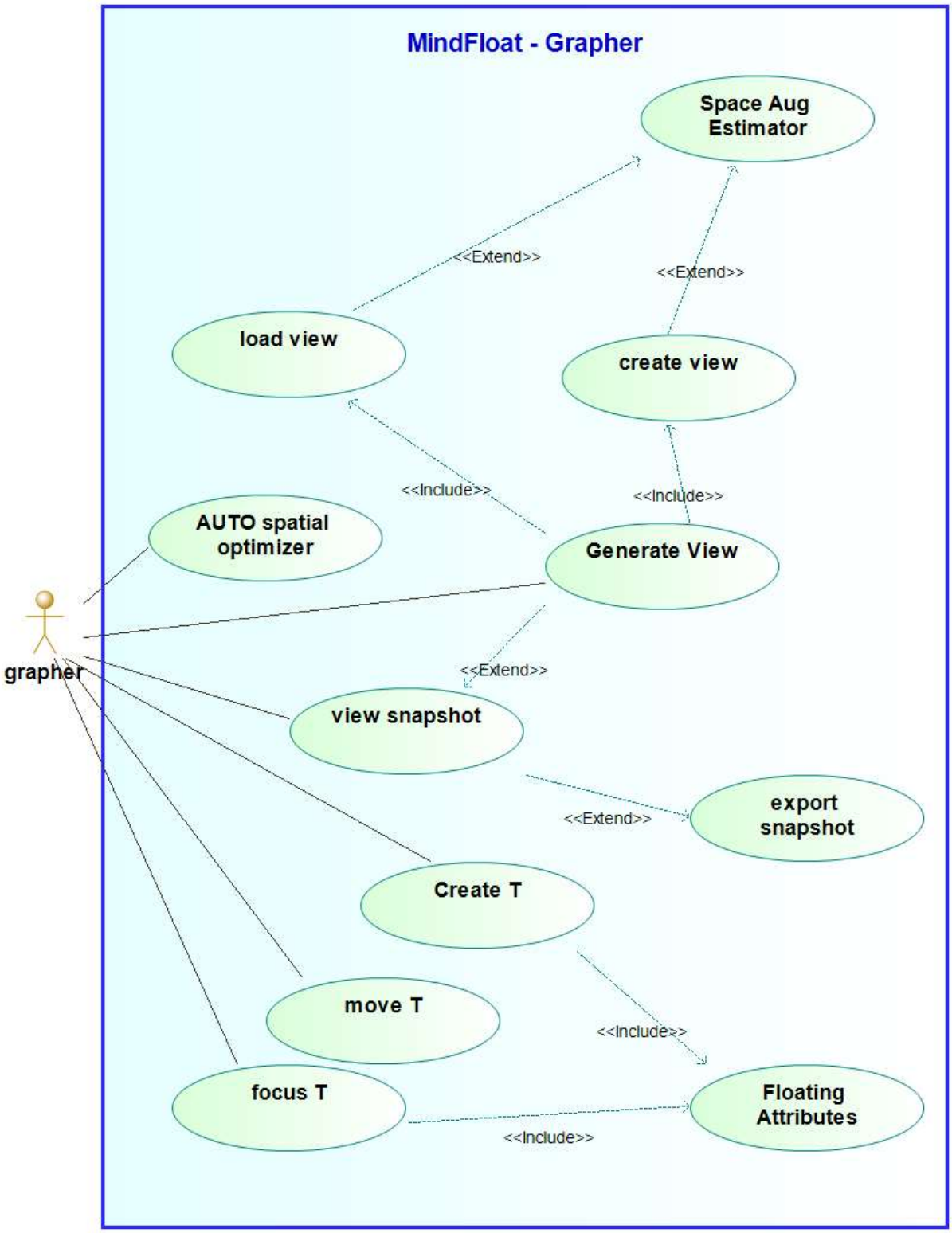
4.25.2.2 Stimulus: The user selects to a more t blocks and presses the grouping button
Response: A feature summary shows up. High constructive interference of attribute vectors are shown the grouping vector / vectors are shown. A global small graph of all the attributes to be generated / updated are shown minimum

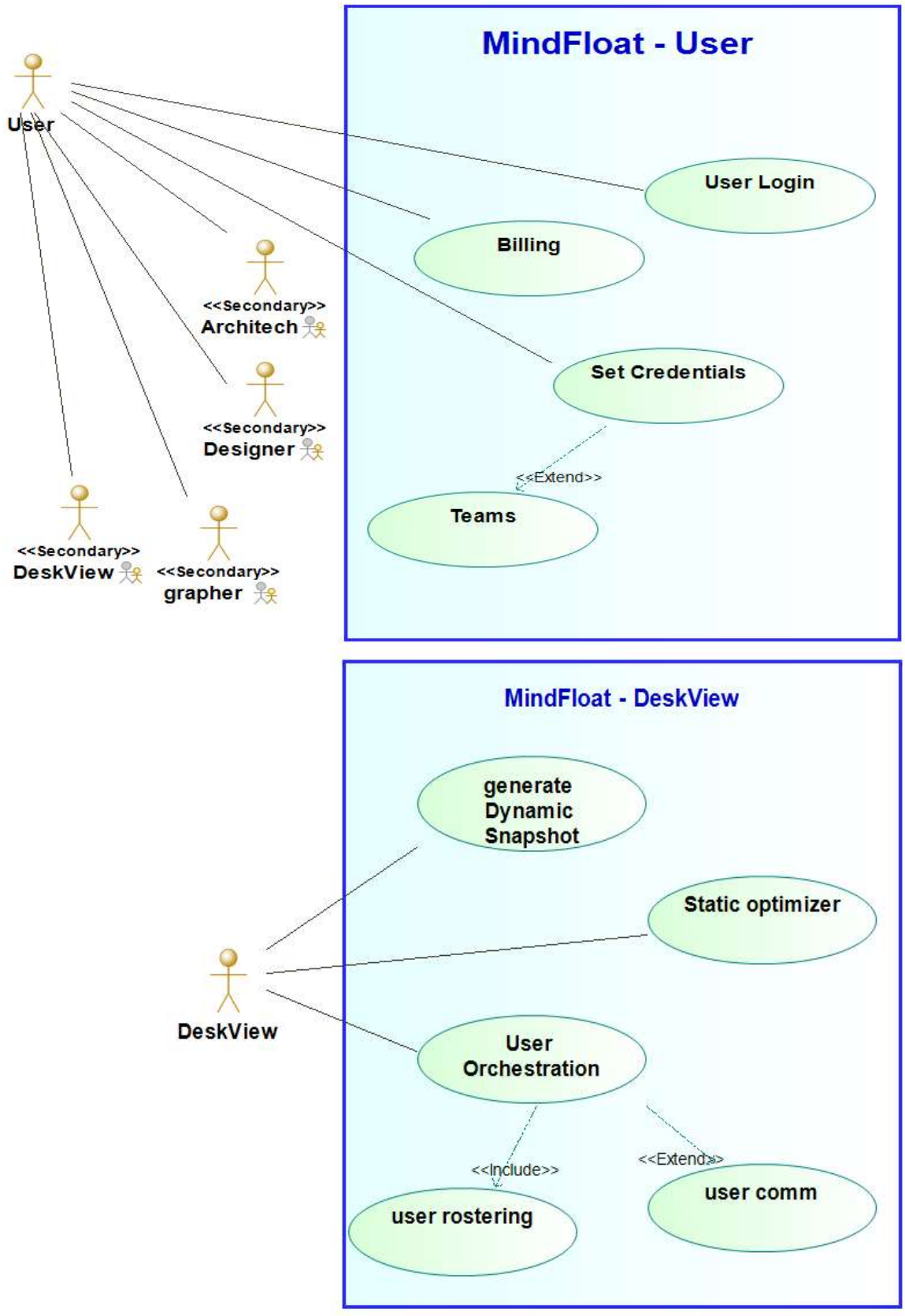
4.25.2.3 Stimulus: The user conforms the changes
Response: The user is promred to optimise the gaf again using the systems auto symatic optimizer

4.26 Auto Semantic Optimization

4.26.1 Description and Priority

Autosmantic optimizer is a passive feature which utilisers all the user input provided in the architecture contacts contacts contact as hole. It rise to optimise the spatial arrangement as well as visual treatment of the diagram. Just like special optimizer This will work upon the diagram and generate a more optimised version of it and if the system algorithm allows the user will be provided with a linear slider to navigate in the generated optimizer space manifold


MindFloat - Grapher
Space Aug Estimator
<<Extend>>
<<Extend>>
load view
create view
<<Include>>
<<Include>>
AUTO spatial optimizer
Generate View
grapher
<<Extend>>
view snapshot
<<Extend>>
export snapshot
Create T
move T
<<Include>>
focus T
Floating Attributes
<<Include>>


MindFloat - User
User
<<Secondary>>
Architech
<<Secondary>>
Designer
<<Secondary>>
DeskView
<<Secondary>>
grapher
User Login
Billing
Set Credentials
<<Extend>>
Teams
MindFloat - DeskView
DeskView
generate
Dynamic
Snapshot
Static optimizer
User
Orchestration
<<Include>>
user rostering
<<Extend>>
user comm

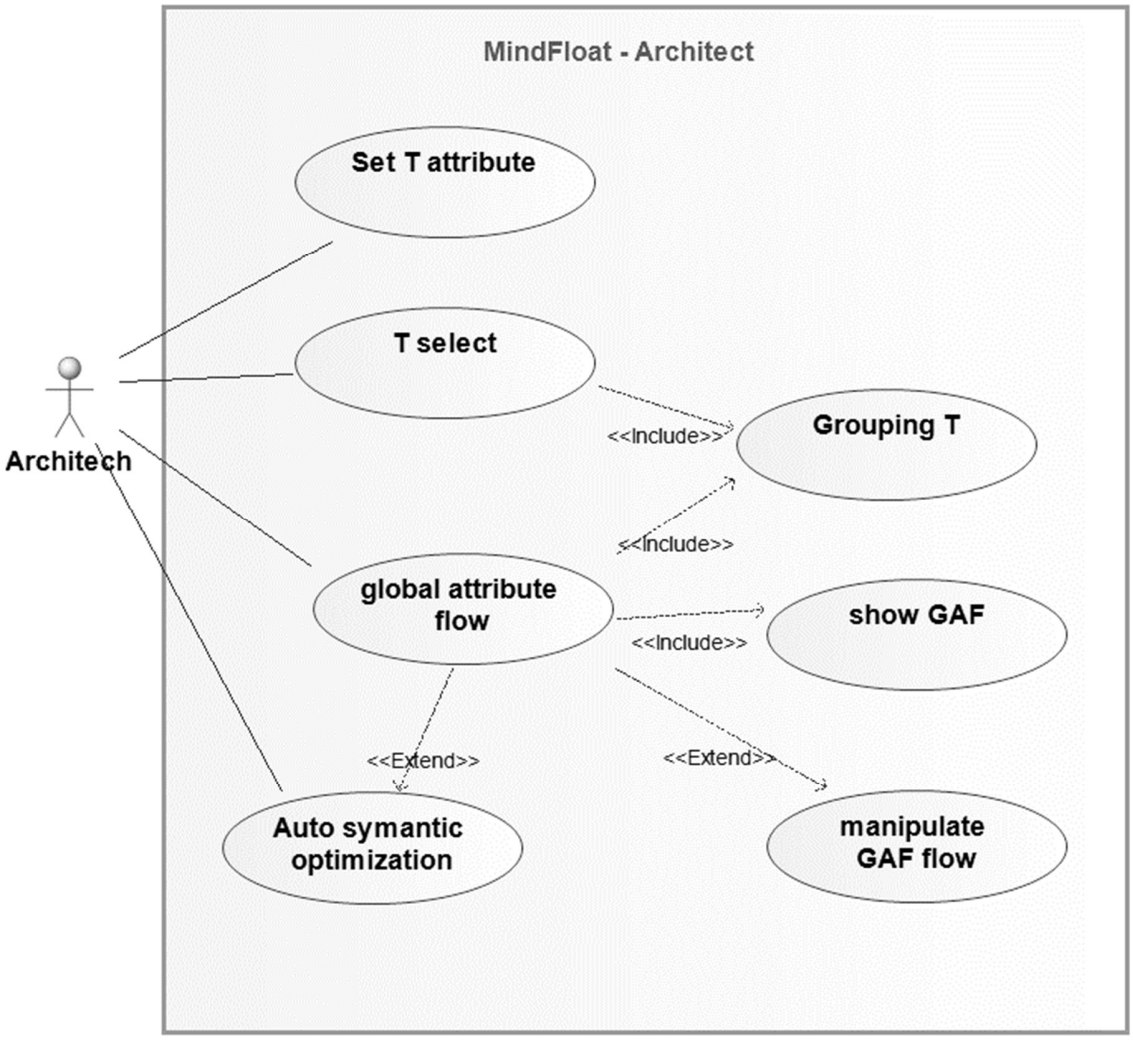

MindFloat - Architect
Set T attribute
T select
Architech
<<Include>>
Grouping T
<<Include>>
global attribute flow
<<Include>>
show GAF
<<Extend>>
<<Extend>>
Auto symantic optimization
manipulate GAF flow

Figure 1 Sequence Diagram: Space-Augmentation Estimator

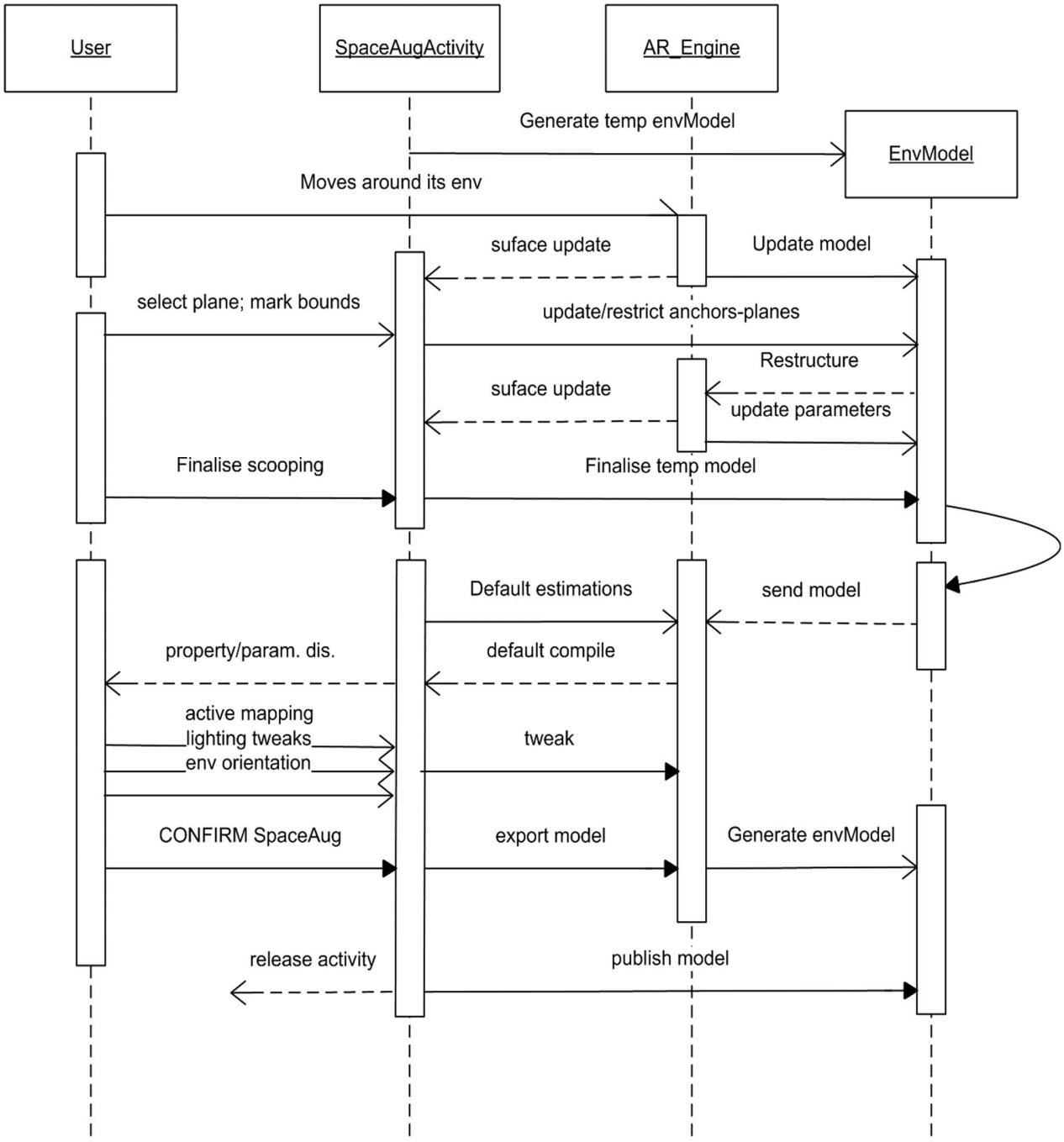


Figure 2 Sequence Diagram: Create View

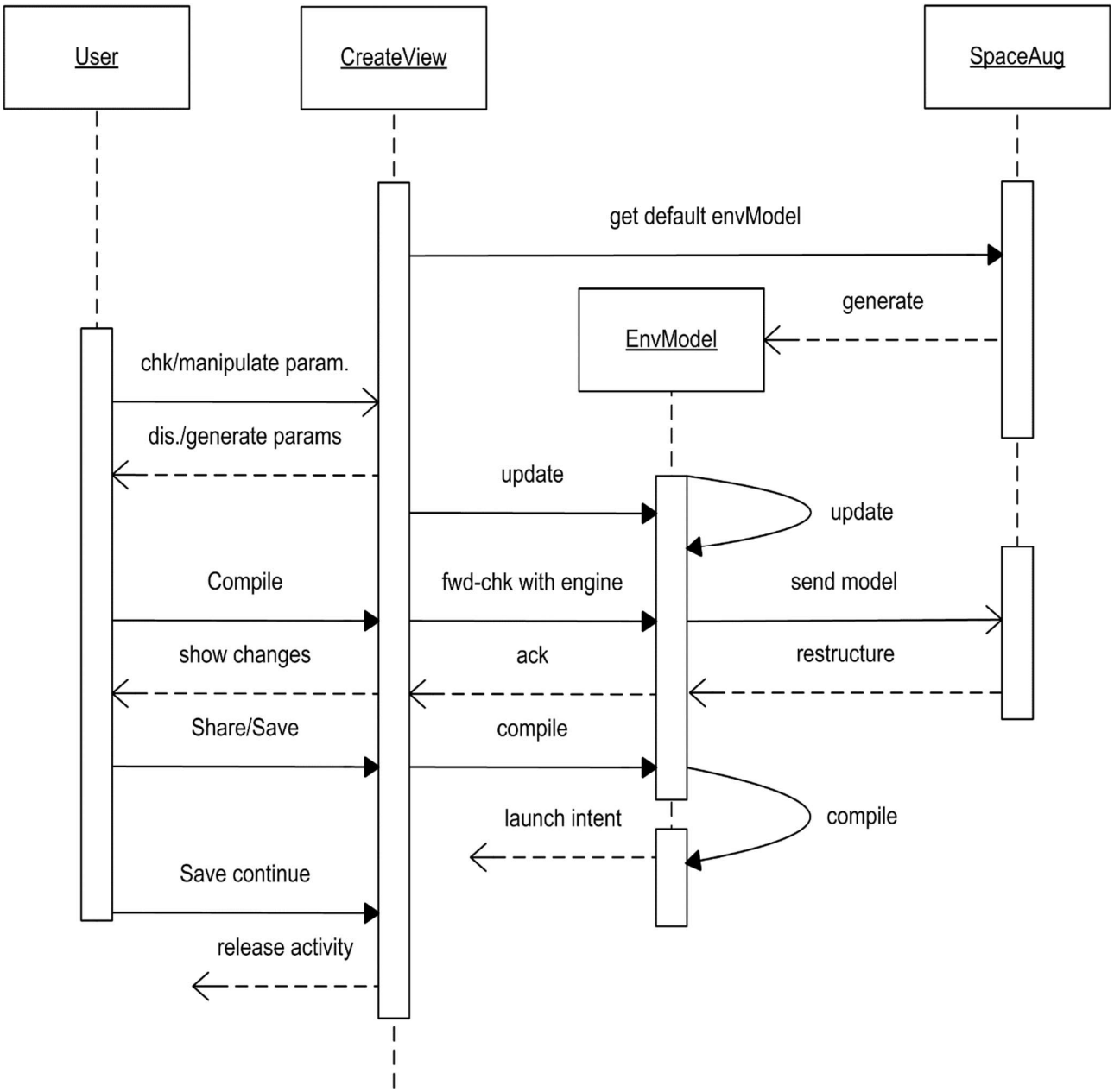


Figure 3 Sequence Diagram: Load View

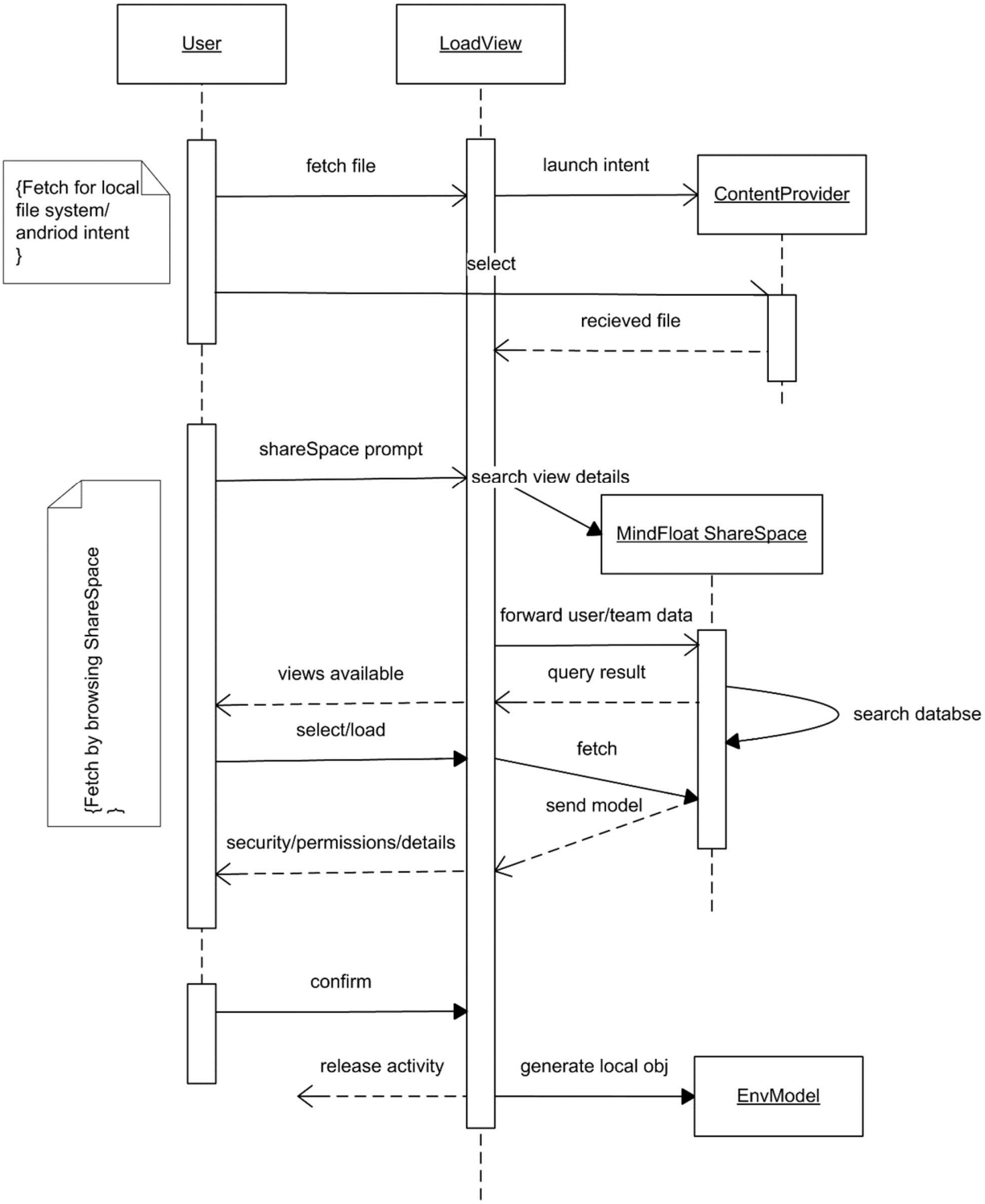


Figure 4 Sequence Diagram: Generate View

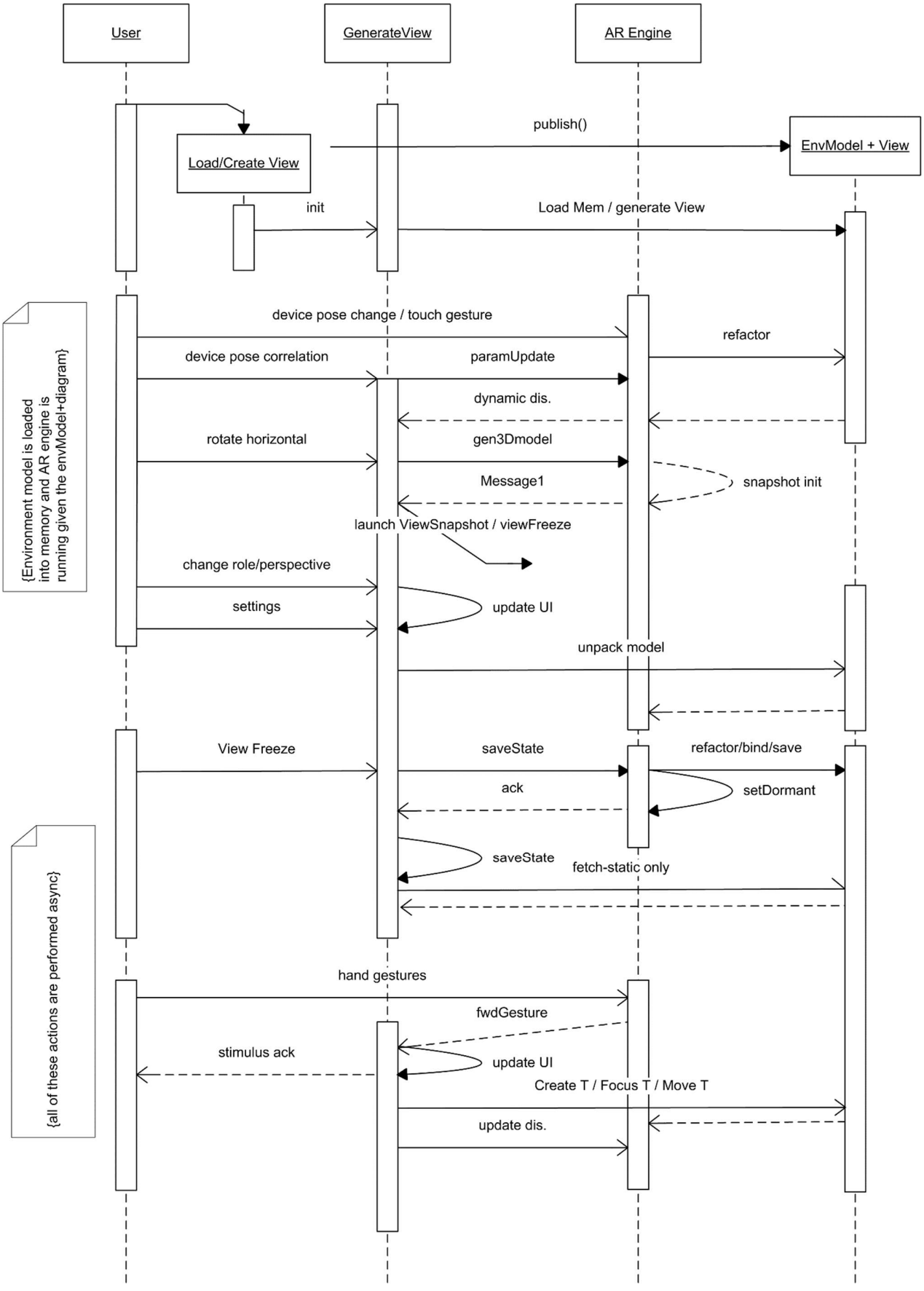


Figure 5Sequence Diagram: View Snapshot

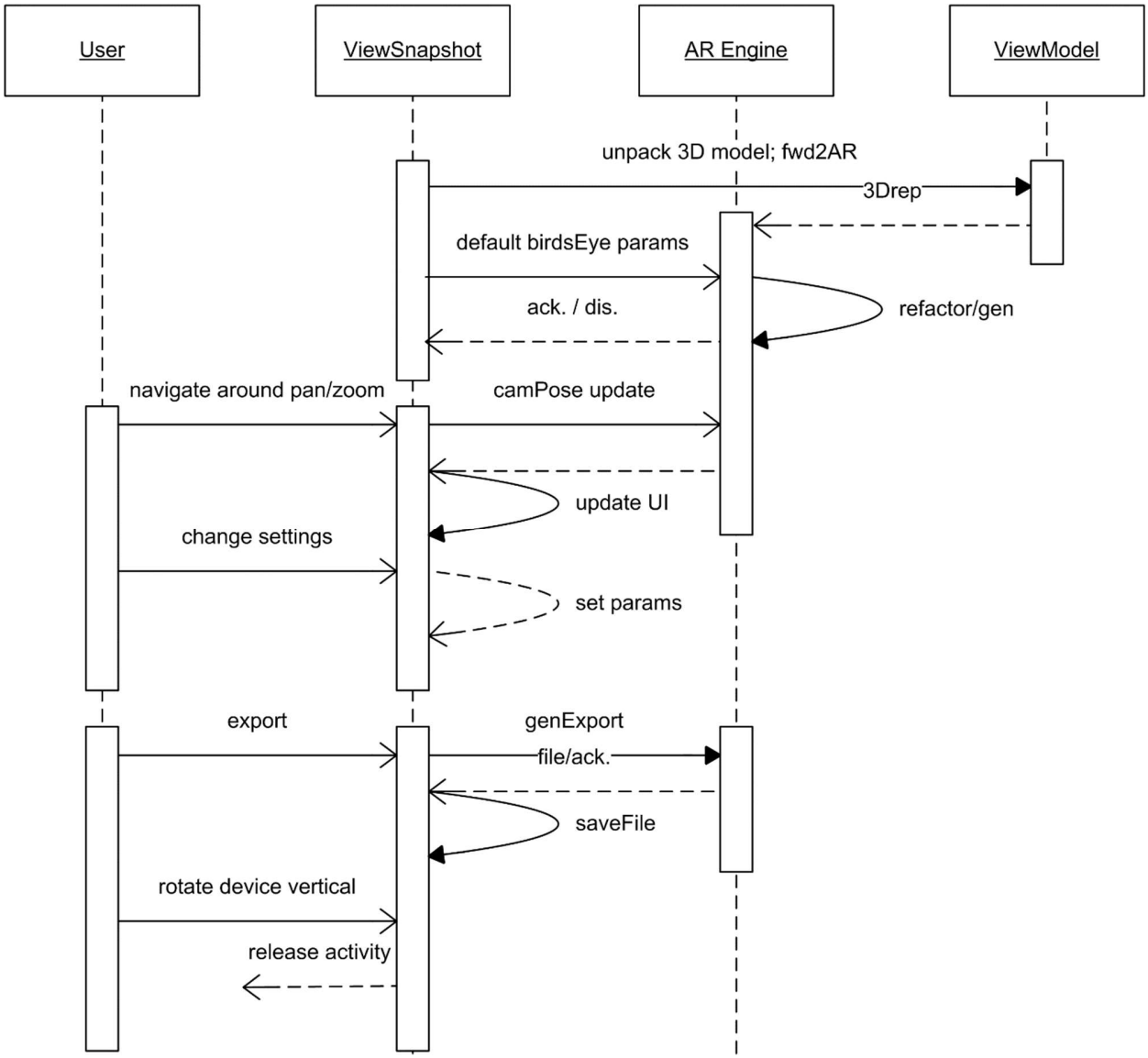


Figure 6: Sequence Diagram: Select T

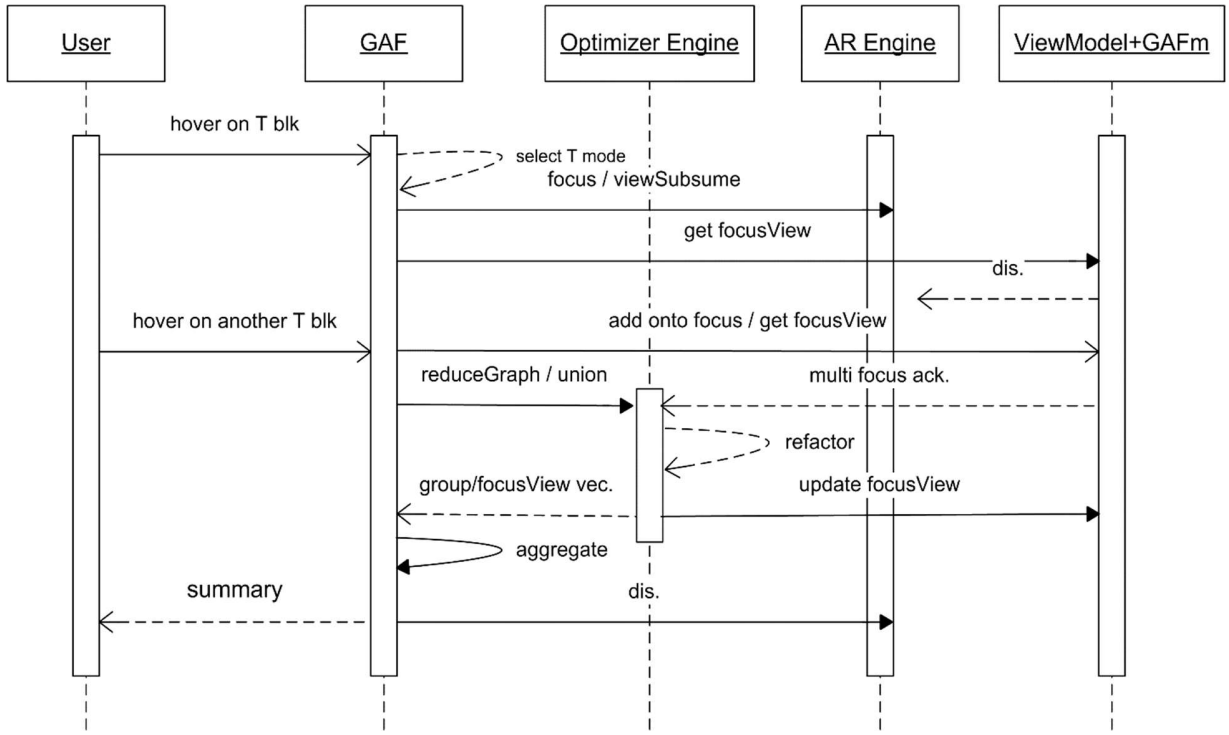


Figure 7: Sequence Diagram: Set T Attributes

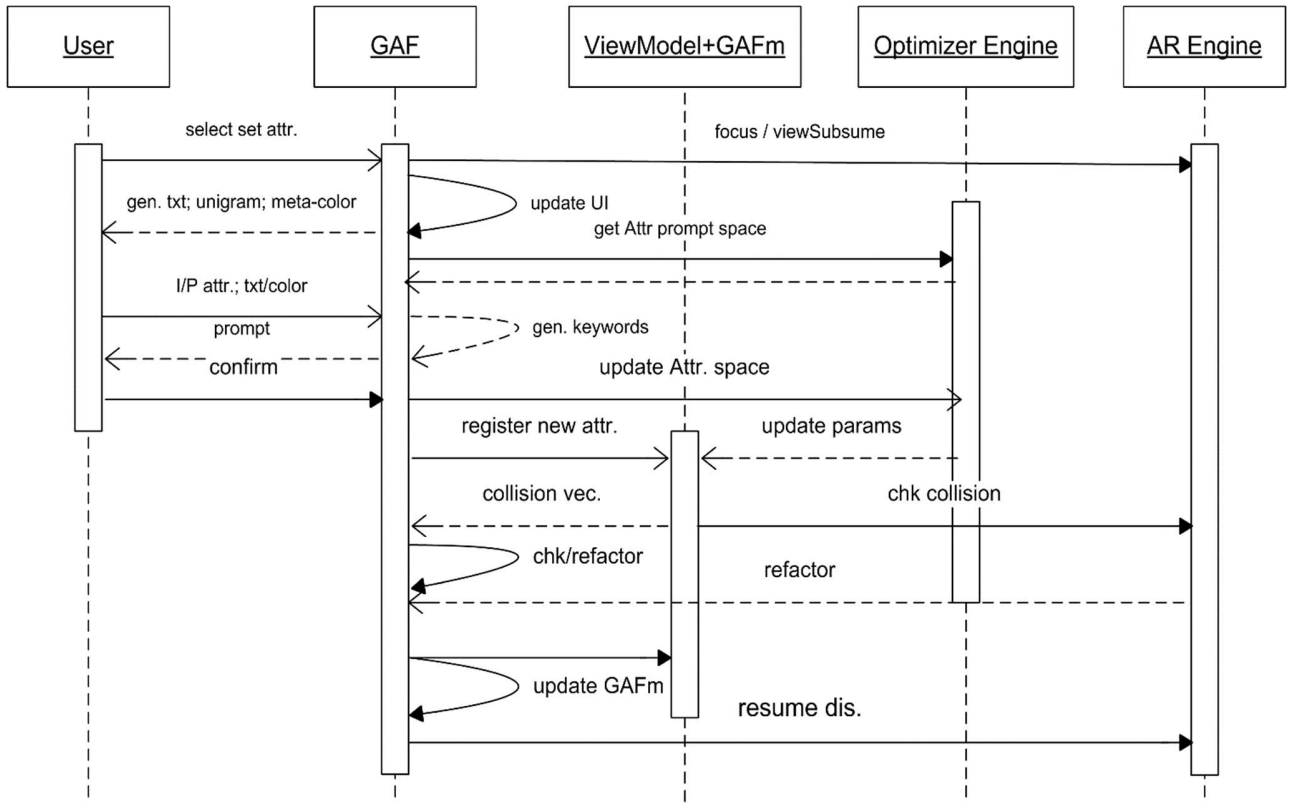
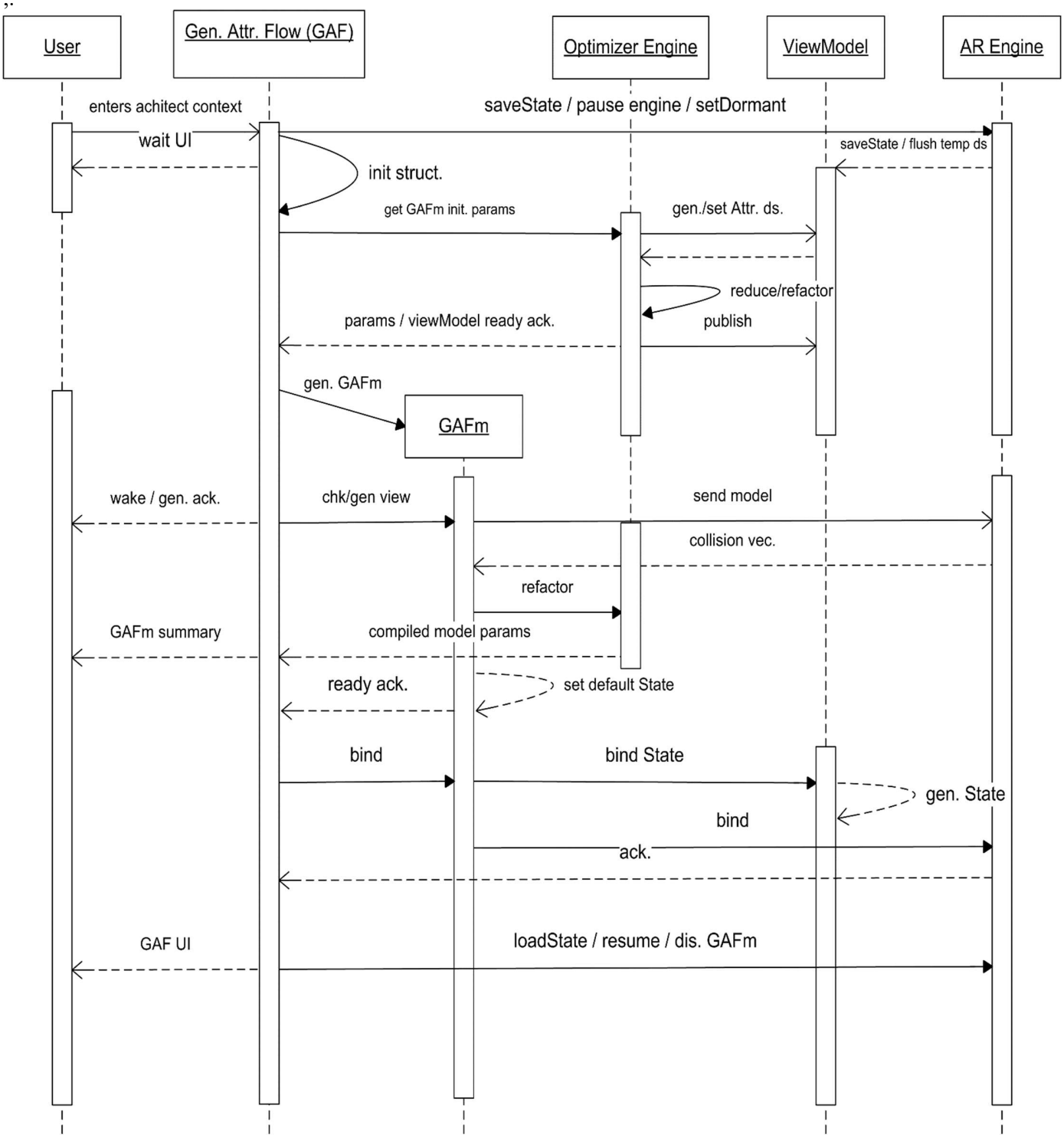
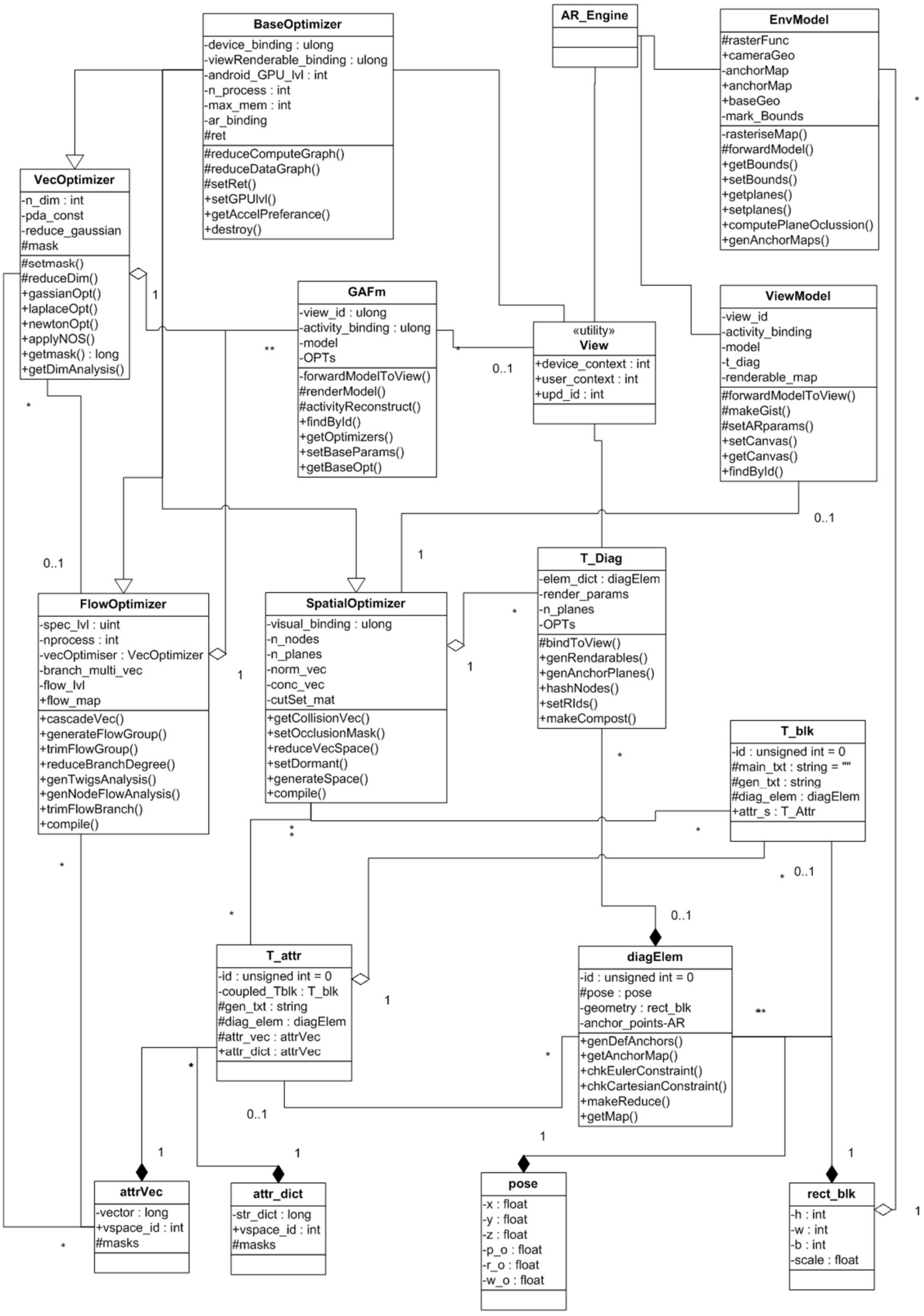


Figure 8: Sequence diagram: Generate GAF

,.




BaseOptimizer
-device_binding : ulong
-viewRenderable_binding : ulong
-android_GPU_lvl : int
-n_process : int
-max_mem : int
-ar_binding
#ret
#reduceComputeGraph()
#reduceDataGraph()
#setRet()
+setGPUlvl()
+getAccelPreferance()
+destroy()
AR_Engine
EnvModel
#rasterFunc
+cameraGeo
-anchorMap
+anchorMap
+baseGeo
-mark_Bounds
-rasteriseMap()
#forwardModel()
+getBounds()
+setBounds()
+getplanes()
+setplanes()
+computePlaneOclussion()
+genAnchorMaps()
VecOptimizer
-n_dim : int
-pda_const
-reduce_gaussian
#mask
#setmask()
#reduceDim()
+gassianOpt()
+laplaceOpt()
+newtonOpt()
+applyNOS()
+getmask() : long
+getDimAnalysis()
GAFm
-view_id : ulong
-activity_binding : ulong
-model
-OPTs
-forwardModelToView()
#renderModel()
#activityReconstruct()
+findById()
+getOptimizers()
+setBaseParams()
+getBaseOpt()
«utility»
View
+device_context : int
+user_context : int
+upd_id : int
ViewModel
-view_id
-activity_binding
-model
-t_diag
-renderable_map
#forwardModelToView()
#makeGist()
#setARparams()
+setCanvas()
+getCanvas()
+findById()
FlowOptimizer
-spec_lvl : uint
-nprocess : int
-vecOptimiser : VecOptimizer
-branch_multi_vec
-flow_lvl
+flow_map
+cascadeVec()
+generateFlowGroup()
+trimFlowGroup()
+reduceBranchDegree()
+genTwigsAnalysis()
+genNodeFlowAnalysis()
+trimFlowBranch()
+compile()
SpatialOptimizer
-visual_binding : ulong
-n_nodes
-n_planes
-norm_vec
-conc_vec
-cutSet_mat
+getCollisionVec()
+setOcclusionMask()
+reduceVecSpace()
+setDormant()
+generateSpace()
+compile()
T_Diag
-elem_dict : diagElem
-render_params
-n_planes
-OPTs
#bindToView()
+genRendarables()
+genAnchorPlanes()
+hashNodes()
+setRIds()
+makeCompost()
T_blk
-id : unsigned int = 0
#main_txt : string = ""
#gen_txt : string
#diag_elem : diagElem
+attr_s : T_Attr
T_attr
-id : unsigned int = 0
-coupled_Tblk : T_blk
#gen_txt : string
#diag_elem : diagElem
#attr_vec : attrVec
+attr_dict : attrVec
diagElem
-id : unsigned int = 0
#pose : pose
-geometry : rect_blk
-anchor_points-AR
+genDefAnchors()
+getAnchorMap()
+chkEulerConstraint()
+chkCartesianConstraint()
+makeReduce()
+getMap()
attrVec
-vector : long
+vspace_id : int
#masks
attr_dict
-str_dict : long
+vspace_id : int
#masks
pose
-x : float
-y : float
-z : float
-p_o : float
-r_o : float
-w_o : float
rect_blk
-h : int
-w : int
-b : int
-scale : float