UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NEO TANDEM TECHNOLOGIES

Eye Tracking

Author:	*Student number:*
Duran Cole	u13329414
Michael Nunes	u12104592
Molefe Molefe	u12260429
Tebogo Seshibe	u13181442
Timothy Snayers	u13397134

May 26, 2015

Architecture requirements

Eye Tracking

Github Link:

Version: Version 0.2 Alpha May 26, 2015

Contents

1 Introduction

1.1 Project Background

The Open Gaze And Mouse Analyzer is a company involved with the development of of eye tracking technology. This technology allows users to interact with a device compatible with the OGAMA software. This also provides another means of communication by providing hands-free communication with the device the user is using. OGAMA uses an eye-tracking device designed and created by The Eye Tribe. This device connects to a user's computer through a USB 3.0 connection. The device tracks the users eye movements through a series of sensors within the front of the device.

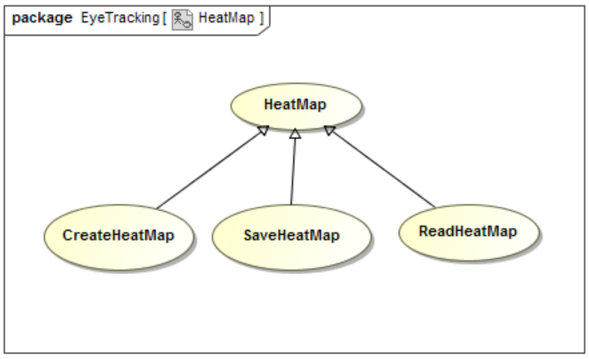
1.2 Project Vision

The aim of this project is to provide an extension to the already existing functionality of the OGAMA software. The extension should be able to track the focus of a user on 3D models. It should be able to generate a heat-map of the users focus that can be applied to the 3D model. An extra functionality should be that the extension can track the focus on a video. Heat-maps should also be possible to render to the video.

2 Use Case/Service Contracts

2.1 Heat Maps

This section will cover all the aspects of the heat maps use case.This use case is chosen as it is a vital part of the entire system.The data that is recorded will be used to create the heat maps that can then later be used to analyse the data and be used for future uses.

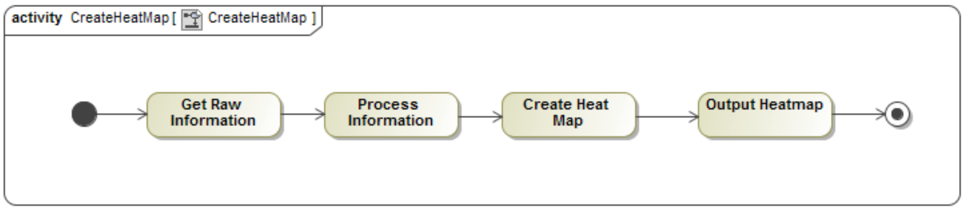


The following sub use cases are included in this use case:

2.1.1 CreateHeatMap

This use case deals with the creation of the Heat map from the recorded data.The data is used to show where the user has viewed the media item the most.

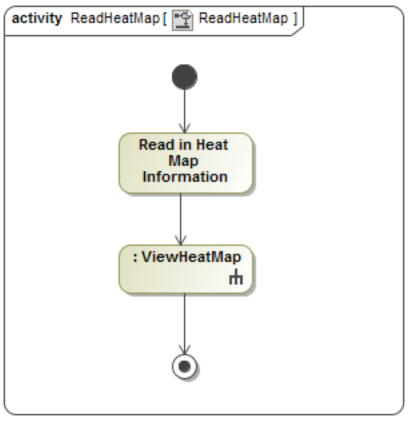
- Pre-condition: Data must be already recorded.
- Post-condition: Heat map of the data is created.
- Request Data Structure: HeatMap.CreateHeatMap(Data[]).
- Return data Structure: Heatmap is created



2.1.2 ReadHeatMap

This use case deals with the viewing of the Heat map from the recorded data.Once the heat map is created the users will be able to see the heat map and view the data.

- Pre-condition: Heat map must have been created.
- Post-condition: Heat map of the data is viewable and can be further analysed.
- Request Data Structure: HeatMap.ReadHeatMap(heatmapID).
- Return Data Structure: Heat map is viewable.

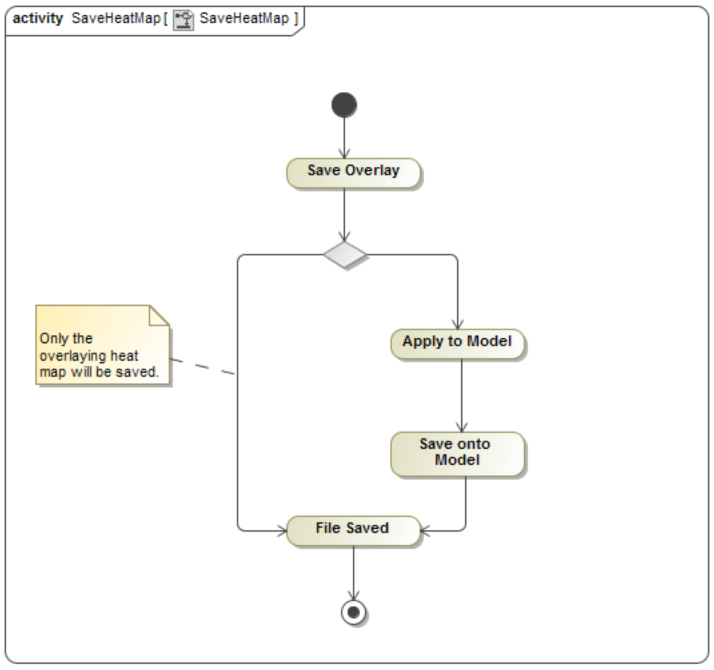

activity ReadHeatMap [ReadHeatMap]
Read in Heat Map Information
: ViewHeatMap

2.1.3 SaveHeatMap

This use case deals with the saving of the Heat map from the recorded data.The heat map will be saved on the users computer to be viewed at a later date.This will save only the heat map.

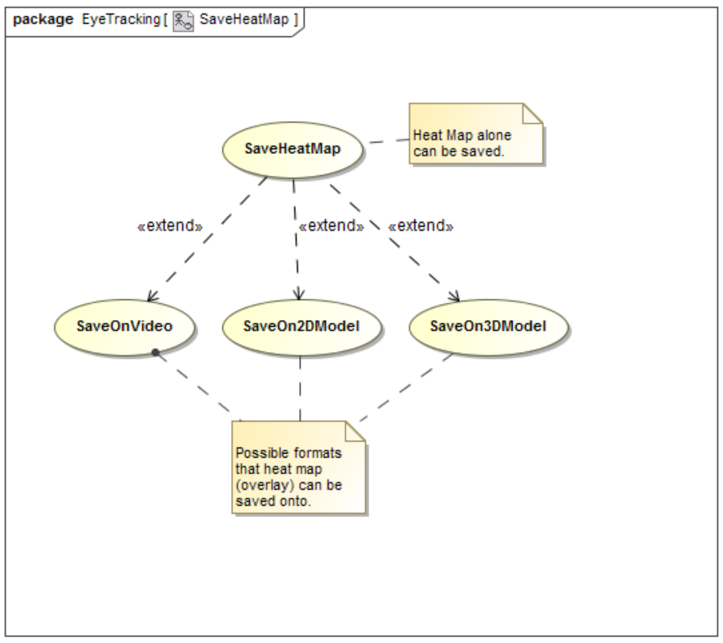
- Pre-condition: Heat map must have been created.

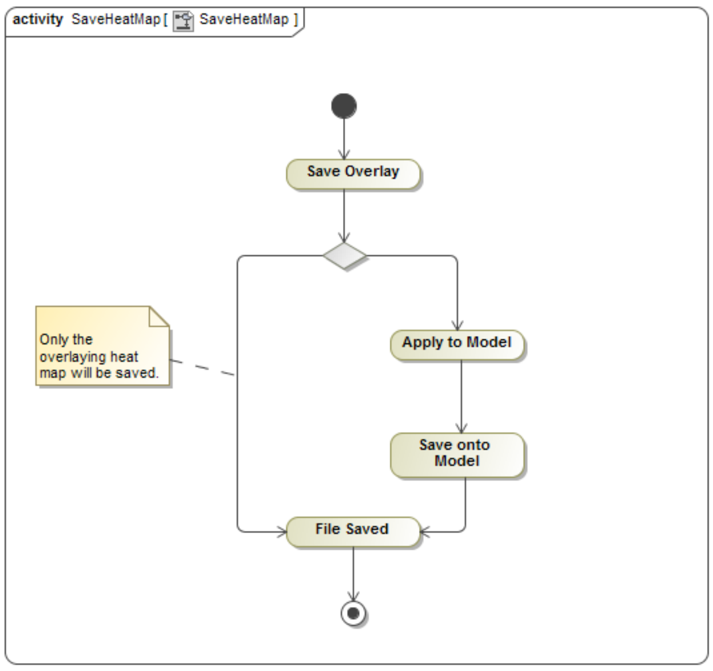
- Post-condition:Heat map of the data is saved.
- Request Data Structure:HeatMap.SaveHeatMap(Data[]).
- Return Data Structure:Heatmap is saved.


activity SaveHeatMap [SaveHeatMap]
Save Overlay
Only the overlaying heat map will be saved.
Apply to Model
Save onto Model
File Saved

2.2 Save Heat Maps

The saving of the heat maps is important to the system as it will allow the users to save data of a whole array of data and then use and compare them on other applications.The heat map will serve as an overlay for the media type and be placed over the media.The saving of heat maps can be divided into three subsections, namely SaveOnImage,SaveOnVideo and SaveOn3DModel.


package EyeTracking [SaveHeatMap]
SaveHeatMap
Heat Map alone can be saved.
«extend»
«extend»
«extend»
SaveOnVideo
SaveOn2DModel
SaveOn3DModel
Possible formats that heat map (overlay) can be saved onto.


activity SaveHeatMap [SaveHeatMap]
Save Overlay
Only the overlaying heat map will be saved.
Apply to Model
Save onto Model
File Saved

2.2.1 SaveOn2DModel

The heat map that is created for a 2DModel media type will be saved and then applied to the 2DModel to see the heat map over the 2DModel to indicate where the user has viewed the most.

- Pre-condition: Heat map must have been created and the 2DModel of the heatmap must be available.
- Post-condition: Heat map of the data is viewable on the 2DModel as an overlay.

- Request Data Structure: HeatMap.SaveOn2DModel(heatmapID,2DModelID).
- Return Data Structure: Heat map of the data is placed over the 2DModel.

2.2.2 SaveOnVideo

The heat map that is created for a video media type will be saved and then applied to the video to see the heat map over the video to indicate where the user has viewed the most.The heat map will follow the video as the video plays and be changed as time passes.

- Pre-condition: Heat map must have been created and the video must be available.
- Post-condition: Heat map of the data is viewable on the video as an overlay and tracks the video position to be changed as the video plays.
- Request Data Structure: HeatMap.SaveOnImage(heatmapID,ImageID).
- Return Data Structure: Heat map of the data is placed over the video.

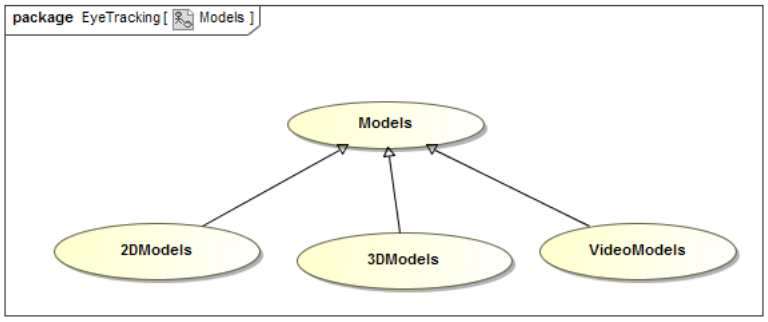
2.2.3 SaveOn3DModel

The heat map that is created for a 3DModel media type will be saved and then applied to the 3DModel to see the heat map over the 3DModel to indicate where the user has viewed the most.The 3DModel can be either a 2D or 3DModel.

- Pre-condition: Heat map must have been created and the 3DModel of the heatmap must be available.
- Post-condition: Heat map of the data is viewable on the 3DModel as an overlay.
- Request Data Structure: HeatMap.SaveOn3DModel(heatmapID,3DModelID).
- Return Data Structure: Heat map of the data is placed over the 3DModel.

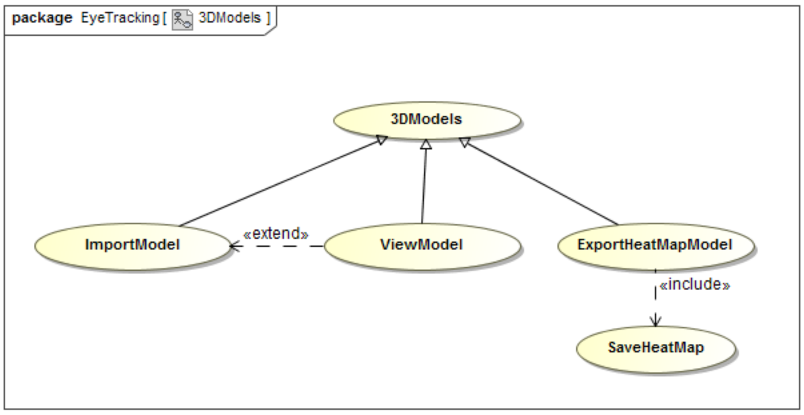
2.3 Models

This section details all the different types of media or models that heat maps can be created for and saved onto.There are three types of models Video,2D and 3D.The heatmaps are created based on the models and their types.



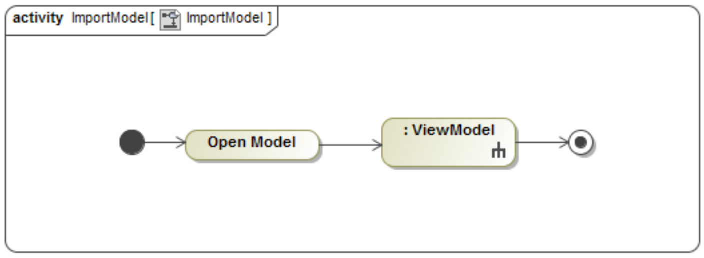
2.3.1 3D Models

3D Models will be taken into the system and then will be rendered and will be viewable to the user.The heatmap will then be allowed to be saved on the 3D model and then from there can be viewed with the heatmap as an overlay.



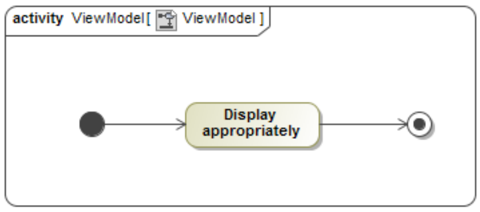
1. ImportModel
   The heat map that is imported for a 3DModel media type will be saved in the system so that eye recording can be done on the media and that a heat map can be created for and be applied to it at a latter stage.

   - Pre-condition: 3D model must exist.
   - Post-condition: Recording can be done on the model.
   - Request Data Structure: HeatMap.3DModeImport(3DModelID).
   - Return Data Structure: Rendered data model is imported.

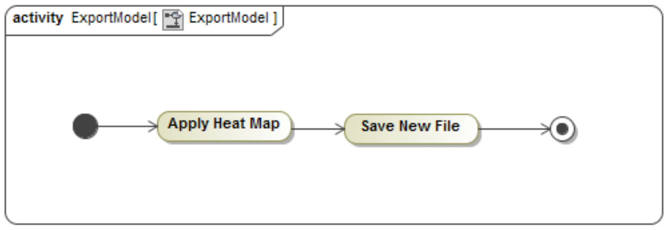


2. ViewModel Once the model is imported the users can view this and a recording of the eye can then be recorded for that specific model.

   - Pre-condition: 3D model must have been imported .
   - Post-condition: 3D model is viewable and can be recorded on.
   - Request Data Structure: HeatMap.View3DModel(3DModelID).
   - Return Data Structure: Viewable 3D model

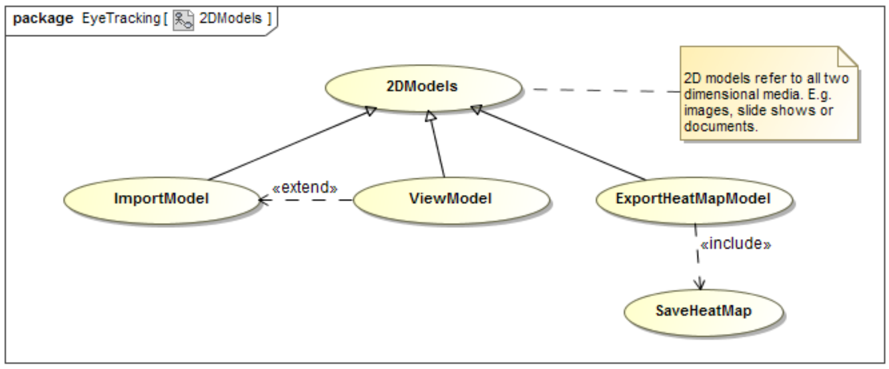


3. ExportHeatMapModel The model will have had recording done to it and then a heat map will be applied to the model.This can then be exported so that it can be viewed at a letter date.
    - Pre-condition: Heat Map must be created and Model Imported.
    - Post-condition: Exported Model With heatmap on it to be viewed.
    - Request Data Structure: HeatMap.ExportHeatMap3DModel(heatmapID,3DModelID).
    - Return Data Structure: Model with heat map.



2.3.2 2D Models

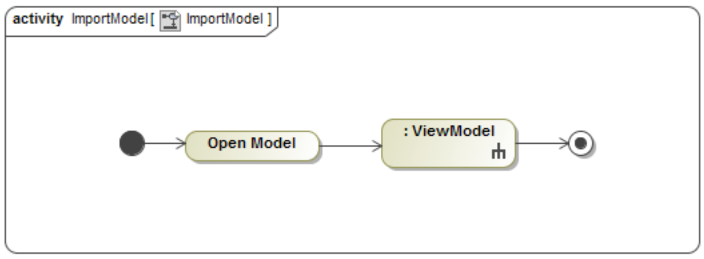
The 2D models can be imported into the system.The model will most often be a image of a jpeg file extension.The image can be imported and have recording done on it.The heat map is created and applied to it.



package EyeTracking[2DModels]
2DModels
2D models refer to all two dimensional media. E.g. images, slide shows or documents.
ImportModel
«extend»
ViewModel
ExportHeatMapModel
«include»
SaveHeatMap

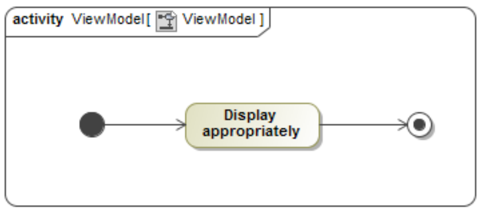
1. ImportModel
   The heat map that is imported for a 2DModel media type will be saved in the system so that eye recording can be done on the media and that a heat map can be created for and be applied to it at a later stage.
    - Pre-condition: 2D model must exist,a image preferable.
    - Post-condition: Recording can be done on the model.

- Request Data Structure: HeatMap.2DModeImport(2DModelID).
- Return Data Structure: Rendered image is imported.



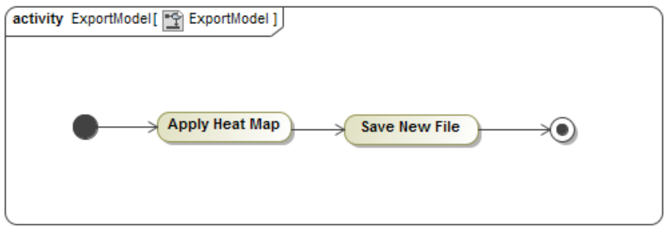
2. ViewModel Once the model is imported the users can view this and a recording of the eye can then be recorded for that specific model.

    - Pre-condition: 2D model(image) must have been imported .
    - Post-condition: Image is viewable and can be recorded on.
    - Request Data Structure: HeatMap.View2DModel(2DModelID).
    - Return Data Structure: Viewable image



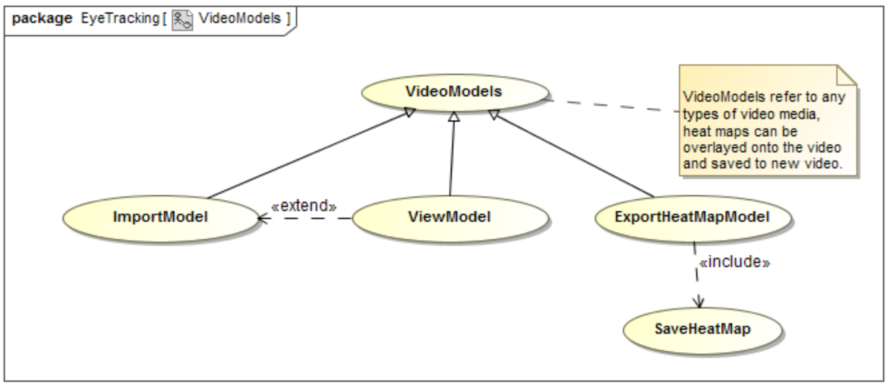
3. ExportHeatMapModel The image will have had recording done to it and then a heat map will be applied to the image or 2D model.This can then be exported so that it can be viewed at a later date and have a heat map placed on it.

    - Pre-condition: Heat Map must be created and Model Imported.
    - Post-condition: Exported Model With heatmap on it to be viewed.
    - Request Data Structure: HeatMap.Export2Dmodel(2DModelID).
    - Return Data Structure: Model with heat map.



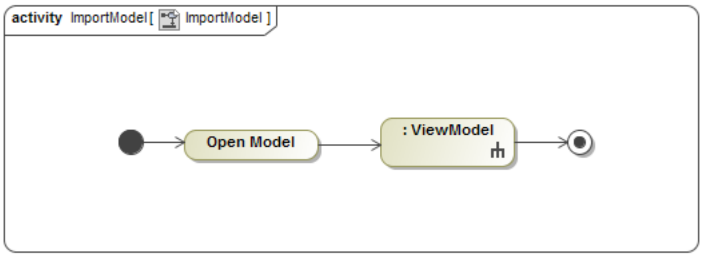
2.3.3 Video Models

A video can be easily imported into the system.The video can have recording done onto it and data stored.The data can then be used to create a heatmap that is then applied to the video second by second.


package EyeTracking [VideoModels]
VideoModels
VideoModels refer to any types of video media, heat maps can be overlayed onto the video and saved to new video.
ImportModel
«extend»
ViewModel
ExportHeatMapModel
«include»
SaveHeatMap

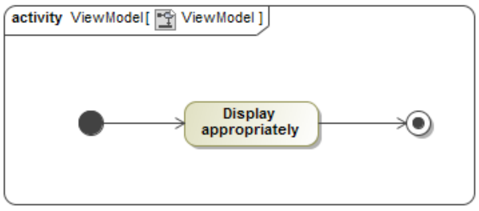
1. ImportModel
   The heat map that is imported for a video media type will be saved in the system so that eye recording can be done on the media and that a heat map can be created for and be applied to it at a later stage.The heatmap will follow the video.

   - Pre-condition: video must exist.
   - Post-condition: Recording can be done on the video.
   - Request Data Structure: HeatMap.VideoImport(videoID).
   - Return Data Structure: Rendered video is imported.



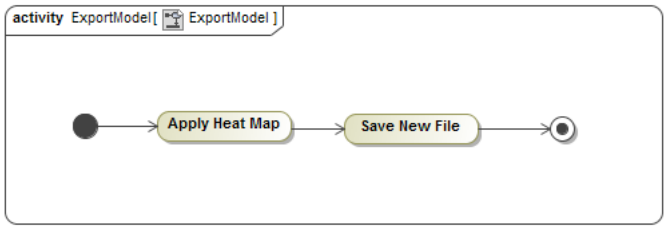
2. ViewModel Once the video is imported the users can view this and a recording of the eye can then be recorded for that specific video.The heat map can be created and applied.

   - Pre-condition: Video must have been imported .
   - Post-condition: Video is viewable and can be recorded on.
   - Request Data Structure: HeatMap.ViewVideo(videoID).
   - Return Data Structure: Viewable video.



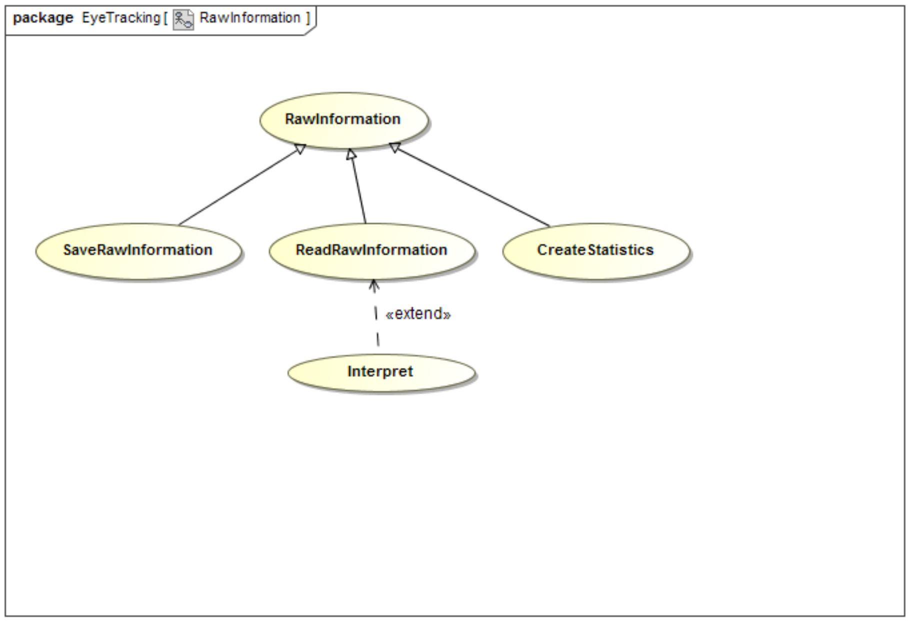
3. ExportHeatMapModel The video will have had recording done to it and then a heat map will be applied to the model.This can then be exported so that it can be viewed at a later date. The exported video will have a heat map on it.

    - Pre-condition: Heat Map must be created and video Imported.
    - Post-condition: Exported video With heat map on it to be viewed.
    - Request Data Structure: HeatMap.SaveOnvideo(videolID).
    - Return Data Structure: Video with heat map.



2.4 Raw Information

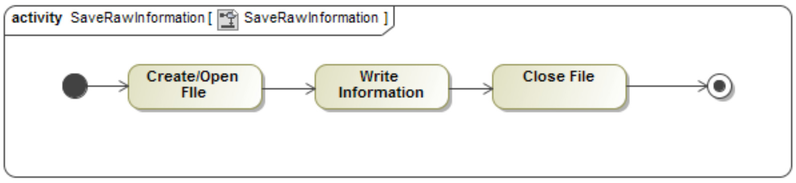
When recording happens on the system the raw information is saved.The raw information is important as it forms the basis for creating heat maps for specific.The raw information is saved on the system.The raw information can then be read as the raw data or put into a statistical formats.



2.4.1 SaveRawInformation

When recording is done on the media the raw information will need to be saved so that it can be used at a later stage.This raw information is only temporarily saved and not saved for the life time of the media it is saved on.

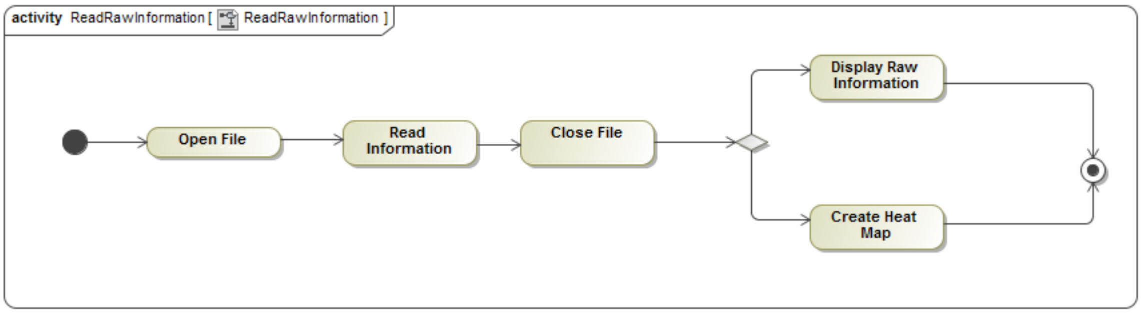
- Pre-condition: Recording on a media type must have happened.
- Post-condition: Data is saved and can be used later.
- Request Data Structure: HeatMap.SaveRawInformation(data[]).
- Return Data Structure: Raw array of data points saved.



2.4.2 ReadRawInformation

Data can be read from a previously saved file.this will allow the user to import the data and then use it to compare to other data files.

- Pre-condition: Recorded data must be saved in a file.
- Post-condition: Data is displayed.
- Request Data Structure: HeatMap.ReadRawInformation(file).
- Return Data Structure: data array with all data points.

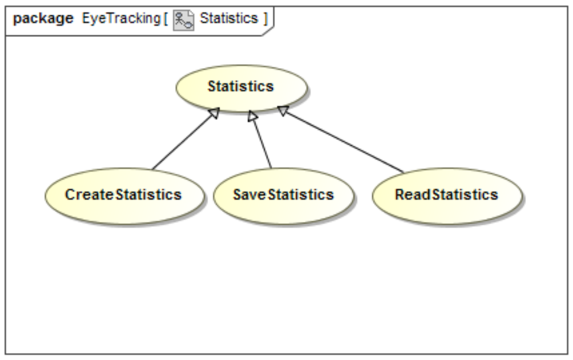


2.4.3 CreateStatistics

The writing of statistics is part of the Statistics use-case and all information can be found there regarding the sub use case

2.5 Statistics

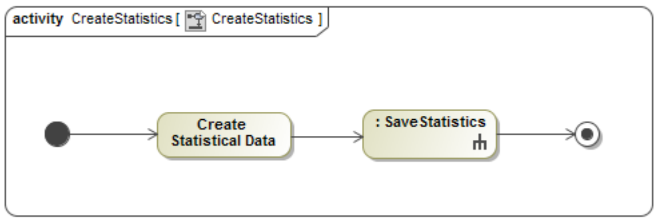
Statistics on all the data is collected and will then be used to make a statistical page that can be analysed and then it can be easily compared at any time.



2.5.1 CreateStatistics

Using the recorded data we can create a statistical analysis that can be viewed and used to compared data.

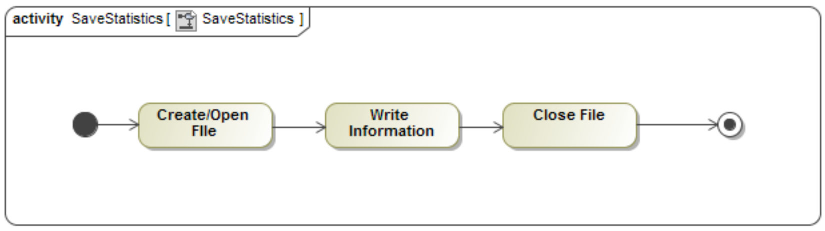
- Pre-condition: Recorded data must exist.
- Post-condition: Data is turned into statistical data.
- Request Data Structure: Stats.CreateStats(data[]).
- Return Data Structure: Statistics in arrays.



2.5.2 SaveStatistics

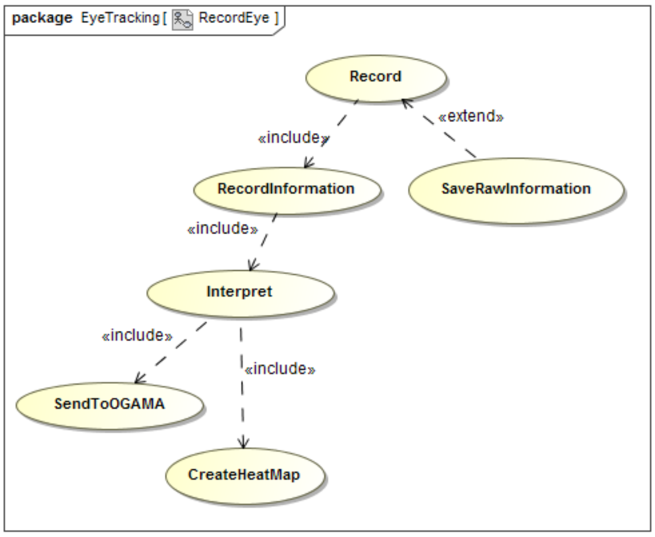
The statistics can be saved into a file so that they can be printed out and can be used as a hard copy.This will take all data and put it in a pdf or csv file.

- Pre-condition: Statistical data must exist.
- Post-condition: Data is put into a report.
- Request Data Structure: Stats.SaveStats(statsdata[]).
- Return Data Structure: Save report in pdf or csv format.



2.6 Record Eyes

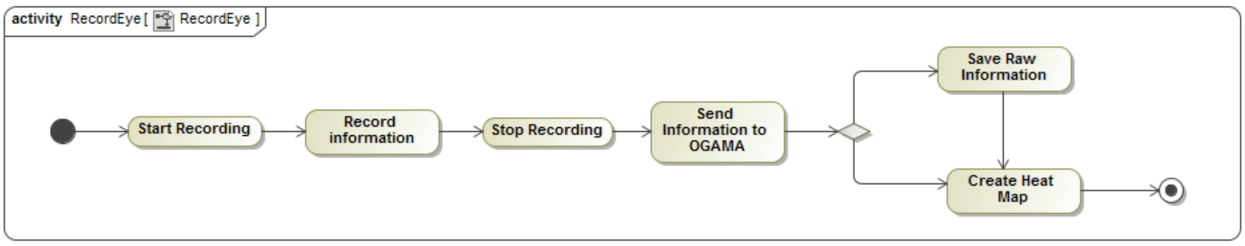
The recording of the eyes will be done using the camera.The data recorded will allow heat maps to be created.The heat maps can then be turned into overlays for the media types.the recording eyes use case uses a lot of sub use cases from other use cases such as SaveRawInformation.The recording of the information is important as this is used throughout.



2.6.1 RecordInformation

The information recorded will be sent to the OGAMA module to then be procesed and then this can be used to carry out functions in the system. The

- Pre-condition: User needs to look at the media.
- Post-condition: Data is recorded about the eye movements.
- Request Data Structure: Recorder.Record(type).
- Return Data Structure: Raw data in x,y and z co ordinates.



3 Required Functionality

3.1 Render 3D scene

Priority: Critical
Given a 3D scene, in a format such as a blend file or an obj file, the scene must be able to be rendered to a display. The rendered scene will be placed in a window for viewing so that the eye-tracker can "see" the scene.
Pre-condition: Open the 3D rendering software software with the 3D scene file.
Post-condition: Have a window open that has the scene loaded and rendered.

3.2 Eye-tracking on a 3D scene

Priority: Critical
The eye-tracking software must be used or augmented in such a way that specific areas in the 3D scene can capture viewing information. This is will be done given information from the eye-tracker itself.
Pre-condition: Have OGAMA and eye-tracking equipment setup and running.
Post-condition: Captured information into a file.

3.3 Create heat-mapped texture

Priority: Critical
From the eye-tracking information a heat map in the form of a texture map must be produced. The format of the texture will depend on the 3D software used.
Pre-condition: Have captured information from OGAMA and eye-tracking equipment.
Post-condition: Captured information made into a heat map image.

3.4 Map heat-mapped texture back onto scene

Priority: Critical
The heat-mapped texture of the object must be able to be mapped correctly back onto the scene, that is the placement must be as was the original texture.
Pre-condition: Have heat-mapped texture..
Post-condition: 3D scene now has new heat-mapped texture applied.

3.5 Real-time heat-mapping

Priority: Nice-to-have
As the scene is being viewed it will also be coloured according to the eye-trackers information.
Pre-condition: Have OGAMA and eye-tracking equipment setup and running.
Post-condition: Captured information used to colour the scene.

3.6 Interactive Maps

Priority: Important
This function involves overlaying a map with data captured by the Eye tribe eye tracking technology and stored using the OGAMA freeware. The data will be retrieved from a database which has already been integrated into the OGAMA freeware that will be used for this project. Through the use of the analytical features provided by OGAMA, its functionality will be extended into the visualization of the information it has stored. This includes the attention maps, scan paths and replay functions.
Pre-condition: Have the Eye tribe eye tracking technology running during the activity.
Post-condition: Output a 2D representation on the interactive map with the necessary data overlaying it.

3.7 In-Video Eye Tracking

Priority: Nice-to-have
This function overlays video frames with eye tracking attention maps while watching a video. This is an especially tricky function due to the fact that OGAMA analytical feature is able to run in parallel with data being viewed, but not render the attention map or scan path at run time.
Pre-condition: Have OGAMA freeware integrated into video viewing software. Have the Eye tribe eye tracking technology running during the activity.
Post-condition: Run time editing of video frames overlaid with attention maps and scan paths.

3.8 Post-video Eye Tracking

Priority: Nice-to-have
This function involves rendering a video with an attention map and scan path overlaid after it has been watched. This function is more achievable than the In-video eye tracking because the OGAMA software is able to record eye gaze and movement on image slideshows thus it can be used on a slideshow of frames from the video to render a video coupled with this analyzed data overlaying it.
Pre-condition: Have OGAMA freeware integrated into video viewing software. Have the Eye tribe eye tracking technology running during the activity.
Post-condition: Output a video with its frames overlaid with attention maps and scan paths.

3.9 Suitable Video Output Formats

Priority: Nice-to-have
This function aims allow the post-video eye tracking video to be outputted in more than one video format. The default video output provided by OGAMA is avi, but through this function it would be possible to output videos in more popular formats such as

mp4, wmv, mkv etc. This allows the product to be used by many more devices, some of which have limited memory or only support specific video formats.
Pre-condition: Same as the Post-video Eye Tracking function.
Post-condition: Output a video in the format that the user has selected.

4 Process Specification

5 Domain Model

Bibliography