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 27, 2015

Architecture requirements

Eye Tracking

Github Link:

Version: Version 0.2 Alpha May 27, 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 (Figure 1). 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.

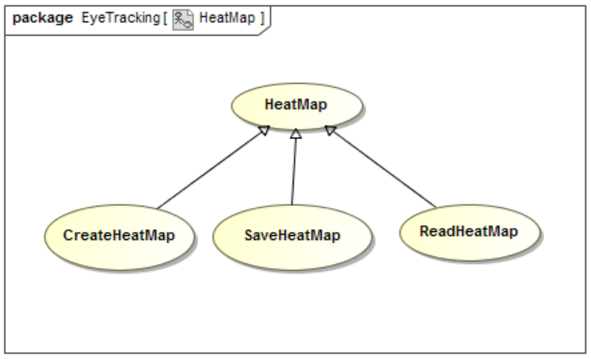

Figure 1: Heat Map Use Case

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

2.1.1 CreateHeatMap

The create heat map use case deals with the creation of the heat map from the recorded data. The heat map shows where on the media item has been viewed the most by the user(s).

- 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

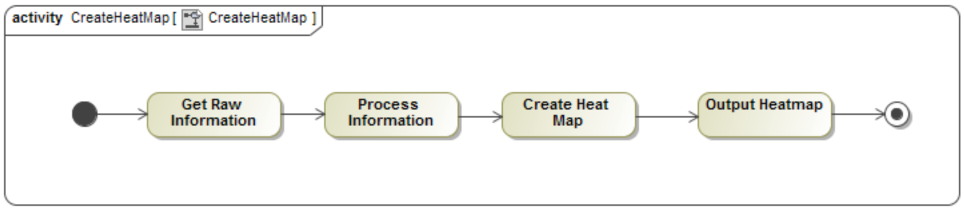

Figure 2: Create Heat Map Activity

2.1.2 ReadHeatMap

This use case deals with the viewing of Heat maps from their 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.

2.1.3 SaveHeatMap

This use case deals with the saving of heat maps that are created from their respective 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 which can later be saved onto a model at a later date.

- 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.

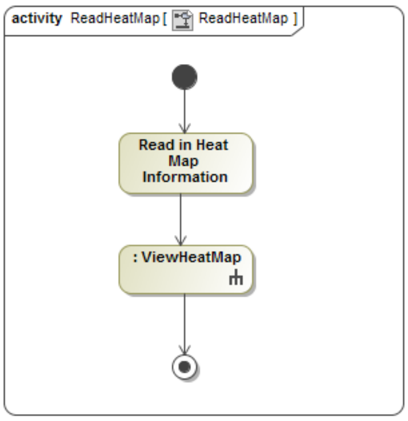
activity ReadHeatMap [ReadHeatMap]
Read in Heat Map Information
: ViewHeatMap

Figure 3: Read Heat Map Activity

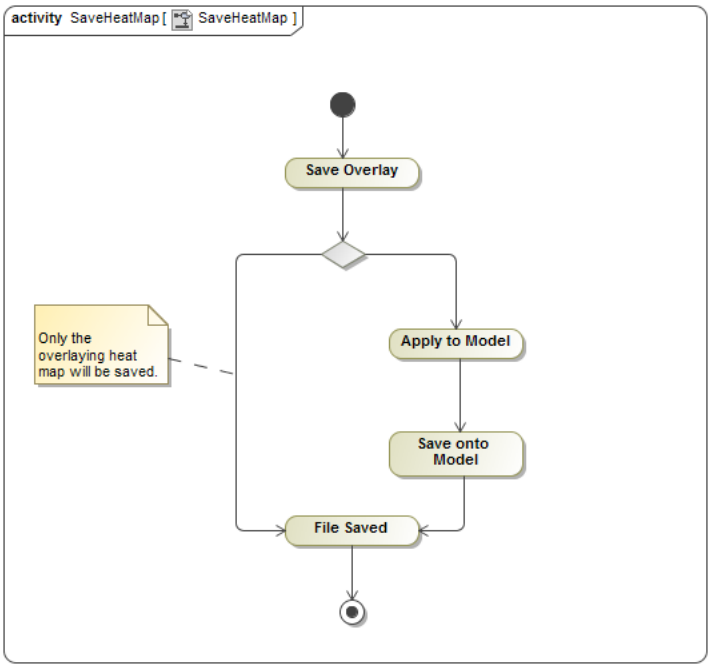
activity SaveHeatMap [SaveHeatMap]
Save Overlay
Only the overlaying heat map will be saved.
Apply to Model
Save onto Model
File Saved

Figure 4: Save Heat Map Activity

2.2 Save Heat Maps

The saving of the heat maps provides important functionality to the system as it will allow the users to save the data which can later be used to either continue research or to compare data.The heat map will serve as an overlay for the media type and be placed over the media model. The saving of heat maps can be divided into three subsections, namely SaveOn2DModel, SaveOnVideo and SaveOn3DModel.

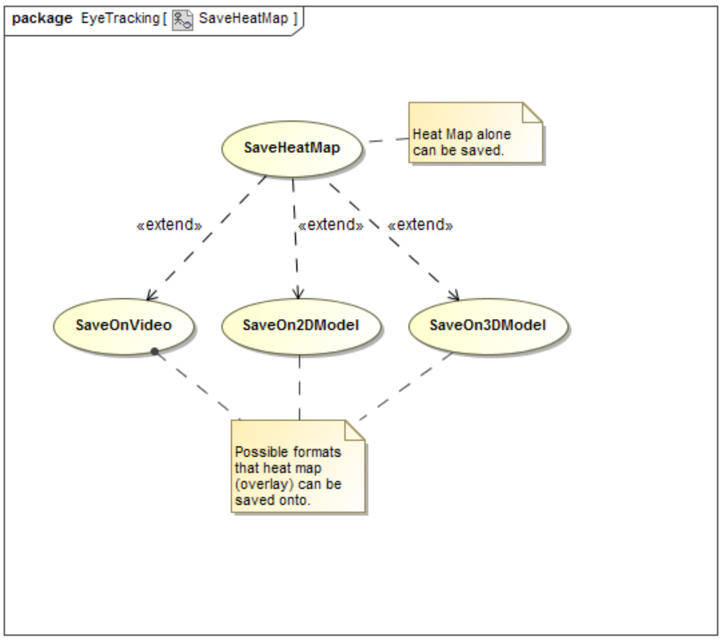
package EyeTracking [SaveHeatMap]
SaveHeatMap
Heat Map alone can be saved.
«extend»
«extend»
«extend»
SaveOnVideo
SaveOn2DModel
SaveOn3DModel
Possible formats that heat map (overlay) can be saved onto.

Figure 5: Save Heat Map Use Case

2.2.1 SaveOn2DModel

The heat map that is created for a 2D model media type will be saved and applied to the 2D model to see the heat map over the model to indicate what has been viewed the most by the users.

- Pre-condition: Heat map must have been created and the 2D model of the heat map must be available.
- Post-condition: Heat map of the data is viewable on the 2D model 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 what has been viewed the most by the users. 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.

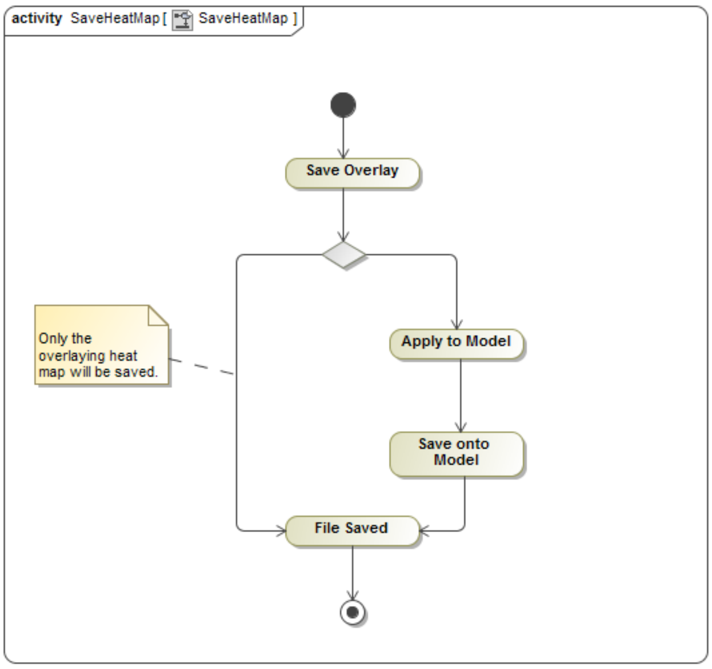
activity SaveHeatMap [SaveHeatMap]
Save Overlay
Only the overlaying heat map will be saved.
Apply to Model
Save onto Model
File Saved

Figure 6: Save on 2D Model Activity

- Request Data Structure: HeatMap.SaveOnImage(heatmapID,ImageID).
- Return Data Structure: Heat map of the data is placed over the video.

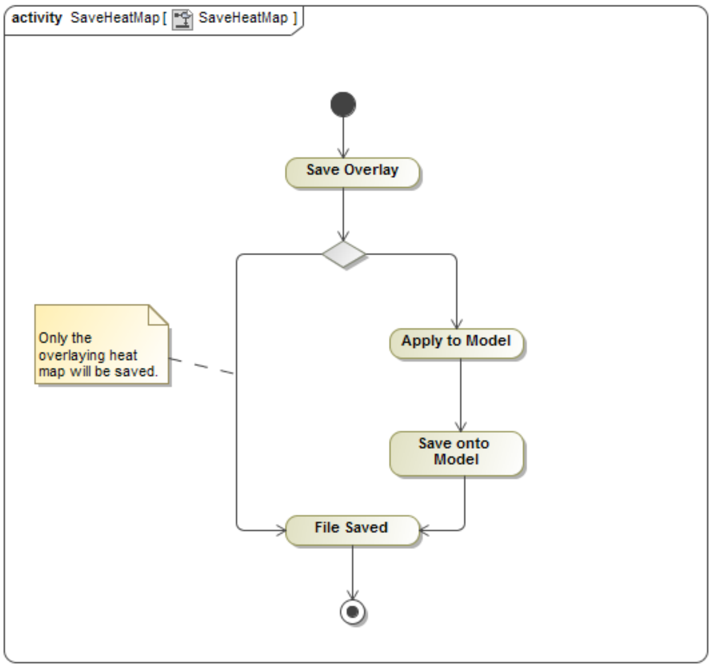
activity SaveHeatMap [SaveHeatMap]
Save Overlay
Only the overlaying heat map will be saved.
Apply to Model
Save onto Model
File Saved

Figure 7: Save on Video Model Activity

2.2.3 SaveOn3DModel

The heat map that is created for a 3D model media type will be saved and then applied to the 3D model to see the heat map over the 3D model to indicate what has been viewed the most by the users. The 3D model with heat map overlay can be saved as either a 2D or 3D model.

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

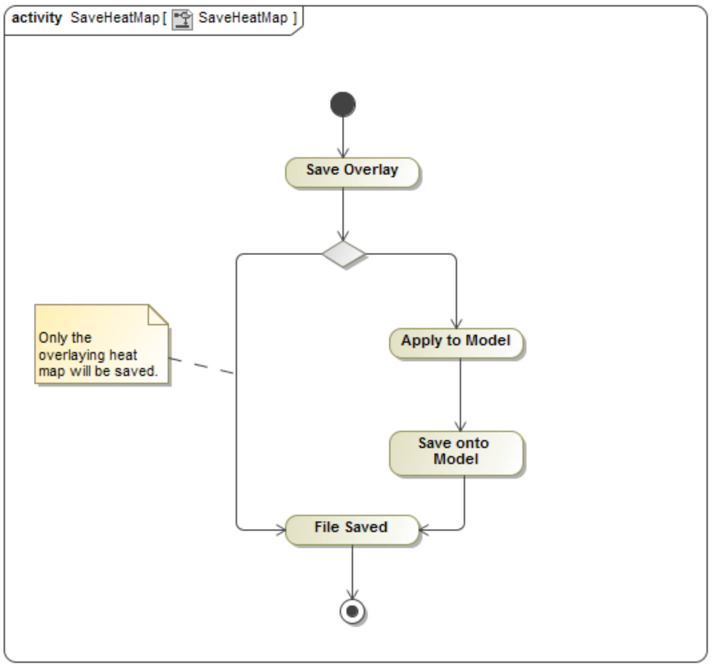
activity SaveHeatMap [SaveHeatMap]
Save Overlay
Apply to Model
Save onto Model
File Saved
Only the overlaying heat map will be saved.

Figure 8: Save on 3D Model Activity

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, namely Video, 2D and 3D. The heat maps are created based on the models and their types.

2.3.1 3D Models

3D Models will be opened and rendered by the system, and will then be made viewable to the user. The heat map can then be saved onto the model as well as opened as the

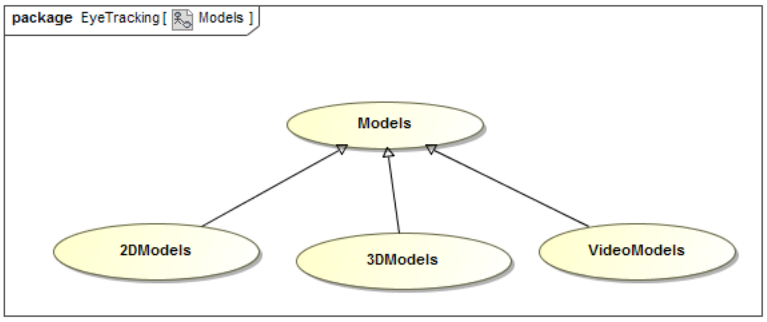

Figure 9: Models Use Case

stand alone heat map.

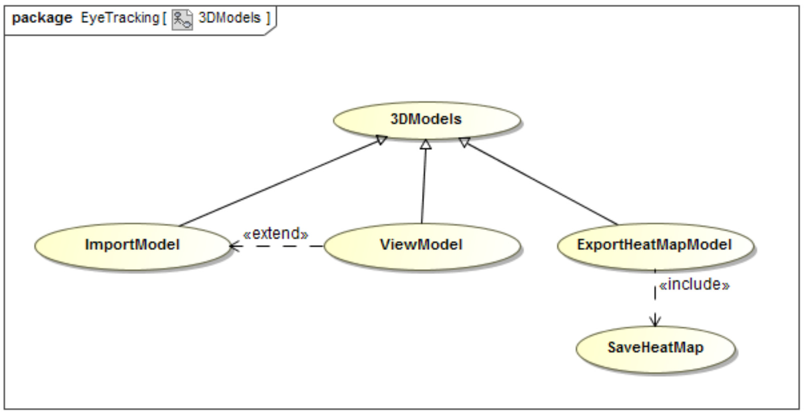

Figure 10: 3D Models Use Case

1. ImportModel
   This will allow for a 3D model media type to be imported into the system for further processing. The imported model can be used for recording data onto and creating a relevant heat map based on it which can then be applied to it at a later 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 model will be rendered for viewing and further processing decided by the user.
   - 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).

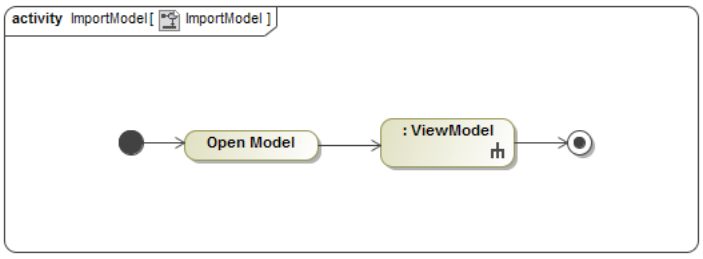

Figure 11: Import 3D Model Activity

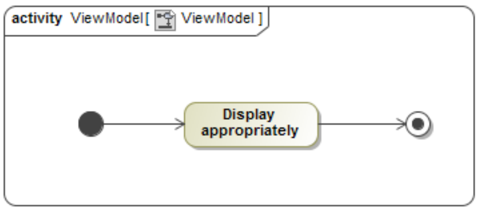

Figure 12: View 3D Model Activity

- Return Data Structure: Viewable 3D model

3. ExportHeatMapModel The exported model will have the heat map applied to it which can then be exported as a new 3D model so that it can be viewed or used at a later date by the user.

    - 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.

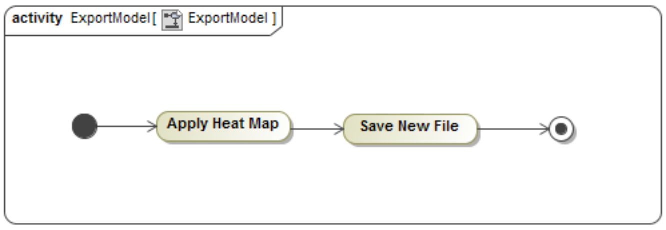

Figure 13: Export 3D Model Activity

2.3.2 2D Models

2D models will typically refer to images but can also refer to any media that is does not show a third dimension of movement but for the scope of this project will only refer to images. The heat map can then be created and applied to the image.

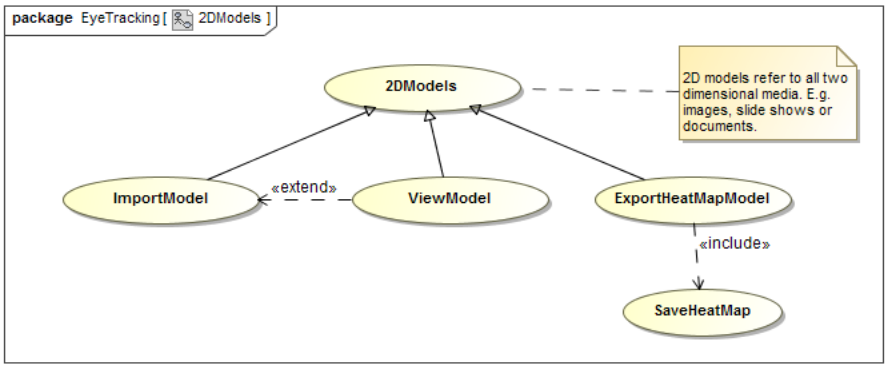
package EyeTracking [2DModels]
2DModels
2D models refer to all two dimensional media. E.g. images, slide shows or documents.
ImportModel
«extend»
ViewModel
ExportHeatMapModel
«include»
SaveHeatMap

Figure 14: 2D Models Use Case

1. ImportModel
   The 2D models can be imported into the system which can then be process further by the user. The imported model can then have recordings done and heat maps created for it.

   - 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.

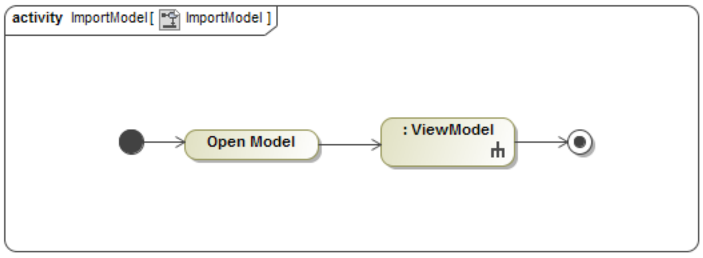

Figure 15: Import 2D Model Activity

2. ViewModel Once the model is imported the users can view the model and eye tracking can then be applied on the model. The model can the be processed further as the user sees fit.

   - 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 exported model will have the heat map applied to it which can then be exported as a new 2D model so that it can be viewed or used at a later date by the user.

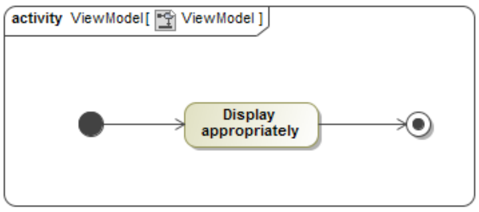

Figure 16: View 2D Model Activity

- 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.

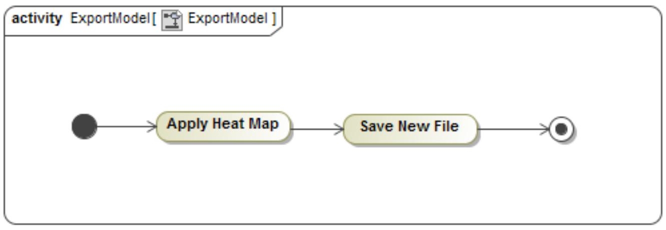

Figure 17: Export 2D Model Activity

2.3.3 Video Models

A video can be easily imported into the system. Eye tracking can then be done to the video model and the recorded data can be stored. The data can then be used to create a heat map that can then be applied to the video second by second.

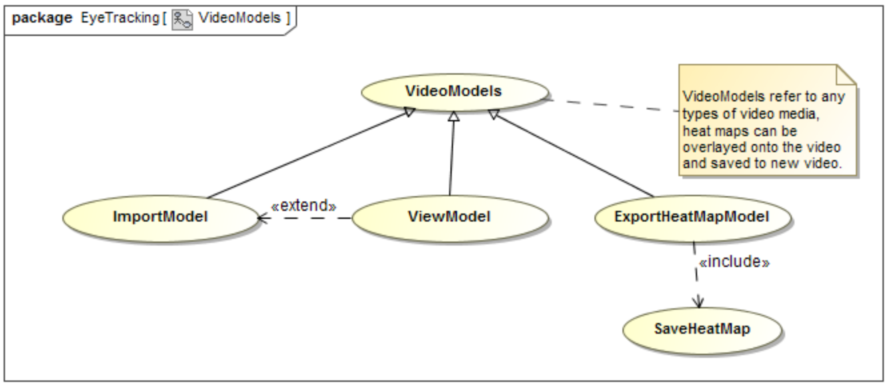
package EyeTracking[VideoModels]
VideoModels
VideoModels refer to any types of video media, heat maps can be overlayed onto the video and saved to new video.
ImportModel
«extend»
ViewModel
ExportHeatMapModel
«include»
SaveHeatMap

Figure 18: Video Models Use Case

1. ImportModel
   The video models can be imported into the system which can then be process

further by the user. The imported model can then have recordings done and heat maps created for it. Heat maps that will be recorded will show a second by second representation of the heat map data.

- 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.

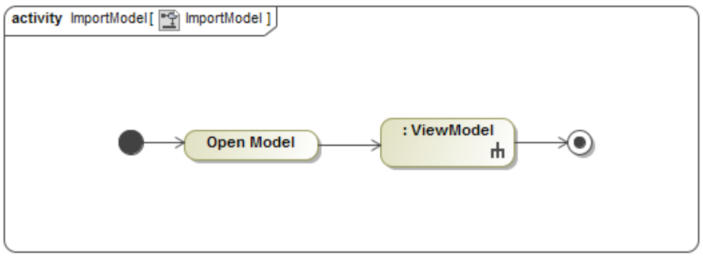

Figure 19: Import Video Model Activity

2. ViewModel Once the model is imported the users can view the model and eye tracking can then be applied on the model. The model can the be processed further as the user sees fit.

3. ExportHeatMapModel The exported model will have the heat map applied to it which can then be exported as a new video model so that it can be viewed or used at a later date by the user. The exported video model will have the heat map recorded onto the original video and saved in the appropriate format.

    - 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 and statistics for specific

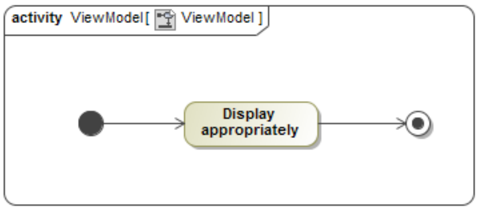

Figure 20: View Video Model Activity

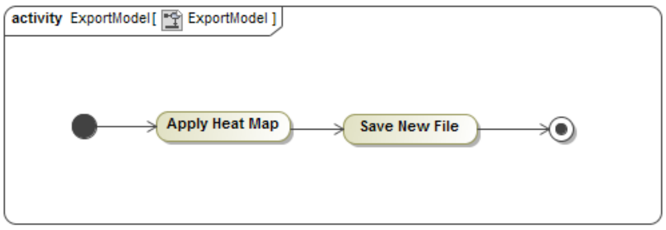

Figure 21: Export 3D Model Activity

models. The raw information is can be saved on the system and processed further by the user. The raw information can then be read as the raw data or put into a statistical formats.

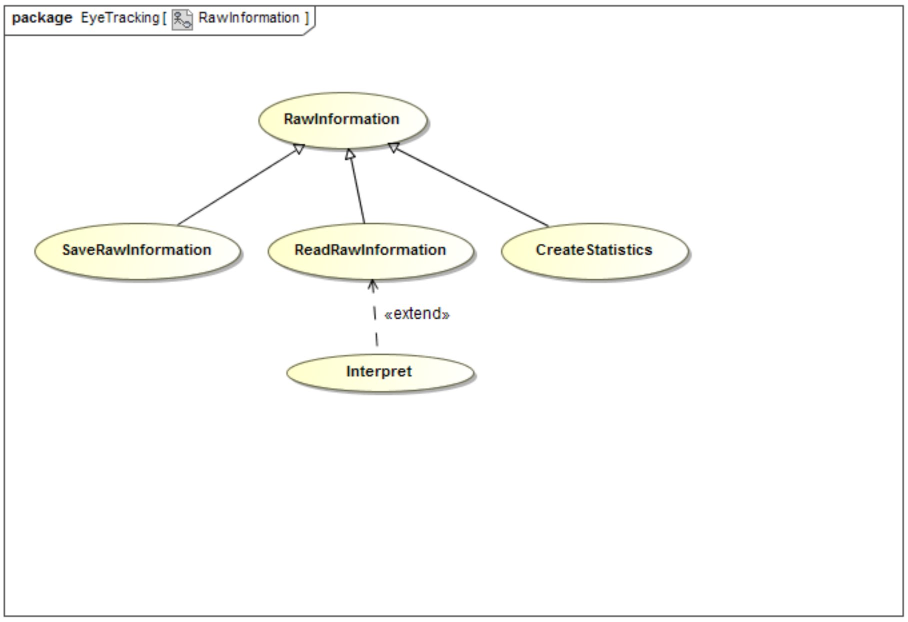

Figure 22: Raw Information Use Case

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 can then be reopened later for the relevant model and used by user for further processing and interpretation.

- 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.

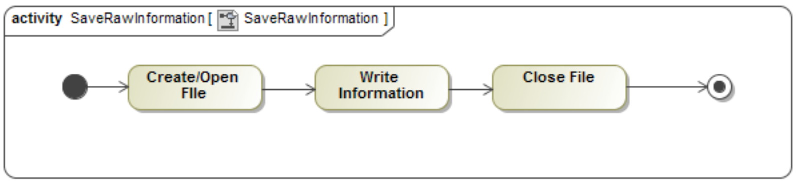

Figure 23: Save Raw Information Activity

2.4.2 ReadRawInformation

Data can be read from a previously saved file. This will allow the user to import the raw information for the relevant model which can then be used for comparison or further processing.

- 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.

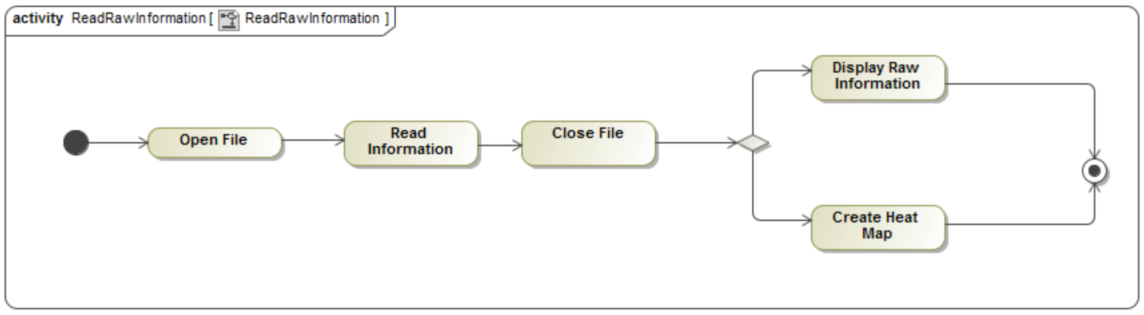

Figure 24: Read Raw Information Activity

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 will be collected and will then be used to make a statistical page based off the model that can be analysed and then it can be easily compared at any time.

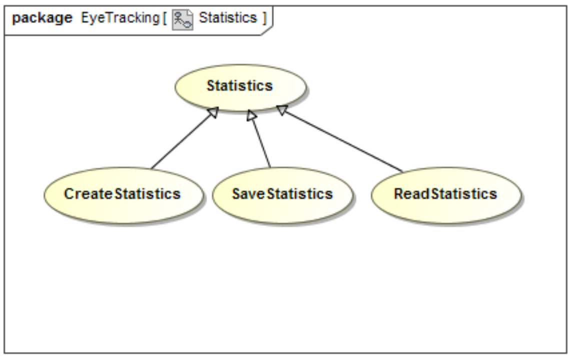

Figure 25: Statistics Use Case

2.5.1 CreateStatistics

Using the recorded data we can create statistical data which can then be used for analysis. This data can then be viewed or 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.

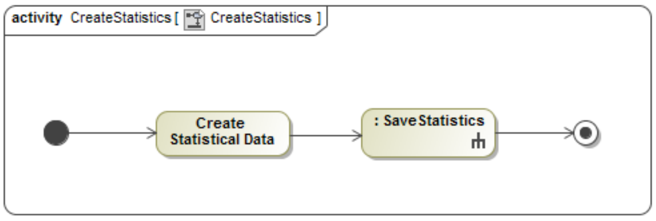

Figure 26: Create Statistics Activity

2.5.2 SaveStatistics

The statistical data can be saved into a file so that they can be printed out and be used as a hard copy. This will take all data and save it to 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.

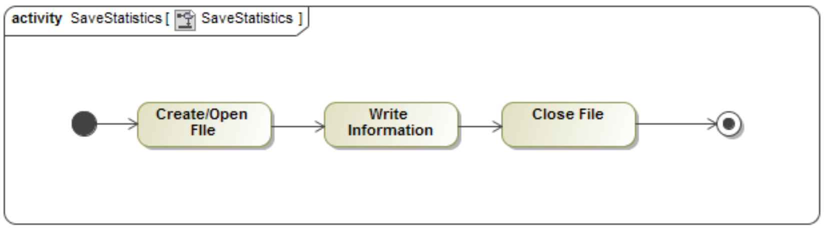

Figure 27: Save Statistics Activity

2.6 Record Eyes

The recording of the eyes will be done using the Eye Tribe camera or any other eye tracking camera. The data recorded in this process will allow heat maps and statistics to be generated. The heat maps can then be turned into overlays for the appropriate media types. The record eye use case (Figure 28) makes use of sub use cases from other use cases that are listed below, such as SaveRawInformation. The recording of the information is important as this is used throughout the project and its functionality.

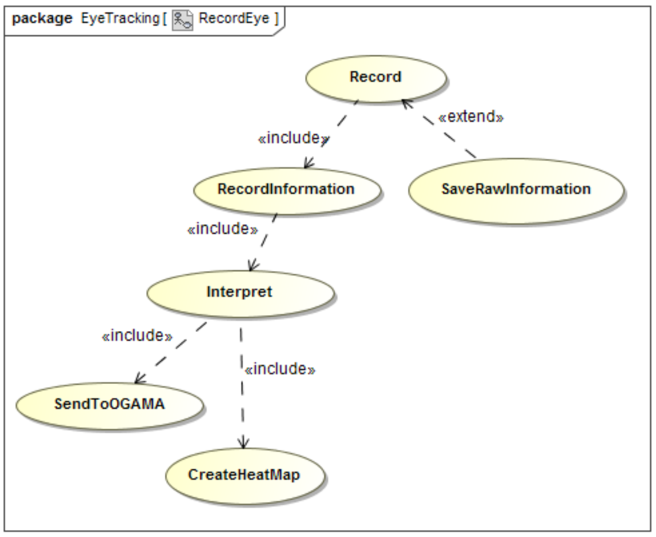

Figure 28: Record Eye Use Case

2.6.1 RecordInformation

The recorded information will be sent to the OGAMA module to be processed and interpreted, which will then return the relevant information which can then 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.

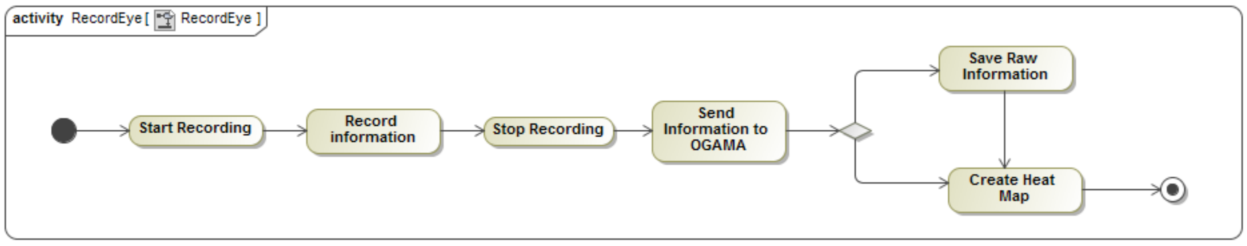

Figure 29: Record Eye Activity

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