A Demonstration of 4D Digital Avatar Infrastructure for Access of Complete Patient Information

Project acronym: MyHealthAvatar

Deliverable No. 8.2
Avatar-Centred Visual Analytics Suite and Evaluation Report

Grant agreement no: 600929

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	MyHealthAvatar
Project Full Name:	A Demonstration of 4D Digital Avatar Infrastructure for Access of Complete Patient Information
Deliverable No.:	D8.2
Document name:	Avatar-centred Visual Analytics Suite and Evaluation Report
Nature (R, P, D, O)[1]	R
Dissemination Level (PU, PP, RE, CO)[2]	PU
Version:	Final
Actual Submission Date:	29/02/2016
Editor: Institution: E-Mail:	Feng Dong BED feng.dong@beds.ac.uk

ABSTRACT:
This deliverable reports deliverable 8.2 – avatar centred visual analytics suite and evaluation report.

KEYWORD LIST:
3D & 2D Avatar Suite, LifeTracker, Dashboard, Diary, MobileApp, Profile

[1] **R**=Report, **P**=implementation, **D**=Demonstrator, **O**=Other

[2] **PU**=Public, **PP**=Restricted to other programme participants (including the Commission Services), **RE**=Restricted to a group specified by the consortium (including the Commission Services), **CO**=Confidential, only for members of the consortium (including the Commission Services)

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 600929.

The author is solely responsible for its content, it does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of data appearing therein.

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Author**
1.0	**25/12/2015**	**Draft**	Prof. Feng Dong
2.0	**26/02/2016**	**Draft**	Stephen Wilson, Xu Zhang, Farzad Parvinzamir, Youbing Zhao
3.0	**29/02/2016**	**Final**	Prof. Feng Dong

Contents

1 Executive Summary

This document reports deliverable 8.2 - Project background

Owing to the highly fragmented health systems in European countries, gaining access to a consistent record of individual citizens that involves cross-border activities is very difficult. MyHealthAvatar is an attempt at a proof of concept for the digital representation of patient health status. It is designed as a lifetime companion for individual citizens that will facilitate the collection of, and access to, long-term health-status information. This will be extremely valuable for clinical decisions and offer a promising approach to acquire population data to support clinical research, leading to strengthened multidisciplinary research excellence in supporting innovative medical care.

MyHealthAvatar will be built on the latest ICT technology with an aim of engaging public interest to achieve its targeted outcomes. In addition to data access, it is also an interface to access integrative models and analysis tools, utilizing resources already created by the VPH community. Overall, it will contribute to individualized disease prediction and prevention and support healthy lifestyles and independent living. It is expected to exert a major influence on the reshaping of future healthcare in the handling of increased life expectancy and the ageing population in Europe. This complies with the priority and strategy of FP7 ICT for healthcare, and constitutes a preparatory action aiming at the grand challenge on a "Digital Patient", which is currently the subject of a roadmap in the VPH community.[3]

The MyHealthAvatar project focuses on research and demonstration actions, through which the achievability of an innovative representation of the health status of citizens, named 4D MyHealthAvatar, is explored. The 4D Avatar is anticipated as an interface that will allow data access, collection, sharing and analysis by utilizing modern ICT technology. It is expected to become the citizen's lifelong companion, providing long-term and consistent health status information of the individual citizen along a timeline representing the citizen's life, starting from birth. Data sharing will be encouraged, which will potentially provide to an extensive collection of population data to offer extremely valuable support to clinical research. The avatar will be equipped with a toolbox to facilitate clinical data analysis and knowledge discovery.

MyHealthAvatar can be described as a personal bag carried by individual citizens throughout their lifetime. It is a companion that will continually follow the citizen and will empower them to look after their own health records. This fits very well into the recent trend of developing patient-centred healthcare systems.

1.1 Avatar Centred visual analytics and evaluation report

This deliverable D8.2 - Avatar-centred visual analytics suite & evaluation report – is a report of task 8.1 Avatar modelling and rendering suite (PM3=>PM18), task 8.2 Key techniques of visual analytics for avatars (PM4=>PM33), task 8.3 A visual analytics suite for the avatars (PM19=>PM33) in WP8 Avatar centred visual analytics (PM3-PM33) . The object of the task is to provide a detailed report of key visual analytics techniques and visual analytics suite for data visualization and analysis in MyHealthAvatar. This report describes the work, achievements and evaluation report of the visual analytics techniques and visual analytics suite which includes 3D avatar, 2D avatar, LifeTracker, dashboard, diary, profile and the mobile app.

[3] MyHealthAvatar project, Description of Work (DoW) document.

2 Introduction

With the rise of wearable sensor technologies, more and more wearable health or even medical sensors have been available on the market - such as Fitbit [Fitbit], Withings [Withings], iHealth [iHealth], etc. - which enables not only people but also doctors to utilise them to monitor people's health status in such a consistent and complete way that the sensors may become lifetime companions of the users. The consistent measurements from a variety of wearable health sensors implies that a huge amount of data from a large number of sensors needs to be processed by observers.

The purpose of health and medical information collection is to promote healthy lifestyles of citizens and to assist clinical decision making. Information analysis is a vehicle by which citizens can make analysis based on knowledge from the platform and medical professionals can augment their clinical knowledge with heterogeneous information from the avatar for clinical decision making and knowledge exploration.

Traditional means of data processing can no longer meet the demands of processing of data from health sensors presently and in the future in such a big volume. Visual analytics provides technologies to promote knowledge discovery and utilisation of big data via mature visual paradigms and well-designed user interactions. As a science of analytical reasoning facilitated by interactive visual interfaces, which should support interactive visual data analysis through a number of visualization means. It plays a significant role in information visualization and has become an indispensable tool in big data analysis. Without proper design of visualisation and user interaction, it is not possible for the user to select, view, understand and gain knowledge from a large collection of data. Consequently, interactive visual analysis is a critical part for the effective utilisation of data collected from mobile sensors and Apps.

MyHealthAvatar believes that healthcare should not only care for patients but also look after the health and wellbeing of all citizens. It needs to be available to healthy people through maintenance of a healthy lifestyle and the notification of early symptoms. Hence, MyHealthAvatar targets both healthy citizens and patients and is designed to be the citizen's lifelong companion, providing long-term and consistent health status information of the individual citizen.

The MyHealthAvatar web and mobile platform is designed to collect and track lifestyle and health data to promote citizen wellbeing. As a lifetime companion of citizens, the amount of data collected will be huge. It is almost impossible for citizen, patients and doctors to view, utilise and understand these data without proper visual presentation and user interaction. Interactive visual analytics of lifestyle data is one of the key features of MyHealthAvatar. This deliverable presents the interactive visual analytics work in both the MyHealthAvatar web and mobile platform to facilitate health and lifestyle data presentation and analysis, including 3D & 2D avatar suite, lifetracker, dashboard, diary, profile in the web platform and overview, statistics views, day views in the mobile app, etc. These visual analytics components are integrated to achieve flexible visual analysis of spatio-temporal lifestyle data in MyHealthAvatar.

3 3D & 2D Avatar Suite

3.1 3D Avatar Suite

3.1.1 Introduction

The visualisation of whole-body anatomy has a variety of applications in health-related analysis and simulation. However, the rendering of complex 3D human anatomy models is generally performed by standalone applications rather than via a web interface, as rendering large 3D models has always been a weak spot of traditional web browsers. Consequently, online access to, and exploration of, the human anatomy in 3D has not been feasible in the past. With the advent of WebGL and HTML5, high performance OpenGL rendering seamlessly integrated with the web interface is now within reach, and this opens the possibility of visualising avatar-centered health data via a web interface. In this section, a WebGL-based 3D Avatar for rendering whole-body anatomy is presented [ZhaoY2013].

The benefits of using a 3D human body model rather than 2D pictures in an avatar web interface include, but are not limited to the following:

- a 3D human model is more realistic and intuitive
- a 3D human model is a natural representation of an avatar
- with user interactions, a 3D human model can represent and gather more information than 2D pictures do
- the user or patient can learn better about his/her health conditions and disease as well as human anatomy.

In the implementation of the avatar rendering suite, we use three.js for web-based 3D graphics as it provides many examples and is easy to start with. In addition, three.js also comes with loaders to load Wavefront OBJ format directly. We also use JQuery [7] for web user interface.

3.1.2 Related Background

A. WebGL

From the early days of the Internet, Web-based 2D and 3D graphics attracted much interest. Java [Java] started as applets in the browser to show interactive graphics. Today, Flash players dominate most browsers' provision of interactive 2D graphics. VRML [VRML] plugins were previously popular for rendering 3D objects in web browsers, but with the evolution of OpenGL [OpenGL] and the advent of HTML5 [HTML5] and WebGL (Web Graphics Library) [WebGL], the 3D graphics capability of web browsers has increased considerably.

WebGL is a JavaScript API for rendering interactive 3D graphics in web browsers. WebGL elements can be mixed with other HTML elements and composited with other parts of the page. The latest versions of Firefox, Chrome and Safari support WebGL, though Internet Explorer will only support WebGL from version 11.

B. Three.js and SceneJS

With WebGL one can write OpenGL programs that render in a web page. However, since version 3.1, OpenGL no longer supports backward compatibility, which implies that the old programming paradigms in OpenGL 1.x and 2.x are no longer favoured, and WebGL users need to write shaders by themselves. This is inconvenient for some programmers who know little about OpenGL shaders.

Three.js [Three.js] is an open-source javascript library, based on WebGL, that is designed to fill the gap by simulating the old OpenGL style programming paradigms such as lighting, material, camera, etc. It is easier to use and does not require knowledge about shaders.

SceneJS [SceneJS], another open-source javascript library based on WebGL, provides a JSON-based scene graph API and supports scene graph management. It was created for the efficient rendering of large numbers of objects.

C. Web-based Human Body Visualisation

There are two major 3D human body websites, one is Zygote Body [ZygoteBody] and the other is BioDigital Human [BioDigitialHuman]. Both provide interactive visualisation of the whole-body male and female anatomy. The model quality of Zygote Body is better than that of BioDigital Human. Zygote Body shows the whole body after model loading, while BioDigital Human displays a skeleton initially and lets the user select other parts to show. The memory management of Zygote Body also outperforms BioDigital Human, and while the latter provides some special visualisations of conditions, none of these aims at health data visualisation.

The user interfaces of Zygote Body and BioDigital Human are shown in Figures 1 and 2 respectively. BioDigital Human uses SceneJS as its WebGL engine.

3.1.3 Model Preparation

We obtained a 3D surface-model set of the whole human body of both the male and the female from an online model vendor. The model set comprises 56 3D Studio Max surface models with textures, each consisting of anatomical sub-object groups identified by their Latin names. The quality of the model is medium but the structure is well organised.

The resolution of the model is too high for direct use in web applications. The loading of the unsimplified models will consume a great deal of memory and will result in the browser freezing or crashing. Consequently, the models have been simplified by processing with the ProOptimizer in 3D Studio Max; sometimes, multi-pass mesh reduction or interactive reduction is needed as automatic processing may not fulfil the task. After simplification, most of the models are reduced to 5-10% of their original size, some even lower. For example, the number of vertices in the skull model is reduced from 222k to 8k.

3.1.4 Model Loading

We use Wavefront OBJ as the model format as plain text OBJ files are easy to read and it has many importers and exporters. Materials and textures are desirable for realistic anatomy rendering, and it supports these well. Three.js provides an OBJ loader with ObjMTLLoader.js and MTLLoader.js. In addition to texture mapping, the code can be easily adapted to support bump mapping, but to save

memory we decided not to enable bump mapping as it almost doubles the texture memory consumption.

In the current implementation, all parts are preloaded. The model names and storage location are retrieved from a database dynamically. There are multiple model files to be loaded and all of them are downloaded in parallel by the OBJMTLLoader. The loading time of the whole model set is dependent on the internet connection and the local cache; it usually ranges between seconds and several minutes. A progress bar is displayed to show the loading progress. Figure 1 shows the 3d avatar skeleton and the user interface of the avatar rendering suite.

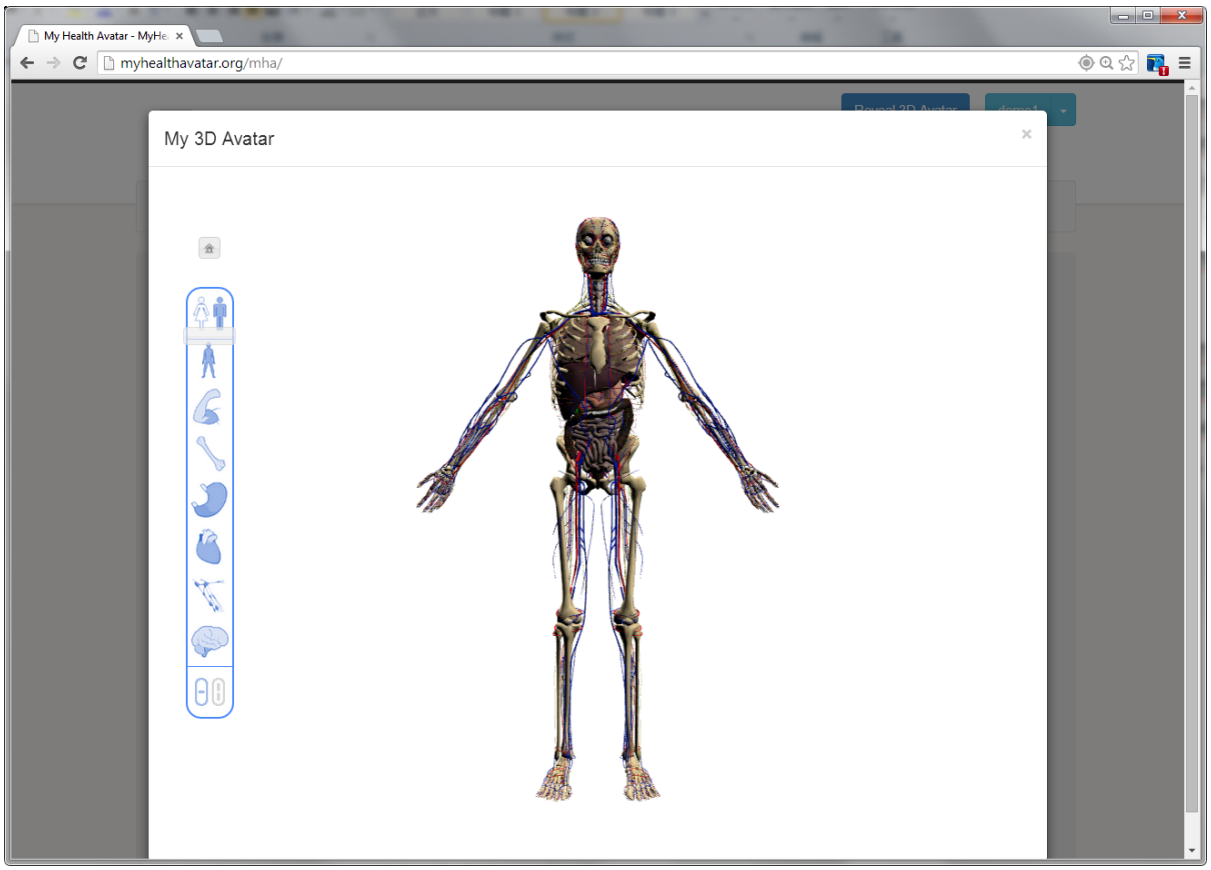

Figure 1 A skeleton after all model parts have been loaded

3.1.5 Interactive Model Rendering

The core components in three.js for rendering are the scene, renderer, camera and lights. All the models that are shown in the browser are stored in the scene; when an object is to be hidden, it is removed from the scene. The renderer is of type THREE.WebGLRenderer and accepts a scene and a camera as parameters to perform the rendering. Lights and camera settings are very similar to those in the old-style OpenGL programming. Trackball controls are attached to the camera to support interactive model exploration with rotating, zooming and panning.

3.1.6 Part Selection

To explore the human anatomy model set efficiently, selective rendering is desired as there are too many body parts and many of them occlude each other.

We use a toolbar similar to Zygote Body to adjust part selection and to set part transparency, as shown in Figure 1. There are two modes for the toolbar, the single vertical slider mode and the multiple horizontal slider mode. Under the single vertical slider mode, the toolbar acts as a vertical toolbar for de-selecting parts from the outside to the interior. Under the multiple horizontal slider mode, the toolbar acts mulitple horizontal sliders which can be adjusted for different parts separately. Consequantly the user can select to show any combinations of parts he desires.

Figure 2 shows a model with a semi-transparent skeleton and non-transparent internal organs after user adjustions.

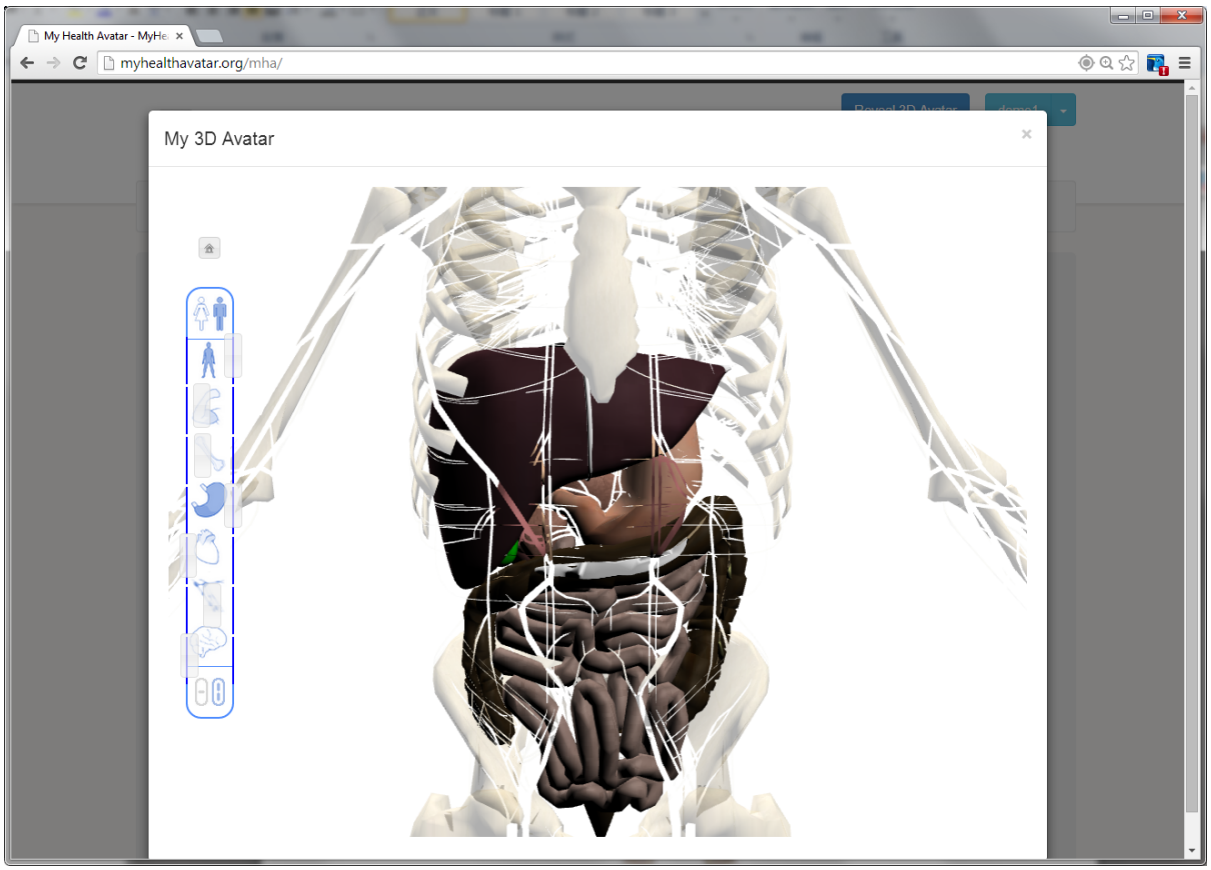

Figure 2 . A semi-transparent skeleton and non-transparent internal organs after user adjusts the slider bar.

3.1.7 Picking and Automatic Focusing

For visual analytics of the digital avatar, displaying body-related health data along with the corresponding body parts is an intuitive and efficient way for both patients and doctors. For example, the colour of a dysfunctional organ can be shown in a colour different from its normal colour. When the patient or doctor picks the part, related information automatically displays. Consequently, for interactive health information exploration, visualisation and analysis, picking is a key step in the human-computer loop.

In our implementation, picking is implemented by the raycaster provided by three.js. All the models in the display are stored in a list which is used by the raycaster together with the position of the user clicks. The raycaster calculates all ray intersections and the first one is the foremost one being picked. A popup message box will show up, which can be used to display health related information;

in the current implementation it shows the name of the picked part. Figure 5 shows the stomach is picked by the user.

When the user picks one part, the 3d avatar suite will automatically zooms to the selected part to fit to and centers on the canvas. This is implemented by a javascript tweening library to animate the camera position and direction to intepolated positions and directions between the origin and target.

Figure 3 shows a highlighted and auto-focused stomach after user picking.

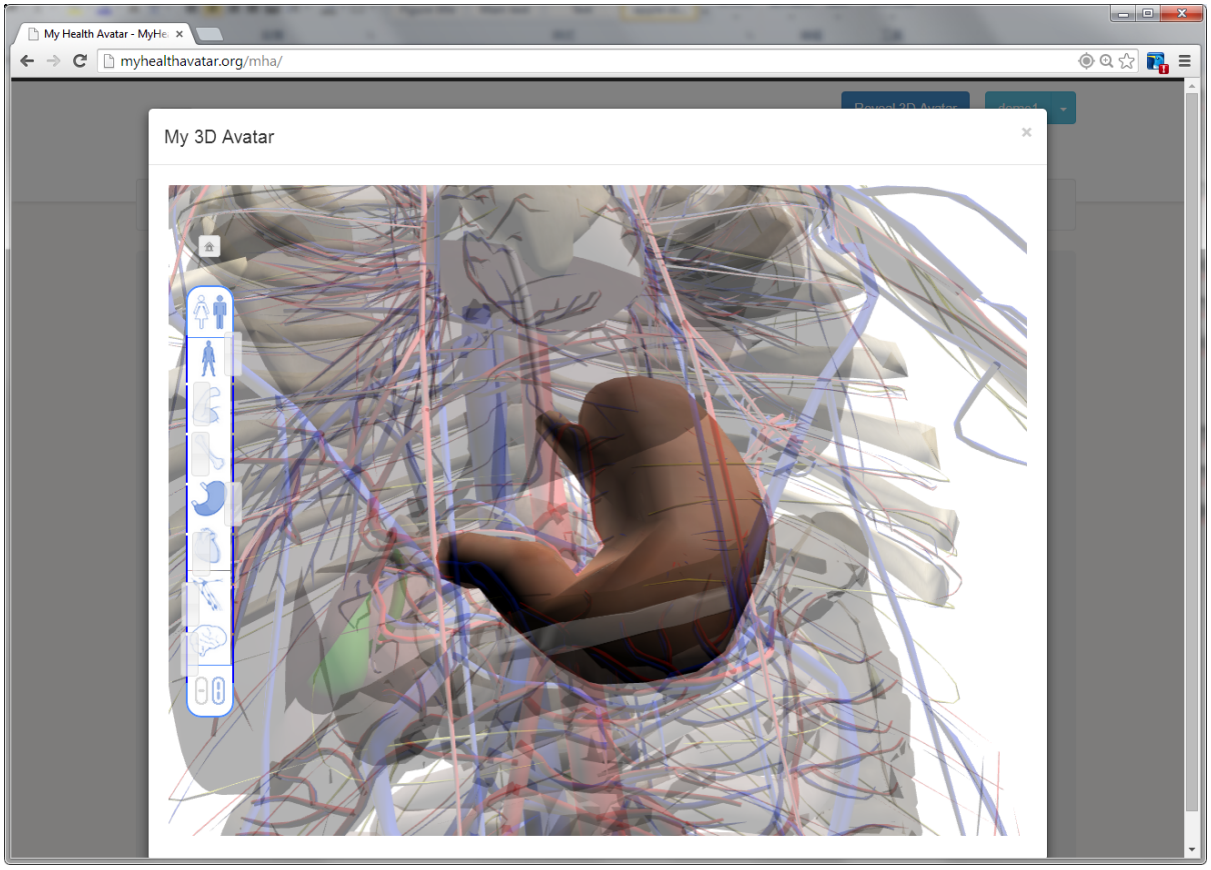

Figure 3 Highlighted and auto-focused stomach after user picking

3.1.8 Memory Management

Most current web browsers do not automatically release all the model memory allocated during reloading and page refreshing, which makes browser memory consumption accumulate and leads to slow-down, freeze or even crash of the browsers.

Memory management in our work comprises two parts: WebGL memory release of geometry, textures and materials by calling the dispose() method in Three.js; model release by setting the relevant model variables to null while page reloading. Test results of an automatically refreshing page in Chrome show that memory consumption accumulation has been removed.

3.1.9 Evaluation

The web-based 3d avatar suite has been integrated into MyHealthAvatar platform and is available to all MyHealthAvatar partners at http://www.myhealthavatar.org. Among the partners, there are

domain experts experts on a variety of subjects, including not only computer science but also medical science, biology, law, etc.

The 3d avatar suite is tested and evaluated in mainstream browsers, including Chrome, Firefox, Safari, Opera, Internet Explorer (version 11). It can run smoothly in all these browsers with WebGL support enabled.

Most partners responded that they can learn how to interact with the 3d avatar without a special learning phase. Trackball interactions with a mouse are intuitive. Picking and interactions with the toolbar are also natural. The 3d avatar gives an overall interactive anatomy model for users and can be accessed via the internet.

Responses from our medical parnters also pointed out the model itself is not very accurate from medical viewpoints and more information can be displayed when the user click on certain organs.

In addition, despite that it has been integrated into MyHealthAvatar platform, the full integration with the health and lifestyle information can only be achieved when the full platform is ready.

3.1.10 Summary

Our web-based 3d avatar shows that, with the support of WebGL, the Internet has become a feasible platform on which to visualise large 3D models such as the complete human anatomy. Embracing this powerful 3D rendering capability in web pages may provide opportunities for a variety of web-based applications, especially avatar-centered health applications, to present more effective and intuitive visualisation of their data to their end users across the Internet.

The web-based 3d avatar suite has been achieved and has been released on the MyHealthAvatar platform [MHAWeb].

3.2 2D Avatar Suite

As stated in previous section 3.1, 3D avatar is a whole-body anatomy visualization which is relatively complex and sometimes intimidate users with its real organ and bones. A 2D avatar is also implemented as a supplement to the 3D version.

3.2.1 Introduction

The 2D avatar is designed to be user friendly and is using a cartoonish design to attract users, the technology used is SVG (Scalable Vector Graphics) and Raphaël.js [Raphaël.js] .

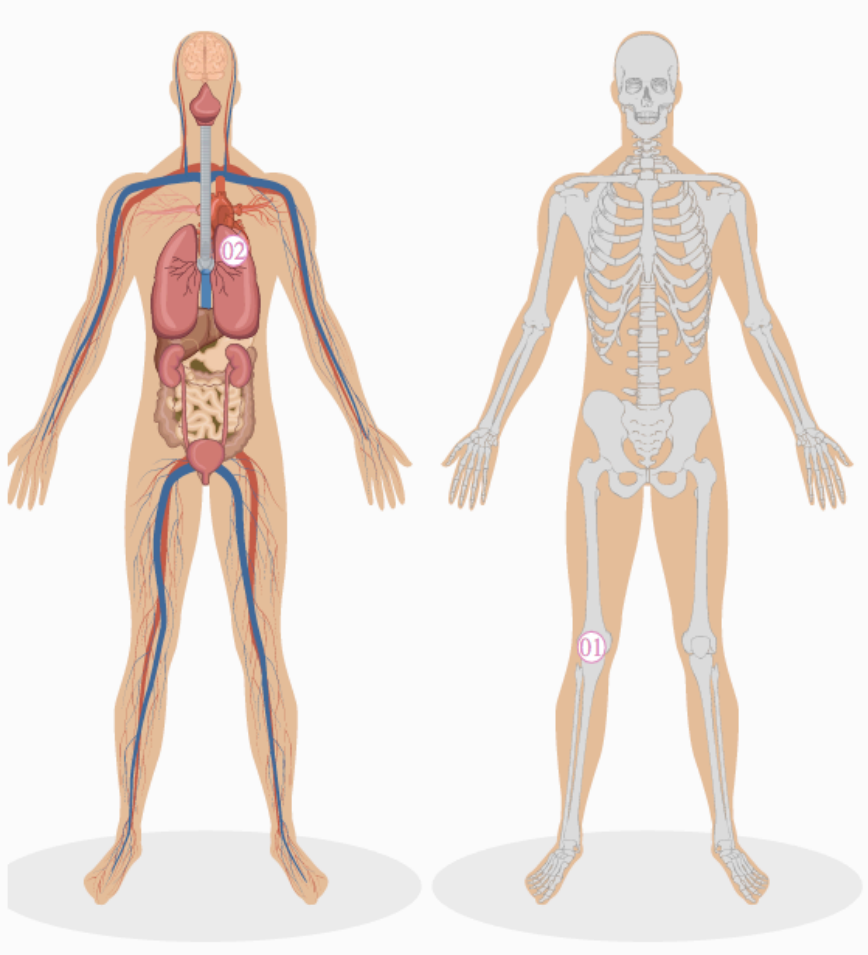

Figure 4 2D avatar overview

SVG is a XML-based language for describing objects and scenes, and SVG elements can fire events, be scripted with JavaScript. SVG contains few useful built-in primitive object types such as texts, circles and rectangles. With recent HTML5 standard, SVG objects can now be embedded directly in HTML page without the need of <object> or <embed> tag. This brings SVG to the same position of HTML5 canvas.

Raphaël.js [Raphaël.js] is a popular SVG manipulate JavaScript library, it allows SVG scenes to be programmatically created. It uses a unified API to create and operate SVG objects when it is supported by client browser, or it will try to use VML (Vector Modeling Language) where SVG is not supported (mainly for Internet Explorer version before IE9).

3.2.2 2D Avatar Interaction with Medical Profile

MyHealthAvatar record user's personal information including medical profile, 2D avatar is a suitable visualization for user's conditions. The visualization will gives user a clear view of the conditions, with interactive interface, the user is able to highlight the related conditions from medical history snapshot section of user's medical profile.

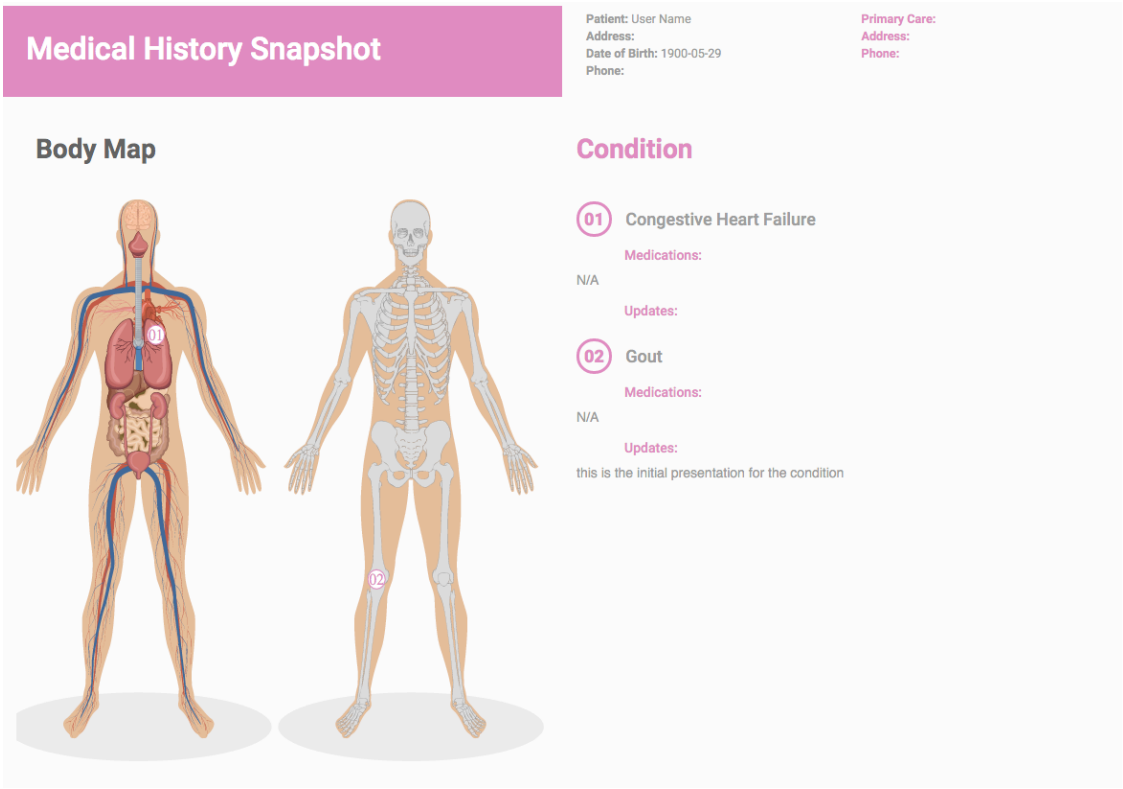

Figure 5 Interaction between Medical Profile and 2D Avatar

As you can observe from above figure, the medical condition of the user (input of import) will be displayed on 2D avatar as circled numbers. The number is placed in the main parts of related condition according to the user's description of conditions. The circle and number themselves are SVG objects, as described before, the browser supports interaction with SVG objects. When user hover the mouse on the circled number, the condition description on the right will be highlight, click on the circle will toggle persistent highlight of description on the right.

3.2.3 Evaluation

Feature	Expected Functionality	Actual
Condition click toggles	Clicking toggle would change toggles appearance and display the relevant condition description on the right	Toggles correctly display their status with highlights. Toggles effectively change background colour of the condition description.
Circle number hover interact	Hover mouse over the circled number on the 2D avatar, should highlight the relevant condition description n on the right.	The right panel's description will change background colour when mouse hover the circled number to allow better focus.

3.2.4 Summary

As a supplement to the 3D avatar visualization, 2D avatar serves its purpose in the medical condition visualization in a user friendly and intuitive manner. The technology stack is explored with SVG interaction with normal HTML contents, more user operations should be allowed in future work.

4 LifeTracker

LifeTracker offers an integrated environment to analyse and present information about an individual's daily activities by using different views – see Figure 6. This is a highly interactive environment as users are able to select an arbitrary day, month or year to view their activities, events and other relevant data.

The interaction within LifeTracker is designed to offer the functionality of "overview", "zoom and filtering" and "details on demand". By overview, users can gain a perspective about their overall activities over a selected period with key events highlighted as alerts. By zoom and filtering, users can select the timescales ranging from years to days and choose the data they wish to view. By details on demand, users are able to explore more details in various ways, for example, they can use tooltips and pop-up windows to display brief information of a selected data segment; they can use the multi-layered timeline to view occurred historical events down to the scale of days; they can also obtain detailed information about their movements on a selected day through the map view coupled with time information.

LifeTracker features a novel integration of a range of analytics and visualisation techniques to support effective exploration of individual lifestyle and patterns from a sequence of self-logging data over years, including:

- A multi-layer timeline allowing for a user to easily define a time period of day, month or year by clicking on the timeline to view activities and events that occurred in the selected time period.
- Techniques for the extraction, summary and ranking of daily events to highlight important information for a given time period, such as key events and life patterns of the user.
- Interactive visualisation allowing for full exploration of information by the users through a number of integrated views including the map, calendar, life patterns, chart, statistics, etc.

4.1 Overview

LifeTracker is designed for individuals to view and explore their life events and activities. The visualisation in LifeTracker follows the principle of "overview first and details on demand". This is achieved by a multi-layer timeline which allows the users to select a time/date point of interest, at a desired timescale (i.e. year, month or day). Overviews are provided to allow users to gain an overall understanding of their life patterns and changes over different time periods. The users are able to explore more details at a finer scale by using mouse hover and clicks. The information is presented through a number of integrated views (see Figure 6), including:

- Life pattern view: show the daily life patterns of a selected day, or the summarised daily life patterns of a selected month or year
- Event list view: show a list of events that occurred within a selected time scale and period.
- Map view: show locations or movements of selected events on a map
- Chart view: show lifestyle and health-related data in charts, such as step count, walking distance, etc.

- Log view: show user uploaded photos or files placed along the timeline
- Statistics view: show statistics of relevant data within the defined time scale and period.

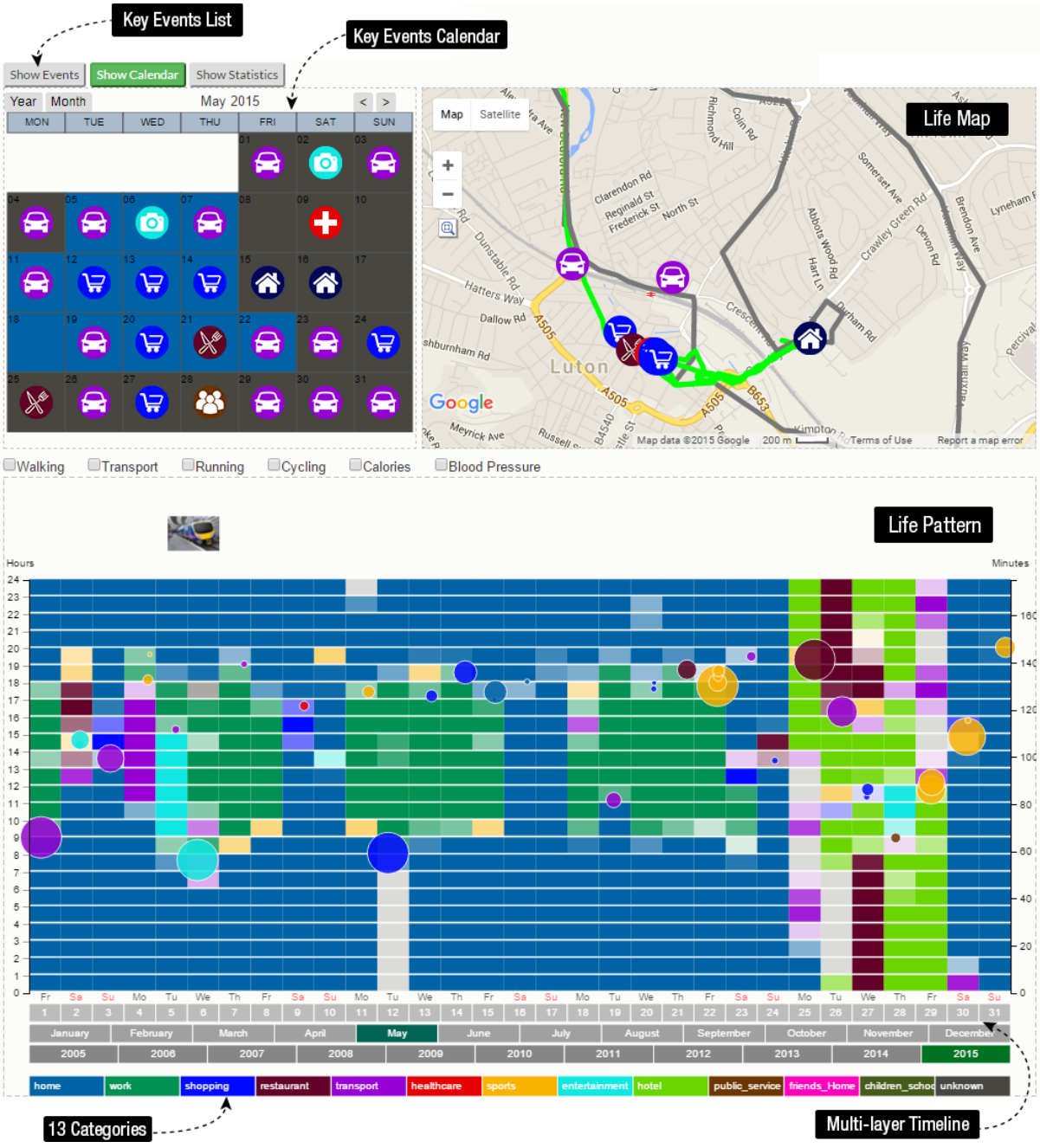

Figure 6 LifeTracker interface consists of calendar, map, and life-pattern

LifeTracker is empowered by a number of techniques for data analytics, including activity extraction, to allow for the recognition of daily activities according to the geo-location and movement data, data ranking techniques for the discovery and highlighting of important data and activities such as the highest step count in a year/month, the results of medical tests, hospital visits, etc., and data summaries at multiple scales, including year and month.

4.2 Data Analysis

Activity Extraction, summary, and data ranking Have been covered in Deliverable D6.4.

4.3 Interactive Visualisation of daily activities

The visualisation and interaction of LifeTracker are carefully designed to maximise its usability, and hence to support user exploration of the data requiring only a very few mouse interactions.

4.3.1 Multi-layer timeline

The timeline in LifeTracker has three layers called year, month and day layers, respectively, representing the time at different scales including year, month and day – see Figure 7.

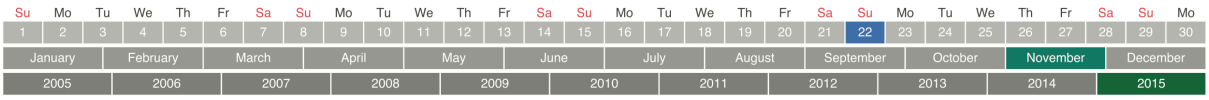

Figure 7 Multi-layer timeline used for selecting the range of time

By default, LifeTracker is set into the LongMode, which presents an overview along the entire timeline. The user is able to perform the following interaction in order to view the data at different time periods.

- YearMode via year selection -- a year selection can be made by clicking on the year layer on the timeline. By this, the time axis is set to the YearMode and displays all the events and data across the 12 months of the year selected.
- MonthMode via month selection -- a month selection can be made by clicking on the month layer of the timeline following the selection of the year.
- DayMode via day selection -- a specific day can be selected by clicking on the day layer following the selection of the month and year.

4.3.2 Calendar view

In LifeTracker, a calendar is placed next to the map. The calendar can be set in two different modes: in the monthly mode, the calendar shows day summary information in colour, together with any key events that occurred on the day; in the yearly mode, it shows monthly summary information in colour together with the key events that occurred in the month. An example is shown in Figure 8.

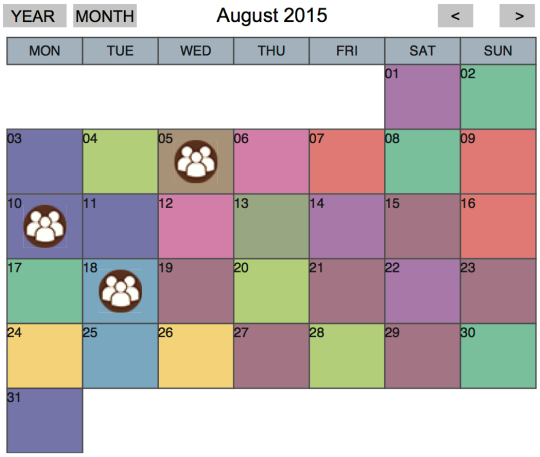

Figure 8 Calendar view shows the summary of each day

The multi-layer timeline is completely synchronised with the calendar -- when the user selects a day from the calendar, the timeline will be also switched to the day in the DayMode.

4.3.3 Life Pattern view

The Life Pattern View is designed to display daily life patterns of the individuals across 24 hours and uses a 2D matrix view, in which the x direction shows the time (i.e. day, month or year) and the y direction shows the 24 hours of a day. Each box in the matrix represents an hour in the day, month or year, and is coloured according to the major activity that took place within the hour. The colour code is given in Figure 9.

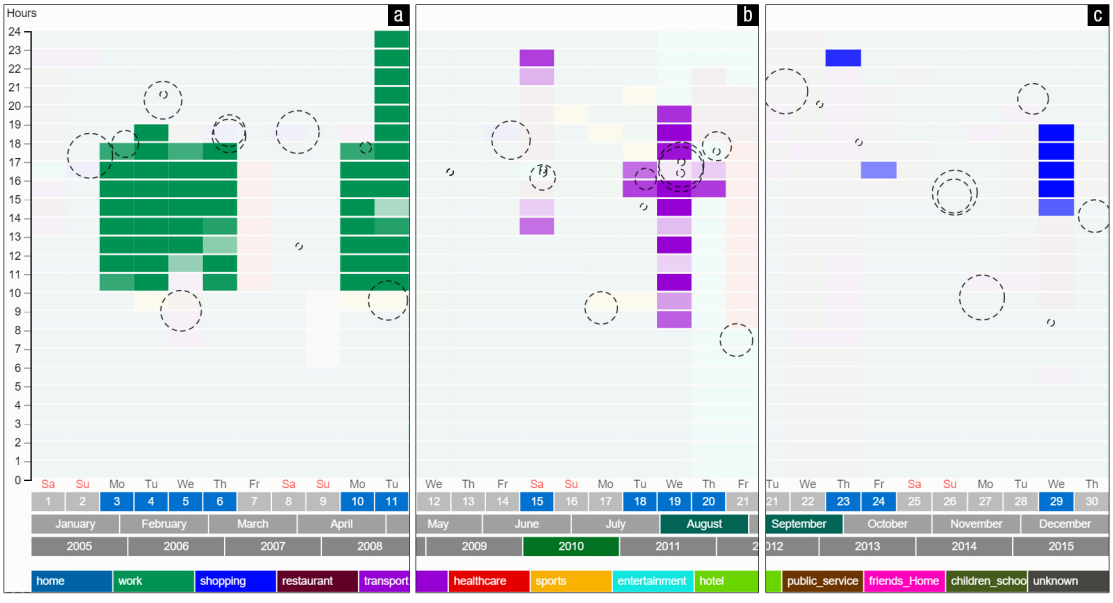

Figure 9 Life Pattern can filter, highlight, and show the specific daily activities by interacting with the provided legends. Hovering the mouse can help user refine the pattern accordingly. a) shows exclusively the user's working pattern during a specific month

The Life Pattern View is presented at multiple time scales. The time scale is defined by the user selection on the multi-layer timeline, and it can be presented either in the year (LongMode), month (YearMode) or day (MonthMode).

item When the timeline is set in the LongMode, a box with coordinate (*i,j*) represents a year summary of the *i* hour within year *j*. The year summary calculation is explained in the above section.

- When the timeline is set in the Year mode, a box with coordinate (*i,j*) represents a month summary of the *i* hour within the month j of the year. Similarly, the calculation of the month summary is given in the section above.
- When the timeline is set in the Month mode, a box with coordinate (*i,j*) represents the *i* hour of day *j* of the selected year and month
- If the user select a day from the timeline, Clock views are used to illustrate the activities on the selected day. The activities are colour coded and placed along the two clocks in order to show the activities within different hours.

In addition, equations $CIMP\_MH_j^m = \sigma_j^m$ and $CIMP\_YR_j^y = \sigma_j^y$ are used to highlight the key events by adding circles on the relevant boxes – see Figure 10.

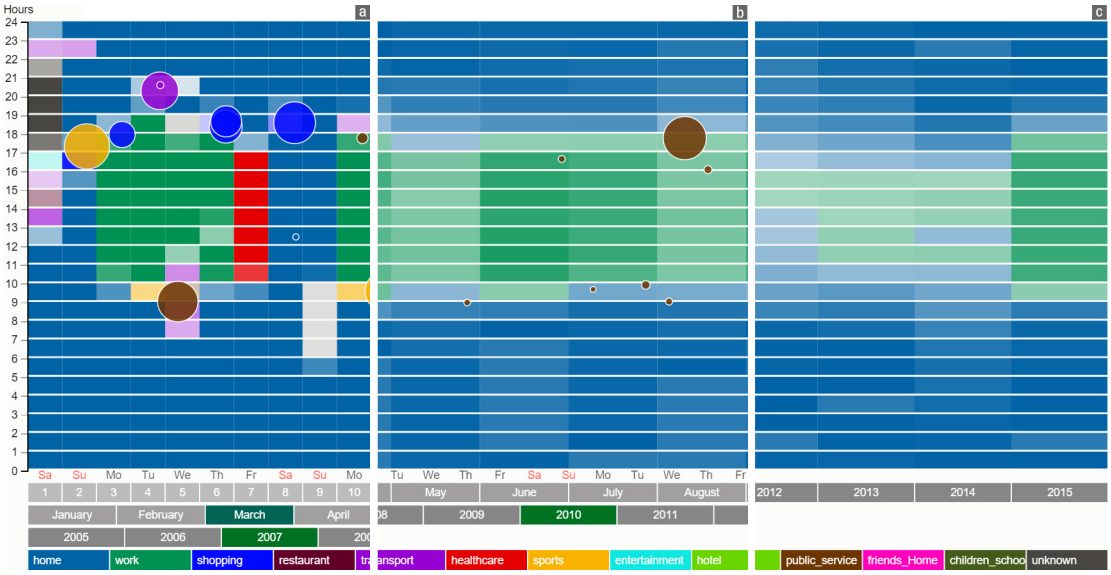

Figure 10 LifeTracker shows individual life patterns on hourly basis at different levels of timescales in matrix views. The y axis represents 24 hours, and the x axis shows the selected timescale. Each element in a matrix shows the hourly activity using pre-define

The Life Pattern view is designed to be highly interactive. The users are firstly presented with the overview (i.e. summary) with key hours highlighted in the LongMode. To explore more details, the users may:

- Use mouse hover on the key event circles for a tooltip that displays more details of the key events.
- Use mouse click on the key event circle to view the key event on the map as well as in the key event list view.

- Use mouse hover on the legend for colour coding to see all the corresponding hours in the view
- Use mouse hover on the events marked on the clocks for more detailed information of the activities
- Use mouse click on the events marked on the clocks to view the events on the map.

4.3.4 Event List view

he Event List view presents events in a list. The list can be ordered at the preference of the users (e.g. according to dates, alphabetical order, etc.). If the user selects an event from the list, the selected event will be shown on the map and s/he can also see an indication of the event on the Life Pattern view.

The Event List view is designed for the display of two types of events. In the LongMode, Year and Month modes the events identified as important are displayed. In Day mode, for a date specified by the user, all the events that occur during a day are presented.

The key events are also highlighted in the calendar, and users may select a key event to view from the calendar.

4.3.5 Map view

The map view is used to show daily activities and locations of the key events geographically. More specifically, it uses

- Straight lines to show the movement path
- Heatmap to show the level of activities at different places - red indicates more frequent activities while green and yellow are used to show areas that are less active.
- Icons to show the key events of the selected time period (via the multi-layer timeline).

The users are able to perform a range of interactions within the Map, including

- Mouse click to explore more information of events and places on the map
- Zoom in/out on the map; zooming in can also be achieved by selecting over a region of interest.

4.3.6 Chart view

The Chart view is designed to show the change of data such as step counts, travel distance, calorie consumption during the selected period of time by using histograms and curves. These histograms and curves are displayed as an overlay on top of the Life Pattern view - see Figure 11.

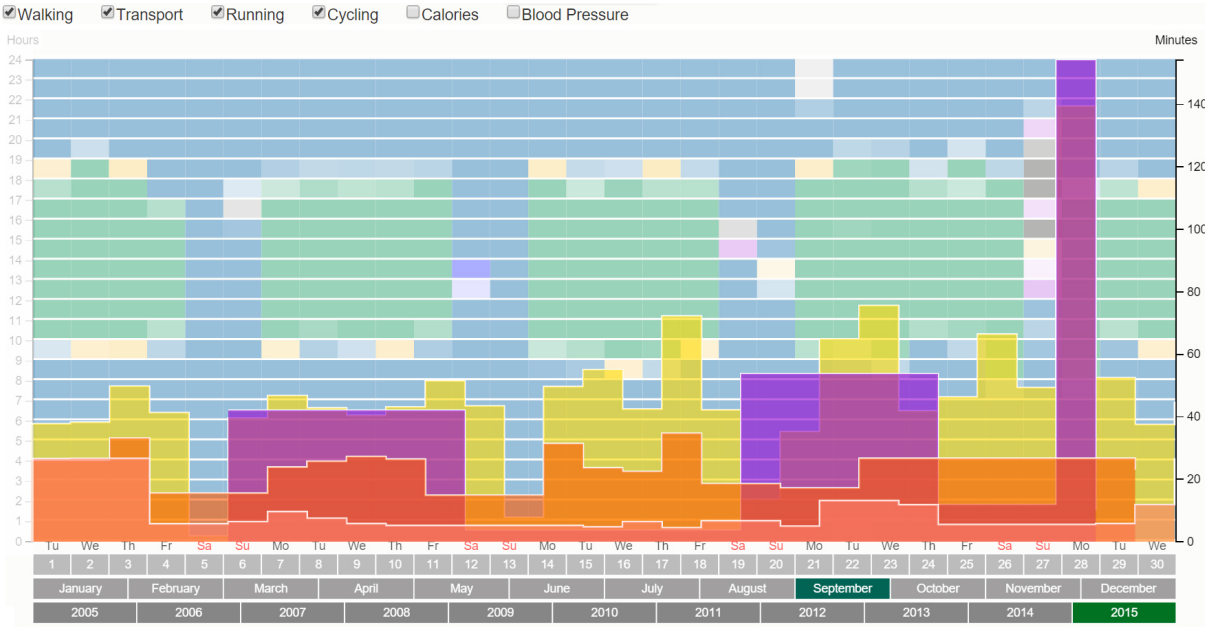

Figure 11 The chart view displays the change of the data and the content of the data can be filtered through the provided facets on the top

4.3.7 Log view

The Log view is designed to allow the users to have easy access to photos or other data files associated with the selected time period. Once a time period is defined via the multi-layer timeline, icons representing the data files that lie within the defined time period are placed along the timeline. The users can click these icons to show more information and to download the files, where necessary.

4.4 Evaluation and Results

A survey was conducted to investigate the effectiveness of LifeTracker in terms of its support to individuals in understanding their life patterns and exploring key events by asking the participants to complete a number of typical tasks. Their performance was recorded and analysed.

In total 15 volunteers were recruited -- 12 males and 3 females, aged between 18 to 60, with different educational backgrounds including business, IT and computer graphics (8), data analysis and English literature. The participants with a computer graphics background had a high level of IT skills but none of them had used any life tracking devices in the past. All of the participants had normal vision without any colour deficiency. The evaluation was conducted via PCs (Intel Core i7 CPU, 32GB Ram, and 8GB Graphic memory) with standard 24-inch desktop monitors. Standard mice and keyboards were provided for the evaluation. The tests were run using the Chrome browser full-screen without any distraction.

4.4.1 Methodology and Procedure

The study began with a brief tutorial to the participants about LifeTracker. The participants were then asked to complete 7 given tasks and their time spent on these was recorded. After their answers were submitted, the participants were asked to complete usability questionnaires. In addition, two participants were interviewed.

The evaluation was designed in three parts: task implementation, usability evaluation and interviews. Seven study tasks covering key functionalities offered by LifeTracker were created, including:

- *Searching for places/visits/activities:* these tasks ask an evaluator to find places visited or activities conducted within a given time period (e.g. day, month or year).
- *Understanding daily life patterns:* these tasks ask for daily life patterns from the data, such as the daily activities during weekdays or weekends, average time of starting and finishing work, etc.
- *Searching for important events:* these tasks search for important events that are identified by the LifeTracker system.

Tasks	Questions
T1	Provide a list of all the overseas visits in July 2015
T2	Provide a list of all the visits to health centres in 2015 (inc. date, time, duration and location)
T3	Provide a list of all the activities conducted on 22/08/2015 in chronological order. (inc. places visited and transport taken)
T4	Look into the daily life pattern during the weekdays in May 2015 and provide the following information: the average start and end time of the work; how many days he did NOT stay at home overnight?
T5	Provide a list of yearly important events of 2015 by describing them in term of dates and categories (one item for each category only).
T6	Find out the longest and the shortest walking duration in the last 5 years and write down their dates and time.
T7	Find out the number of days and the total duration of the shopping activities in March 2015.

For each task, the user was expected to provide a straightforward answer to the associated question, e.g. the number of visits to healthcare centres. The accuracy of the answers and the completion time of the tasks were recorded in order to analyse the performance. Table above shows a list of the tasks involved in the evaluation.

The usability questionnaires were designed to test a number of usability aspects - Functionality, Efficiency, Usability, Reliability, etc. The participants were asked to provide the answers on the scale of 1-5 after their completion of the 7 tasks mentioned above. Answering these usability questions were not timed.

The interviews were used as a means to further explore the user experience and to obtain their feedback to the system, which are difficult to capture using multi-choice questionnaires. We conducted the interview with two selected interviewees -- a male aged 50 (high IT skills and IT background) and a female aged 27 (moderate IT skills and business background). They were asked to explain how LifeTracker would helps them identify hidden parts of the data and if it provides sufficient insight. All the comments were used to improve the usability of Lifetracker.

4.4.2 Results

Accuracy

Overall, the participants achieved 78% accurate answers to the tasks. Figure 12 provides more detail.

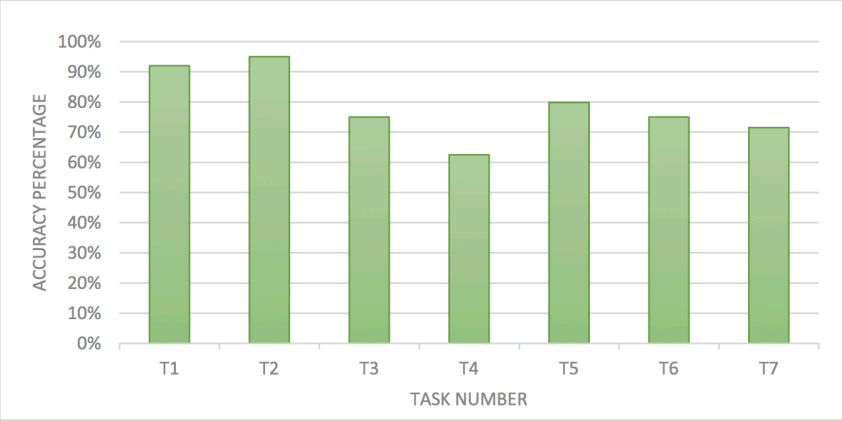

Figure 12 Tasks accuracy in details

T1 and T2 had the highest accuracy, with participants able to use the combination of different views to finish them successfully. We observed that some of the participants tried to double-check their answers via functionalities that we had not expected. T3 needed more focus on the daily activities in which more than 15 items were involved. The participants completed this task in 2 different ways. 78% used the daily ClockVis to get hold of all the activities while 22% answered the task via sorting the event list according to time. The second group was much faster in writing the summary of the day but they had to click more frequently in order to see information regarding specific events, whereas the users who used the ClockVis found the information by hovering the mouse.

T4 required the participants' attention on two different aspects at the same time. We learnt that the legend interaction did not sufficiently support the user needs during the task. The accuracy on T5 satisfied our expectations. Most of the participants used the life pattern view and the detected rare-event bubbles to complete the task. In T6, the participants faced unfamiliar interpolation and needed more focus to complete the task. We understand the interpolation was slightly misleading to the users to find out the exact day of an event but this did not have any effect on finding the event duration. Although, T7 was performed smoothly, we found some wrong answers to day counting.

Performance

The average performance for completing tasks correctly was analysed and the minimum and maximum time for completing each task was found. Figure 13 shows the range of the times over which the participants completed each task.

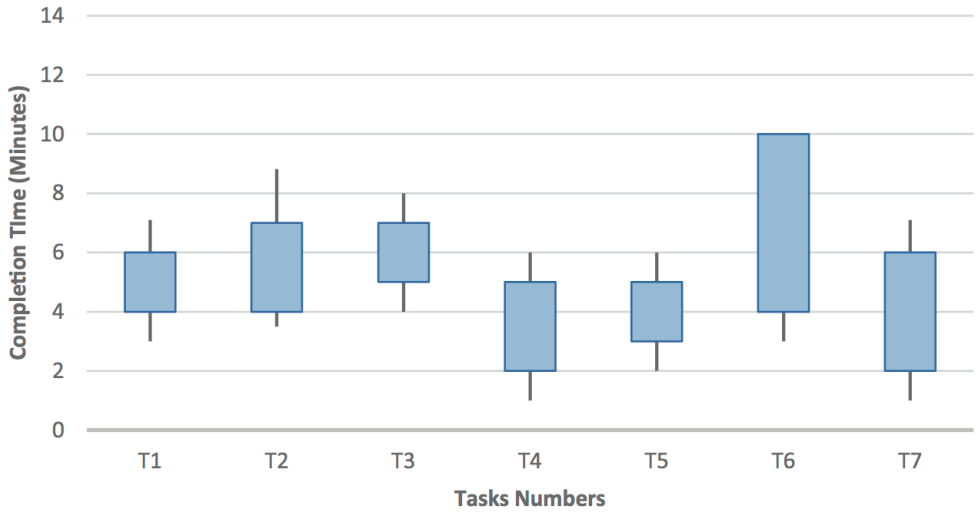

Figure 13 The completion time range for each task plus the expected minimum and maximum time

On average, each task required between 4 to 5 minutes, and the most time-consuming task was T6.

Usability

The usability survey consisted of 4 different parts. Each part is rated between 1 (Low) to 5 (High). In the following presentation we transfer the score into the percentage.

Functionality: 80.3% of the participants were satisfied with the LifeTracker functionalities in general. The non-expert participants needed more assistance with using the functionalities provided properly. Figure 14 gives more detail.

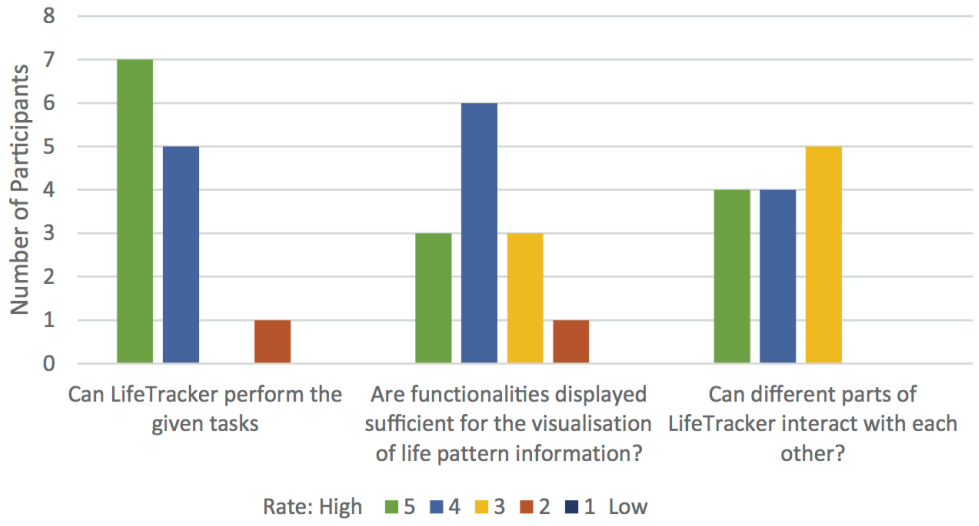

Figure 14 Functionality of LifeTracker rated by participants

Efficiency: to some extent the efficiency was affected by two factors, age and IT skills. Unsurprisingly, young users with higher IT skills finished the tasks faster than the others. The overall efficiency came to 83.75%, which met our expectation Figure 15.

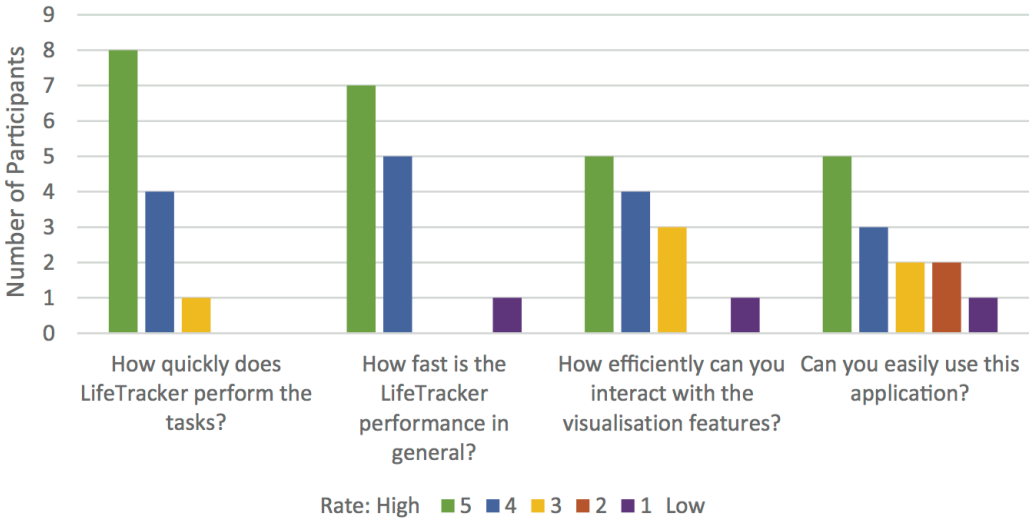

Figure 15 LifeTracker efficiency based on the participants rate

Usability: The average score was 71.8%. We learnt from the comments that some users felt that the options provided were a little overwhelming.

Reliability: We asked the participants if they encountered errors during the evaluation session. Only the participants with the high IT skills answer this question. On average LifeTracker achieved 87% on reliability.

4.5 Discussion and future work

LifeTracker provides an overview, and detailed life patterns in 24 hours based on 13 pre-defined activity categories. We use a 24x13 matrix to analyse the data and fit them to the associated hours and categories. LifeTracker uses colour, alpha intensity, and glyphs to perform the visualisation. It displays the dominant activity during the specific hour of a day while storing all the other activities in matrices.

Based on the evaluation results, further improvements can be performed on:

- the event-detection engine to assess data based on the user preference to provide more effective results;
- the colour schemes for the activity categories
- the pre-computation of data to deal with large-volume daily logged data

There are many ways in which our approach can be extended. The event detection can be made customisable based on user preferences (e.g. some of the users see transportation as an important event whilst the others may not). Also, we can offer more controls to show/hide the desired functions; this could help users to become familiar with subsets of the system before moving to full usage, thus making the initial experience less overwhelming.

5 Dashboard

5.1 Dashboard

There are many components and data that can be accessed by the user from the web-based MyHealthAvatar web portal. However, as there is a variety of data sources and data types, it is very difficult for a user to grasp an overview with important notifications from the scattered health status information. To present the user a quick overview of their health status, MyHealthAvatar provides a dashboard as the front page. The dashboard provides a summary of the user's latest health status and may present important notifications. It may include several simple visualisation components to depict data for a relatively recent period. Figure 16 shows the example dashboard with data tiles, map and a timeline. The user can interact with the map and the timeline to obtain more detailed information.

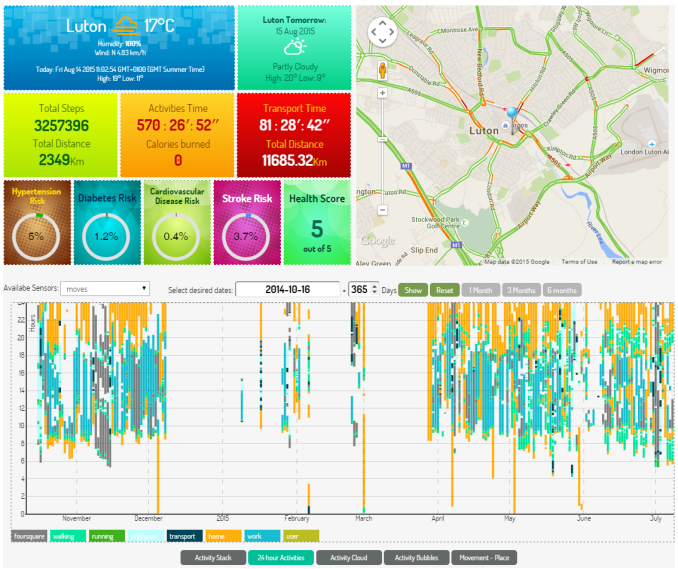

Figure 16 Example screenshot of dashboard

5.2 Timeline

In the dashboard there is a daily activity timeline. A timeline is a traditional method to visualise time-varying data and events in a linear layout. Compared to a calendar, a timeline is more suitable for visualising continuous variables which cover a relatively long period, such as health indicators and medical measurements. Activity events which are time dependent can also be shown in a timeline if a longer time scale is desired to view daily activity events and activities. In the current implementation, the timeline supports interactive visualisation of Fitbit/Withings sensor data as well as Moves data. There are five different visualisation styles including activity stack, 24-hour activity, activity cloud, activity bubbles and movement-place. Activity stack shows activities directly on the timeline in a form similar to stack bar charts. A 24-hour activity organises the activities on a daily basis for easier comparison of daily activity changes. The activity cloud uses concentric disks of different radius to represent the activities; activity bubbles use bubbles of different colour and radius. Movement-place shows the movement and place in the user's Moves data.

In addition to interactive time period selections and zooming, the timeline supports interactive filtering and automatic clustering of events as the number of events may be too large for web-based applications. **Error! Reference source not found.** shows an example of daily activity events visualised in a timeline.

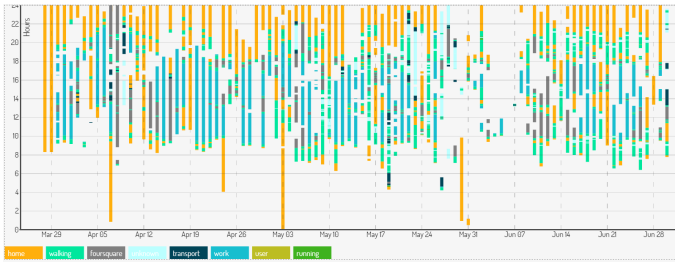

Figure 17 A timeline view of daily events in a 24-hour style

6 Diary

6.1 Introduction

The purpose of health and medical information collection is to promote healthy lifestyles of citizens and to help to assist clinical decision making. Information analysis is a vehicle by which citizens can make analysis based on knowledge from the platform and medical professionals can augment their clinical knowledge with heterogeneous information from the avatar for clinical decision making and knowledge exploration.

As a project focusing on web-based health and lifestyle data collection, access, analysis and sharing, there are always new data in the process of collection from a number of data sources. The users may have different filtering and analysis requirements of analysis. Without proper design of visualisation and user interaction, it is not possible for the user to select, view, understand and gain knowledge from a large collection of data. Consequently interactive visual analysis is a critical part for the effective uitilisation of data collected and stored on MyHealthAvatar platform.

This section presents the interactive visual analytics components of the diary in MyHealthAvatar to facilitate health and lifestyle data presentation and analysis, including calendar, information panel, clock view and map. These components can be integrated to achieve flexible visual analysis of spatio-temporal lifestyle data.

6.2 Related Background

With the increasing popularity of the fitness and health sensors in recent years, there have seen devices or mobile apps on the market such as Fitbit [3], Withings [4], iHealth [5], Moves [6], etc. providing internet-enabled fitness data collection. However, as fitness collecting devices, they only provides very basic visualisation of a limited set of data sources.

On the other hand, there have been increasing interests in building centralised health and fitness data repository. Google Health was a personal health information centralisation service introduced by Google in 2008 but later discontinued in 2011 [7]. HealthVault [8] from Microsoft integrated a

number of fitness sensors on the market to provide central repository and APIs for data access but its emphasis is on data storage and does not provide powerful interactive visualisation tools for data analysis. Fluxtream [9] is an open source project to provide data storage of fitness sensors and basic visualisation tools, but unlike MyHealthAvatar it does not support connections to medical systems thus the visualisation only focuses on fitness data.

6.3 Data

MyHealthAvatar data are health and lifestyle oriented and can be categorised into health sensor data and medical data. As the medical data is managed by medical partners in the project and is imported from the hospital information system, such as Optima, they are not the focus of this paper. Here we are more interested in sensor data, including data from mobile apps.

A. Sensors and Apps

In recent years there has been a boom in the health sensor market. Sensors evolved from traditional devices such like the step meter to internet-enabled devices such as Fitbit [Fitbit], Withings [Withings], iHealth [iHealth], etc. They measure steps, walking distance, calories, sleep quality, heart rate, weight, etc., based on the device model. Some devices measure more health oriented data, such as blood pressure, glucose, etc. The collected data can be upload to the servers and shared via APIs to other internet applications. Meanwhile, with the rise of smart phones, fitness mobile apps also become an important data source of health and lifestyle data. Mobile phones with proper sensor installed are capable of not only measure the step number with the accelerometer but also record the location of the user with GPS sensor, thus keeping track of both the fitness data and daily lifestyle data of the user.

Moves [Moves] is a very popular app for fitness recording. Moves automatically records the step number and location of the user and calculates calories burned and distance of the movement. It automatically recognise the activity type, such as walking, running, cycling, transport, etc. The user can either view the distance, duration, steps, and calories data on the mobile phone or export the data from Moves server. An automatic daily storyline with time and location are recorded and shown on the map in Moves App.

Currently MyHealthAvatar supports importing data from user's Fitbit, Withings and Moves accounts via APIs provided by these devices.

B. Human Input Data

In addition to data automatically collected by fitness and lifestyle sensors and apps, there are data that may need manual input such as event data, food data, etc. Manual input is a very important data source which can not only be used for data input but also can be used for data editing and error fixing. Consequently, MyHealthAvatar supports human input data in the mobile app in addition to sensor data and medical data.

6.3.1 Data Format

6.3.1.1 Fitbit Activity Data

The fitbit activity data stores information of steps as well as other information, especially data derived from steps, such as distance, calories, etc. The main data fields from the fitbit activity API used in MyHealthAvatar is as follows:


```
{
        "activityCalories":230,
        "caloriesBMR":1913,
        "caloriesOut":2143,
        "elevation":48.77,
        "fairlyActiveMinutes":0,
        "floors":16,
        "lightlyActiveMinutes":0,
        "marginalCalories":200,
        "sedentaryMinutes":1166,
        "steps":0,
        "veryActiveMinutes":0
}
```


6.3.1.2 Fitbit Heart Rate and Sleep Data

As Fitbit uses dedicated APIs for sleeps, heart rate and body/weight, one needs to use different APIs to fetch these data. Currently the heart rate and sleep data are used and their data structure is as follows:


```
Heart Rate:
{
    "activities-heart": [
        {
            "dateTime": "2015-08-04",
            "value": {
                "customHeartRateZones": [],
                "heartRateZones": [
                    {
                        "caloriesOut": 740.15264,
                        "max": 94,
                        "min": 30,
                        "minutes": 593,
                        "name": "Out of Range"
                    },
                    {
                        "caloriesOut": 249.66204,
                        "max": 132,
                        "min": 94,
                        "minutes": 46,
                        "name": "Fat Burn"
                    },
                    {
                        "caloriesOut": 0,
                        "max": 160,
                        "min": 132,
                        "minutes": 0,
                        "name": "Cardio"
                    },
                    {
```

```
                        "caloriesOut": 0,
                        "max": 220,
                        "min": 160,
                        "minutes": 0,
                        "name": "Peak"
                    }
                ],
                "restingHeartRate": 68
            }
        }
    ]
}

Sleep:
{
    "sleep": [
        {
            "isMainSleep":true,
            "logId":29744,
            "efficiency":98,
            "startTime":"2011-06-16T00:00:00.000",
            "duration":28800000,
            "minutesToFallAsleep":0,
            "minutesAsleep":480,
            "minutesAwake":0,
            "minutesAfterWakeup":0,
            "awakeningsCount":0, // deprecated
            "awakeCount":0,
            "awakeDuration":0,
            "restlessCount":0,
            "restlessDuration":0,
            "timeInBed":480
            "minuteData":[
                {
                    "dateTime":"00:00:00",
                    "value":"3"
                },
                {
                    "dateTime":"00:01:00",
                    "value":"2"
                },
                {
                    "dateTime":"00:02:00",
                    "value":"1"
                },
                &amp;lt;...&amp;gt;
        },
        {
            "isMainSleep":false,
            "logId":29745,
            "efficiency":93,
            "startTime":"2011-06-16T14:00:00.000",
            "duration":3600000,
            "minutesToFallAsleep":20,
            "minutesAsleep":38,
            "minutesAwake":0,
            "minutesAfterWakeup":2,
            "awakeningsCount":0,
            "awakeCount":0,
```

```
            "awakeDuration":0,
            "restlessCount":0,
            "restlessDuration":0,
            "timeInBed":60,
            "minuteData":[
                {
                    "dateTime":"14:00:00",
                    "value":"3"
                },
                &amp;lt;...&amp;gt;
            ]
        }
    ],
    "summary":{
        "totalMinutesAsleep":518,
        "totalSleepRecords":2,
        "totalTimeInBed":540
    }
}
```

6.3.1.3 Withings Activity Data

Withings provides a similar activity data structure to Fitbit, including steps, distance, calories etc. , as shown in the following data structure.

```
{
   "status": 0,
   "body": {
       "date":"2015-04-10",
       "steps":6523,
       "distance":4600,
       "calories":408.52
       "elevation":18.2,
       "soft": 5880,
       "moderate": 1080,
       "intense": 540,
       "timezone":"Europe/Berlin"
    }
}
```

6.3.1.4 Withings Body Measurement Data

Withings keeps various body measurement data in a single data structure and uses different type IDs to represent different measurement variables. Each measurement is organised into one group, as shown in the following data structure.

```
{
           "status": 0,
           "body": {
               "updatetime": 1249409679,
               "timezone": "Europe/Paris",
               "measuregrps": [
                   {
                       "grpid": 2909,
                       "attrib": 0,
                       "date": 1222930968,
                       "category": 1,
                       "measures": [
```

```
                        {
                            "value": 79300,
                            "type": 1,
                            "unit": -3
                        },
                        {
                            "value": 652,
                            "type": 5,
                            "unit": -1
                        },
                        {
                            "value": 178,
                            "type": 6,
                            "unit": -1
                        },
                        {
                            "value": 14125,
                            "type": 8,
                            "unit": -3
                        }
                    ]
                }
            ]
        }
    }
```


The type IDs represent the following variables:

```
1 : Weight (kg)
4 : Height (meter)
5 : Fat Free Mass (kg)
6 : Fat Ratio (%)
8 : Fat Mass Weight (kg)
9 : Diastolic Blood Pressure (mmHg)
10 : Systolic Blood Pressure (mmHg)
11 : Heart Pulse (bpm)
54 : SP02(%)
```

6.3.1.5 Moves Daily Activity Storyline

The Moves app [Moves] provides more detailed recording of daily activities with geographical coordinates. A Moves daily storyline is organised into segments, each segment may either be a “place” or “move”. A “place” is a fixed location with names, start and end time. A “move” is a track of movement points. As Moves also records the steps, it can recognize the move type as “walk”, “running”, “cycling” or “transport”. An example moves data structure is shown as follows:


```
[

    {
        "date": "20121212",
        "segments": [
            {
                "type": "place",
                "startTime": "20121212T000000+0200",
                "endTime": "20121212T071430+0200",
                "place": {
```

```
            "id": 1,
            "type": "unknown",
            "location": {
                "lat": 55.55555,
                "lon": 33.33333
            }
        },
        "lastUpdate": "20130317T121143Z"
    },
    {
        "type": "move",
        "startTime": "20121212T071430+0200",
        "endTime": "20121212T074617+0200",
        "activities": [
            {
                "activity": "walking",
                "group": "walking",
                "manual": false,
                "startTime": "20121212T071430+0200",
                "endTime": "20121212T072732+0200",
                "duration": 782,
                "distance": 1251,
                "steps": 1353,
                "calories": 99,
                "trackPoints": [
                    {
                        "lat": 55.55555,
                        "lon": 33.33333,
                        "time": "20121212T071430+0200"
                    },
                    {
                        "lat": 55.55555,
                        "lon": 33.33333,
                        "time": "20121212T072732+0200"
                    }
                ]
            },
            {
                "activity": "transport",
                "group": "transport",
                "manual": false,
                "startTime": "20121212T072732+0200",
                "endTime": "20121212T074616+0200",
                "duration": 1124,
                "distance": 8443,
                "trackPoints": [
                    {
                        "lat": 55.55555,
                        "lon": 33.33333,
                        "time": "20121212T072732+0200"
                    },
                    {
                        "lat": 55.55555,
                        "lon": 33.33333,
                        "time": "20121212T074208+0200"
                    },
                    {
                        "lat": 55.55555,
                        "lon": 33.33333,
```

```
                                "time": "20121212T074617+0200"
                            }
                        ]
                    }
                ],
                "lastUpdate": "20130317T121143Z"
            },
            {
                "type": "place",
                "startTime": "20121212T074617+0200",
                "endTime": "20121212T100051+0200",
                "place": {
                    "id": 2,
                    "type": "unknown",
                    "location": {
                        "lat": 55.55555,
                        "lon": 33.33333
                    }
                }
            }
]
```


6.4 Diary Components

6.4.1 Calendar and information panel

MyHealthAvatar provides health data collection, storage and access to end users. The data can either be automatically collected or manually input. For lifestyle and health tracking, the data are often time dependant, especially date dependant. A natural form of date-based data organisation, display and editing is a calendar, which is a traditional way to visualise daily events. In MyHealthAvatar, a calendar-based diary is used for daily data display as well as daily event input, editing and planning. Figure 18 shows an example view of the calendar with the event editor. The calendar displays a brief summary of the fitness data such as daily steps, walking and transportation distance, as well as calories burned.

With the information panel the user can view detailed information, including Fitbit activity data, heart rate and sleep data, Withings activity data and body measurement data and Moves summary data.

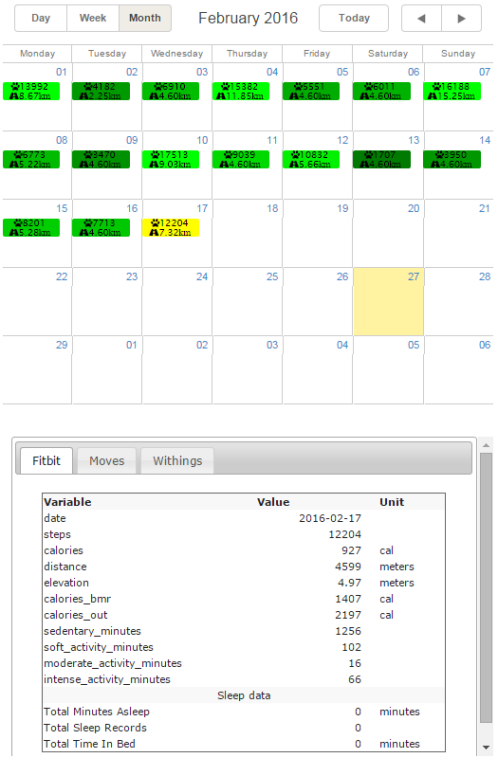

Figure 18 An example view of the calendar and information panel

6.4.2 Clock View

For daily activities, timeline provides visualisation over a relatively long period. Interactive timelines can provide zooming to smaller scales. However, the linear layout may make it difficult for the user to understand and compare daily events. A fine-grained view of activities within one day is better visualised in a radial layout. A natural, real-life way of radial daily time representation is the clock. MyHealthAvatar uses a similar radial layout called ClockView to visualise daily events. Movements and places from Moves data are visualised in the radial layout. Activity types are marked by icons and colours. When the user hovers the mouse over the icons more detailed information will be displayed, as shown in Figure 19.

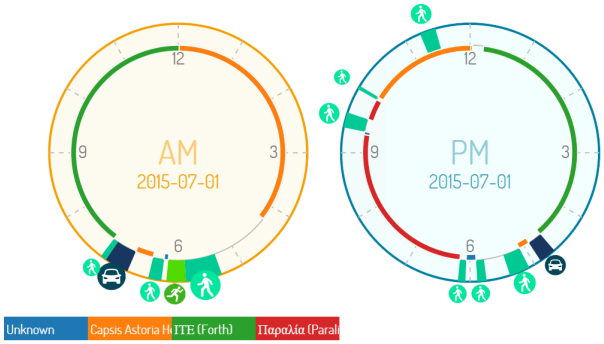

Figure 19 ClockView example

6.4.3 Map

While calendar, timeline and clock view are majorly designed for visual analysis of temporal data, they are hardly suitable for visualization of spatial locations and events. A map is a natural choice to provide intuitive spatio-temporal visualisation and analysis of the user's locations and routes for better understanding and knowledge discovery of the lifestyle. The map implementation is based on Google Maps [GoogleMap]. In MyHealthAvatar the map is used for visualisation and analysis of the Moves data only but it is capable of supporting other location-sensor-based apps. Figure 20 is the map visualisation of Moves daily data using line segments. A heat map is also shown to highlight the most visited areas.

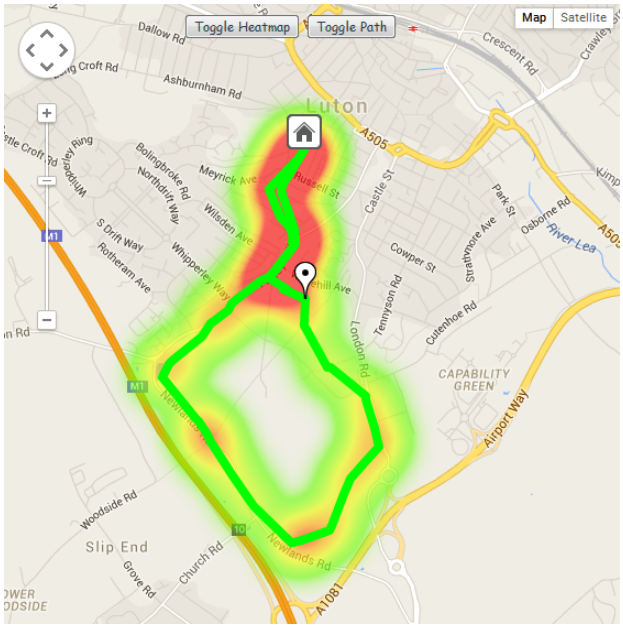

Figure 20 Map visualisation of routes and locations of daily activities from Moves

6.5 Integrated Diary Visualisation

MyHealthAvatar uses an integrated view to visualise and analyse events and activities, including diary, map and clock view, as shown in Figure 21. The advantage of this compound view is that it provides integrated spatio-temporal visualisation and analysis. The page itself provides the user an extensive view of data collected from different sources and the user does not need to refer to multiple pages to view and analyse related spatio-temporal data collected and stored on the MyHealthAvatar platform.

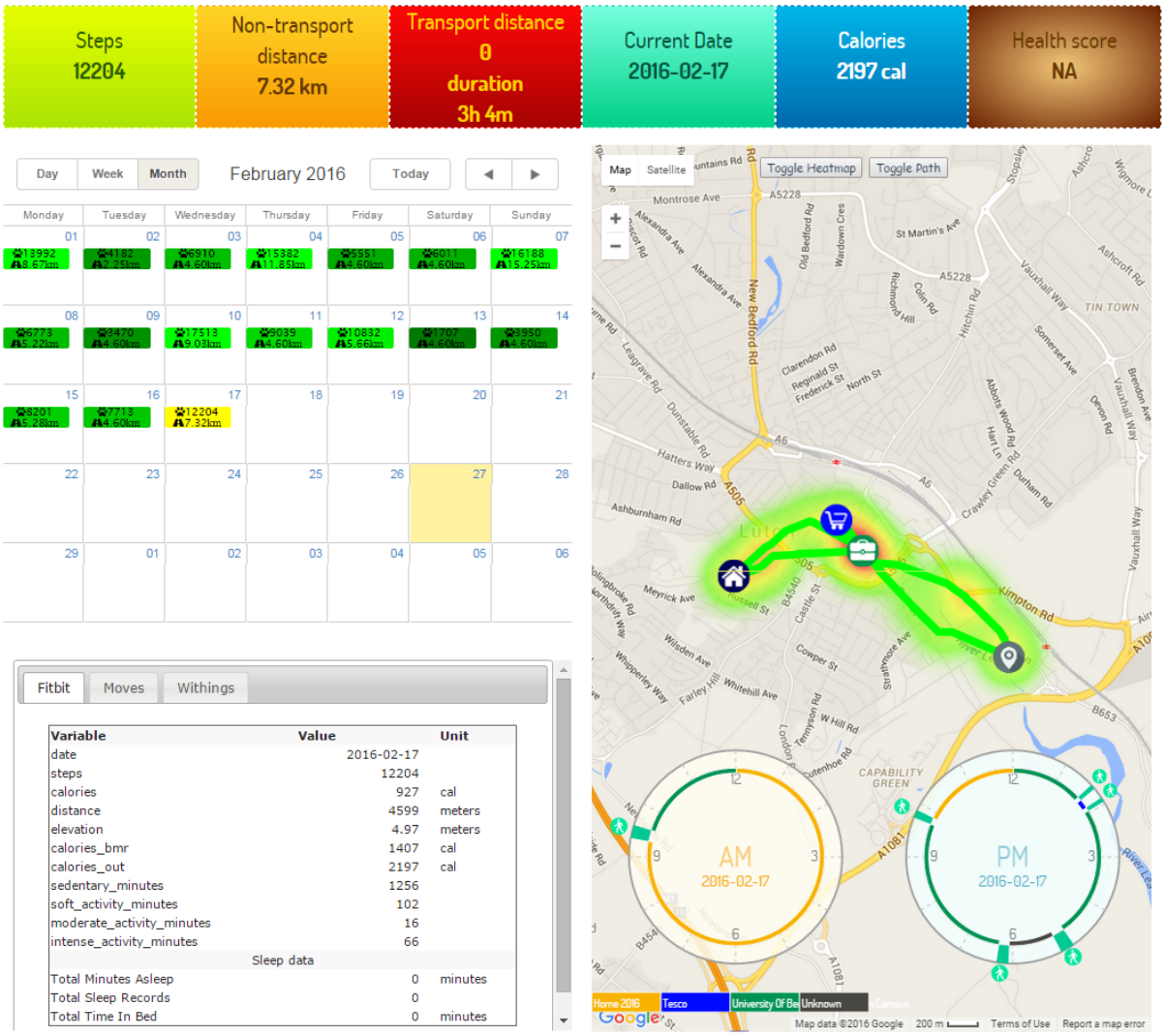

Figure 21 Integrated view of diary, map and clock view

6.6 Evaluation

Feature	Expected Functionality	Result
Fitbit activity data, heart rate and sleep data display	Fitbit activity data, heart rate and sleep data displayed in the information panel	Achieved, and a timeline visualisation is desired
Withings activity and body measurements data display	Withings activity and body measurements data displayed in the information panel	Achieved, and a timeline visualisation is desired
Fitbit/Moves/Withings Activity data display on the Calendar	Daily step count, distance, calories are shown in the calendar	Achieved
Moves daily route visualisation	Moves daily activity visualised on the map	Achieved
Moves daily route visualisation	Moves daily activity visualised in the clockview	Achieved
Place annotation	Annotate Moves place information on the map	Achieved

6.7 Summary

MyHealthAvatar is a project designed to collect and track lifestyle and health data to promote citizen wellbeing. As a lifetime companion of citizens, the amount of data collected will be huge. It is almost impossible for citizen, patients and doctors to view, utilise and understand these data without proper visual presentation and user interaction. Interactive visual analytics of lifestyle data is one of the key features of MyHealthAvatar.

In MyHealthAvatar the diary is designed to visualise the daily fitness and lifestyle data. The calendar, the information panel and the clockview are used to visualise time dependent fitness data and the map is used to interactively visualise daily activity data. An integrated view is used to display all the important lifestyle data related to the user.

7 Overviews on the Mobile App

The overview screen within the MyHealthAvatar mobile application is designed to provide immediate feedback on a user's general activity progress at the time of viewing, as well as provide quick access to features pertinent to the overviews functionality as well as convenient application navigation buttons.

The overview page is comprised of the following elements:

- Animated goal dial: Procedurally generated circular dial that converts MHA's own step service data into percentile values, set against user defined goals.
- Interactive activity readout text. Textual readout of values collected by MHA's step service with click to update functionality that triggers a feedback animation.
- Weather widget: Powered by a combination of a GPS latitude and longitude query, and the OpenWeatherMap API. This widget provides a textual description of current weather conditions as well as local temperature.
- Goal Dialog button: This button launches a dialog fragment which can invoke an interface with the main overview fragment to trigger an update.
- My Journal quick link: This button invokes a method within the parent Activity to facilitate a fragment change in the Activities fragment manager.

7.1 Overview page in the scope of the application:

The overview page is a Fragment, a Fragment in the context of native Android development is a reusable UI element or behavior that must be embedded in a parent Activity. Its life cycle is directly controlled by its parent, and it can be invoked or manipulated through Fragment transactions.

7.2 Goal dial implementation:

Description: The goal dial is a custom Android View that inherits from a standard View, overriding the on-Measure and on-Draw behaviors of its parent class to dynamically resize itself within its parents Layout. Upon the Fragment being created the custom goal view waits for the on-Measure

method to be called, indicating the dimensions of the view are finalized. The process requires this step as the view has no concrete dimensions until this method is called.

Initialization: With the width and height known the on-Measure method calls an Init function that calculates the largest possible dimension that the dial can occupy and maintain 1:1 aspect ratio, then calculates the top left corner and the center in screen space Cartesian coordinates that serve as the starting point.

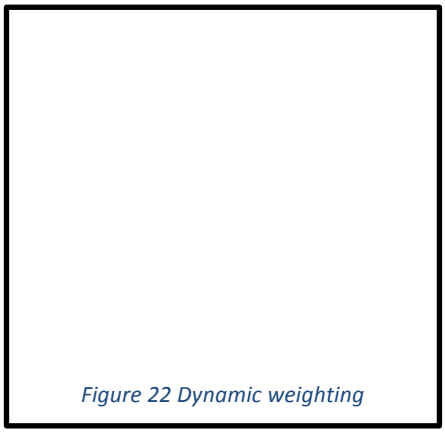

Figure 22 Dynamic weighting

Dynamic Weighting: With explicit dimensions and a reference point the app initializes a number of variables that represent each visual element's dimensions within the dial based on percentage values. For example a master value for the dial zone (DIAL WEIGHT) is defined as 0.7F which drives all other values as seen in Figure 22.

Populating the dial: The dial queries a Vector array of Goal objects and loops through the array using the loop's iterator as a component in the bounding calculations, Starting from the outer edge of the dial zone, stepping inwards by 0.03% of the 1:1 view bounds per valid goal. This method allows the number of individual goals to increase inwards towards the center, although there are only three active goals at present. These bounding Rects are then added to the goal object and a pair of Icon offset values are calculated that provide a new height and size target for icons that are overlapping.

Drawing the dial: The overridden on-Draw allows the custom dial View to draw objects into a Canvas. Each goal in the Goal array is converted into the basic components that comprise a draw call within a valid Canvas. Each goal contains its own outer bounds in the form of a Rect, a completion curve Rect, an icon Rect and a line starting point and a Bitmap image. Android provides a number of useful primitives for rendering a percentile complete curve, such as the arc function which is used by converting the percentage of goal completeness to a corresponding angle ((percent-complete /100)*360). The dial also sorts goals into completeness order before calculating all the required bounds, this is to ensure that icons never have to cut through other goal completion arcs.

Animating the dial: The animations are all controlled by one master Value-Animator within the overview fragment, that when activated animates a float value between zero and one over three thousand milliseconds, using a fixed delta time instead of program cycles ensures a consistent effect across all target devices. The animations required more complicated behavior than simply moving in a constant arc around the dial, early versions of the dial suffered from issues with overlapping icons, one solution was to lock each icon into its own fixed channel, but this forces all icons to be very small

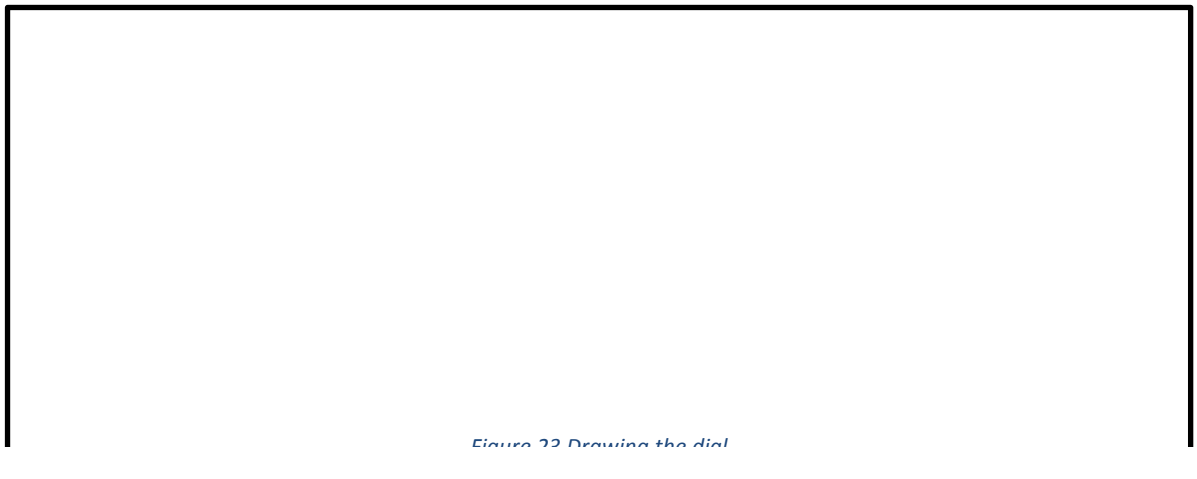

Figure 23 Drawing the dial

all the time and not make use of all available space. The solution was a combination of angle based collision detection and pre-defined alternative size and positions.

Each animation frame a loop checks every icon against every other icons percentage complete value and applies an equation that returns between zero and one, one representing a complete overlap within a bound angle range of between 5 and twenty five degrees, an angle of five or less degrees is considered a total overlap. This overlap factor can then be used to interpolate the icons between their pre-defined offset positions and sizes. The result illustrated in Figure 23 shows that an icon will interpolate to full size if it is twenty five degrees away from any neighbor.

7.3 Activity readout implementation:

Readout text fields: The steps, distance and calories readout is a new subtle replacement of an over complicated view paging spinner that occupied the top of the screen in earlier versions of the application. The new system is built from text-view objects within a relative layout, with on-click listeners attached to the title text they trigger a shake animation when clicked, this provides immediate feedback to the user informing them that their click has been registered.

BMR & calorie calculation: Under the calorie heading there are three sub values, calories in, calories burn and BMR (Basal Metabolic Rate). Calories in makes an SQL database call that adds up all calorie input values from the target day and returns the total and then subtracts the total calories burned to reveal a calorie surplus. Calories burn reads from the parent Activities intent any extra data attached by the step counting service. The BMR calculation looks for a stored activity level value of between 0-4, representing extreme inactivity to highly active. If this value does not exist the fragment launches an Asynchronous task to query the MyHealthAvatar API for the users current activity level. The function then uses a formula to calculate gender specific BMR values.

7.4 Goal dialog implementation:

The goal editing menu is a custom Dialog-Fragment launched from within the overview fragment. It provides a grid layout of active and long term goals. The dialog is populated by querying the local cache manager for a User-Goals object which is a custom data structure capable of storing all supported goals. The dialog fragment provides a programmatic interface used in the parent overview fragment which allows the goal dialog to send any changes the user made to their goals back to the overview. Once new goals have been received the overview launches two Asynchronous tasks, one to push the goal up to the MHA API and the other to put the same values into the local goal database, then the callback restarts the dial animation forcing the new changes to be visualized.

7.5 Weather Widget implementation:

The overview page weather widget is comprised of an Image-View and two Text-Views within a Relative-Layout. The image has an on-Click behavior which queries MHA's GPS location service for the user's longitude and latitude. If these values cannot be acquired the user is prompted to enable their GPS service. If the longitude and latitude are acquired then the behavior initiates an Asynchronous task and sends to it the relevant views that make up the widget. The Asynchronous

task will be accessing a REST web service call openWeatherMap to make a weather query based on the users current position, then it will output the results directly into the views it was passed earlier.

7.6 Testing

Feature	Expected Functionality	Actual **tested on: Nexus 4, Nexus 9, Huawei x2, HTC M9, HTC M8**
Goal dial dynamic scaling	Goal dial will scale to fit any Android mobile screen.	Dial scales correctly.
DEPRECIATED: Goal editing dialog fragment list	Dialog fragment appears and allows user to view and edit goals, dialog is removed when canceled or accepted.	Dialog appears and facilitates goal editing and viewing. ISSUE: on devices with a small screen, goals were hidden below the bottom of the screen.
Goal editing dialog fragment grid	Dialog fragment appears and allows user to view and edit goals, dialog is removed when canceled or accepted.	Dialog appears and facilitates goal editing and viewing.
Adding new goals	Dial will update with new goal, web API receives new goal values and updates	Dial updates immediately with new goal ring and Icon.
Goal dial animating	Dial animates activity icons to the correct corresponding angle to represent a percentage completion. Icons avoid overlapping each other. Animation re-starts on goals being changed or the page is re-loaded.	Dial animates correctly, dynamic sizing could be a little tighter with no adverse effects. ISSUE: icons at 100% overlap icons at 0%
REST web service weather widget	Widget displays a weather icon, temperature and condition text.	Widget appears as described.
DEPRECIATED: Activity readout view pager	Scrolling view pager with all goals listed with up to date progress values.	Pager scrolled as intended, with extra navigation chevrons to indicate its scrolling nature. ISSUE: due to paging behavior values within each subpage updated later than expected. ISSUE: users tended not to use or notice the scrolling behavior of the pager.
Activity readout text view	Serve as a replacement to the activity view pager feature. Show the three main active goals and their corresponding values. Steps, Distance & calories.	Values appear as expected. ISSUE: BMR currently displaying “Disconnected”

8 Statistics Views on the Mobile App

The statistics screen within the MyHealthAvatar application is designed to facilitate the viewing and filtering of the activity data and measurement data of the user. It achieves this by requesting and caching large amounts of activity data into a large cache of drawable objects representing all filtered views spanning the selected time range.

The statistics page is comprised of the following elements:

- Filter toggle button row: Custom toggle button views that switch on and off the corresponding data chart and chart scale guides.
- Date range navigation and display bar: Displays a textual representation of the currently active date range and two chevron navigation buttons to initiate a time step of the value of the active time step period.
- Charts view: Procedurally generated custom view, displaying a combination of bar charts and line chart overlays with two dynamic scales.
- Date range selection button row: three time step toggle buttons that control the behavior of the time step navigation buttons.

8.1 Statistics page in the scope of the application:

The statistics page is a Fragment parented to the Main-Activity of the MHA application.

8.2 Filter toggle row implementation:

Description: The filter button row is comprised of extended Image-Buttons with custom coded behaviors for handling toggling. The custom buttons are also designed to be defined by an XML style and attribute layout. The toggling behavior is triggered simultaneously with the corresponding filter action.

Filter changes: Once a filter is activated by an on click action a hash-map is generated with toggle names and active states, this is used by the custom charts view to set cached activity charts to be visible or hidden.

Enforced toggling: The chart displays both bar and line charts simultaneously, but due to the dynamic nature of the scales only one of each is allowed to be active at once. This is to prevent cluttering and labels overlapping.

8.3 Date range navigation bar implementation:

Description: The date navigation bar is comprised of two navigation chevrons and a textual readout of the current visible date range. The behavior of the navigation buttons is to trigger an Async-task to query the activity cache, this will cause the caching system to validate a local store or acquire the new activity data from the MHA API and inform the statistics page only when the job is done. Once caching has been confirmed a new Async-task is created to calculate and populate the rendering caches off the main UI thread, once this task is done it updates the custom chart view directly. The date navigation range is also constrained to dates on and before the current date.

8.4 Charts view implementation:

Description: The charts view is an extended View within Android, inheriting all the View parents' behaviors and adding and overriding new and existing behaviors.

Chart frame: The frame of the charts view is procedurally generated in much the same fashion as the overview goal dial fig 6.2.a, the available screen space is determined during the on-Measure phase and then the areas of the chart are multiplied against scales and rounded to integers to give a pixel perfect result that should be consistent between mobile devices.

Bar charts: The bar charts are comprised of multiple rendered lines and rectangle primitives. Each bar has the potential to represent current activity levels, the user defined goal, as it was on the date in question and an alternative shade of bar to represent activity that either surpassed the goal or the deficit to the goal (Figure 24).

Figure 24 Bar charts

Bar data: The data that is used to generate the bar charts is the explicit value of the activity as reported by the MHA API and recorded in the mobile applications local database and then scaled into the available screen space. For the purposes of rendering, all charts start and end coordinates are rounded to integers to ensure a crisp and clear rendered boarder, this arguably could lead to a small amount of loss of accuracy, but the effect would likely be negligible as the scales would also be rounded the same way.

Goal approximation: Goals are not set explicitly every day, but due to the MHA app caching all goal changes in a local database the application can step backwards through this database until it finds a goal record, or it exhaust the list and returns no goals set (Figure 25).

Figure 25 Goal approximation

Line charts: Filters that represent data points that have the potential to be intermittently entered by the user are represented by line charts. Many of the data types are entered multiple times in a single day, for example insulin levels could be entered before and after each meal, meaning a potential of six entries per day. In these cases data points are averaged out to reduce the complexity of the chart. On days where no data exists an interpolated data point is added by scanning ahead and behind this time period, if no past data points exit then the value is assumed to be the same as the next explicit point, if no data point exist after the date then the line continues at the same level as the last explicit point, this simply means that there is no trend prediction, and all interpolation is linear (Figure 26).

Figure 26 Line charts

Scales: The scales to the left and right are procedurally generated to attempt to solve the problem of unreadable pre-defined scales. To achieve this behavior all the charts have to be built in two stages, the data gathering stage pulls data from the cache manager and populates an array of indexed references to each day. Then the population of the rendering arrays occurs, this is where values are converted into screen space coordinates. By caching all the raw data first the highest values of each measurement item can be found, these values are then used to adjust the scaling factors for rendering to ensure all data fits within the chart and the highest value can be processed to find appropriate scale points to add value labels.

8.5 Date range selection implementation:

The date range selectors are Image-Buttons with some extra logic to change their appearance to indicate which time step is active. The buttons listen for on-Click events and apply logic that sanity checks the new date ranges and then create a new Async-task that performs the same logic as described in “Date range navigation bar”.

8.6 Caching and filtering:

The filtering of views within the app is made faster by pre-caching all possible viewing data within the time range. This process works by calculating all lines, polygons and rectangles for every filter whenever there is a change in date range. This method takes longer than just creating the drawable components of the currently active toggle but it makes toggling between groups quicker. The cost in processing and device memory is offset by the performance gain of seamless toggling.

8.7 Testing

Feature	Expected Functionality	Actual
Activity toggles	Clicking toggle would change toggles appearance and display the relevant bar chart or line in the chart view	Toggles correctly display their status with highlights. Toggles effectively change the active data within the chart view.
Date range navigation	Button click would initiate a cache update and chart refresh based on the time step toggle setting.	The chart view updates after a brief loading screen.
Dynamic chart view scaling	The chart view would scale correctly to fit any screen size it is deployed on.	Chart maintains a consistent scaling across devices. ISSUE: Text scaling is not always consistent.
DEPRECIATED: Current goal value label & line	Line would highlight the most recent active goal value and provide a scale for the charts	Line was obscured by other graphical elements and hard to interpret.
Time step buttons	Button click would initiate a cache update and chart refresh based on the time step toggle setting.	The chart view updates after a brief loading screen.
Dynamic scales, implemented to replace the feature “Current goal value label & line”	Scales would resize themselves to provide legible values to gauge the charts data by	Scales react well to varying data ranges. ISSUE: scale on calories not dynamically scaling.

9 Day Views on the Mobile App

The day view within the MyHealthAvatar application is designed to allow a user to dig down into the specific events of a day by category e.g. events, planner and charts. This screen is actually several embedded Fragments allowing the user to scroll or "Page" through their data.

The day view is comprised of the following elements:

- Date navigation bar: Provides a text readout of the active date and access to two chevron buttons that allow forward and backwards scrolling through time.
- Tabbed view pager: Hold three sub Fragments with their own behavior and logic.
    - Events Fragment
        - List of all events from a wide range of data sources, headed by a parent geo-location node.
    - Planner Fragment:
        - List of all planner events that occurred or will occurred on the listed target day.
    - Charts Fragment:
        - Filter toggles: Buttons that not only toggle their own appearance but also remove entire chart elements from the page.
        - Chart view: bar charts that show activity data in-line with a value label.

9.1 Events tab

Description: The event tab is used to show the activity diary for a special single day. Every activity event is listing in an "*Extend ListView"* in Android according to the time factor. Users can check the activities in the Google Map and also annotate the place for other functions as described in section 4 and 5.

In terms of "Extend ListView", compared with the normal "ListView", "Extend ListView" supports two layers, parent and child layer, rather than the single layer in "ListView". The reason why MHA app implements this view is because of the data structure. The parent layer is used to display the main event and the child layer is used to display the specific sub-events as shown in Figure 27. The sub-event may be walking in a place or the special event which is generated by the app itself. Moreover, the "Extend ListView" could disable the child layer if there are no special sub-events.

Figure 27 ListView

Date resource: There are two data resource providers, such as Moves and MHA, using the list in the event tab.

Moves data records any movement (walking, cycling, running and transport) and place data [Moves, introduction]. Once the user links their MHA account with their Moves account through the website, the app then can collect the data from our API service by using Async-task. After that the app filters the data and display it in the list.

Figure 28 Data filtering

In terms of data filtering, there are four steps to achieve the function as shown in Figure 28.

1. To create the Moves event list object and the special event list object, then sort them by time in Ascending.
2. Base on the Moves event list, to create the parent list.
3. Base on the time range of Moves event data, to find out any special event that happened during the place and movement time period. Then to save the special event as corresponding parent's child item.
4. If the special event didn't happen during the time period of the Moves event, then create a new parent item and add it to parent list.

MHA data is based on MHA step sensor. It offers convenience for the users who have no Moves data. Currently, the MHA data can detect the user's movement and place. The Moves data offers more specific data types (e.g. running), but the service only can provide users the ability to review their data from yesterday. Compared with the Moves service, MHA data can provide real time information for users.

In terms of place and movement detection for MHA data. It is very easy to detect the annotated place and its geographical coordinates (GPS location, latitude and longitude). The difficult part is to detect the user's movement and unannotated place due to no user interaction.

The flowchart of the proposed method is shown in Figure 29

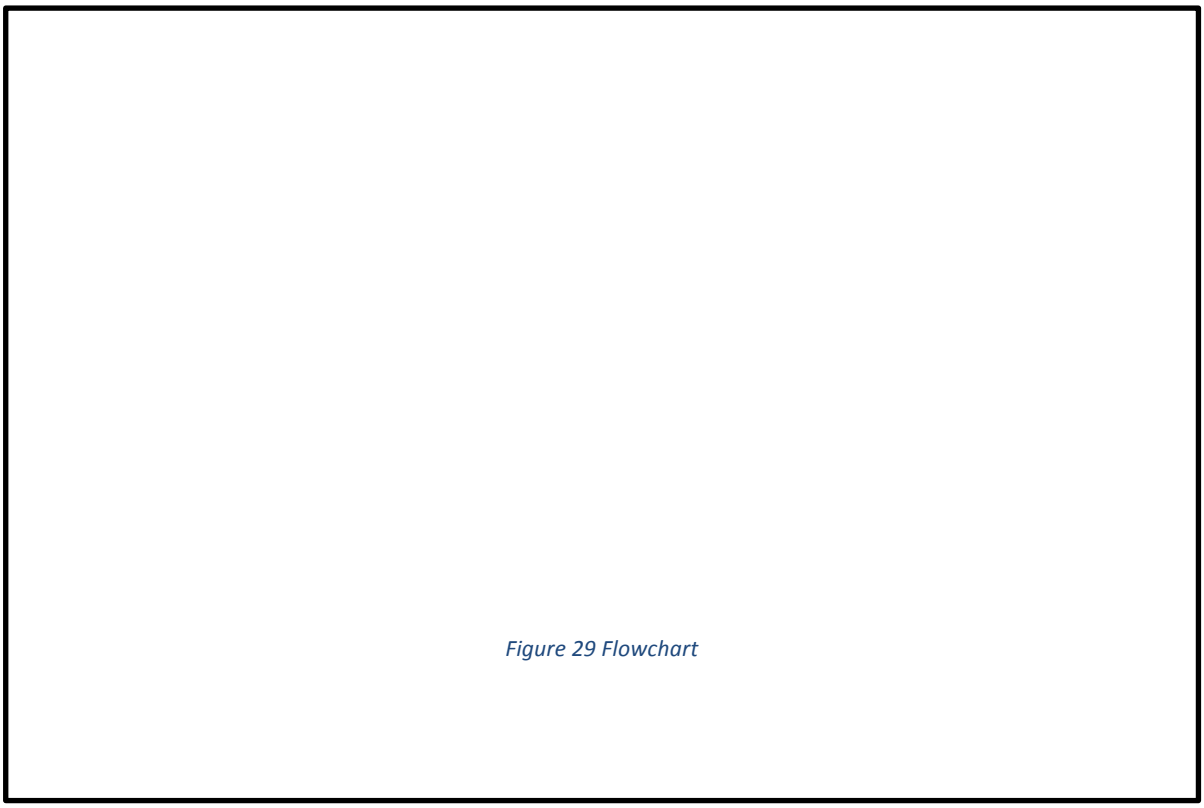

Figure 29 Flowchart

The method to detect the user's movement and place is described below.

1. Set IntentService to obtain GPS location from GPS service on 5 mins per 1 request. The IntentService is a base class for Services that handle asynchronous requests (intents). Services is an application component that can perform long-running operations in the android system background. It does not provide a user interface, and also continues running during the sleep mode of the mobile. [Android, Services]

   Currently, in most of mobile devices, the internal GPS update rate can reach to 1 Hz (one request per one second) due to the hardware GPS limitation [Mobile Device, GPS Update Rate]. The higher the frequency ab app updates the GPS location, the more power consumption the app has to face. Moreover, the MHA app does not need to achieve the navigation functions. Thus, the GPS update rate is set as one request per five mins.

2. Check the new GPS location with MHA place location database. If the two locations are within 30*30 square meters, the app will annotate two locations as the same place and set the place name according to the database.

   The reason why the app is using 30*30 square meters to distinguish different places is that:

GPS hardware gives 10m horizontal accuracy since SA is turned off according to policy directive in May 2000. In 2015, high-quality FAA grade Standard Position Service (SPS) GPS receivers provide horizontal accuracy of better than 3.5 meters. [GPS, Horizontal Accuracy]

According to [Android, GPS Accuracy], people have routinely recorded a 2 meter accuracy with certain Android phones like Motorola Droid. Take into consideration: weather, surrounding high buildings and device on the move, it makes sense to assume the Moves place location has an accuracy of 10 meters on its supported phones.

Moreover, According to [Greenwich], at Greenwich longitudinal length equivalents of 0.0001 degree is about 6.95 meters, which is about the range of the accuracy 10 meters of GPS. A place of small shop, friends home, etc. normally have a box size of 10* 10 square meter, however university, train stations can have a box size larger than 30*30 square meter. Thus, the app uses the 30*30 square meters to distinguish different places.

3. After the system sets the place name, the app will check the previous record.

   If there is no previous record, then the system will consider the current location is “Start New Place” which means that the user has arrived at a new place.

   If there is a previous record, then the system will check the previous record status is “Move” or “Place”. If the status is “Move”, the system will consider the current location is “Start New Place”. If the status is “Place”, the system will consider the current location is “Continuing place” which means that the user is still staying in the same place.

4. Check the new GPS location with MHA place location database. If there is no annotated place the same as the given location. Then check the previous record.

   If there is no previous record, the system will consider the current location is “Start New Place”. This means that the user just start to record the location and due to every movement is start at status “Place”, the status will set as “Place”.

   Check the previous location, if two locations are not within the 30*30 square meter, and the status of the previous record is “Move”, the app will set the current location status as “Move” and the current location is “Continuing Movement” which means that user is still moving. If the status of the previous record is “Place”, the system will set the new location status as “Move” and the current location is “Start Movement” which means that user has started to move.

   Check the previous location, if two locations are within the square area, and the status of the previous record is “Move”, the app will set the current location status as “Place” and name it as “Unknown” and the current location is “Start New Place” which means that user has completed his movement and arrived at a new unannotated place. If the status of

previous record is “Place”, the app will set the current location is “Continuing Place” and name of current place is “Unknown”.

5. Update the data record for “Move” status data. Since those “Move” date records have all the movement locations. If a user has traveled for 5 hours, then the app will record at least 60 records. It makes sense to optimise the “Move” data.

   Since the “Place” and “Move” activities will alternately appear, it is very easy to collect the group “Move” records between two “Place” records. Once the app find the group records, it will evenly delete the number of records in the group according to the size of the group.

Special Event: The special event is generated by Journal functions in MHA app. As the main interface of user and app interaction, the Journal can record the information of a user’s food, where, mood, pain, questionnaire, and 12-weeks weight reduce program. Once the user has interact with app, the action will be recorded and displayed on the activity diary (Event tab). Therefore, users can easily check their special event time, location, and detail.

As mentioned before, the special event is an overlap with the activity diary, if there are many events happened in a place, the parent item will show the maximum of five icons to indicate the events as shown in Figure 30.

Figure 30 Event visualisation

Marker Animation: The animation of the markers on the Google Map in the event tab are achieved by using the List of Markers and camera operation [Camera and Views].

Since the activity diary has already record a series number of locations, according to the records, it is very easy to create the Marker list with the specific icons, messages and GPS locations.

Figure 31 Marker animation

Moreover, Google Map offers two methods which are especially for short moves to achieve animating the change. By creating "*CameraPostion*" object to set the target location, zoom and bear value. The bear means Direction that the camera is pointing in, in degrees clockwise from north. [Google-Camera]

Then using method "*GoogleMap.animateCamera*" to let the camera move smoothly to the new location according to "*CameraPostion*" object. This method offers three arguments:

1. "*cameraUpdate*" describing the location that camera needs to move to which uses the "*CameraPostion*" object.
2. "*callback*" means that the app needs to implement "*GoogleMap.CancellableCallback*" to achieve continued animation (Loop). Moreover, within this method, the app can easily cancel the animation and stop the loop.
3. "*duration*" indicating the duration of each movement in the animation.

Therefore, by using those elements, the method process of animation is shown in Figure 31.

As shown in Figure 31, according to the time factor, by using the marker detail in the marker list, the system can create a new "*cameraPostion*" object and then implements the "*GoogleMap.animateCamera*" which starts the animation from the current location and then moves to the new location. Every time the camera reaches a new location, the "*cameraPostion*" object will be updated to the next position, and "*GoogleMap.animateCamera*" will continue running the animation until the loop has finished or broken.

Figure 32 Map visualisation

The visualization of animation is shown in Figure 32.

Place Annotation: The app supports user customisation and annotation for unknown places and also support to use Four-square to annotate the place.

Foursquare is a search and discovery location service which provides many results in order to let the user choose the most likely one as the result to annotate the place. [8.9] The MHA app uses this third party API to assist user to annotate the place. The process is described below:

1. When the user starts to annotate the place, the user can choose to use four-square data as a recommendation.
2. The app will use the location information as a parameter to call the four-square API service in an Async-task.
3. Four-square will return many results in the form of a list.
4. The User selects the result.
5. The app updates a local database according to the selected result, and also sends the update request to the MHA Place API through an Async-task.
6. Change the data in activity diary cache and update the event-tab list.

9.2 Planner tab

Description: The planner tab is comprised of a List-View populated by an Async-task. The Asycn-task queries a local planner event database and find any explicit event that occurs on the target day, as well as running a more advanced algorithm to detect if there are any active repeating (implicit) events that could occur on this day. The result are loaded into an adapter and given a custom layout.

Editing functionality: The planner tab is the only way of editing historical planner events, each row view in the list has a two stage on-Click behavior. Single click will load a pop out Dialog Fragment with a summary of the event, but a click and hold re-launches the original planner event dialog interface with the original settings re-loaded, here the user may edit or delete the record.

9.3 Charts tab

Description: The Charts tab is designed to provide an overview of activity metrics for a single day, not only providing a visual representation but also an explicit value or quantity for the activity. The chart is also designed to allow customization in the form of persistent toggling chart elements.

Chart scales: The current version of the charts scales all data to provide a consistent uniform effect when rendering the charts visually, half the chart bar is reserved for activity below a user's goal and everything above this point is reserved for activity beyond a user's goal. This design method makes the charts uniform but at the cost of distorting the ratios of the bar. For example an activity with a value of 50 and a goal of 10 will still be visualized as half yellow and half green.

Toggling behavior: The chart view allows for persistent toggling by storing a Hash-map of toggle states in the devices shared preference file. The reason for this extra behavior is because users are highly likely to scroll the date, and due to the charts page being a Fragment it would have re-initialized for each new day, effectively re-setting the toggles. By storing the toggle states in a persistent file, the user can scroll and compare a custom group of activity bar charts.

Interpolation: Much the same as the line charts in the statistics page some elements such as weight, BMI or blood pressure may not be recorded as regularly as other tracked activity, as such their value

on a day with no explicit recorded value is linearly interpolated from data before and after this point in time.

9.4 Testing

Feature	Expected Functionality	Actual
Extend ListView Visualisation	Data has completely and accurately mapped in the parent and children layers	Data are correctly display in the "Extend ListView" ISSUE: on devices with a small screen, it is difficult to see the parent and child layer entirely, if the parent has many children layers
Moves Data and Special event	Special events have completely and accurately mapped in "Extend ListView" with Moves data together	Data are correctly display in the list
MHA Diary Builder	The user's activity diary based on MHA is Complete and accurate display in Event tab	Data are correctly display in the list
MHA data and Special events	Special event has completely and accurately mapped in "Extend ListView" with Moves data together	Data are correctly display in the list
Google Map	Displaying the right location	Data are correctly display in the Google Map.
Location Animation Visualisation	Displaying the animation continuously and completely	Animation works fine. ISSUE: on devices with a lower memory, sometimes the map loading is slower than the animation movement which cause the few seconds blank map page.
Location Animation Operating	Pause and play the animation	Pause and play functions works fine.
Google Map Marker	Markers (Name and category) are completely and accurately displaying in Google Map	Markers are correctly displaying in the Google Map ISSUE: on devices with a small screen, sometimes many markers are overlap together, and it is difficult to click the marker, thus the user has to zoom in the Google map. Solution: The user can choose an individual location from the "Extend ListView" to annotate.
Place Annotation and marker	Easily to annotate place by using the marker click event	Each marker is correctly correspond to the right annotation dialog.
Place Annotation dialog	Dialog appears and allows user to view and annotate the place.	Dialog appears and facilitates place annotating and viewing.
Four-square Information	Four-square place information is correctly	Data are correctly display in the annotation

	for the specific place (Marker on the map)	dialog
Planner Event Visualisation	Planer events have completely and accurately mapped in the list.	Data are correctly display in the list
Planner Event Editing	Dialog fragment appears and allows user to view and edit planer, dialog is removed when cancelled or accepted.	Dialog appears and facilitates planner editing and viewing.
Dynamic chart view scaling	The chart view would scale correctly to fit any screen size it is deployed on.	Chart maintains a consistent scaling across devices.
Activity toggles	Clicking toggle would change toggles appearance and display the relevant bar chart or line in the chart view	Toggles correctly display their status with highlights. Toggles effectively change the active data within the chart view.

10 Profile Summary and Visualization

10.1 Introduction to profile

User profile in MyHealthAvatar stores user's general, health and medical information, there is also an overview tab for the profile which contains a user friendly visualization of user's important information. User is able to import medical profile from CCR (Continuity of Care Record) style XML format file which might be obtained from clinical systems.

There are several models based tools to help users analyse cardiovascular disease, hypertension, diabetes and stroke risks. User profile support the model by store and supply user information, so that user can get always get result which in line with the health profiles.

The overview tab provides a place to show important summary of information either edit/input directly from MyHealthAvatar or imported from other clinical systems (upload as CCR file). Please note that the system parses the uploaded CCR file and read information which is useful to MyHealthAvatar profile module only. There is a separate module of MyHealthAvatar which deal specifically with clinical data in Toolbox -> Clinical Data page.

10.2 General, Health and Medical Profile

The general, health and medical profile provide information displaying as well as editing to user, they utilize mainly browser's default text input, textarea, checkbox and radio buttons. This provides maximum compatibility to different desktop and mobile browsers, also the users are more familiar with how to use the default component to provide information to MyHealthAvatar system.

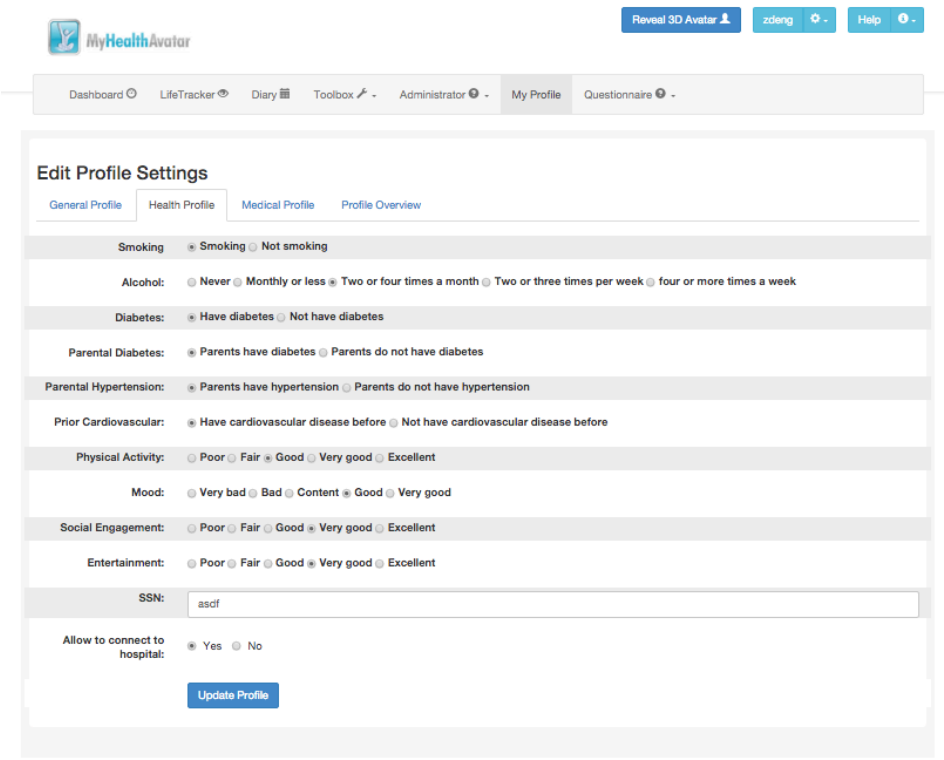

Figure 33 MyHealthAvatar health profile

As part of the project work, HTML5 components of input like date and email is used, also maximum and minimum range limits of input is applied according to HTML5 standard. A promising technology which seems the future of web called web components [Web Components] are evaluated during the implementation of profile. Web component provides an encapsulated way to put HTML, CSS and JavaScript together, and allow easy code reuse without affect other visual elements on the same page. However, due to browser support limitation and the technology is still not mature during experiment, it is considered as potential future work.

10.3 Overview of the Profile and PDF export

The overview of the profile is located at the forth tab of the user profile page, it visualizes the user profile as a PDF file view to the user. And export this as a real PDF file is supported by MyHealthAvatar, so that user can print out profile information on paper to bring it to places where no internet access is available.

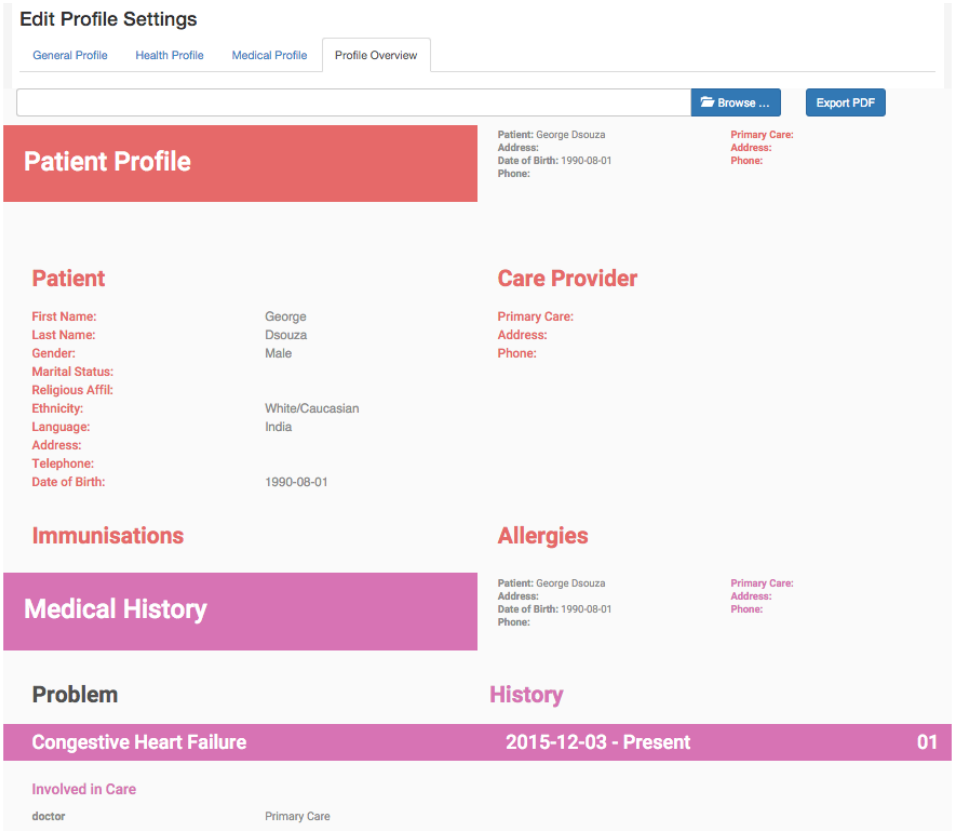

Figure 34 MyHealthAvatar Profile Overview

The profile overview breaks the user's information into different sections which includes general profile, medical history, medical history snapshot, medications summary and schedule, labs, etc. A colorful scheme is designed to make sections stand out and attract users attention to import part of the sections.

10.4 Evaluation

Feature	Expected Functionality	Actual
View different profile tabs	User can toggle different profile view by click on the tab header, this should not trigger browser reload	Different type of profile view is visible by click the tab
Update general profile	Edit the page by change general profile content and click on update, which should successfully update user general profile	Edit and update works by click on update on all fields on general profile page
Profile overview synchronisation	When user profile is updated from general profile tab, the profile overview should automatically updated to reflect the changes	After update general profile, switch to profile overview tab, updated value is displaying correctly.
Update health profile	User health profile should be edit and update by click the update button after edit	Edit and update of health profile works as expected
Risk model synchronisation	After user update health profile, the risk model should have corresponding fields updated	Edit and update the health profile and then switch to Toolbox -> Risk models.

		Updated content sometimes requires page refresh to display new values.

10.5 Summary

The user profile is function as user requirements analyses, the visualization is provided in a simple and meaningful manner. User interactions are baked in when need, especially in the 2D avatar of the overview tab. While the profile has achieved the designed goal, it sometimes takes time to load all the information cause users to wait few seconds before the page is loaded completely. Even better user experience could be achieved by improve the loading time, and in the future web component is suggested to replace vanilla browser inputs.

11 Future Work – LifeLoop

11.1 Introduction

Life pattern mining are especially useful for lifestyle detection, health monitoring, crime analysis, traffic management, urban planning, etc. A daily storyline is the full sequence of places and activities taken by a person in his one day life. A loop is an important life pattern in daily activity storylines. To recognize the daily activity patterns, in the first stage loops in the daily activity storylines may be recognized and analysed. A lifeloop is one of the special activity patterns which starts and ends at the same place. The definition of a loop type can be specified by the user. Similarity definition is the key for loop definition and recognition. There may be other types of activity patterns which can be specified by the user for the future work.

The objectives of Lifeloop include:

- Detect all the loops from the movements data based on user specification.
- Define an event as a loop by detecting the “semantics” of the loop, together with its granularity (the granularity is used to define the detail of the events)
- Allow queries for the events from temporal and spatial dimensions
- Define similarity between different events (namely the loops)
- Clustering of the events based on the similarity
- Outlier detection of the events
- Statistics of the events (Similarity-based)
- Importance ranking of the events

Details of Lifeloop generation has been introduced in Deliverable 6.4.

The property of a life loop for visual analytics may include:

1. Repeatability of individual pattern objects

2. Internal arrangements – internal sequence structures in the pattern object

3. Intelligence or knowledge can be discovered based on both the repeatability and internal structure of the pattern objects.

11.2 Lifeloop Visualisation

The life loops can be directly visualised as graphs, as shown in Figure 33.

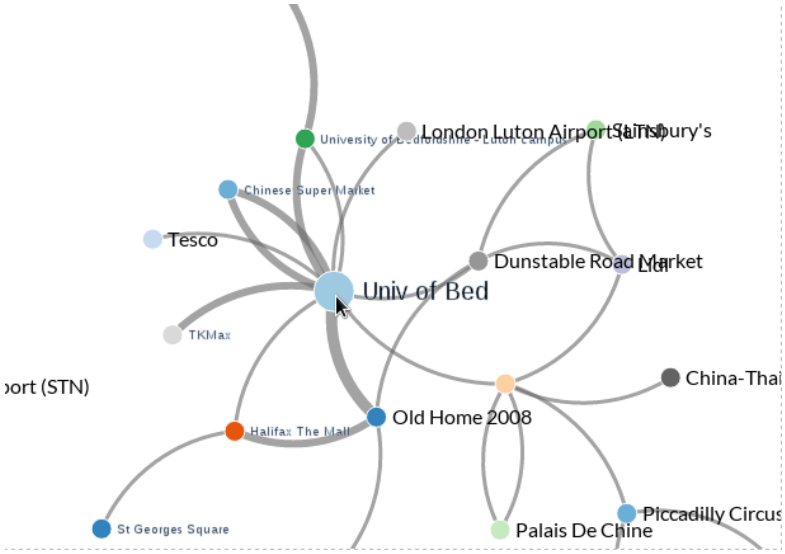

Figure 35 Storyline graph in the current Lifeloop interface

11.2.1 Lifeloop User Interface

The current Lifeloop user interface is composed of the following components, as shown in Figure 34 :

- Detail map
- Overview map
- Loopview
- Time range selector
- Radar chart user control

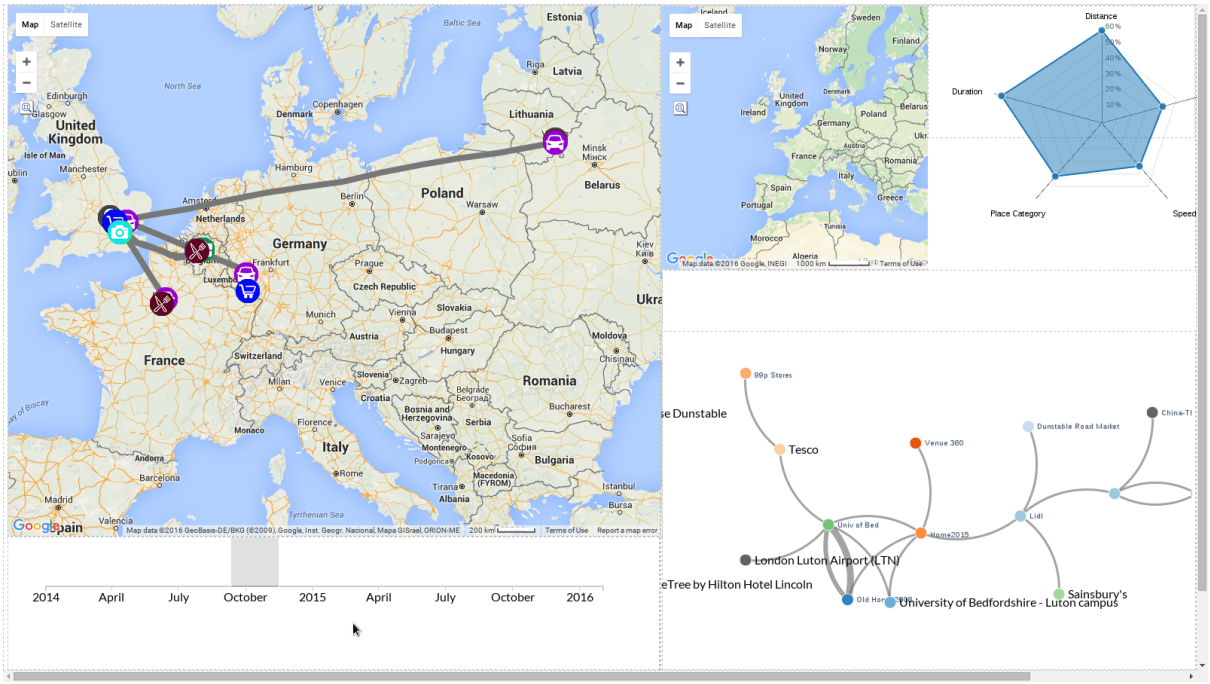

Figure 36 Lifeloop interface: initial design

11.2.2 User Interaction Design:

Envisaged user interactions are :

- Input time/date, find & show the loops with in the defined time period
- Input a few locations, find & show the relevant loops (events)
- Input semantics, find and show loops with relevant semantics
- User interaction to show links between the Map and Loop View
- A granularity bar to control how much detail of the loops is displayed (very good)
- A statistic view to show how often a loop is repeated within a time period selected
- Show important (typical) life pattern loops

11.3 Lifeloop Visual Analytics

The planned tasks of lifeloop visual analytics are listed as follows:

11.3.1 Define similarity between different loops

The similarity between two loops can be defined according to

The semantics of the events

Follow the same places (may consider duration as well)

Follow the same places & also be able to tolerate difference at nested loops

11.3.2 Clustering of the events based on the similarity

Clustering can take place based on the similarity defined above to show a number of typical trips (i.e. the center of the cluster)

11.3.3 Outlier detection of the events

An outlier can be defined as a loop that does not belong to any of the detected clusters.

11.3.4 Statistics of the events

We can apply statistics to events through a few different ways

Sum up according to the semantics of the events

Sum up according to the loop similarity

Sum up according the clustering outcome

11.3.5 Importance ranking of the events

Follow the TF-IDF scheme

Work out importance according to outliers via clustering

11.3.6 Understand the purpose of the loop

Investigating all the visited places (i.e. place, category, duration, time, date), and select the important ones (e.g. using a ranking method)

Create semantics for each sub-loop

11.4 Current Status

Currently the Lifeloop component is still under development and has not been integrated into the MyHealthAvatar web platform. The user interface of the development version is shown in Figure 34. The major features will be available soon.

12 Conclusion

MyHealthAvatar is a project designed to collect and track lifestyle and health data to promote citizen wellbeing. As a lifetime companion of citizens, the amount of data collected will be huge. It is almost impossible for citizen, patients and doctors to view, utilise and understand these data without proper visual presentation and user interaction. Interactive visual analytics of lifestyle data is one of the key features of MyHealthAvatar.

The purpose of MyHealthAvatar visual analytics is to promote healthy lifestyles of citizens and to assist clinical decision making. Visual analysis is a vehicle by which citizens can make basic analysis based on knowledge from the platform and medical professionals can augment their clinical knowledge with heterogeneous information from the avatar for clinical decision making and knowledge exploration.

MyHealthAvatar offers significant assistance to users by performing visually assisted data analysis (i.e. visual analytics) to extract clinically meaningful information from the heterogeneous data of individual/shared avatars, such as the patterns of symptoms, experience of treatments, medicines, self-care guidelines, risk factors etc.

MyHealthAvatar project has built an integrated web-based system [MHAWeb] and a mobile app, with a visual analytics suite featuring the following:

- 3D Body health information platform 3D Avatar
- Summary information visualization with the dashboard
- Life pattern visual analytics with the LifeTracker
- Daily activity information extraction, annotation and visualization from the data repository with the diary
- Profile summary display

In addition, MyHealthAvatar provides a powerful MobileApp to support health and lifestyle information collection, analysis and visualization, featuring the following:

- Overview health information display
- Detailed health statistics information recording and visualization
- Day view for lifestyle data recording and visualization

Testing and evaluation reports of individual components have shown that they have reached their design goals and formed an integrated visual analytics toolbox for people's health information collection, visualization and analysis.

13 References

[3DMax] Autodesk 3D Studio Max, http://www.autodesk.com/products/autodesk-3ds-max

[Android, GPS Accuracy] http://stackoverflow.com/questions/1567443/how-accurate-is-android-gps

[Android, Services] http://developer.android.com/guide/components/services.html

[BioDigitalHuman] BioDigital Human, https://www.biodigitalhuman.com/

[Camera and Views] https://developers.google.com/maps/documentation/android-api/views

[Cockburn2009] Andy Cockburn, Amy Karlson, Benjamin B. Bederson: A review of overview+detail, zooming, and focus+context interfaces. ACM Computing Surveys (CSUR) Surveys 41(1) (2009)

[Fitbit] Fitbit: http://www.fitbit.com/

[Fluxstream] Fluxstream: https://fluxtream.org/

[Foursquare] https://foursquare.com/

[Fruchterman 1991] T. Fruchterman, E.R.: Graph drawing by force-directed placement. Software - Practice & Experience(Wiley) 21(11), 1129–1164 (1991)

[Google-Camera] https://developers.google.com/android/reference/com/google/android/gms/maps/model/CameraPosition

[GoogleHealth] Google Health (discontinued) : http://www.google.com/intl/en-GB/health/about/

[GoogleMap] Google Maps: https://www.google.co.uk/maps

[GPS, Horizontal Accuracy] https://en.wikipedia.org/wiki/Global_Positioning_System#Selective%5Favailability

[Greenwich] https://en.wikipedia.org/wiki/Geographic_coordinate_system

[HealthVault] Microsoft HealthVault: https://www.healthvault.com/

[Holten2006] Holten, D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. IEEE Transactions on Visualization and Computer 12(5) (2006)

[iHealth] iHealth: http://www.ihealthlabs.com/

[Inselberg1990] A. Inselberg, B.D.: Parallel coordinates: A tool for visualizing multi-dimensional geometry. In: Proc. the 1st IEEE Symposium On Visualization. pp. 361–378. IEEE (1990)

[JQuery] JQuery, http://jquery.com/

[Keim2010] Keim D., Kohlhammer J., Ellis G., et. al. (Eds): Mastering the Information Age - Solving Problems with Visual Analytics. Eurographics Association (2010)

[Liu2014] Shixia Liu, Weiwei Cui, Y.W.M.L.: Parallel coordinates: A tool for visualizing multi-dimensional geometry. The Visual Computer 30(12), 1373–1393 (2014)

[MHA] MyHealthAvatar Project, http://www.myhealthavatar.eu/

[MHAWeb] MyHealthAvatar Platform, http://myhealthavatar.org

[Mobile Device, GPS Update Rate] https://learn.sparkfun.com/tutorials/gps-basics

[MobileGPS] Mobile Device, GPS update rate, https://learn.sparkfun.com/tutorials/gps-basics

[Open3dViewer] Open-3d-viewer: https://code.google.com/p/open-3d-viewer/

[OpenGL]OpenGL, http://www.opengl.org/

[Raphaël.js] http://raphaeljs.com/

[Riehmann2005]Patrick Riehmann, Manfred Hanfler, B.F.: Interactive sankey diagrams. In: Proc. IEEE Symposium on Infomation Visualization, InfoVis 2005. pp. 233–240. IEEE (2005)

[SceneJS] SceneJS, http://scenejs.org/

[Web Components] http://webcomponents.org/

[Three.js]Three.js, http://threejs.org/

[WebGL] WebGL, http://www.khronos.org/webgl/, http://en.wikipedia.org/wiki/WebGL

[Moves] Moves: https://www.moves-app.com/

[Withings] Withings: http://www.withings.com

[ZhaoY2013] Y Zhao, X Zhao, F Dong, G Clapworthy, N Ersotelos, E Liu, WebGL-based interactive rendering of whole body anatomy for web-oriented visualisation of avatar-centered digital health data. BIBE 2013: 1-4

[Zygote Body] Zygote Body, http://www.zygotebody.com/