Universiteit Leiden
Opleiding Informatica

Evaluating the feasibility of HTML5

for the visualization of bio-imaging and image-model data

Name:	Prashand Ramesar
Date:	04/09/2016
1st supervisor:	dr. ir. Fons J. Verbeek
2nd supervisor:	dr. Enrique Larios Vargas

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Evaluating the feasibility of HTML5 for the visualization of bio-imaging and image-model data

P. S. Ramesar

Abstract

The Imaging & Bioinformatics research group at LIACS processes large quantities of raw biological data into images, 3D models and videos. In this thesis, new methods will be developed to process, store, retrieve and natively visualize biological data with the help of HTML5 and up-to-date ICT tools. This must be done because older standards are becoming obsolete and have the disadvantage of requiring software installations before any data can be accessed by third parties. Evaluating the feasibility of HTML5 for server-based visualization can be accomplished by drawing an HTML5-based server architecture. This HTML5-based server architecture can then be qualitatively compared to the current Java3D-based server architecture to determine the differences between both methods of 3D model visualization. Furthermore, the capabilities of HTML5 have to be explored to determine how biological images and videos can be displayed natively in most web browsers. This research discovers that native server-based visualization of biological models can be achieved by extending the *Bio-Visualization web service* with Xia's parser to parse all contour information from the retrieved .PIX file into the generated .XML file. This .XML file can then be loaded into a web page that makes use of HTML5 and `Three.js` (i.e. Xia's template) to accomplish native web browser visualization. The difference between both methods of model visualization is that the HTML5 method requires slightly more computation on the client-side. Furthermore, it is also found out that biological images can be visualized natively in most web browsers using the <img> tag, provided the images are converted to a supported format with an image editing tool, such as *Pinta*. In addition, biological videos can be constructed from gene products using *Huygens Essential*. In order to make these videos ready for native HTML5-based visualization, they have to be converted to a supported format (such as .MP4) using a video editing tool. This video editing tool should also enable one to provide the video with a text watermark for the purpose of intellectual property protection. After the videos are stored in the file directory of the server, they can be retrieved by querying on their respective metadata and annotative information, which must be stored in the GEMS database. In summary, the visualization of biological images, models and videos is feasible and this thesis can hence be used as an argument to support the transition to HTML5.

Acknowledgements

First of all I would like to especially thank my supervisor dr. ir. Fons Verbeek, who took the time to guide me during the Summer months without any hesitation whatsoever. The timely completion of this thesis is largely due to the efforts and willingness on his part to make it happen. I always took his advice to heart and it paid off for me until the very end of the project.

I would also like to thank dr. Enrique Larios Vargas for graciously accepting the role of second supervisor despite his tight schedule. It was a very funny and unforgettable experience to witness him become my supervisor just hours before my defense. I thoroughly enjoyed the fun conversations we have had throughout the duration of my bachelor project.

Contents

List of Abbreviations

3NF	Third Normal Form
CLSM	Confocal laser scanning microscopy
DAOZ	Developmental Anatomy Ontology of Zebrafish
FISH	Fluorescent *in situ* Hybridization
FPS	Frames Per Second
GEMS	Gene Expression Management System
GO	Gene Ontology
IB	Imaging & Bioinformatics
LIACS	Leiden Institute of Advanced Computer Science
MAGE-ML	MicroArray Gene Expression Markup Language
TDRML	3D Reconstruction Markup Language
ZFIN	Zebrafish Information Network

List of Tables

List of Figures

Chapter 1

Introduction

The IB (Imaging & Bioinformatics) research group at LIACS (Leiden Institute of Advanced Computer Science) concerns itself with the acquisition and analysis of large volumes of biological data [38]. This thesis will explore new ways to keep biological data available to third parties. This chapter shall first briefly address the problem at hand in section 1.1, followed by the research questions that will attempt to solve the problem in section 1.2. Section 1.3 will go into the methodology taken to answer the research questions. In section 1.4, the chapter will conclude with an overview of the entire thesis.

1.1 Problem statement

The IB research group has been generating and visualizing images, models and videos from biological data [20]. It is the ambition of the IB research group to consistently keep these images, models and videos available online, so that third parties can have access to the data. Currently (anno 2016), the IB research group is largely visualizing 3D models online using Java3D-based technology [31]. As time goes on, older techniques become obsolete whereas emerging standards such as HTML5 bring with them a set of convenient advantages that could be considered for adoption [1]. This thesis will argue that newer technologies such as HTML5 can be used as a means to sustain and expand the availability of biological data to third parties. This is important because it is desirable to be as platform independent as possible by using methods that are supported by the most popular web browsers. To this end, the capabilities of HTML5 will be explored in order to understand to what degree HTML5 meets the demands of server-based visualization of biological data.

1.2 Research questions

As described in section 1.1, the capabilities of HTML5 will be explored and evaluated as a means to keep biological data available to third parties. In order to do this, a main research question will have to be formulated and answered. The research question is as follows:

Research question: *In what way is it possible to keep biological data available to third parties with the help of up-to-date ICT-tools?*

It is worth noting that the main research question is not limited to just HTML5, but rather to "up-to-date ICT-tools". This is important because the biological data may have to undergo preprocessing stages that are done with other tools before HTML5-based visualization is possible. In order to sufficiently answer the main research question, two additional research sub-questions will have to be formulated. The two sub-questions together provide an answer to the main research question:

Research sub-question 1: *What are the differences between server-based visualization of resources with Java3D and server-based visualization of resources with HTML5?*

Research sub-question 2: *What needs to be done to ensure the availability of video and image resources in HTML5?*

1.3 Methodology

First, relevant literature will have to be read in order to gain a thorough understanding of the current activities of the IB research group. This understanding will serve as the theoretical framework on which this research is based.

In order to answer the first research sub-question sufficiently, a new client-server architecture will have to be designed to determine in what way it is possible to keep biological data available to third parties. Subsequently, a qualitative comparison will be made between the current Java-based client-server architecture and the HTML5-based client-server architecture so that it becomes clear what the advantages and disadvantages are for each architecture. It is worth noting that this must be done to evaluate HTML5's feasibility to visualize biological *model* data.

The second sub-question is focused on HTML5's feasibility to visualize biological *video* and *image* data. In order to sufficiently answer this sub-question, a protocol will have to be developed that is focused on:

1. Producing videos from biological 3D data with the help of "standard" software.
2. Saving the produced video data in a specifically designed database structure.
3. Retrieving and visualizing video and image data with the help of database technology.

After answering these two research sub-questions, it will be clear how HTML5 deals with server-based visualization of biological *image*, *model* and *video* data. Together they provide an answer to the main research question. Furthermore, it will also be clear how the HTML5-based techniques compare to the techniques that are currently in place.

1.4 Thesis overview

In chapter 2, this thesis will go into the relevant theory to conduct the intended research. All research regarding the first research sub-question will be discussed in chapter 3. In chapter 4, all research regarding the second research sub-question will be discussed. The answer to the research sub-questions and, by extension, the main research question will be given and discussed in chapter 5. The final conclusions of this thesis can lastly be found in chapter 6.

Chapter 2

Background information

This chapter concerns itself with the necessary background information which will serve as the theoretical basis for this thesis. A proper understanding of the activities (i.e. acquisition, storage, management, annotation and retrieval of raw biological data) of the IB research group at LIACS is required to be able to answer the research questions. Section 2.1 will explain how raw biological data are acquired. Section 2.2 will provide information on how the biological data are stored and managed. Section 2.3 will then explain how the biological data are annotated and retrieved.

2.1 Acquisition of raw biological data

The first step of visualizing 3D gene expression patterns is the acquisition of raw biological data. In the world of bio-imaging and microscopy, there are several techniques to (re)construct 3D images [40]. One of these techniques is CLSM, or confocal laser scanning microscopy. Using the CLSM-technique makes it possible for one to visualize gene expressions by tagging fluorescent molecules. For example, for the zebrafish this is done at different stages of development: 24, 36, 48 and 72 hpf (*hours post-fertilisation*) [46].

2.2 Storage and management of raw biological data

Once the biological data have been acquired, all data need to be stored in a database and managed. To this end, the IB research group has produced a 3D Atlas of Zebrafish Development. With the help of the 3D atlas of Zebrafish Development, any produced gene expression pattern can be mapped on bodily structures [16,20,46].

The IB research group at LIACS has also constructed an accessible storage system called Gene Expression

Management System, or GEMS. Gene expression patterns are found by conducting *in situ* experiments. *In situ* means that the zebraFISH (FISH = Fluorescent *in situ* Hybridization) protocol is followed. The resulting gene expression patterns are 3D images that are submitted to the GEMS in the form of raw data. A benefit of this storage method is that a user can potentially process and morph the data according to their needs. There also exists a GEMS browser that visualizes the contents of the GEMS database [3].

The 3D Atlas and the GEMS enable researchers to store and manage information about gene expression patterns that are *spatio-temporal*. *Spatio* refers to the location on the anatomical structure (e.g. *head* or *tail*, whereas *temporal* refers to the time frame in which the gene expression occurs (e.g. *hours post-fertilisation*). The purpose of the construction of this information management system is to be able to openly publish gene expressions that are compatible with other information systems. To this end, the gene expressions in the images are described using the Gene Ontology, or GO [3]. More information about ontologies will be given in section 2.3.

In order to store reconstruction data (e.g. section images, contour information and metadata), the IB research group also established a 3D reconstruction data repository which will be accessed to reconstruct the 3D models [30].

On top of the two databases (the 3D reconstruction data repository and GEMS), TDR and GEMS web services have been established in order to provide access to the repositories. With these web services, application clients can be developed to provide users with a way to acquire and modify the reconstruction and gene expression patterns data from the databases. Potikanond et al. constructed the *Bio-Visualization web service*, which serves as an intermediate web application that enables users to browse and query for gene expression patterns and 3D reconstruction models in the online 3D digital atlas of zebrafish development. The *Bio-Visualization web service* is also used by the web application to acquire data from the ArrayExpress Atlas in order to visualize gene expression data within 3D reconstruction models over a network via Java applications [30].

When a user sends a query to the server, the query is executed by the web services (TDR and GEMS). This triggers the *Bio-Visualization web service* to send all relevant data to the web applications. Once this is done, the data are subsequently passed on to TDRViewer, a Java-based web application that delivers the visualization to the users [30]. The data transfer is done by sending XML-based files from the databases to the end-user due to XML's portability, extensibility and platform independence [5].

2.3 Annotation and reconstruction of raw biological data

In the world of bioinformatics, ontologies are used to help collect, organize and publish biological data [32]. To this end, the IB research group at LIACS established the Developmental Anatomy Ontology of Zebrafish, or DAOZ. This ontology is used to annotate collected biological data in the GEMS and to assign relevant annotation to bodily domains in the 3D Atlas. Structuring the data in this order enables users to write complex queries to consistently retrieve data for analysis [3].

For biological models, the next step is the 3D reconstruction phase, which is done with TDRViewer [17]. The TDRViewer also graphically annotates the relevant boundaries of bodily domains and/or spatial gene expression patterns in the 3D image dataset. Produced contour information is then used to reconstruct a 3D surface of each annotated "domain". During this stage, it is still not possible to enrich the model with data from other external sources. This is where the textual annotation from standard ontologies such as DAOZ is required. The ontologies are used to annotate anatomical domains with relevant terms. Similarly, gene expression patterns are annotated with genomic terms (such as gene name and gene symbol) in the GEMS. The vocabularies from DAOZ and GEMS are derived from standard ontologies such as ZFIN Anatomical Ontology and Gene Ontology [2,4,29].

After the retrieval of raw data from the databases and the annotation of anatomical structures, the 3D reconstruction models can be regarded as single instances of data that are described by a model description, which is done in an XML-based language, i.e. the 3D Reconstruction Markup Language (TDRML) . A TDRML-file contains information about annotated domains, section images and metadata. The domains are attached with 3D surface data and contour information [29].

The 3D reconstruction model also visually integrates a quantitative microarray analysis of patterns of gene expression. The quantitative analysis data are acquired from an external source, namely the ArrayExpress Atlas [7,29].

With TDRViewer, the user is able to visualize the dataset two- or three-dimensionally. The 2D view delivers a section image, along with its relevant graphical annotation of all relevant domains, i.e. the bodily structures for the atlas dataset and the areas where genes are expressed. One can also determine the zooming levels, along with which section image to view. TDRViewer offers three modes to view the visualizations, namely the contour view, the solid view and the surface view [30].

Chapter 3

Server architecture: a comparison

This chapter concerns itself with answering the first research sub-question. The current server architecture will be discussed in section 3.1 . The same will be done for the HTML5-based server architecture in section 3.2. The chapter concludes with a qualitative comparison of the two server architectures in section 3.3.

Research sub-question 1: *What are the differences between server-based visualization of resources with Java3D and server-based visualization of resources with HTML5?*

3.1 The current architecture

A display of the current server architecture can be found in Figure 3.1 below.

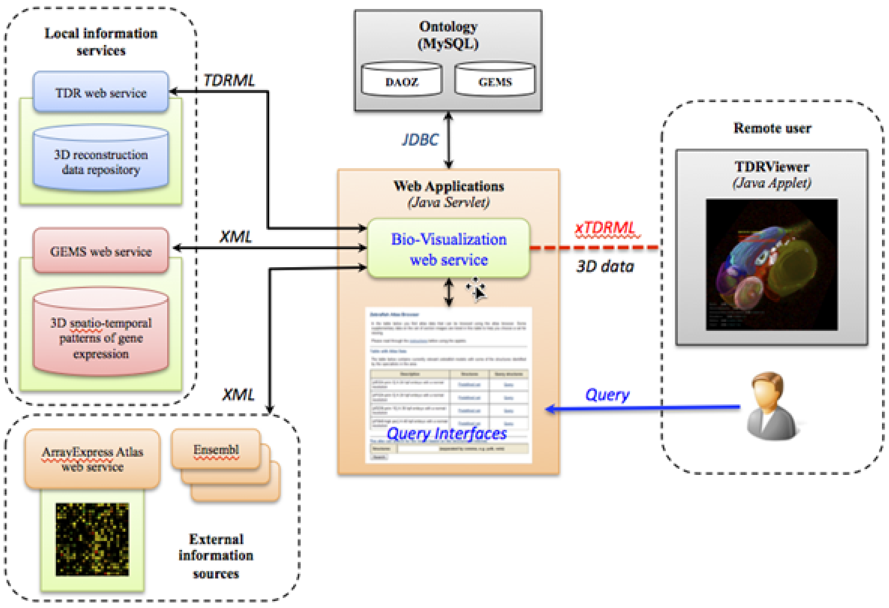

Local information services
TDR web service
3D reconstruction data repository
GEMS web service
3D spatio-temporal patterns of gene expression
TDRML
XML
XML
Ontology (MySQL)
DAOZ
GEMS
JDBC
Web Applications (Java Servlet)
Bio-Visualization web service
Query Interfaces
xTDRML
3D data
Query
Remote user
TDRViewer (Java Applet)
ArrayExpress Atlas web service
Ensembl
External information sources

Figure 3.1: The current server architecture, as illustrated in Potikanond et al. [29].

As one can see in Figure 3.1, the user is able to send queries through an online query interface. These queries are picked up by a Java Servlet called the *Bio-Visualization web service*. The *Bio-Visualization web service* then requests the relevant information from the web services, i.e. the TDR web service, the GEMS web service and the ArrayExpress Atlas web service. These web services retrieve the data from the underlying repositories. The ArrayExpress Atlas provides data from an external source [29].

The TDR web service ensures the exchange of data by receiving the queries from the *Bio-Visualization web service* and sending the relevant reconstruction data (such as the contour information, surface reconstruction information and the section images) forward to the client-side. These data are transmitted as an XML file (extended TDRML format) [29].

The GEMS web service sends back gene expression data in XML format. The gene expression data are stored based on annotation. This means that the data are linked to specific genes, hpf and the part of the bodily structure of the zebrafish where the expression of genes occurs [29].

The ArrayExpress Atlas web service is the only web service that does not contain locally stored data but rather imports microarray gene expression data externally, and sends this back as Microarray Gene Expression Markup Language format, also known as MAGE-ML, which is also an XML-based format [29].

The *Bio-Visualization web service* collects all the incoming data and annotates them using terms from the ontologies DAOZ and GO (for GEMS). All these data are then passed on to the TDRViewer. As mentioned previously in Chapter 2, TDRViewer is a Java-based web application that enables the user to view the 3D models in the web browser. [29].

3.2 The HTML5-based architecture

In this section, the relevant capabilities of HTML5 will be explored so that it can be determined to what degree HTML5 is able to take over the functionalities that are currently in place. This section will look into HTML5 as a modern standard for client-side development. After this, it will be explored which steps have to be taken on the server-side to support the HTML5-based client-side.

It has been verified by Xia et al. that HTML5 is suitable to visualize 3D brain models using `Three.js` [49]. `Three.js` is a library that simplifies WebGL-based visualization [25]. In order to visualize a model, the client needs to get the relevant section images and dimensions of the anatomical structure that needs to be visualized. To reduce the strain on the network connection, data transfer is minimized by sending the anatomical structure data as Freeman chain code to the client (in XML format), where the model will be reconstructed [42]. Xia's method also shows that an "overlap" can be drawn on top of the section images. This can be used to graphically annotate the section images [49].

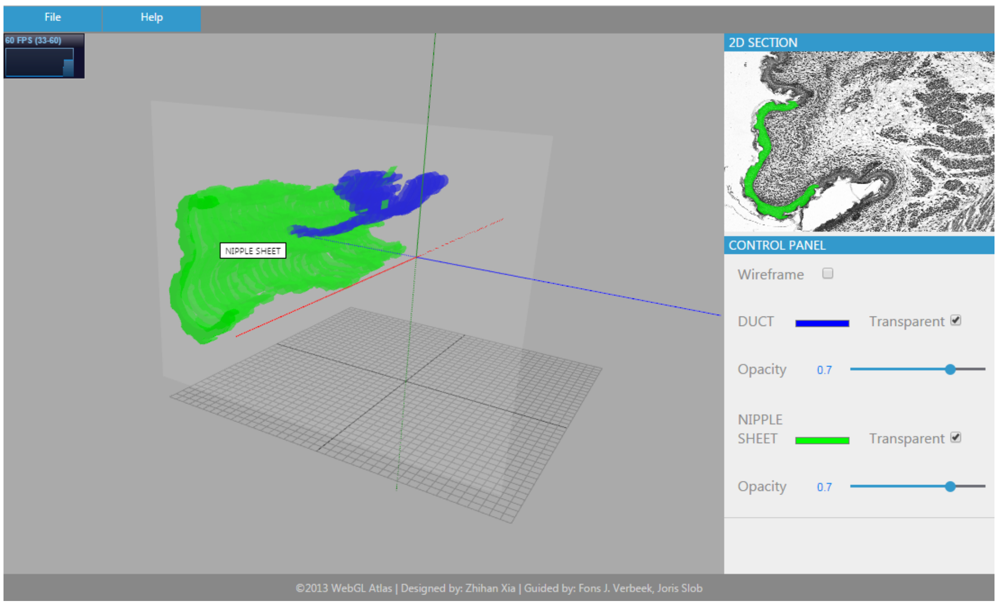
The HTML5-based architecture is very similar to the current architecture. The server-side of the model stays almost entirely intact, whereas the actual visualization done at the client-side will be renewed. The actual client-side visualization can be done in using a WebGL-based engine, such as `Three.js` along with HTML5 [25].

The contour information is encoded in .PIX format and cannot be processed immediately in HTML5. To this end, Xia developed a parser that can merge the contour information into the .XML file. This makes it possible to draw the relevant data points in the 3D canvas using `Three.js` [49].

Experiment

In order to check if the visualization technique by Xia works for models other than the brain, a prototype model has to be constructed from different biological data. Doing so will also provide insight into the exact workings of his technique and how it can be used to conduct automated server-based model visualization.

For this experiment, the mammary gland of the *Mus musculus* (house mouse) is visualized using the model's .PIX and .XML file. This experiment is conducted by first using the parser to merge the contour information from the .PIX file into the related .XML file. Thereafter, the enhanced .XML file is drawn using Xia's HTML5 "template". The result of this experiment can be found below in Figure 3.2.



Wireframe ☐

DUCT Transparent ✓

Opacity 0.7

NIPPLE SHEET Transparent ✓

Opacity 0.7

©2013 WebGL Atlas | Designed by: Zhihan Xia | Guided by: Fons J. Verbeek, Joris Slob

Figure 3.2: The visualization of the mammary gland of the *Mus musculus* using Xia's template.

The labels (DUCT and NIPPLE SHEET), colours, section images and contour information are all exported to the browser where everything is drawn. After providing some bug fixes, the model viewer now correctly scales the section images and it also correctly draws the contour information on the section images. Furthermore, it is now possible to rotate the image around its own origin. Another bug involving the incorrect loading of the .XML files has also been fixed. This shows that there are a few steps that can be taken to produce a server-based visualization service using this technique.

The first step is to extend the client-server model. Not only should the section images, the .PIX file and the .XML file be retrieved by the *Bio-Visualization web service*, it should also produce the "enhanced" .XML by using Xia's chain code parser to integrate the .XML file with the contour information from the .PIX file. This makes visualization of biological 3D data in HTMl5 feasible, because the enhanced .XML file and the section images can then be loaded into Xia's HTML5 template. This can all be done by extending the *Bio-Visualization web service* to perform the tasks. The template consists of a folder with therein:

1. A `css` folder that contains lay-out information. The *Bio-Visualization web service* does not adjust this folder.
2. A `js` folder that contains the necessary JavaScript files (such as `Three.js`). The *Bio-Visualization web service* does not adjust this folder.
3. A `2D_PIC` folder that contains the section images and the enhanced .XML file. The *Bio-Visualization web service* has to put the recently queried section images along with the .XML file in this folder.
4. An .HTML file that draws the entire application. The *Bio-Visualization web service* has to adjust this file where it refers to the location of the enhanced .*Bio-Visualization web service* file and where it refers to the folder that contains the section images (`2D_PIC` by default). It should additionally provide the correct dimensions (height and width in pixels) of the section images to the .HTML file.

For clarity, the .HTML file has to be adjusted by the *Bio-Visualization web service* on lines 38 and 452 to load all the retrieved files into Xia's template. These two lines are explicitly highlighted below.

```
var xmlfilename = "./2D_PIC/b.xml"; //line 38: refer to the extended .xml


for (var il = 0; il <= layerNum-layerCounter; il++) {
        var srcStr = xmlDoc.getElementsByTagName("img")[il].getAttribute("src");
        imgSrcs[il] = "./2D_PIC/"+srcStr;//line 452: refer to the folder that
}                                             //contains the section images
```

The .HTML5 file also needs to receive the correct dimensions of the section images on lines 72, 77, 764, 766, 807 and 810. An illustration of this process can be found below. This must be done to correctly scale the section images and the contour information to the size of the image viewer window, i.e. 200 ∗ 300 pixels.

```
context.scale(300/(width_section_image,200/height_section_image); //lines 72,77,766,810
context2.scale(width_section_image/300,height_section_image/200);//lines 764,807
```

The process described in the previous part of this section are sufficient to load the images and models into the template. More visualizations using this technique can be found in Appendices A, B, C and D. It should be noted that Xia's template only deals with the visualization of the section images and does not contain annotation from the ArrayExpress Atlas web service. A possible workaround is to extend Xia's parser to not only parse the contour information from the .PIX file into the original .XML file, but to also include the relevant annotation from the .XML file that is produced by the ArrayExpress Atlas web service. The annotation could be put into the labels of the section images. These section images are labeled in the original .XML file with the "name" attribute of the <biolstruct> tag. An example is as follows:

```
<biolstruct color="0,0,255" label="1" name="DUCT">
```

The "name" attribute can be used to graphically annotate section images by adding the relevant annotation that is taken from the ArrayExpress Atlas web service to the attribute. The annotation can be further extended by drawing a text overlay on the 3D canvas to include gene expression data from the ArrayExpress Atlas. This text overlay can, for instance, be drawn with `Angular` and `TypeScript` [8].

After the files have been loaded into the template, the entire server-side process is complete and the web page is loaded on the client-side (the 3D Atlas of Zebrafish Development). A web page on the bio-imaging website should contain the hyperlinks to all the models that can be visualized. When a user clicks on a hyperlink, a request is made to the server that prompts the *Bio-Visualization web service* to query for the relevant biological model data. This new visualization method requires a slightly extended server-client architecture, because it incorporates the use of Xia's parser on the server-side. Additionally, the enhanced .XML file and the relevant biological section images have to be loaded in Xia's HTML5 template on the client-side. A high-level concept of the HTML5-based client-server model can be found in Figure 3.3. A larger version of this architecture is included in Appendix E.

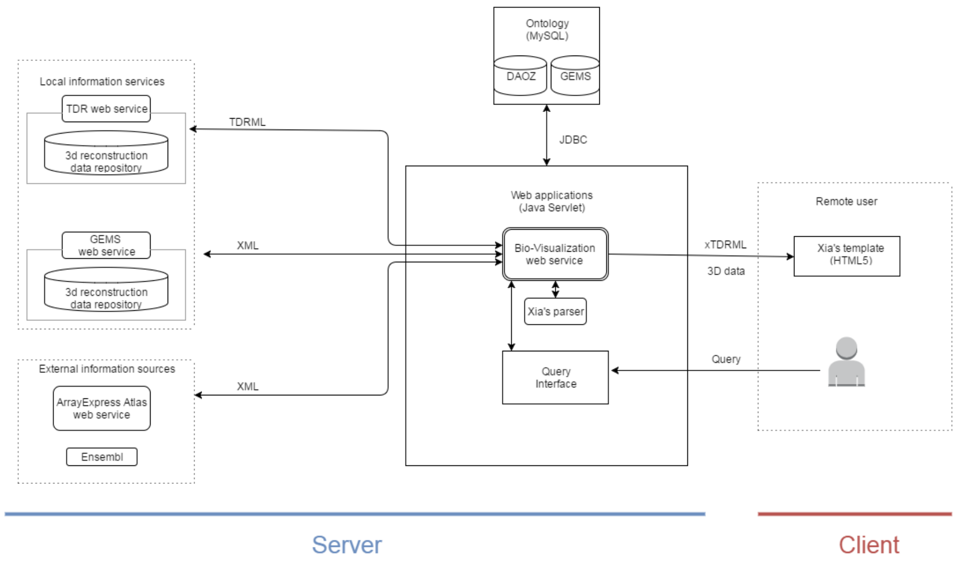

Ontology (MySQL)
DAOZ
GEMS
Local information services
TDR web service
3d reconstruction data repository
TDRML
JDBC
Web applications (Java Servlet)
GEMS web service
3d reconstruction data repository
XML
Bio-Visualization web service
xTDRML
3D data
Remote user
Xia's template (HTML5)
Xia's parser
External information sources
ArrayExpress Atlas web service
Ensembl
XML
Query Interface
Query
Server
Client

Figure 3.3: The new, HTML5-based server architecture.

Note that the new architecture is an adaptation of the MVC-architecture. The *Model*-part contains the logic, the model data and all the processing steps (such as generating the .XML file) that have to be taken. The *View*-part contains the files that are meant to be loaded on the client-side, i.e. in Xia's template. The *Controller*-part is basically the *Bio-Visualization web service* that receives the HTTP-requests from the client and prompts the *Model*-part to prepare the data according to the user's request by loading the data into the *View*-part [3,6].

To elaborate further, the extension of the *Bio-Visualization web service* can be achieved by using JavaServer Pages (JSP) to generate dynamic pages by the server [37]. These web pages integrate the required information from the databases (section images, metadata etc.) into the template. It can also be used to call the parser to correctly parse the contour information from the .PIX file into the .XML file.

To illustrate, the web server gets an HTTP request from a user to load a certain model in the developed template (which is now a .JSP file). The .JSP file should contain queries to retrieve the required information from the underlying databases (mostly from the TDR database). The .JSP file needs to know *the folder that contains the section images and the .XML file*. Additonally the .JSP file needs to know the *height* and *width* of the section images. All this information is different for each model, which is why the use of JSP is required to generate the page "on the fly". This .JSP page is then transformed into a servlet (which is a .JAVA file), and thereupon converted into an executable .CLASS file. Upon running the .CLASS file, all relevant information (.XML file, section images and metadata) are queried for and loaded into the template. This template is then transformed into a static .HTML file, which can be sent back to the user as an HTTP response [36].

3.3 Comparing the new and current architectures

In this section the current and HTML5-based architectures are compared to each other at the server- and client-side. Additionally, the quality of the server architectures are compared.

Considering the *server-side* of the architectures, not much has changed. The *Bio-Visualization web service* will have to call Xia's parser before loading the enhanced .XML file into the client side. Xia's parser is a very light program that only reads two files to produce another file.

Considering the *client-side* of the architecture, a lot has changed, relatively speaking. The TDRViewer could be considered outdated technology since Oracle has released JavaFX, a modern way to build web applications in Java [27]. However, HTML5 holds the advantage that it is a new standard because it is supported by most modern web browsers over multiple operating systems [12–15]. Furthermore, Java3D and JavaFX require a client-side installation of Java, which means that fewer devices are able to view the visualizations in the web browser, whereas no extra installations are necessary whatsoever when the visualization is done in HTML5.

In order to determine the quality of the architectures, quality attributes can be used to assess the degree to which each architecture meets them. These quality attributes are based on O'Brien et al. [26]. Some quality attributes are left out because they share no relevance to this specific server setup. The following quality attributes will be used to compare the architectures:

- Interoperability
- Message and server reliability
- Availability
- Data granularity
- Normal usability operations
- Confidentiality
- Integrity
- Performance
- Scalability
- Extensibility
- Modifiability

In order to be able to visualize the model data, the server needs to transform the data (from .PIX to .XML). This indicates that the *interoperability* of the server architecture is acceptable, but not very high. The *Bio-Visualization web service* performs well when integrating different types of (XML-based) data from different sources. However, in the event that the server architecture needs to deploy the model data to another client-side platform, it may be the case that the data require another stage of preprocessing before visualization is feasible.

Considering the minimal changes in the "new" server architecture, there is no noticeable reduction in *message* and *service reliability*. The probability that the server fails at the data preprocessing stages (the parsing of the data) can be considered negligible to zero, because the parser is a small program (5.7 kB in size, extracted) that performs very light tasks.

The server should also see no decline in *availability* due to the server architecture's extensions for the same reasons listed as the *message* and *service reliability*. Additionally, the changes made to the architecture do not otherwise interfere with the server's availability.

As for the *data granularity*, this quality attribute does not impact the quality of the architectures either. This is because the data are only transformed into a different format, which means that the data hold the same properties as before. The *normal usability operations* are also not impacted, because both visualization methods are designed to perform the same tasks.

The *confidentiality* is decent to high for both server architectures. The GEMS database, for instance, requires authentication before any data can be submitted or changed. Generally, anonymous users are only able to view the data [3].

The *integrity* of the data lies in the hands of the users that are authorized to submit data to the databases. This holds true for both server architectures and indicates that the integrity is decent to high, because these users understand what properties the data must possess.

The *performance* for both server architectures is acceptable, but not high. This is because the data are quite substantial in size and have to be sent from multiple repositories to the end users. Additionally, both server architectures are .XML-based, which require extra steps (parsing, validation and transformation of data) to perform the tasks [26]. This also indicates that the HTML5-based architecture takes a slightly longer time to prepare the data than the Java3D-based architecture, because of the extra preprocessing step that is required.

Since biological data are substantial in size and data are transmitted in .XML-based formats, the *extensibility* is not high for both server architectures. Should it happen that there are many more end users submitting and retrieving data at the same time (e.g. the server load is increased by a factor of 100), then it may lead to the need of hardware and software upgrades [26]. As to how much the system will change in size and volume in the foreseeable future remains an open question.

The *modifiability* of both server architectures is quite high thanks to the use of .XML-based formats. XML has the benefits of being flexible, extensible and can embed metadata [26]. This means that it is relatively easy to change the system (e.g. making a change in the user interface) because it is possible to morph the data.

There are different kinds of model representations [39, 41]. Aside from the contour view in TDRViewer, `Three.js` is also able to load a surface view by simply querying for the model's .OBJ file (which is present in the TDR database) and loading it using the OBJLoader in `Three.js` [35]. Relevant annotation can be retrieved from the ArrayExpress Atlas web service and visualized using a text overlay. This illustrates that it is possible to keep the model data available using up-to-date tools.

A potential downside to using HTML5 instead of Java to visualize models is that HTML5 is a relatively new standard which means that the native support for 3D visualization in web browsers (WebGL) is still in development whereas Java3D has been around since 1998 [11]. This implies that there exists more material such as documentation and samples for Java3D. It also means that it is not clear at which point HTMl5 stops developing which could raise slight concerns for future development. It could happen that a future extension will not be supported by HTML5, however, HTML5 (and more specifically, `Three.js`) is able to at least visualize and graphically label 2D sections and 3D models by querying for the relevant data from the databases and deploying them to the client-side. With regards to the vast and global usage of HTML5 [23], the expectation remains that it is a future-proof and extensible platform for online visualization.

Another problem with HTML5-based visualization is that it takes a relatively long time to load the model, the section images and all of the required information into the web browser. The difference is that the Java3D-based technique *caches* and *preloads* the information in smaller "chunks" to improve loading speed whilst the HTML5-based technique loads everything all at once, which puts a strain on the user's computer and their internet connection. It would therefore be recommended to develop methods which ensure the preloading and caching of information for the HTML5-based technique.

Chapter 4

Developing a protocol to visualize biological video and image data

In this chapter, an answer to the second research sub-question will be given, namely a protocol that states in what way videos from biological data can be made, stored and retrieved. The chapter is divided in two different sections. Section 4.1 will explain how the videos will be made to effectively capture videos using biological data and section 4.2 will then explain how biological videos and images can be stored and retrieved.

Research sub-question 2: *What needs to be done to ensure the availability of video and image resources in HTML5?*

4.1 Capturing videos using Huygens Essential

This section concerns itself with capturing videos using *Huyens Essential*, a scientific tool that is used to analyze and render biological data [33]. Capturing and publishing videos of gene products can be very helpful to scientists such as bioinformaticians [9].

After *Huygens Essential* is started, open the file that needs to be rendered into a video. One can split the file into its respective channels, as is shown in Figure 4.1.

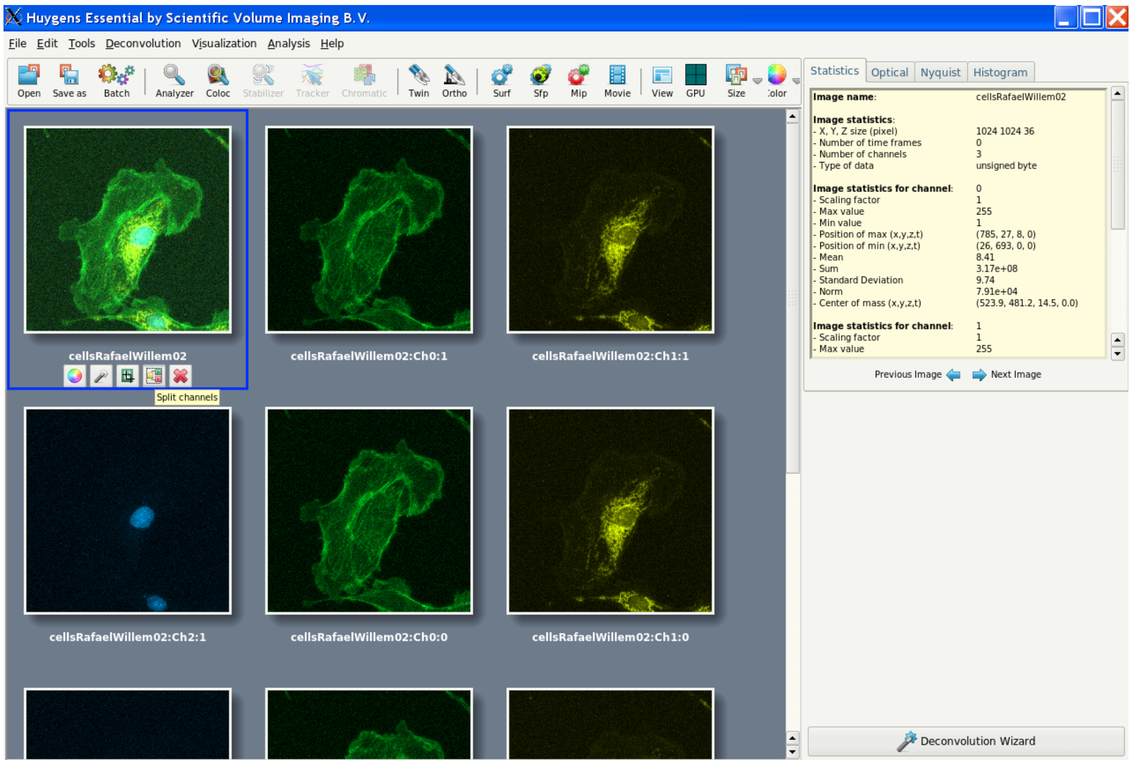

Huygens Essential by Scientific Volume Imaging B.V.
File Edit Tools Deconvolution Visualization Analysis Help
Open Save as Batch Analyzer Coloc Stabilizer Tracker Chromatic Twin Ortho Surf Sfp Mip Movie View GPU Size Color
Statistics Optical Nyquist Histogram
cellsRafaelWillem02
Split channels
cellsRafaelWillem02:Ch0:1
cellsRafaelWillem02:Ch1:1
cellsRafaelWillem02:Ch2:1
cellsRafaelWillem02:Ch0:0
cellsRafaelWillem02:Ch1:0
Previous Image
Next Image
Deconvolution Wizard

Image name:	cellsRafaelWillem02
Image statistics:	
- X, Y, Z size (pixel)	1024 1024 36
- Number of time frames	0
- Number of channels	3
- Type of data	unsigned byte
Image statistics for channel:	0
- Scaling factor	1
- Max value	255
- Min value	1
- Position of max (x,y,z,t)	(785, 27, 8, 0)
- Position of min (x,y,z,t)	(26, 693, 0, 0)
- Mean	8.41
- Sum	3.17e+08
- Standard Deviation	9.74
- Norm	7.91e+04
- Center of mass (x,y,z,t)	(523.9, 481.2, 14.5, 0.0)
Image statistics for channel:	1
- Scaling factor	1
- Max value	255

Figure 4.1: Splitting the channels using *Huygens Essential*.

After splitting the channels, a recommended approach is to make a video of every channel following the same standard, because the videos can then be easily compared with each other. Click on the object of choice and select the option *Movie maker*, as shown in Figure 4.2.

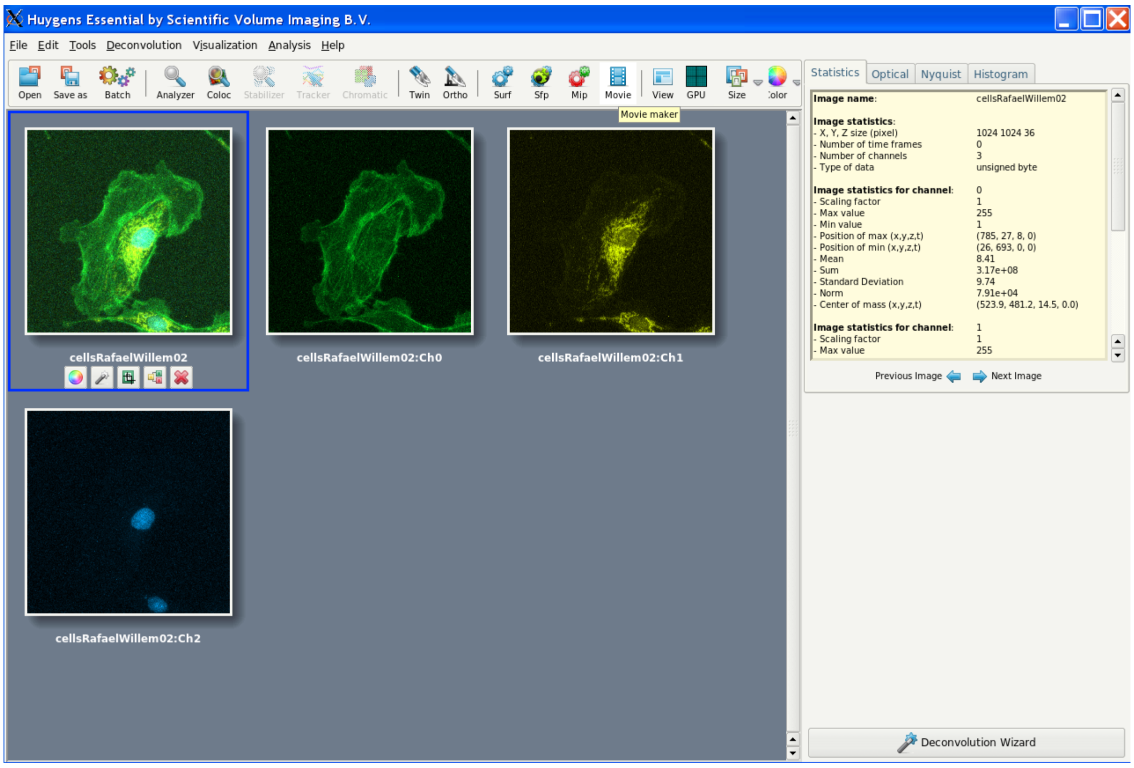

Huygens Essential by Scientific Volume Imaging B.V.
File Edit Tools Deconvolution Visualization Analysis Help
Open Save as Batch Analyzer Coloc Stabilizer Tracker Chromatic Twin Ortho Surf Sfp Mip Movie View GPU Size Color
Movie maker
Statistics Optical Nyquist Histogram
cellsRafaelWillem02
cellsRafaelWillem02:Ch0
cellsRafaelWillem02:Ch1
cellsRafaelWillem02:Ch2
Previous Image
Next Image
Deconvolution Wizard

Image name:	cellsRafaelWillem02
Image statistics:	
- X, Y, Z size (pixel)	1024 1024 36
- Number of time frames	0
- Number of channels	3
- Type of data	unsigned byte
Image statistics for channel:	0
- Scaling factor	1
- Max value	255
- Min value	1
- Position of max (x,y,z,t)	(785, 27, 8, 0)
- Position of min (x,y,z,t)	(26, 693, 0, 0)
- Mean	8.41
- Sum	3.17e+08
- Standard Deviation	9.74
- Norm	7.91e+04
- Center of mass (x,y,z,t)	(523.9, 481.2, 14.5, 0.0)
Image statistics for channel:	1
- Scaling factor	1
- Max value	255

Figure 4.2: Selecting the *Movie maker* in *Huygens Essential*.

When the *Movie maker* is opened, a renderer has to be selected. Select the *Sfp* renderer in the *Movie maker* window, as shown in Figure 4.3.

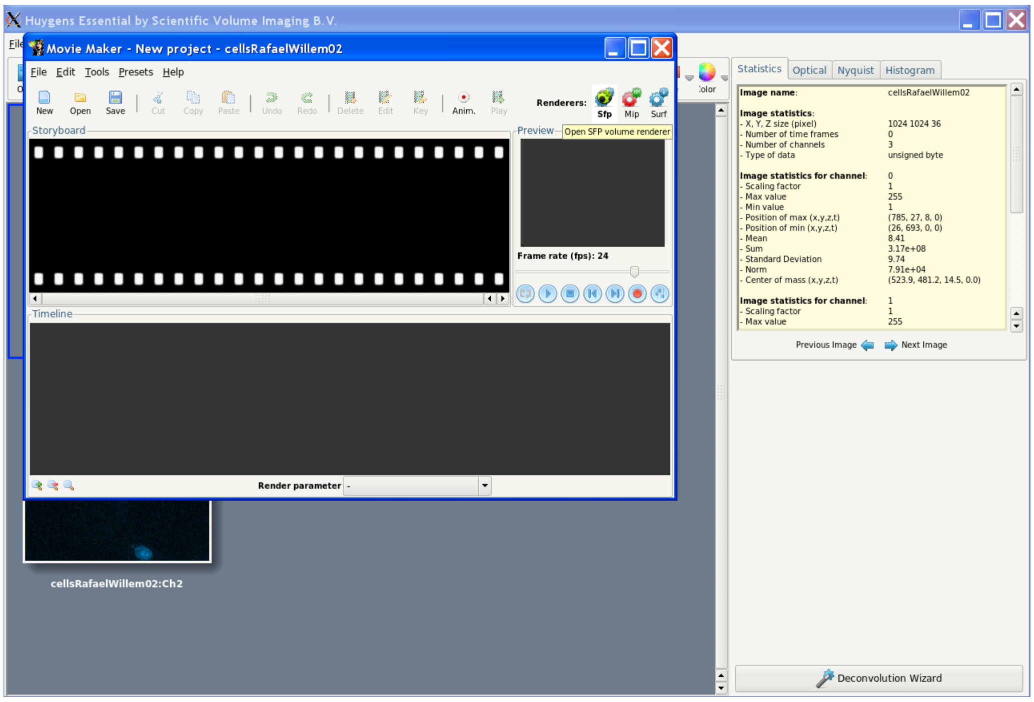
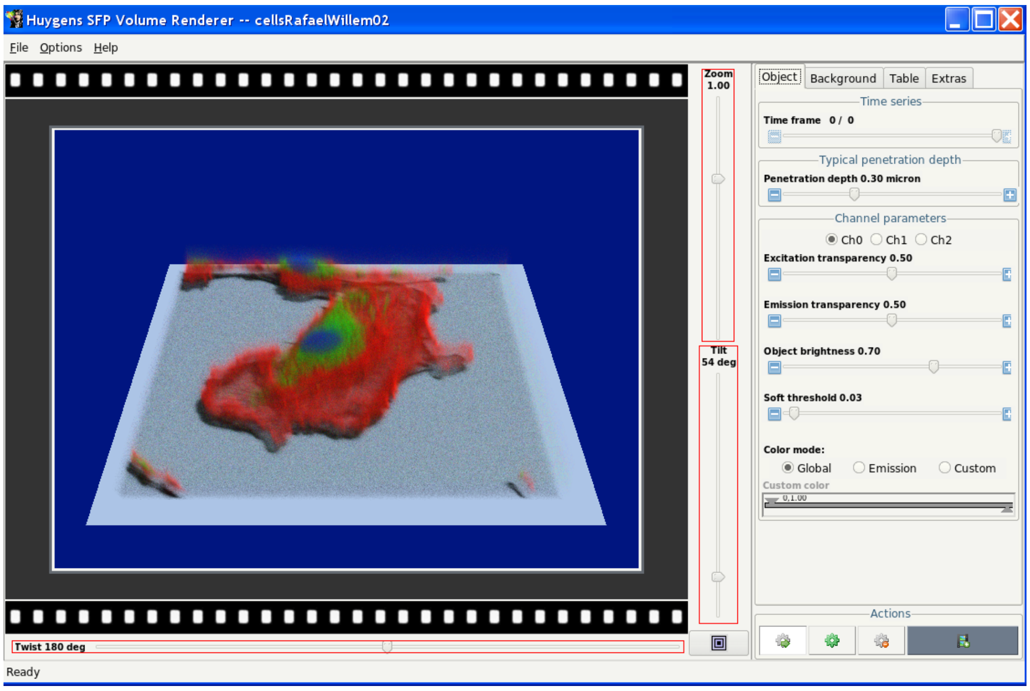

Huygens Essential by Scientific Volume Imaging B.V.
Movie Maker - New project - cellsRafaelWillem02
File Edit Tools Presets Help
New Open Save Cut Copy Paste Undo Redo Delete Edit Key Anim. Play
Renderers: Sfp Mip Surf
Storyboard
Preview
Open SFP volume renderer
Frame rate (fps): 24
Timeline
Render parameter
cellsRafaelWillem02:Ch2
Statistics Optical Nyquist Histogram
Previous Image
Next Image
Deconvolution Wizard

Image name:	cellsRafaelWillem02
Image statistics:	
- X, Y, Z size (pixel)	1024 1024 36
- Number of time frames	0
- Number of channels	3
- Type of data	unsigned byte
Image statistics for channel:	0
- Scaling factor	1
- Max value	255
- Min value	1
- Position of max (x,y,z,t)	(785, 27, 8, 0)
- Position of min (x,y,z,t)	(26, 693, 0, 0)
- Mean	8.41
- Sum	3.17e+08
- Standard Deviation	9.74
- Norm	7.91e+04
- Center of mass (x,y,z,t)	(523.9, 481.2, 14.5, 0.0)
Image statistics for channel:	1
- Scaling factor	1
- Max value	255

Figure 4.3: Selecting the *Sfp* renderer in *Huygens Essential*.

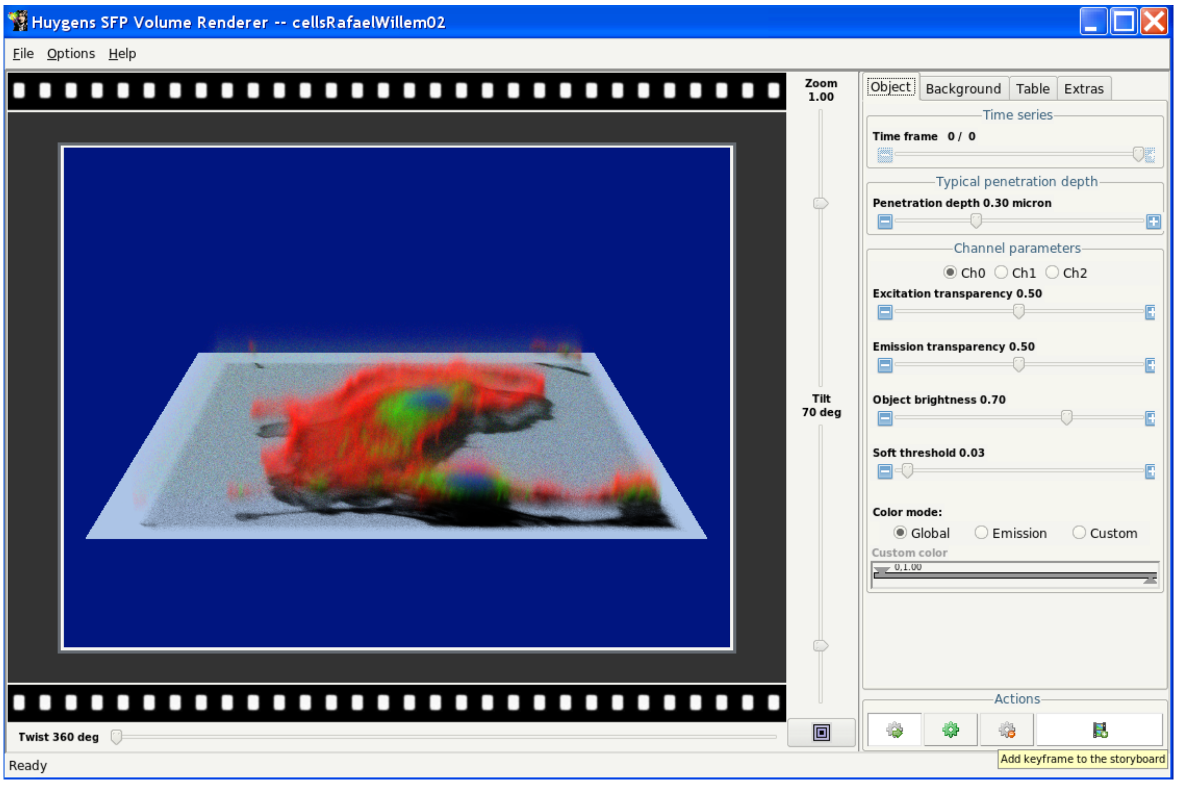
Once the *Sfp* renderer has started, it is time to make videos. Notice that the *Sfp* renderer has three axes to adjust the view of the object, namely *Zoom*, *Tilt* and *Twist*, which can be seen in Figure 4.4.


Huygens SFP Volume Renderer -- cellsRafaelWillem02
File Options Help
Object Background Table Extras
Time series
Typical penetration depth
Channel parameters
Color mode:
Custom color
Actions
Ready

Zoom 1.00
Tilt 54 deg
Twist 180 deg
Time frame 0 / 0
Penetration depth 0.30 micron
[✓ Ch0] Ch1 Ch2
Excitation transparency 0.50
Emission transparency 0.50
Object brightness 0.70
Soft threshold 0.03
Color mode: [✓ Global] Emission Custom

Figure 4.4: The red boxes show *Zoom*, *Twist* and *Tilt* within the *Sfp* renderer.

The purpose of the video is to capture the entire object. Therefore the proposed method is to first capture the object in the renderer from a *side view*. The view should then be rotated 360 degrees to capture every angle from the side. The camera should then capture a *top-down view* of the object to get a good picture of the entire object. The first frame to capture is at the setting {Twist=360deg;Zoom=1.00;Tilt=70deg}. Once the *Sfp* renderer is set to this setting, select *Add keyframe to the storyboard* in the bottom right corner of the *Sfp* renderer, as shown in Figure 4.5 below.


Huygens SFP Volume Renderer -- cellsRafaelWillem02

Zoom 1.00

Tilt 70 deg

Twist 360 deg

Time series: Time frame 0 / 0

Typical penetration depth: Penetration depth 0.30 micron

Channel parameters: [✓ Ch0] Ch1 Ch2

Excitation transparency 0.50

Emission transparency 0.50

Object brightness 0.70

Soft threshold 0.03

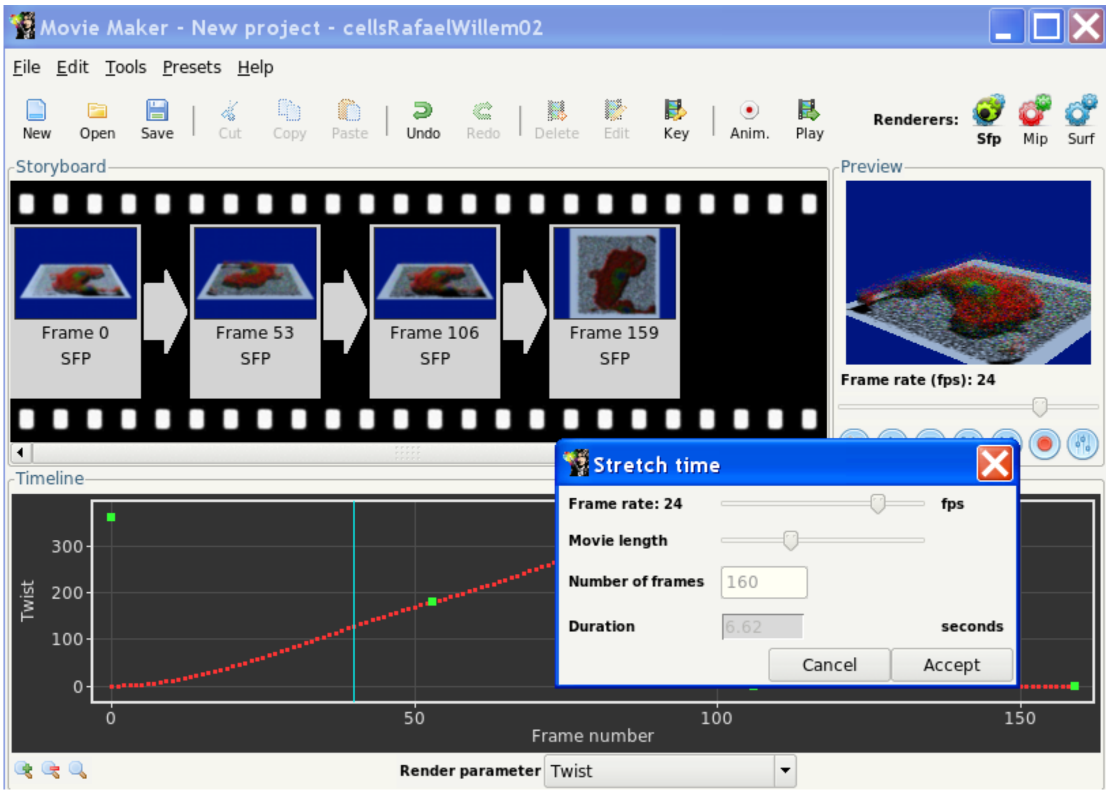
Color mode: [✓ Global] Emission Custom

Ready

Figure 4.5: Adding keyframes to the storyboard using *Huygens Essential*.

Leave the *Zoom* and *Tilt* settings the way they are (at {Zoom=1.00;Tilt=70deg}), set *Twist* to 180 degrees and add the keyframe to the storyboard in the *Movie maker* window. Repeat this step once more, turning *Zoom* 180 degrees to the right, so that the setting {Twist=0deg;Zoom=1.00;Tilt=70deg} is also added to the storyboard. At this point the entire *side view* is captured. The next step is to capture one last keyframe at the setting {Twist=0deg;Zoom=1.00;Tilt=0deg}. This ensures that a *top-down view* of the object is captured.

After the keyframes have been added to the storyboard, go to the *Movie maker* window and first select *Tools→Stretch movie* to open a new window. Make sure that the frame rate is set at the desired frames per second (FPS) and that movie length is set to a desired amount of frames, as shown in Figure 4.6. Include a sufficient amount of frames to ensure that the transitions in the movie do not pass too quickly.



Frame rate: 24 fps
Movie length
Number of frames 160
Duration 6.62 seconds

Figure 4.6: Adjust the frame rate and number of frames to stretch the movie in *Huygens Essential*.

After this step is completed, click on the red *record* button in the *Movie maker* window, as shown in figure 4.7.

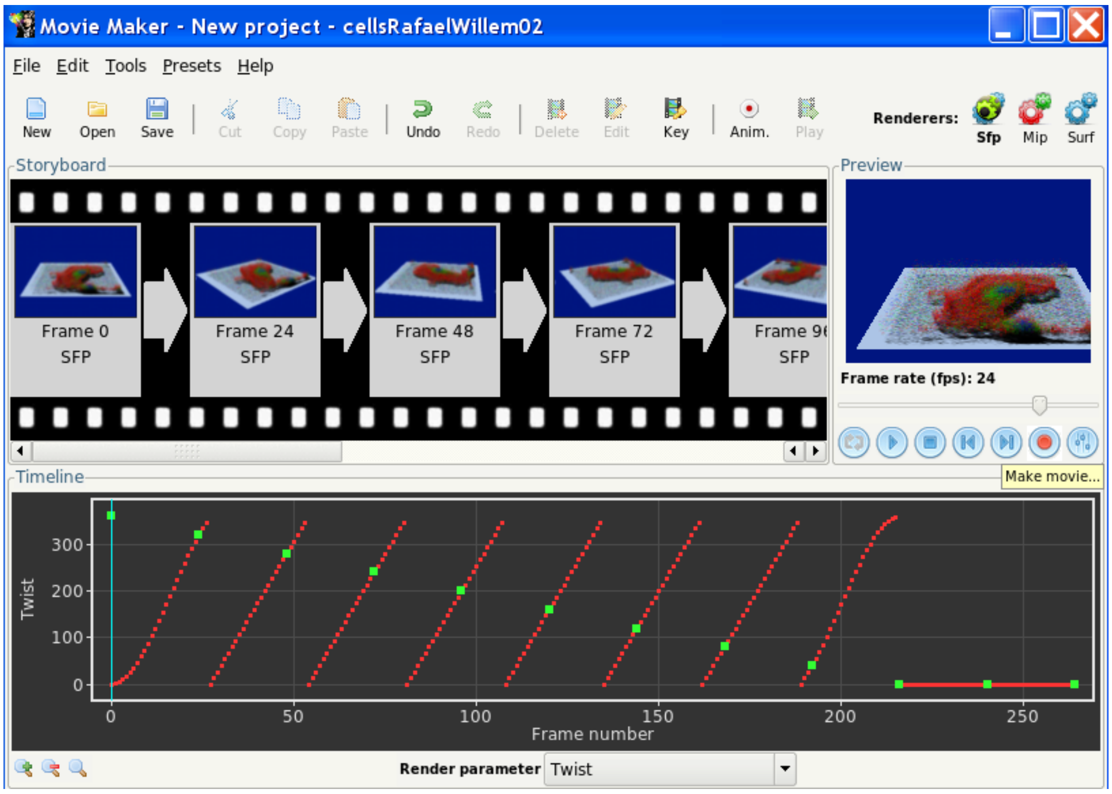
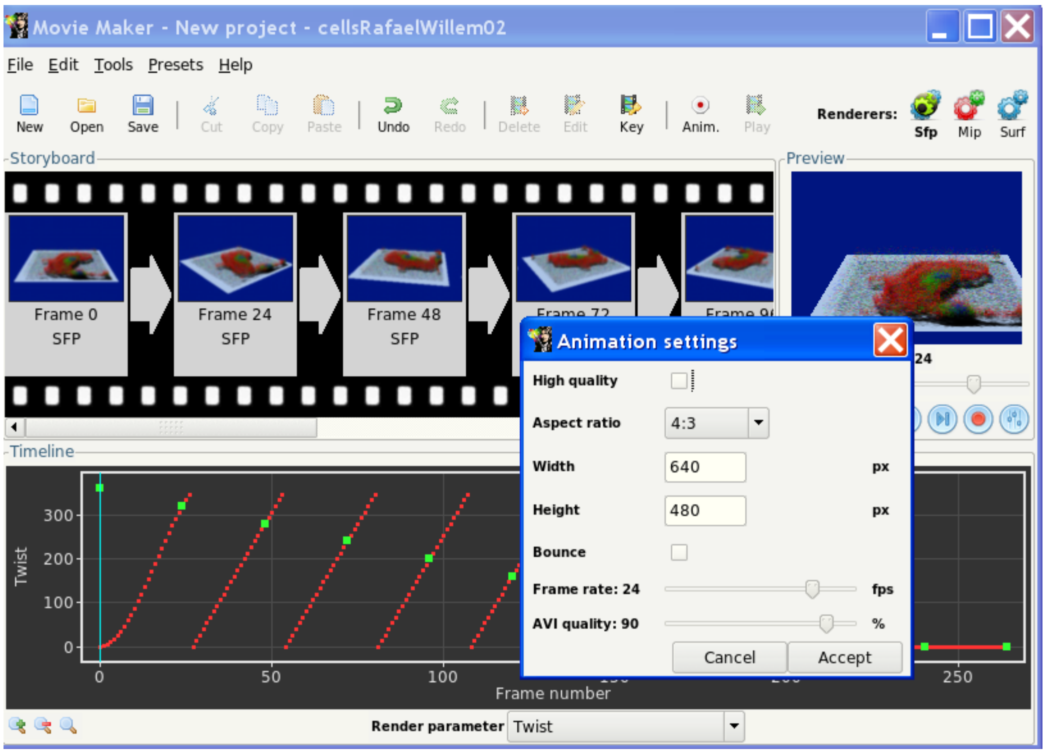


Figure 4.7: The red button is used to record a movie.

When clicking the record button, a window will pop up asking for the animation settings. It is possible to save memory space by keeping the file size small. This can be done by unchecking the 'High quality' option and by leaving the settings to an aspect ratio of 4:3, a video width of 640 pixels and a video height of 480 pixels as shown in Figure 4.8. Make sure to give up a desired frame rate and a desired AVI quality. Press *Accept* to continue after making sure that the settings are correct.



High quality	☐	
Aspect ratio	4:3	
Width	640	px
Height	480	px
Bounce	☐	
Frame rate: 24		fps
AVI quality: 90		%

Figure 4.8: Make sure the animation settings are correct to save local storage space.

After confirming the animation settings, a client pops up asking for the name of the movie and the location where it should be saved. Make sure to save the movie in .AVI format. Repeat this process for every channel until all channels are done. Note that this is a laborious task and it may take a while before the videos are rendered. After this step is completed, make sure to use a tool to convert the .AVI movie to .MP4, .Ogg or .WebM format since these formats can be played natively in HTML5 [45]. An easy way to solve this problem is to use a freely available tool, such as *Windows Movie Maker*. Another benefit of using *Windows Movie Maker* is that it is also properly suited to provide a text watermark to a video [48]. A downside to using *Windows Movie Maker* is that it is only available for the Windows platform. In this thesis, *Windows Movie Maker 2012* will be used for the conversion and branding of the movie.

In summary, the following steps require a recent version of *Windows Movie Maker* and the video (.AVI) that has been rendered by *Huygens Essential*. Open *Windows Movie Maker* and select *Start→Add videos and photos* to add the video to the program, as shown in Figure 4.9.

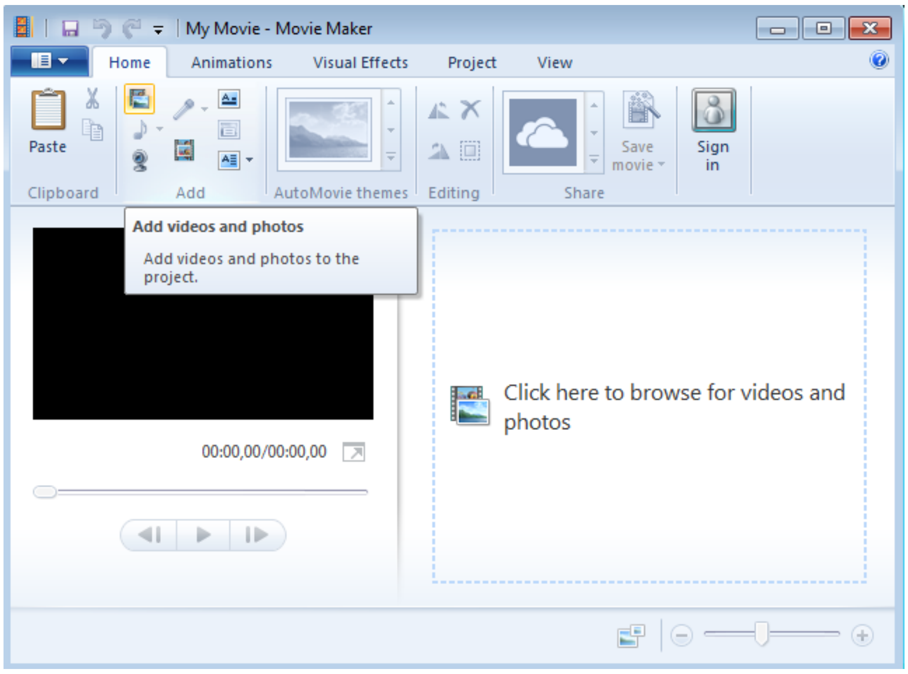


Figure 4.9: Adding a video to *Windows Movie Maker*.

Once the video is added to the program, select *Start*→*Add Caption* to provide the video with a text watermark. Select a desired font family, font size and font color. It is recommended to use a small font size so that a small portion of the screen is occupied. It is also recommended to set the font colour to *white* which improves readability due to its contrast with the video's colours. When this is done, click on the text field that is on the video, change the text to "Universiteit Leiden" and then drag the text field to the bottom right corner. The result of this process can be seen in Figure 4.10 below.

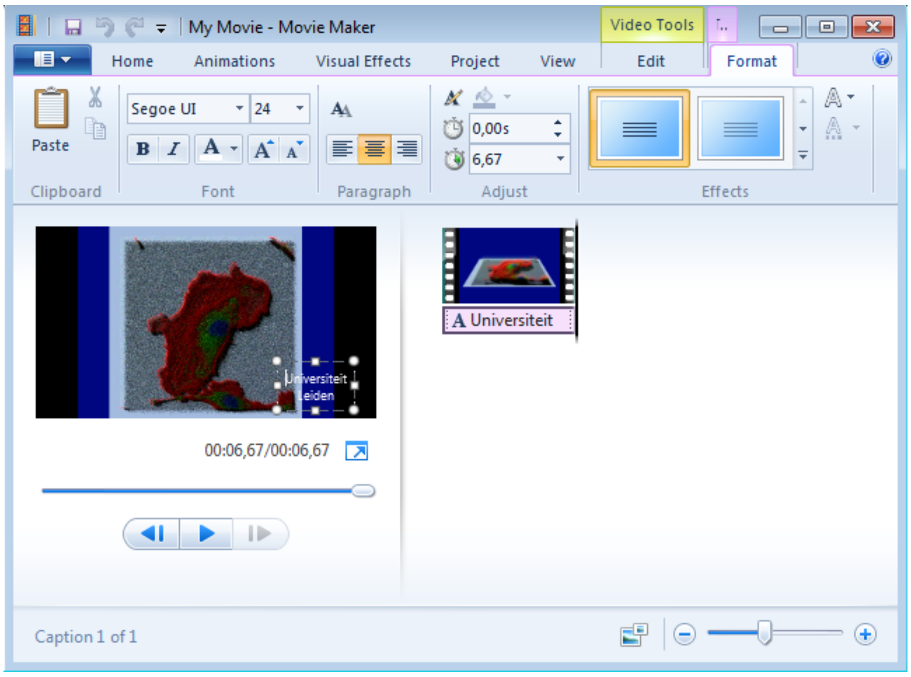


Figure 4.10: Adding a text watermark to the video in *Windows Movie Maker*.

After adding the text watermark to the video, make sure to select *Project→Standard (4:3)* (or *Widescreen 16:9*) to retain the initial aspect ratio, as shown in Figure 4.11 below.

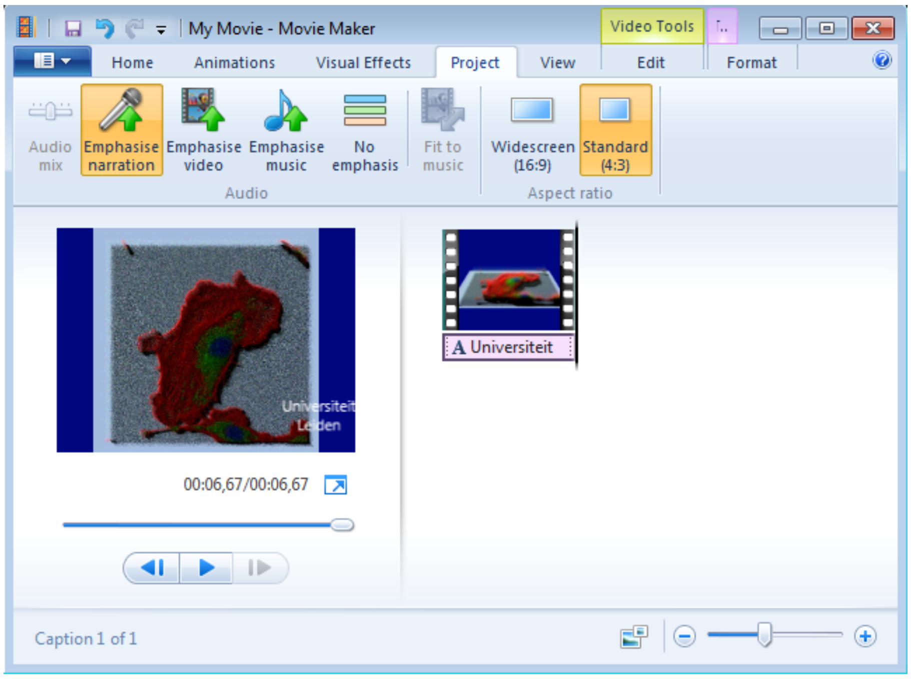


Figure 4.11: Changing the aspect ratio to 4:3 in *Windows Movie Maker*.

When all these steps are completed, it is time to render the video. Select *Movie maker→Save movie→For Computer* (as illustrated in Figure 4.12), and make sure to save the video as MPEG-4/H.264. The video is now provided with a text watermark and can be played natively in a web browser that supports HTML5.

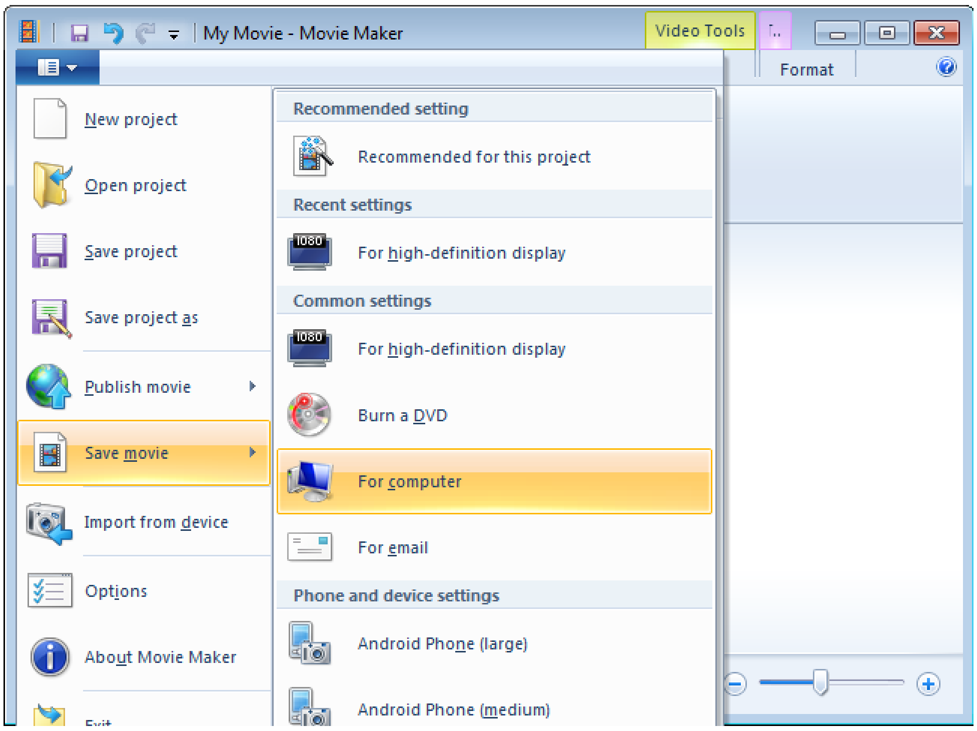


Figure 4.12: Exporting the file to the correct format in *Windows Movie Maker*.

To summarize the steps in this section: make sure the keyframes in Table 4.1 are added to the storyboard in *Huygens Essential* before stretching the video and exporting it in .AVI format to a local directory.

Keyframe	Twist	Zoom	Tilt
1	360 deg	1.00	70 deg
2	180 deg	1.00	70 deg
3	0 deg	1.00	70 deg
4	0 deg	1.00	0 deg

Table 4.1: The keyframes that have to be added to the storyboard in *Huygens Essential's* movie maker.

The file then needs to be converted to a suitable format for playability purposes. Use *Windows Movie Maker* to provide the video with a text watermark and make sure to save the file as MPEG-4/H.264. The video is now ready to be played natively in an HTML5-supported web browser.

4.2 Storing and retrieving biological videos and images

This section concerns itself with the storage and retrieval of the videos that have been constructed from biological data (gene products) in section 4.1. This section will also briefly go into HTML5-based visualization of 3D images. This means that the current database has to be extended with a multimedia section that also retains the annotative properties for clarity and retrieval purposes. The intention is to portray the videos in the GEMS browser, which is driven by the GEMS database. The data in the GEMS database are described using gene and spatial information, which are derived from GO and DAOZ respectively. Since the videos are based on gene products, they can be annotated using terms from GO [3].

Server-side

One approach to storing and retrieving the produced videos is by storing the videos separately in a local folder on the server to ensure that they are independent from the database and the server's retrieval mechanisms [43].

Additionally, a database table will have to be constructed that contains all the important attributes regarding the videos. Considering the nature of the videos and their methods of retrieval, it is important that the table contains a unique ID and all relevant annotative information. Since the annotation is based on GO, the database can be searched on the relevant gene symbol or the gene name. Aside from the annotation, metadata attributes should also be included for completeness and retrieval purposes [43]. Relevant metadata attributes for the constructed biological videos are **file_size**, **file_type**, **video_height**, **video_width** and **location** (path to the stored file on the server). Adding a video to the local folder on the server means that the database table also needs to be updated by providing a new table record. A concept of the database table can be found in Table 4.2.

video_ID	video_name	file_type	video_height	video_width	location	gene_name	gene_symbol
1	file.MP4	MP4	640	480	/srv/webapps-data/GEMSData/Videos/	fibroblast growth factor 8	fgf8

Table 4.2: The database table containing the attributes which are needed for the storage and retrieval of videos.

The database table contains all necessary information for retrieval. However, it should be normalized further to reduce the difficulties of updating, deleting and inserting records. Database normalization is important because it reduces database inconsistencies [34]. Normalization to 3NF can be achieved by identifying the functional dependencies between the attributes in Table 4.2. The following functional dependencies are identified:

1. {gene_name, gene_symbol}→{video_ID}
2. {video_ID}→{video_name, file_type, video_height, video_width, location}

Thus, the database table can be split into two tables to achieve a database design that is in 3NF [24]. The first table table concerns itself with the identification of the videos based on their annotative information. A concept of the gene annotation table can be found in Table 4.3 below. Not only will normalization provide better oversight, it also contributes to a simple and scalable database design. Should the video collection, for instance, be extended with videos that require different forms of annotation, then only the table containing the annotative information will have to be updated, leaving the rest of the database intact.

video_ID	gene_name	gene_symbol
1	fibroblast growth factor 8	fgf8

Table 4.3: The gene database containing the relevant annotative information for querying purposes.

In addition to the gene annotation table, there also needs to be a table that contains the metadata. A concept of this metadata table can be found below in Table 4.4. Together with Table 4.3, these two tables form the 3NF equivalent of the database illustrated in Table 4.2.

video_ID	video_name	file_type	video_height	video_width	location
1	file.MP4	MP4	640	480	/srv/webapps-data/GEMSData/Videos/

Table 4.4: The database table containing the metadata that are needed to portray the videos natively in an HTML5-supporter web browser.

When a user clicks on a hyperlink to view a video, a request is made and sent from the client-side to the server, which prompts the server to consult the table containing the annotative information (i.e. Table 4.3) to retrieve the correct video_ID. The server will then use the video_ID to query for the metadata in the metadata table (i.e. Table 4.4). The metadata are then subsequently used for the visualization on the client-side. It is worth noting that it is crucial that the entries in both tables correspond with each other, i.e. the video_ID's from both tables match, otherwise the incorrect videos are retrieved by the server.

Client-side

On the client-side, the videos are to be visualized in the GEMS browser on the bio-imaging website of the IB research group at LIACS [18]. In the GEMS browser, one can search by gene symbol or per 'cluster', i.e. searching by stage of development, functional system and/or location. This will prompt the server to load thumbnails into the GEMS browser, which a user can click on to load a page containing more detailed information [3].

The queries that are involved generating this more detailed page, should then be extended with queries that retrieve all the relevant videos and metadata (**video_name**, **video_height**, **video_width** and **file_type**) that are related to the discussed gene. The video and metadata can then be loaded in that web page using HTML5's native <video> tag. The general format of the <video> tag is as follows [45]:

```
 <video width="video_width" height="video_height" controls>
  <source src="video_name" type="video/file_type">
</video>
```

It is possible that a user's web browser does not support a certain file type (such as .MP4). It is therefore recommended to store the Huygens video on the server in multiple formats (e.g. .MP4, .Ogg and .WebM) and retrieve all of these using the <source> tag to further promote platform independence [45]. The "on the fly" generation of these web pages can be done using JSP, which has already been discussed in chapter 3.

Currently, most of the videos on the bio-imaging web pages are available as separate download links [19, 21, 22]. It can be argued that the transition to native visualization in the web browser requires less effort on the user's end and should therefore be promoted.

HTML5 is also capable of visualizing 3D images of gene expression patterns in a web browser, because these images are already stored in the GEMS database. This means that the database tables containing the images are already constructed and that these images are already correctly annotated.

The images are stored in .JPG and .TIF format. HTML5 inherently supports the display of .JPG files with the <img> tag. The general basic format of the <img> tag is as follows [44]:

```
<img src="file_name.jpg">
```

However, .TIF files have very limited web browser support [47] and have the tendency to be very large in file size. Therefore it is highly recommended to convert all images to a supported format, such as JPG, before visualizing these images in the GEMS browser. The conversion can be done by any image editor program that can open .TIF files and store them as .JPG. One tool that is capable of performing this task is *Pinta* [28].

Chapter 5

Results and Discussion

This chapter concerns itself with providing and analyzing the answers to the research questions. This chapter is divided in four sections. Section 5.1 and section 5.2 will answer the research sub-questions and section 5.3 will answer the main research question. Lastly, section 5.4 will discuss the main results and limitations of this research.

5.1 Answering research sub-question 1

Research sub-question 1: *What are the differences between server-based visualization of resources with Java3D and server-based visualization of resources with HTML5?*

Since HTML5 is an umbrella term for several client-side technologies [1], the server-side is almost identical for both the Java-based platform and the HTML5-based platform. The main difference for the server-side is that the *Bio-Visualization web service* needs to prepare the data differently for the HTML5-based platform by parsing the Freeman chain code from the retrieved .PIX file directly into the generated .XML file. This is important because `Three.js` cannot read .PIX files to interpret the Freeman chain code independently, a task that Java3D is able to carry out.

As for the client-side, the main difference is the transition from running a Java3D-based application to running an HTML5-based (mainly HTML, CSS and JavaScript) web page in the web browser to visualize the biological model data. Another difference is that the HTML5-based web page can be run natively in any browser that offers HTML5-support, which means that the HTML5-based web page requires no further installation of software. In contrast, Java3D-based applications do require an installation of Java before any content can be accessed which may result in compatibility issues.

In addition, quality attributes have been used to evaluate the quality of both server architectures. The comparison shows that the architectures do not differ wildly in quality since most of the quality attributes receive the same assessment. The *performance* of the HTML5-based architecture is slightly lower than the Java3D-based architecture because it incorporates an extra preprocessing stage at the server-side.

5.2 Answering research sub-question 2

Research sub-question 2: *What needs to be done to ensure the availability of video and image resources in HTML5?*

The videos can be made from gene products using *Huygens Essential*, which produces .AVI formatted files. However, the HTML5 video player that is capable of running natively in the web browser does not support this file extension. This means that the .AVI file has to be converted to a supported format (e.g. .MP4) using a video editing tool such as *Windows Movie Maker*. Not only will it be able to convert the file to a proper format, it can also provide the video with a watermark for the protection of intellectual property.

Using HTML5's <video> tag requires additional knowledge of the video's dimensions (height and width in pixels) along with its file name and file extension. Consequently, this information has to be stored in a database table that contains all the metadata. Additionally, the database must also contain a table that keeps track of all the videos, their location on the server and their annotation (gene symbol and gene name). A request for a video is essentially an annotation-based query sent to the server, which in turn returns the relevant metadata that can be embedded into the web page using the <video> tag.

As for visualizing biological *images*, supported file types such as .JPG can be natively embedded using the <img> tag. The only disadvantage is that the GEMS database consists of images that are not only stored as .JPG, but also as .TIF, which is an unsupported file type for many web browsers. This means that the .TIF files have to be converted using an appropriate image editing tool, such as *Pinta*.

5.3 Answering the main research question

Research question: *In what way is it possible to keep biological image data available to third parties with the help of up-to-date ICT-tools?*

In order to keep biological data available to third parties, a dedicated server that hosts the interaction between the databases and the HTML5-based web pages is required. The key is to build a dynamic server system with clearly and transparently constructed databases. This allows for easy annotation-based storage and retrieval. Biological data have the tendency to require substantial preprocessing before the data can be visualized.

The biological *models* require a parser (i.e. Xia's parser) that embeds the Freeman chain code from the acquired .PIX file into the .XML file that is delivered by the *Bio-Visualization web service*. The model can then be visualized in the 3D Atlas of Zebrafish Development using a web page that makes use of `Three.js` and HTML5 (i.e. Xia's template).

Furthermore, biological *images* and *videos* can be visualized natively in the GEMS browser by using the <img> and <video> tags respectively. However, these tags require supported file types. To this end, image and video editing software (e.g. *Pinta* and *Windows Movie Maker*) are required to convert the images and videos to supported file types (e.g. .JPG and .MP4). The videos and images are retrieved based on annotation. The videos additionally require a metadata table in the GEMS database that contains the necessary attributes to support the <video> tag.

5.4 Discussion and limitations

All in all, it is feasible to keep biological data such as images, 3D models and videos available using HTML5 and up-to-date ICT-tools. However, it does call for an efficient storage system and a scalable database design to store the biological data, because the data have the tendency to be quite substantial in size. This method also requires some preprocessing of the data in order for it to be suitable for HTML5-based visualization, which may result in data loss and, by extension, loss of perceived quality [10].

The methodology that has been followed in this thesis is sufficient to provide an answer to the main research question. Each type of biological data (*images*, *videos* and *models*) has its own methods of preprocessing, storage, retrieval and visualization. Since the visualization of *images* has been natively supported by web browsers for longer than the release of HTML5, it does not necessarily need its own research sub-question. The biological *models* and *videos*, however, both required more research to determine what has to be done at the preprocessing, storage and retrieval stages for native visualization in the web browser. Since this is different for both types of data, it has been a good practice to separate the main research question into two different sub-questions.

Regarding the limitations of this research, it is focused on *how* native visualization of biological data can be accomplished. This means that the research scope is limited to understanding the capabilities of HTML5, applying this understanding to biological data and evaluating the feasibility. This also means that this research is not focused on *how well* the visualization can be done in HTML5 in terms of image, model and video quality. It is also not clear which visualization techniques are preferred by end users.

Chapter 6

Conclusions and future work

Developing up-to-date methods to perform server-based visualization is important to ensure the availability of biological data to third parties. This thesis shows how biological images, models and videos can be visualized natively in the web browser using up-to-date techniques and the HTML5-platform.

Firstly, biological models can be made available in the 3D Atlas of Zebrafish Development using `Three.js`. There exists a template (i.e. Xia's template) that integrates section images and .XML data to visualize the models natively in the web browser. However this can only be accomplished when the *Bio-Visualization web service* first calls a parser (i.e. Xia's parser) to integrate the contour information from the retrieved .PIX file into the retrieved .XML file. This "enhanced" .XML file can then be used by the template to visualize the biological models. This technique requires no installation of additional software (as opposed to the Java3D-based technique), which leads to an improvement of platform independence. However, this technique does require more computation on the server-side to prepare the models for native visualization.

Secondly, biological videos can be constructed from gene products using *Huygens Essential*. The software produces an .AVI file, which must then be watermarked and converted to a file format that is natively supported by HTML5's <video> tag, such as the .MP4 format. The conversion and watermarking can be accomplished using video editing software, such as *Windows Movie Maker*. The videos have to then be stored on the server's file system. In the database, tables need to be constructed that contain the videos' metadata and annotative information for storage and retrieval purposes. The biological videos can then be displayed in the GEMS browser.

Lastly, biological images can also be made available in the GEMS browser using HTML5's native <img> tag, provided the images are converted to a supported file format, such as .JPG. This can be done by using image editing software, such as *Pinta*.

Future work

The next step is to further improve the server-based visualization by developing techniques to generate the pages "on the fly" using `JSP`. Once this is finished, new methods need to be found to improve web browser loading times. A recommendation is to find out whether it is possible to cache and/or preload information to ensure a better interaction with the HTML5-based application.

Another recommendation is to evaluate the perceived quality of the *images*, *models* and *videos* that the HTML5-based system delivers. This can be accomplished by comparing it to the perceived quality of the biological data that the Java3D-based system delivers. It should also become clear which visualization method (Java3D or HTML5) is preferred by the end user. In doing so, improved methods can be developed to ensure the highest perceived quality of *images*, *models* and *videos* as possible.

Appendix A

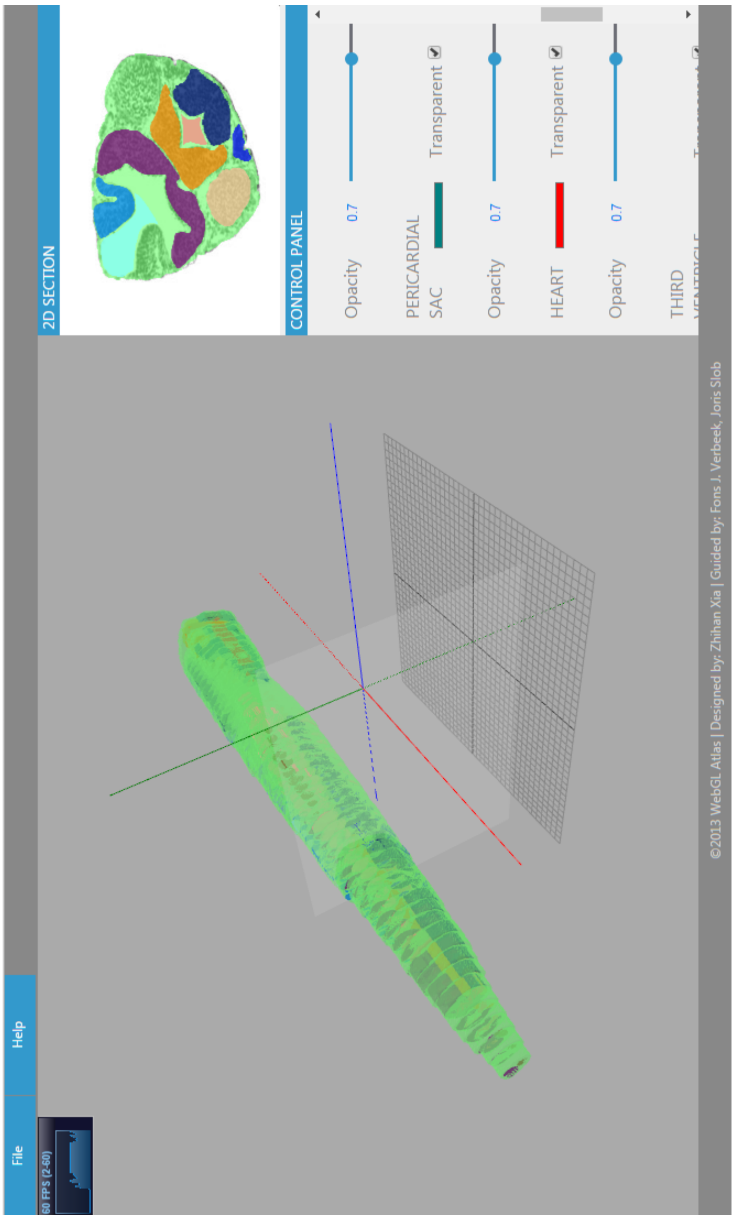
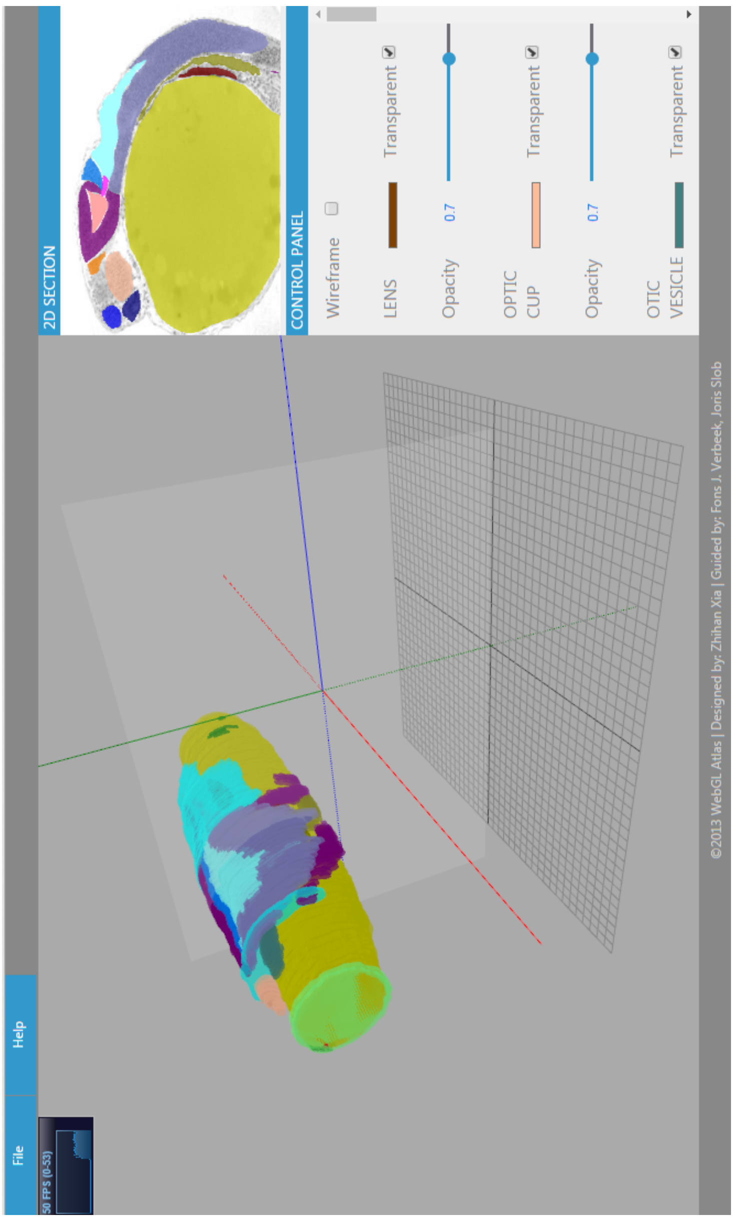


Figure A.1: A model of the *zebrafish* (zf0048) visualized using the HTML5-based technique.

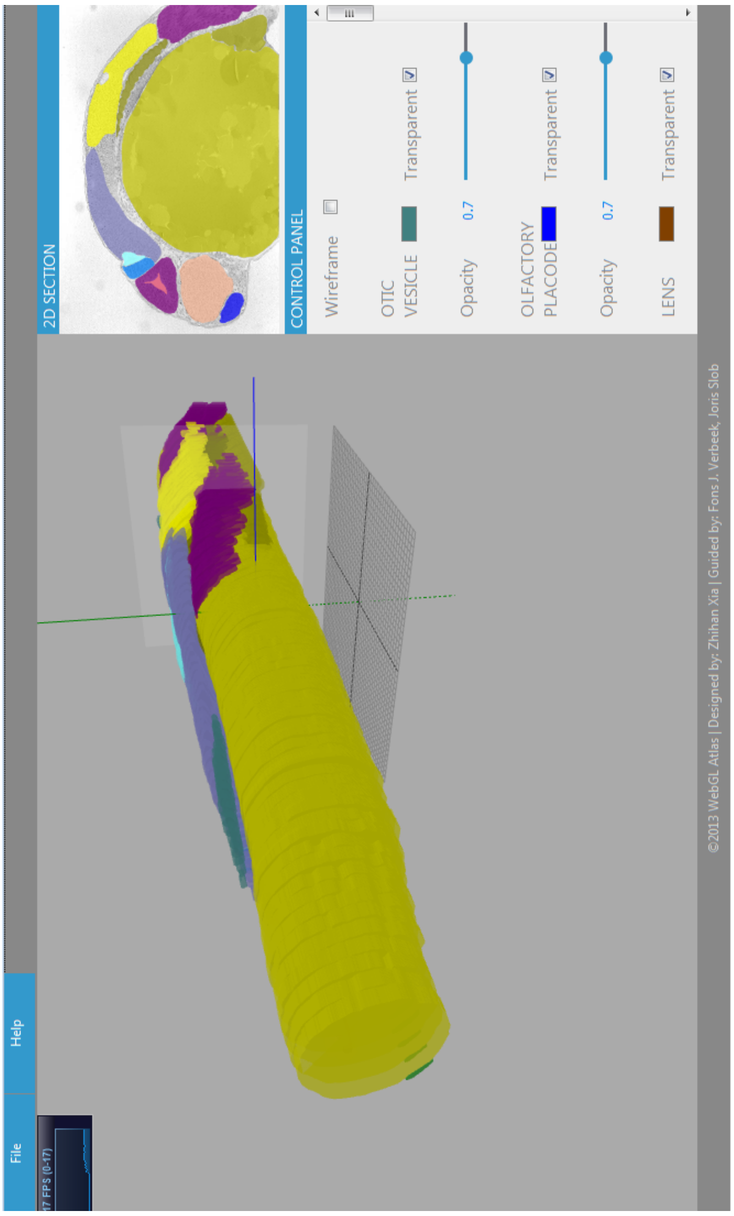
Appendix B



Wireframe
LENS Transparent
Opacity 0.7
OPTIC CUP Transparent
Opacity 0.7
OTIC VESICLE Transparent

Figure B.1: A model of the *zebrafish* (zf0324) visualized using the HTML5-based technique.

Appendix C


2D SECTION
CONTROL PANEL
File
Help
17 FPS (0-17)
©2013 WebGL Atlas | Designed by: Zhihan Xia | Guided by: Fons J. Verbeek, Joris Slob

Wireframe ☐

OTIC VESICLE 🟩 Transparent ☑

Opacity 0.7

OLFACTORY PLACODE 🟦 Transparent ☑

Opacity 0.7

LENS 🟫 Transparent ☑

Figure C.1: A model of the *zebrafish* (zf1124) visualized using the HTML5-based technique.

Appendix D

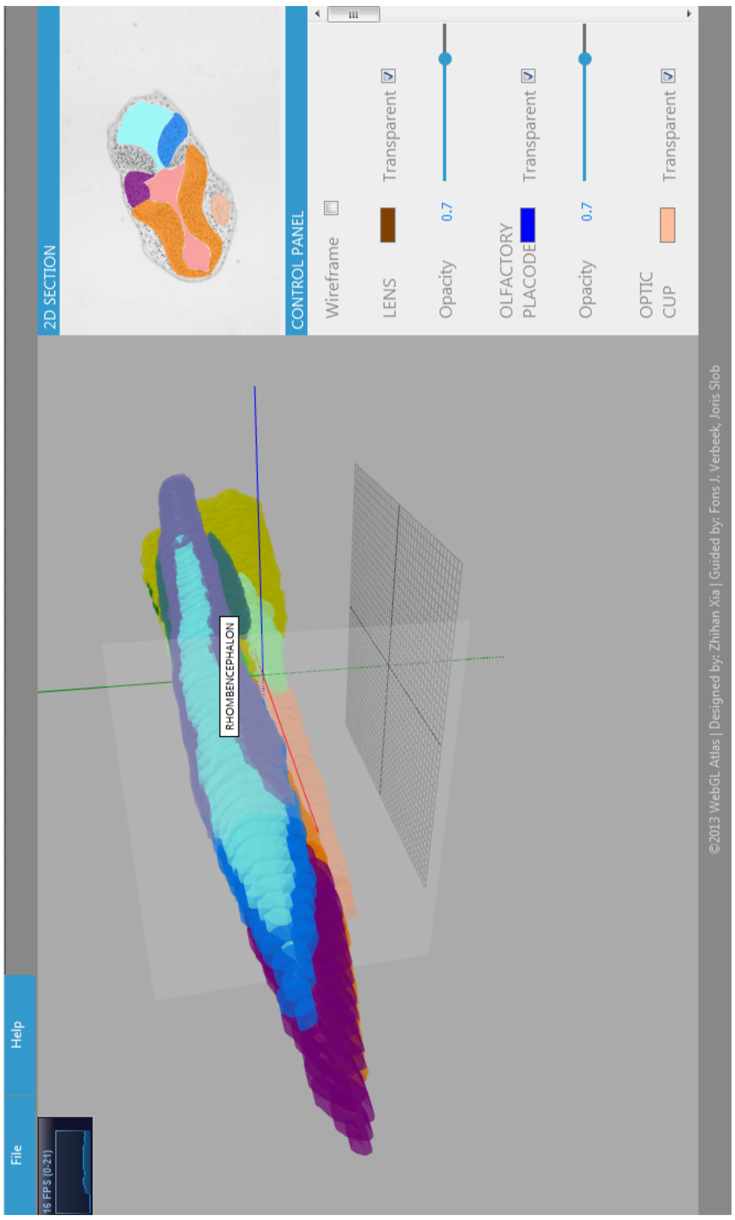


Figure D.1: A model of the *zebrafish* (zf1524) visualized using the HTML5-based technique.

Appendix E

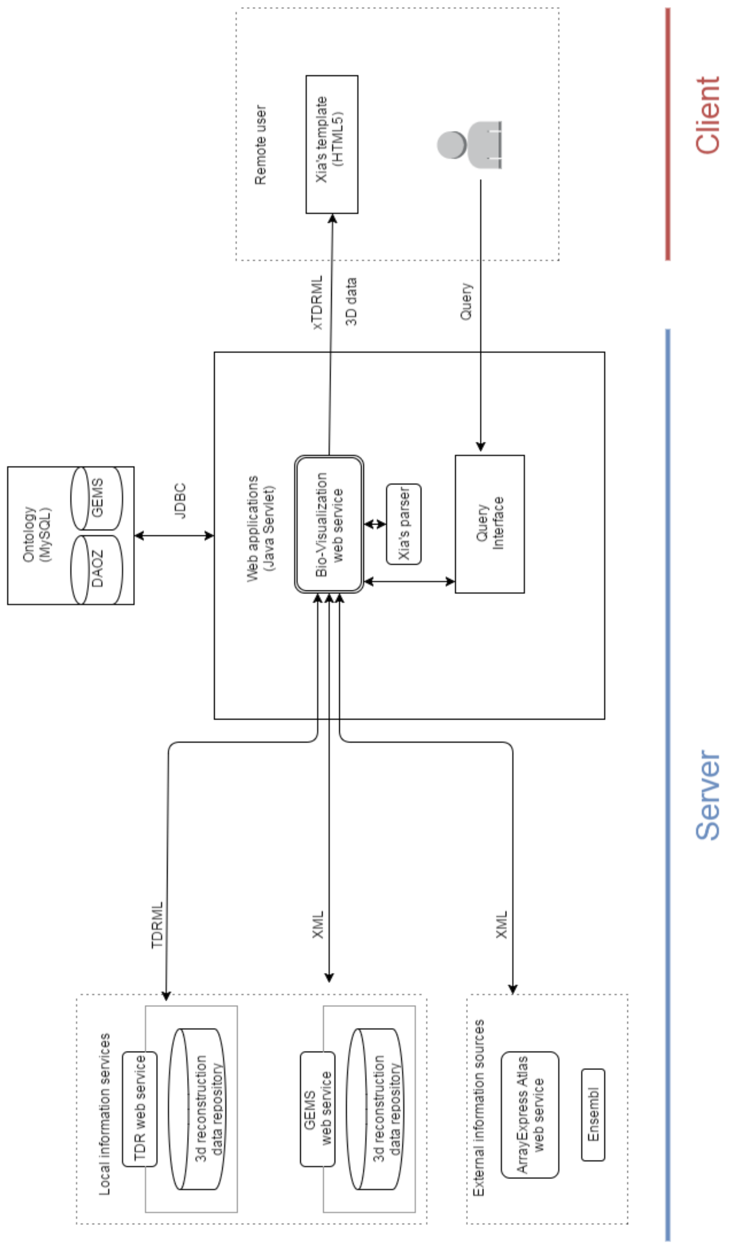

Remote user
Xia's template (HTML5)
Client
xTDRML
3D data
Query
Ontology (MySQL)
GEMS
DAOZ
JDBC
Web applications (Java Servlet)
Bio-Visualization web service
Xia's parser
Query Interface
Server
TDRML
XML
XML
Local information services
TDR web service
3d reconstruction data repository
GEMS web service
3d reconstruction data repository
External information sources
ArrayExpress Atlas web service
Ensembl

Figure E.1: The new, HTML5-based server architecture.

Bibliography

[1] G. Anthes. Html5 leads a web revolution. *Commun. ACM*, 55(7):16–17, July 2012.

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[3] M. Belmamoune et al. *Spatio-temporal framework for integrative analysis of zebrafish developmental studies*. PhD thesis, Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University, 2009.

[4] M. Belmamoune and F. J. Verbeek. Developmental anatomy ontology of zebrafish: an integrative semantic framework. *Journal of Integrative Bioinformatics*, 4(3):65, 2007.

[5] P. Boon, N. Eminovic, A. De Vos, B. Buitendijk, E. Van Raaij, M. Den Broeder, W. Hage, and F. Verbeck. A virtual lab-notebook for multidimensional microscope images. In *Information Visualization, 2000. Proceedings. IEEE International Conference on*, pages 187–191. IEEE, 2000.

[6] J. Deacon. Model-view-controller (mvc) architecture. *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*, 2009.

[7] European Bioinformatics Institute (EMBL-EBI). Expression atlas: Differential and baseline expression. `https://www.ebi.ac.uk/gxa/home`, 2016. [Online; accessed 04-July-2016].

[8] P. Feldman. Text overlay for threejs with angular and typescript. `https://phifel.wordpress.com/2015/02/20/text-overlay-for-threejs-with-angular-and-typescript/`, 2015. [Online; accessed 15-August-2016].

[9] Gene Ontology Consortium. What is a ’gene product’? `http://geneontology.org/faq/what-gene-product`, 2015. [Online; accessed 15-August-2016].

[10] L. Goldmann, F. De Simone, F. Dufaux, T. Ebrahimi, R. Tanner, and M. Lattuada. Impact of video transcoding artifacts on the subjective quality. In *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*, pages 52–57. IEEE, 2010.

[11] G. D. Guttmann. Spilling the beans on java 3d: a tool for the virtual anatomist. *The Anatomical Record*, 257(2):73–79, 1999.

[12] HTML5test. Desktop browsers. `https://html5test.com/results/desktop.html`, 2016. [Online; accessed 27-July-2016].

[13] HTML5test. Mobiles. `https://html5test.com/results/mobile.html`, 2016. [Online; accessed 27-July-2016].

[14] HTML5test. Other. `https://html5test.com/results/other.html`, 2016. [Online; accessed 27-July-2016].

[15] HTML5test. Tablets. `https://html5test.com/results/tablet.html`, 2016. [Online; accessed 27-July-2016].

[16] Imaging & Bioinformatics Research Group of LIACS. The analysis of patterns of gene expression. `http://bio-imaging.liacs.nl/liacsgems.html`, 2005. [Online; accessed 04-July-2016].

[17] Imaging & Bioinformatics Research Group of LIACS. Tdr-3dbase: software for 3d reconstruction. `http://bio-imaging.liacs.nl/tdr3dbase.html`, 2005. [Online; accessed 04-July-2016].

[18] Imaging Research Group of LIACS. The gene expression management system (gems) web browser. `http://bio-imaging.liacs.nl/gems/`. [Online; accessed 04-August-2016].

[19] Imaging Research Group of LIACS. Animations of the zebrafish embryo. `http://bio-imaging.liacs.nl/animationstart.html`, 2005. [Online; accessed 12-August-2016].

[20] Imaging Research Group of LIACS. The atlas of zebrafish development. `http://bio-imaging.liacs.nl/liacsatlas.html`, 2005. [Online; accessed 04-July-2016].

[21] Imaging Research Group of LIACS. Gallery in collaboration with dept. medical pharmacology, leiden university. `http://bio-imaging.liacs.nl/galleries/mmhippocampus/HippocampusGallery.html`, 2009. [Online; accessed 12-August-2016].

[22] Imaging Research Group of LIACS. Gallery pages for the european pond turtle (emys orbicularis). `http://bio-imaging.liacs.nl/galleries/emysorbicularis/TurtleModel08.html`, 2010. [Online; accessed 12-August-2016].

[23] INCORE Digital Agency. Html5 popularity among fortune 500 companies. `http://www.incore.com/Fortune500HTML5/#infographic`, 2013. [Online; accessed 27-July-2016].

[24] S. Kandasamy. Third normal form (3nf) with example. `https://exploredatabase.blogspot.nl/2014/02/third-normal-form-3nf-with-example.html`, 2014. [Online; accessed 12-August-2016].

[25] D. Lyons. Intro to webgl and three.js. Front Porch Conference, 2014.

[26] L. O'Brien, P. Merson, and L. Bass. Quality attributes for service-oriented architectures. In *Proceedings of the international Workshop on Systems Development in SOA Environments*, page 3. IEEE Computer Society, 2007.

[27] Oracle. Getting started with javafx 3d graphics. `http://docs.oracle.com/javase/8/javafx/graphics-tutorial/javafx-3d-graphics.htm#JFXGR256`, 2016. [Online; accessed 27-July-2016].

[28] Pinta. Pinta: Painting made simple! - pintaproject/pinta, 2016. [Online; accessed 09-August-2016].

[29] D. Potikanond and F. Verbeek. Visualization and analysis of 3d gene expression patterns in zebrafish using web services. In *IS&T/SPIE Electronic Imaging*, pages 829412–829412. International Society for Optics and Photonics, 2012.

[30] D. Potikanond, F. Verbeek, et al. 3d visual integration of spatio-temporal gene expression patterns on digital atlas of zebrafish embryo using web service. In *Int. Conf. on Advances in Communication and Information Technology*, 2011.

[31] D. Potikanond, F. Verbeek, et al. Visual integration of spatio-temporal data from a 3d digital atlas and patterns of gene expression in zebrafish, 2012.

[32] N. Schuurman and A. Leszczynski. Ontologies for bioinformatics. *Bioinformatics and biology insights*, 2:187, 2008.

[33] Scientific Volume Imaging B.V., Laapersveld 63, 1213 VB Hilversum, The Netherlands. *Huygens Essential User Guide for version 15.10*, 2015.

[34] Studytonight. Normalization of database. `http://www.studytonight.com/dbms/database-normalization.php`, 2016. [Online; accessed 12-August-2016].

[35] three.js / docs. Objloader. `http://threejs.org/docs/index.html?q=obj#Reference/Loaders/OBJLoader`, 2016. [Online; accessed 27-July-2016].

[36] tutorialspoint. Jsp architecture. `http://www.tutorialspoint.com/jsp/jsp_architecture.htm`. [Online; accessed 04-September-2016].

[37] tutorialspoint. Jsp overview. http://www.tutorialspoint.com/jsp/jsp_overview.htm. [Online; accessed 04-September-2016].

[38] Universiteit Leiden. Computer systems, imagery and media (csi). https://www.universiteitleiden.nl/en/science/computer-science/computer-systems-imagery--media. [Online; accessed 13-July-2016].

[39] F. Verbeek, M. De Groot, D. Huijsmans, W. Lamers, and I. Young. 3d base: a geometrical data base system for the analysis and visualisation of 3d-shapes obtained from parallel serial sections including three different geometrical representations. *Computerized medical imaging and graphics*, 17(3):151–163, 1993.

[40] F. J. Verbeek and P. J. Boon. High-resolution 3d reconstruction from serial sections: microscope instrumentation, software design, and its implementations. In *International Symposium on Biomedical Optics*, pages 65–76. International Society for Optics and Photonics, 2002.

[41] F. J. Verbeek and D. Huijsmans. A graphical database for 3d reconstruction supporting (4) different geometrical representations. In *Medical Image Databases*, pages 117–144. Springer, 1998.

[42] F. J. Verbeek, D. Huijsmans, R. Baeten, N. Schoutsen, and W. Lamers. Design and implementation of a database and program for 3d reconstruction from serial sections: A data-driven approach. *Microscopy research and technique*, 30(6):496–512, 1995.

[43] A. P. Vries. *Content and multimedia database management systems*. University of Twente, Centre for Telematics and Information Technology (CTIT), 1999.

[44] W3Schools. Html images. http://www.w3schools.com/html/html_images.asp, 2016. [Online; accessed 09-August-2016].

[45] W3Schools. Html <video> tag. http://www.w3schools.com/tags/tag_video.asp, 2016. [Online; accessed 02-August-2016].

[46] M. C. M. Welten et al. *Spatio-temporal gene expression analysis from 3D in situ hybridization images*. PhD thesis, Leiden Institute of Advanced Computer Science, group Imaging and Bio-informatics, Faculty of Science, Leiden University, 2007.

[47] Wikipedia. Comparison of web browsers: Image format support—Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Comparison_of_web_browsers#Image_format_support, 2016. [Online; accessed 09-August-2016].

[48] J. C. Wong. How to watermark your videos in windows movie maker. http://www.makeuseof.com/tag/watermark-videos-windows-movie-maker/, July 2010. [Online; accessed 08-August-2016].

[49] Z. Xia. A webgl based 3d atlas viewer, July 2014. Unpublished report—LIACS, Universiteit Leiden.