Animation and Modeling of Cardiac Performance for Patient Monitoring

1 Introduction

The goal of Cardiac Rehabilitation is to reduce the risk of another cardiac event and to control the current heart condition of a patient and stop it from deteriorating. Most current Cardiac Rehabilitation procedures include counseling, to help the patient understand their disease more clearly and manage the disease process. As well as helping the patient to understand (and hopefully modify) risk factors e.g. high cholesterol, diabetes, smoking, lack of physical activity and high blood pressure. Development of exercise programs and lending emotional support are also standard of most Cardiac Rehabilitation Procedures. Many patients treated for heart problems do not follow correct rehabilitation procedures because they have a poor understanding of their disease and how it is affected by their behaviour and lifestyle. The aim of this project is to introduce a further tool for use in Cardiac Rehabilitation programs. By providing an animated model of a heart which reflects the cardiac disease of the patient it is hoped that patients will gain a better understanding of their disease and its consequences and allow them to gain more insight into why correct rehabilitation procedures are imperative. In addition the animated heart model will enable the patient to visually monitor the rehabilitation process which might lead to several psychological benefits (increased motivation to follow the rehabilitation procedures, a more positive attitude towards life after a heart attack, etc.). The impetus of the project is to design a model of the human heart that visualizes changes in cardiac performance and rehabilitation, rather than to design a physiologically and anatomically correct model.

2 Previous Work

In the past there has been a plethora of research in the area of biomedical visualization. Research has mainly focused on providing physiologically correct models of various areas that make up the heart, for example the heart's arteries or ventricles. Some early research on providing 3D visualizations of carotid arteries was conducted by Lorensen [Lorensen 1995]. Robb [Robb 1989] initially dealt with patient specific 3D organ visualizations. Jolesz and Kikinis [Jolesz and Kikinis 1995] have also provided some early publications on 3D biomedical visualizations.

Modelling of the heart's left ventricle has been quite successful in the past with a common approach focusing on modelling with iso-surfaces using proloate spheroidal coordinates. The use of B-splines have also been common for modelling the left ventricle.

Not much research on using 3D heart models as part of cardiac rehabilitation programs has been done in the past.

3 Background

Previous research, especially in the area of heart model construction, has used tools such as OpenGL or DirectX to build and display the visualization. Our heart model was constructed using a scripting language called CMGUI which was developed by the Auckland University Bioengineering Department. CMGUI is an extension of CMISS (also developed by the Bioengineering Department) which is a "mathematical modeling environment that allows the application of finite element analysis, boundary element and collocation techniques to a variety of complex bioengineering problems".[1] CMISS consists of a graphical front end (CMGUI) with advanced 3D display and modeling capabilities and a computational backend that may be run remotely on powerful workstations or supercomputers.

Development with CMGUI runs under the Linux operating system. All the files needed to install CMGUI can be obtained from the above website. CMGUI reads .com files which contain all the scripting commands for the program. Model data is stored in .exnode (Nodes) and .exelem (Elements) files. A typical CMGUI program will read in some model data and then add/modify certain elements in the scene. In CMGUI, every element group has a graphical rendition - a list of attributes such as lines, surfaces etc. that describes how it is to look. Combined, this 'graphical element group' monitors changes in its nodes and elements and automatically updates the graphics to reflect the changes. By default, every group is drawn using lines, i.e. as a wireframe mesh.

To run a CMGUI program at the command prompt you type:

cmgui -example <*filename*>

Where <*filename*> corresponds to the name of the .com script file you have written. Typically this should be located in a folder called cmiss_input which is located under a folder which has the same name as the filename (without the .com extension).
In CMGUI all graphics commands begin with 'gfx' followed by the appropriate tokens to achieve what is desired. As an example the following code would read in the nodes of a cube from a file named cube.exnode and display the CMGUI graphics window.

```
gfx read nodes $example/cube.exnode;
gfx create window 1
```

The above window is named '1'. The cube would be displayed as a wireframe mesh by default. These commands could be entered directly into the CMGUI interface or they could be read from a '.com' file for convenience.

The above has described CMGUI as a development tool under the Linux operating system.

[1] More information about CMISS and CMGUI can be found at the Bioengineering Department's website http://www.bioeng.auckland.ac.nz/home/home.php

CMGUI now has the ability to be embedded directly into web based applications through the use of a ZINC plug-in application developed by Carey Stevens of the Bioengineering Department. ZINC is an extension for the Mozilla based browsers that incorporates the CMGUI 3D visualization environment. This allows applications that have been developed with CMGUI under Linux to be used under windows operating systems as well.
ZINC provides the ability to allow interaction to take place between the users and the system. With a web based user interface patients can input certain required data and then the output of the system can be displayed based on the attributes that were entered.[2]

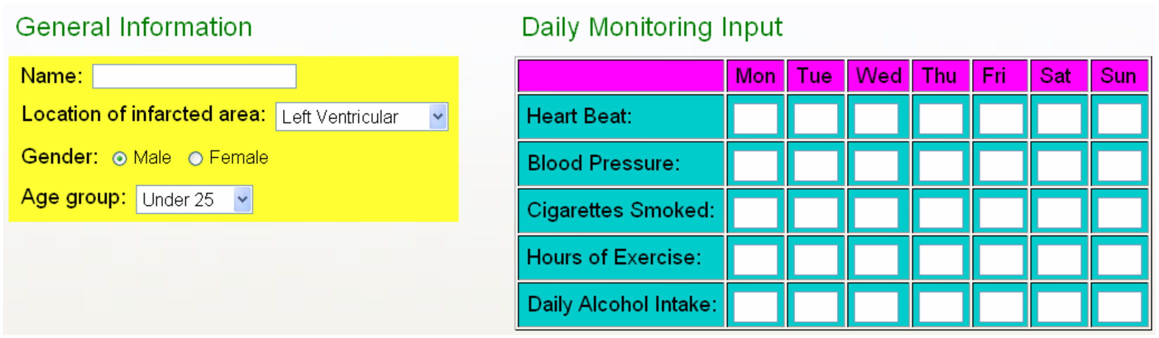
CMGUI applications can be embedded into web based interfaces through the ZINC plug-in by including the appropriate tags in the script that describes the webpage e.g. html or xml. The format that has been used for the current project uses the XML User Interface Language (XUL) to build the interface. XUL is a markup language for describing user interfaces. With XUL you can create rich, sophisticated cross-platform web applications easily. XUL is part of the Mozilla browser and related applications; it is designed to be portable across operating systems.[3] To create truly dynamic interfaces XUL can be combined with Javascript to handle action and events (as has been done for this project).

4 Cardiac Rehabilitation and CMGUI

The aim of this project was to provide an animated model of a heart which displayed specific information about the health of a cardiac rehabilitation patient. This was achieved by obtaining certain monitoring data from the patient, which was then processed and the heart model was displayed depending on this data. The state of the patient's heart is interpreted through the size and location of infarcted regions (areas where damage has occurred to the heart's surface). The patient's approximate heart beat and damage to the heart's coronary system also provide information about the patient's current health. For the system, as has been developed so far, a web based user interface is provided to cardiac rehabilitation patients. General and daily information is obtained from the patient through the interface. The general information that the patient enters includes the patient's name, age and gender as well as the location where a possible infarction (damage) has occurred to the patient's heart. The current options are 'Left Ventricular Infarction' (damage has occurred to the left ventricle), 'Right Ventricular Infarction' (damage to the right ventricle) or 'Mid Section Infarction' (damage to the septum or in between the left and right ventricles). The general information section is intended to obtain static information about the patient which will be used to affect the outcome of the heart visualization. More importantly, however, daily monitoring information is obtained from the patient, which will have a greater, dynamic effect on the corresponding 3D visualization. The daily data that is collected from the patient includes the patient's average heart beat and blood pressure. If the patient is a smoker then the amount of cigarettes smoked is recorded. The amount of exercise a patient receives and the patient's alcohol intake is also recorded. Once the data are input into the system they all contribute

[2] For more information about ZINC or to download the plug-in visit www.physiome.net/software/ZINC
[3] Extra information and tutorials on how to construct XUL based applications can be found at www.xulplanet.com

to the final visualization the patient sees describing the state of their heart. Figure 1 shows a screen shot of the current prototype prompting the user for the general and daily monitoring information.



General Information

Name:	
Location of infarcted area:	Left Ventricular
Gender:	[✓ Male] Female
Age group:	Under 25

Daily Monitoring Input

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Heart Beat:							
Blood Pressure:							
Cigarettes Smoked:							
Hours of Exercise:							
Daily Alcohol Intake:							

Figure 1: On the left of the screen the patient's general information is obtained. On the right the patient enters the data obtained from daily monitoring.

For the current system, once the data has been entered into the interface it is processed through a scoring system (at the moment this is a temporary prototype system) which calculates for each day the improvement or damage the patient is doing to their health through their lifestyle choices. This is based on the data entered for cigarettes smoked, alcohol intake, exercise etc. This information is then portrayed to the patient through the heart model by affecting the state of an infarcted region on the heart surface. A weekly comparison of models is then made available to the patient through the interface which displays this information for each day that the patient has entered data so that they may get an indication of how their cardiac rehabilitation is going. If the patient has participated in harmful lifestyle choices this may be depicted in the model by enlarging the infarcted region, increasing the patient's heart beat or increasing the damage to the coronary system whereas if the patient has been living a healthy lifestyle then this would be shown by decreasing the infarcted region/heart beat etc.

It is important to note, however, that the visualization provided to the patient is not meant to describe an anatomically correct heart, but rather it provides various visual cues (through infarcted regions, heart beat rate and coronary system damage) which the patient can interpret as harmful or beneficial thereby providing them with information about the status of their rehabilitation process. It is hoped that by providing the patient with a visualization that displays the damage they may be causing their heart this would increase their motivation to get healthy and stay healthy. Together with standard cardiac rehabilitation procedures such as counseling, exercise and lending emotional support this tool would hopefully be a worthy addition to the cardiac rehabilitation process.

5 Implementation

The overall structure of the system at present begins with the main index page for the web application. This is simply a script describing a standard web page, which is written using the XML User Interface Language (XUL). The index page is made up of two panels. At first the only panel visible contains various input elements, including textboxes and drop down menus to accept the input from the user as well as buttons used to reset the information entered or submit the information for processing. Included at the beginning of the script is a reference to a javascript file which will handle the processing of the information received from the page.

```
<script type="application/x-javascript" src="rehab1.js"/>
```

The above line just specifies the type of script to include i.e. a javascript application and specifies the name of the file, 'rehab1.js'.

Once all the information is entered into the web page and the user clicks 'submit' the javascript application will validate and process the data. Control is then handed back to the index page which now shows the next panel which displays the heart model and a scroll bar to scroll through the various days of the week so that the patient can compare the state of their heart on different days. The tag in the index file that embeds the CMGUI 3D environment into the application is as follows:

```
<html:embed type="application/zinc-plugin" width="400" height="400"
id="cubePlugin" scene="default" uuid="f5a.."/>
```

This specifies to include an html 'embed' element (html:embed). The type is a ZINC plug-in. The width and height are specified and various identification information is recorded. The 'scene' attribute refers to the scene in the CMGUI com file that the model was rendered in. In this case it is just the default scene. One last thing to note about the index file is the use of 'oncommand' attributes used in some tags in the script.

```
<button label="Submit" oncommand="submit();" />
<button label="Reset" oncommand="reset();"   />
```

These just refer to functions which have been written in the javascript file to handle what should happen when the various elements are activated in any way. For example, in the above code if the 'Submit' button is clicked on by the user then the submit() function will be called in the javascript file.

There are various functions in the javascript file e.g. Load_file_to_memory(), executeCommand() and openFile(), which handle the setup of the CMGUI environment so that it can be embedded into the web application. For CMGUI to run correctly all the files that are associated with a certain project need to first be read into memory and then have the memory released. This includes the CMGUI com script all the exnode and exelem files that describe the model/animation data and all the texture/picture files used.

```
// Stream data into memory
Load_file_to_memory('example_cardiac_rehab.com','example_cardiac_rehab.com');
Load_file_to_memory('full_heart_press_0_def.exnode','/full_heart_press_0_def.exnode');
Load_file_to_memory('full_heart_press_0_def.exelem','/full_heart_press_0_def.exelem');
Load_file_to_memory('../../../../../rc_text2.jpg','/rc_text2.jpg');

executeCommand("open comfile memory:example_cardiac_rehab.com exec;");

// Free memory
Free_memory_block('example_cardiac_rehab.com');
Free_memory_block('/full_heart_press_0_def.exnode');
Free_memory_block('/full_heart_press_0_def.exelem');
Free_memory_block('/rc_text2.jpg');
```

The above section of code is an example taken from our javascript file which loads the CMGUI com file (example_cardiac_rehab.com), some model data (full_heart_press_0_def.exnode and full_heart_press_0_def.exelem) and a texture (rc_text2.jpg) into memory through the Load_file_to_memory() function. The executeCommand() function actually opens and runs the CMGUI script that has been written which displays the 3D heart visualization and then all the memory is released using the Free_memory_block() function. All this occurs in our script in the load_data() function. This function is called after validation and processing of the input entered by the user from the main index page (which is invoked when the user clicks on the Submit button). If the above process completes successfully the init_app() function is called which simply sets up some viewing parameters in the scene. A timer is then activated to call the incrementTime() function every 50 milliseconds:

```
      IntervalId = setInterval("incrementTime();", 50);
```

A special id is assigned to the global variable IntervalId to identify the timer. This incrementTime() function handles the animation of the model. In a normal CMGUI project a 'timekeeper' object can be used to display an animation simply by issuing the 'play' command. Once the play command is issued for the 'timekeeper' object all the animation frames are displayed from beginning to end. This provides the animation capabilities in a CMGUI script. At the moment, however, the 'timekeeper' functionality is not available with the ZINC plug-in so we have manually coded the animation by explicitly controlling the current time and then specifying what the CMGUI timekeeper's time should be e.g.

```
      executeCommand("gfx timekeeper default set_time " + time);
```

By making a call to the 'executeCommand()' function we can invoke a CMGUI command in the javascript file. The command that we executed above tells CMGUI's timekeeper object what it's current time value should be. This time value is associated with a specific animation frame (which has been read in through the .exelem files describing the animation). So the overall effect is the display of the next frame in the animation.

```
      time += beat_rate;
      if(time > 58)
            time = 0;
```

The time is then incremented by a certain delta value and if it exceeds the total amount of animation frames it is reset to the beginning to allow the animation to loop around.

Finally, the way we have implemented the display of several heart models (one for each day) is through the scrollbar object in the web application. The scrollbar has an event handler associated with it which calls the dayChanged() function whenever an event occurs. This function checks which day it should be displaying and calls the controlHeartBeat() and controlInfarctionSize() functions with the various input information the user entered into the index page as their parameters. These functions simply receive the information and modify the state of the heart accordingly. Take the controlInfarctionSize() function for example.

```
        function controlInfarctionSize(smoked, exercise, alcohol){
        ...
        }
```

The amount of cigarettes smoked, exercise done and alcohol intake for a particular day is sent to the function. A very simple scoring system has been devised which calculates a health score based on this information. At present this health scoring system is a rough prototype and in the final system a medical professional should have exactly specified how the input values could be interpreted in the display of the heart model. Once a score is calculated the corresponding size and location of the infarcted area is displayed on the heart model by calling specific functions which have been written in the CMGUI com file. For example, if the health score is large (size_score >= 4 as used by the current algorithm) and the location of the infarction is on the right ventricle then the following code is invoked:

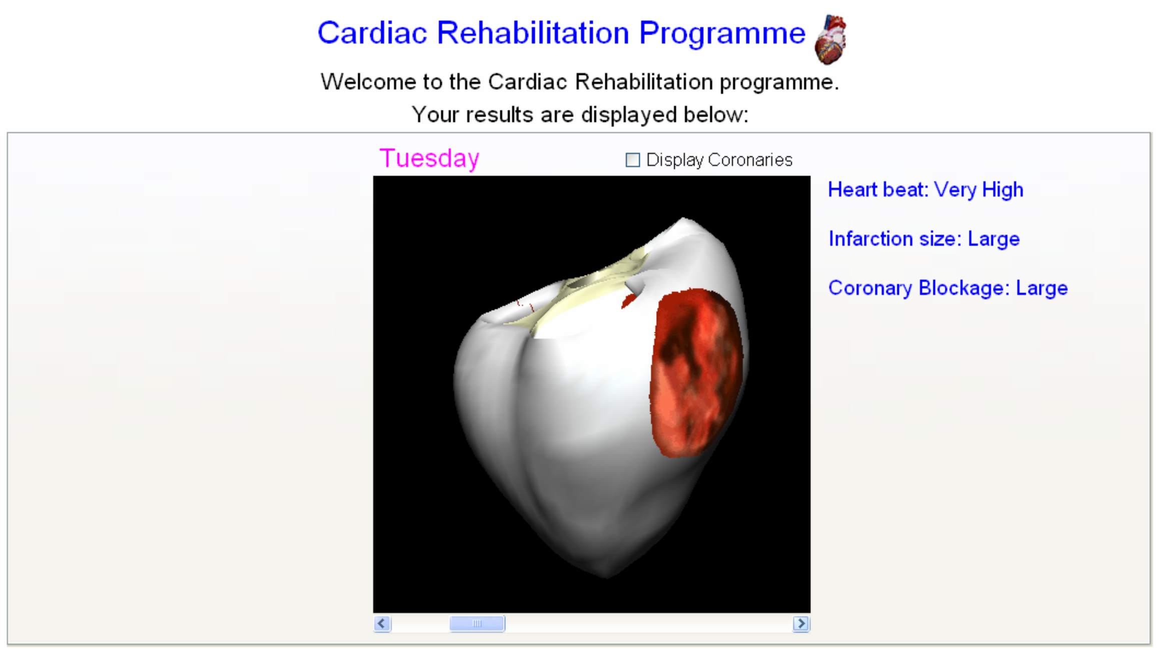
```
if(size_score >= 4){
        executeCommand("scale_large");

        switch(locn){
                case "left": executeCommand("scale_large_lv"); break;
                case "mid": executeCommand("scale_large_mid");
                                if(show_coro)
                                        executeCommand("display_large_mid_coro");
                             break;
                case "right": executeCommand("scale_large_rv");
                                 if(show_coro)
                                        executeCommand("display_large_rv_coro");
                              break;
        }

        stats.value = "Infarction size: Large";
        coro_stats.value = "Coronary Blockage: Large";
}
```

The above code uses a switch statement to control the location of the infarction. The executeCommand("scale_large") line calls a function scale_large() which has been implemented in the CMGUI com file. This scale_large() function specifies that a large infarction texture should be used on the heart model, indicating a large amount of damage to the heart surface. At the moment, this effect is achieved by using a separate image file for large, medium and small infarction sizes, however a better design would be to scale one main texture thereby ensuring a large range of infarction sizes could be achieved.

Because the location of infarction is on the right ventricle in our example the “right” case statement would be executed. The first command executeCommand(“scale_large_rv”) for this case once again calls a function, scale_large_rv(), which has been implemented in the CMGUI com file. This function uses the modification of texture coordinates to control where the infarction will be displayed i.e. over the right ventricle. More on using texture coordinates for translation is described below. The if statement tells CMGUI to display the heart’s coronary system. Finally, the last two lines of code outside the switch statement update the description of the heart’s state which is displayed in the user interface. An example of the output that this situation may produce in the current system is displayed in Figure 2 below.


Cardiac Rehabilitation Programme

Welcome to the Cardiac Rehabilitation programme.
Your results are displayed below:

Tuesday

☐ Display Coronaries

Heart beat: Very High

Infarction size: Large

Coronary Blockage: Large

Figure 2: Displays the output for a left ventricular infarction on the day Tuesday. A large infarction is displayed and information about the heart state is summarised on the right.

The CMGUI com file is the final script which makes up the current system. This is the file that handles the display of the heart model in the embedded CMGUI environment in the web application. First all the heart and coronary model and animation data (nodes and elements) are read in. Below is a segment of the code used to read in some model data.

```
# First read in initial heart data
gfx read node $example/full_heart_press_0_def time 0;
gfx read element $example/full_heart_press_0_def;

gfx read node $example/MRP01;
gfx read element $example/MRP01;
gfx read element $example/MRP01_base;
```

```
# Read in coronary data
gfx read nodes $example/pigsmooth.exnode;
gfx read elements $example/pigsmooth.exelem;
```

The above code uses the same tokens we described above about reading nodes and element files. It is split into three sections, the first reads nodes and elements for the static (non-animated) heart model. The second reads in data for a structure called MRP01. This is just the same as the heart surface data. Lastly, the coronary system is read in. This will allow the application to display the heart's coronary system as well as the heart's surface. For this project data from a pig's heart was read in and modified. Extra coronary groups are created to display various amounts of damage to the coronary system:

```
gfx create egroup small_mid_coro add 213..215;
gfx create egroup med_mid_coro add 211..215;
gfx create egroup large_mid_coro add 208..215;
```

The above commands tell CMGUI to create extra element groups. The numbers refer to the actual nodes which should be included in these groups. These separate groups each refer to a subset of the coronary system which are used to vary the amount of coronary damage. An example of various blockages to the right ventricle is displayed in Figure 3 below.

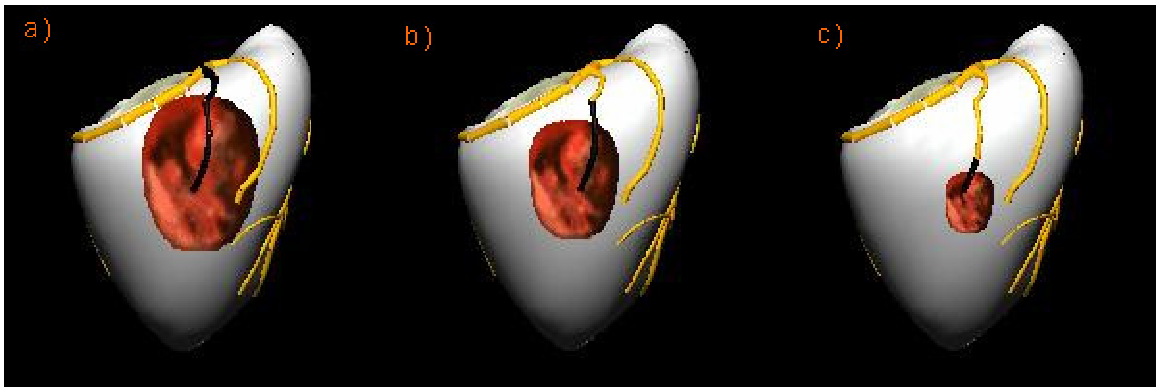


Figure 3: The above figures show a side on view of the left ventricle. Various amounts of blockage are shown to the coronary system (a) large, (b) medium, (c) small.

For the wrapping of the various textures around the heart surface Rectangular Cartesian coordinates are transformed into Cylindrical coordinates. This is achieved by defining extra fields in CMGUI which are slightly modified from previous existing fields e.g.

```
gfx define field cylindrical_texture_coordinatesA coordinate_system rectangular_cartesian
offset field coordinates offsets 0 0 -99.9464;
```

The above is the first in a series of field modifications we have used in our script. It says to define a new field called cylindrical_texture_coorindinatesA which is a coordinate_system. At the moment it is still a Rectangular Cartesian coordinate system and the modification to occur is to offset another coordinate system called 'coordinates' by approx. -100 in the z-axis. The last line in our modifications:

```
# Create an extra field which will be used for interaction
gfx define field tex_coords coordinate_system rectangular_cartesian scale field
cylindrical_texture_coordinates scale_factors $width $height 1;
```

produces a superfluous field called 'tex_coords' which will handle the translation of the infarcted region by varying the $width and/or $height parameters. By varying these parameters it has the effect of moving the current size infarction over the surface of the heart. Various functions have been setup e.g. translate_lv(), translate_rv(), in the CMGUI com file which are called from the javascript file when necessary to handle this ability. An example of a translation function is as follows:

```
sub translate_left(){
      $width -= $delta;
      gfx define field tex_coords coordinate_system
      rectangular_cartesian scale field cylindrical_texture_coordinates
      scale_factors $width $height 1;
}
```

The above function provides the functionality to translate the infarcted region to the left by some specified value ($delta). Using the tex_coords field the x and y coordinates are set to the current $width and $height values, respectfully.

The CMGUI com file also provides various functions for scaling the infarcted region e.g. scale_large_lv(), scale_med_mid(). As well as displaying the coronary system, display_coronaries(), and displaying the various amounts of damage to the coronary subsets, display_large_mid_coro(), display_small_rv_coro(). All of these functions are called from the javascript file when a modification to the heart model needs to be displayed to the user.

6 Results

Using daily monitoring input parameters such as average heart beat, exercise, cigarettes smoked and alcohol intake the system processes this data and produces a 3D visualization of a patient's heart with various visual cues describing the state of cardiac health of the patient. These visual cues include the speed of the animation describing the patients average heart beat rate, the location and size of an infarcted region on the heart's surface and damage to the coronary system of the heart by displaying blockages which have occurred at various areas in the heart's coronaries. At the present the system collects data from the patient such as location of infarcted region, age, average heart beat, daily alcohol intake etc, but the system could be expanded to incorporate further input data that would effect the state of the patient's heart. Moreover, additional information could be displayed to the patient through the visualization thereby increasing the scope of the system, for example a possible improvement might be to display certain enlarged areas of the heart or to display contraction of the heart's arteries.

In the current system a separate model is displayed for each day that data is collected. This is to give the patient a basis for comparison and allow them to physically monitor the impact that various behaviors have on their heart's state. Figure 4 shows a comparison between various heart states for a sample patient and how the state of the heart is portrayed to the user.

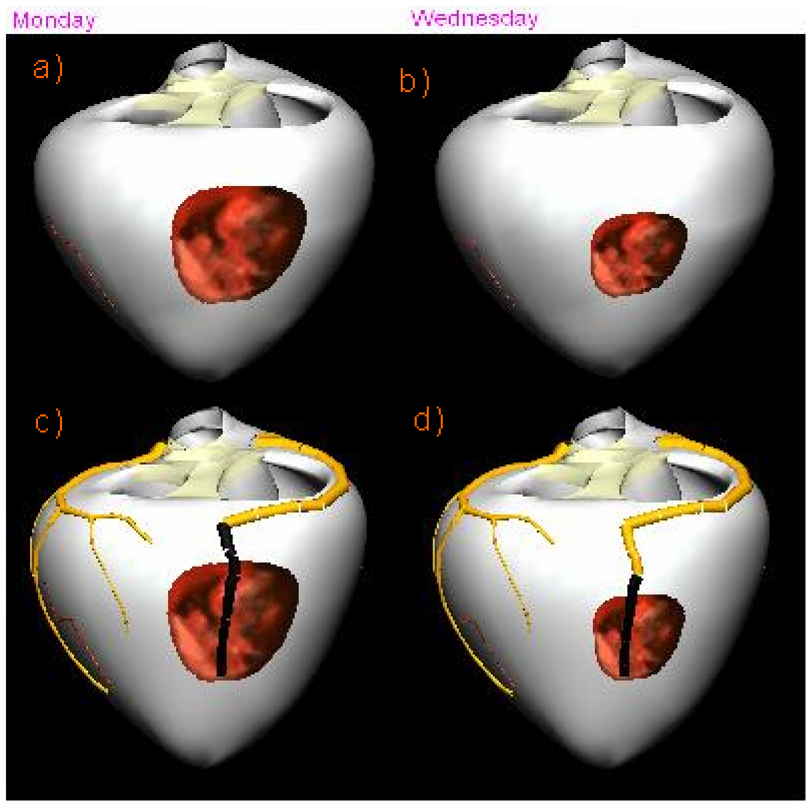


Figure 4: Heart model comparisons for Monday and Wednesday (a) and (b).
The comparison shows a large infarcted middle section region for Monday and a less infarcted region for Wednesday.
The coronary system is displayed for models (c) and (d).

As can be seen above the current model displays the infarcted region across a white heart's surface. One of the main drawbacks of the current system is that at the present it does not provide multi-texturing capabilities which would allow the infarction to be translated and scaled on top of a general heart texture.

Figure 5 shows the desired outcome that will be possible once multi-texturing is enabled under CMGUI. Figure 5 also incorporates per pixel lighting and bump mapping techniques for increased realism.

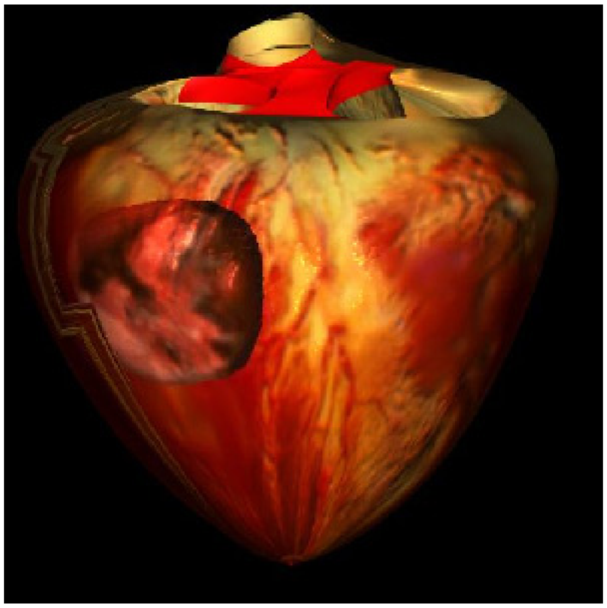


Figure 5: Shows a heart model with a left ventricular infarction. The above model uses per pixel lighting and bump mapping effects to create a more realistic image.

7 Future Work

At the moment the system displays the infarcted area on the heart surface and the rest of the texture is transparent. Ideally, the infarcted region should be visible on top of a normal, healthy heart surface texture that has been wrapped around the whole heart (as demonstrated above in Figure 5). This multi-texturing ability requires modification to the CMGUI system (which is currently in progress). The infarction can then be scaled and translated around the healthy texture to provide information about the patient's state of health.

A further improvement would be to apply CMGUI's ability to use per pixel lighting and bump mapping to the heart surface to provide a better quality visualization model to the user (once again this has been shown in Figure 5). At the moment the system uses standard gouraud shading. However, certain hardware requirements are then imposed, which should be taken into account.

As mentioned above, there are extra attributes which could be included as input into the system. General details such as age, gender and weight could have a slight impact on areas such as infarction size, coronary damage and heart beat rate whereas daily data input (e.g. alcohol intake) would have more of an affect on the results.

Another improvement involves the way the infarcted region is currently scaled. At the moment three different texture images are used to control the size of the infarction. A better implementation would use a more continuous design to scale the infarction, for instance modifying the actual texture coordinates. Masking could also be used to control the shape of the infarction. At the moment the actual shape does not change, but by using masks this could be controlled, if required.

At the moment the ZINC extension which allows the CMGUI environment to be embedded into web based applications can only be used with Mozilla based browsers. A further improvement would be to provide this ability for other web browsers e.g. Internet Explorer.

Bibliography

Kikinis R, Langham Gleason P, Jolesz FA. Surgical planning using computer-assisted three-dimensional reconstructions. In: Taylor R, Lavalle S, Burdea G, Mosges R, editors. Computer integrated surgery. Cambridge, MA: MIT Press, 1995. p. 147 – 54.

Lorensen WE, Jolesz FA, Kikinis R. The exploration of cross-sectional data with a virtual endoscope. In: Morgan K, Satava RM, Sieburg HB, Mattheus R, Christensen JP, editors. Interactive Technology and the new paradigm for healthcare. Ohmsha: IOS Press, 1995. p 221 – 30.

Robb RA, Barillot C. Interactive display and analysis of 3-D medical images. IEEE Transactions on Medical Imaging 1989;8(3):217-26.

Robb RA. Three-dimensional biomedical imaging. Principals and practice. New York: VCH, 1995.