Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ Proyecto Fin de Grado ▪

Computación

Generación de patrones de costura mediante Kinect y Makehuman.

Autora:

Carla Aldabaldetrecu Orús

Directora:

Blanca Rosa Cases Gutiérrez

Septiembre 2016

informatika fakultatea
facultad de informática

Me gustaría agradecer a Blanca Rosa Cases Gutiérrez el apoyo, la ilusión y la orientación que me ha ofrecido durante la realización del proyecto y del curso. Has conseguido que disfrute aún más de mi especialidad.

A mis compañeros de clase, en especial a Laura Gran, Olivia Olazabal, Leire Roa y a mi novio Andoni Sánchez por ofrecerme cuatro años maravillosos llenos de diversión, apoyo y complicidad que nunca olvidaré.

A mi familia, Mariano, Maite, Irma y Ángel, por darme la oportunidad de estudiar aquello que tanto me gusta y animarme a ello. En especial a mi padre, Mariano Aldabaldetreku, tantos sacrificios que has hecho por mí, por Irma y por la Ama para que podamos tener un futuro próspero. Aunque lamentablemente ya no estés aquí presente espero que te sientas orgulloso de mí y que de alguna forma este proyecto me acerque más a ti, a tu profesión, a la sastrería, que tanto esfuerzo te ha costado levantar.

Summary

This document corresponds to the End of Bachelor's Project Report (Memoria del Proyecto de Fin de Grado), Generation of sewing patterns using the Kinect and MakeHuman, developed for the Faculty of Computer Engineering at the UPV-EHU university of Donostia-San Sebastián under Blanca Cases supervision, teacher at UPV-EHU University.

The project consists in obtaining the measurements of someone by using the Kinect's depth sensor (which was previously achieved by Ibon Olabarria using only the image that ended up causing precision loss), creating the model with the client's form and measurements on MakeHuman and, after selecting the clothes, adapting the pattern automatically to the user's measurements. This would generate a life-size sewing pattern in a pdf.

Index

List of Figures and Tables

FIGURES

TABLES

1

Introduction

The idea behind this project is to make available a tool that helps people to create sewing patterns that will fit their body and to help industries developing their own patterns. The problem is that standard sizes usually do not fit everyone properly, two people can share that same clothing size but, because their bodies do not have the exact same measurements, these do not look the same, and so, changes have to be done on the clothes. Tailors come to work in those situations but this means that more money has to be spent. Another solution is to create your own pattern from scratch but it may take a lot of time and effort. Taking your own measurements, learning how to create a pattern, drawing it or using a program to create it would be essential.

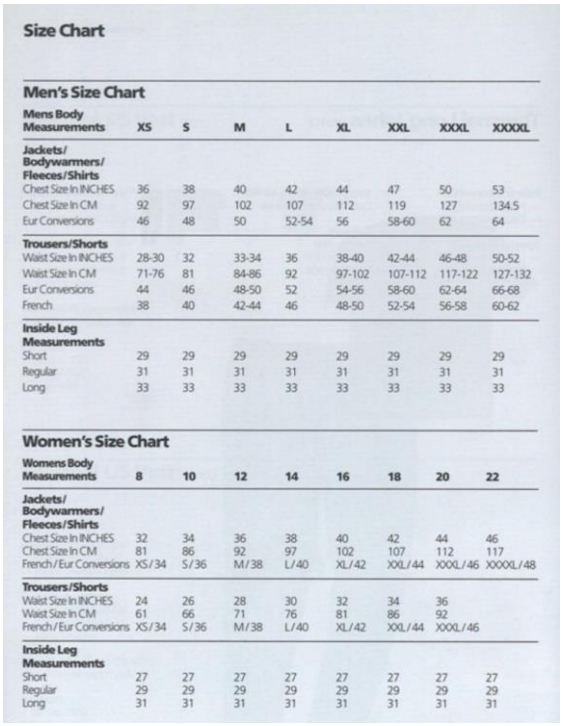
The proposed solution, which is the combination of Geogebra with Kinect, Matlab, Blender and MakeHuman, follows all the steps needed to the creation of your own pattern so that the client does not need to spend time or money on it. Measurements will be taken automatically and by selecting the wanted clothing and, filling in the corresponding measurements, the pattern will be created. Furthermore, a mannequin with the possible clothes will be shown to the user. Finally, the client will be able to download the pattern so that later he/she can sew it.

This project will follow the lead of Ibon Olabarria's "Sistema de comercio electrónico para el sector textil: generar maniquíes 3D a partir de imágenes del sensor Kinect" Project. He managed to achieve measurements thanks to the Kinect's depth sensor.

1.1. Pattern making production

Pattern making/cutting is the core of the clothing production; it unifies the idea of the designer with the three-dimensional result. Not many people are specialized in this field which requires investigating and drawing ideas, cutting the patterns and identifying the final size; creating and finishing the garment and also selling it. There is no doubt that tailors need quite a lot of skills in order to succeed in the fashion world. In order to make things easier for them technology has been brought into it.

During production creators follow standard sizing tables which depend on where they are making it. Though they are different depending on the country/continent there is a way to calculate the equivalence size in other cities:



Size Chart

Men's Size Chart

Mens Body Measurements	XS	S	M	L	XL	XXL	XXXL	XXXXL
Jackets/ Bodywarmers/ Fleeces/Shirts								
Chest Size In INCHES	36	38	40	42	44	47	50	53
Chest Size In CM	92	97	102	107	112	119	127	134.5
Eur Conversions	46	48	50	52-54	56	58-60	62	64
Trousers/Shorts								
Waist Size In INCHES	28-30	32	33-34	36	38-40	42-44	46-48	50-52
Waist Size In CM	71-76	81	84-86	92	97-102	107-112	117-122	127-132
Eur Conversions	44	46	48-50	52	54-56	58-60	62-64	66-68
French	38	40	42-44	46	48-50	52-54	56-58	60-62
Inside Leg Measurements								
Short	29	29	29	29	29	29	29	29
Regular	31	31	31	31	31	31	31	31
Long	33	33	33	33	33	33	33	33

Women's Size Chart

Womens Body Measurements	8	10	12	14	16	18	20	22
Jackets/ Bodywarmers/ Fleeces/Shirts								
Chest Size In INCHES	32	34	36	38	40	42	44	46
Chest Size In CM	81	86	92	97	102	107	112	117
French/Eur Conversions	XS/34	S/36	M/38	L/40	XL/42	XXL/44	XXXL/46	XXXXL/48
Trousers/Shorts								
Waist Size In INCHES	24	26	28	30	32	34	36	
Waist Size In CM	61	66	71	76	81	86	92	
French/Eur Conversions	XS/34	S/36	M/38	L/40	XL/42	XXL/44	XXXL/46	
Inside Leg Measurements								
Short	27	27	27	27	27	27	27	27
Regular	29	29	29	29	29	29	29	29
Long	31	31	31	31	31	31	31	31

Figure 1.1 Size Chart

As you can see on the image above women's and men's sizes are not equivalent. In men a shirt size 38 is not put together with a trouser size 38, also, sizes for men's shirts start at 46. The size of a men's shirt can be calculated by dividing his chest circumference by two. For example, if the chest circumference is size 100cm then the size of my shirt is 50.

But no everyone has the exact chest size and that is when tailors come into the picture. People are not like mannequins with the exact sizes and symmetric bodies so every inch of the body has to be measure, this will be discussed in section Kinect and Matlab

Once all the needed measurements are taken the tailor can start creating the pattern with the required equipment: scissors, tailor's chalk, set-square, measuring tape and the tailor's rule. This last one is really important in order to create the scye's and neck's curves.

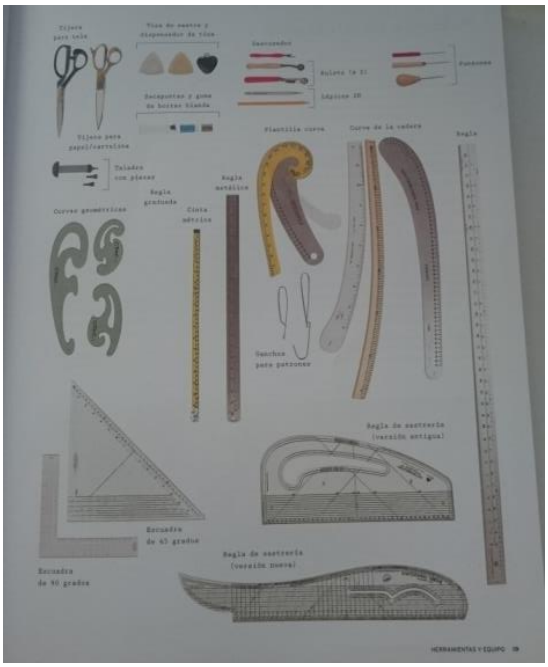


Figure 1.2. Tailor's equipment

1.2. Existing Programs

There already exit programs that can help designers/tailors design their creations. What I am trying to achieve is a similar result to the ones obtained by these softwares and, also, add the element of obtaining measurements. Another aim of this project is to avoid the costs of these programs, as they are neither free nor cheap. Here is a list containing the names, prices and a brief description.

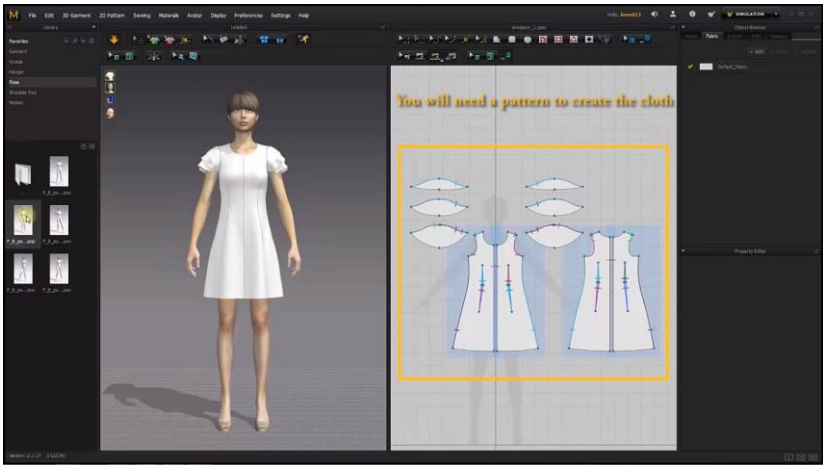


Figure 1.2.1 Marvellous Studio Designer interface

Name	Price (€)	Description
Marvellous Studio Designer	53.71	Allows creating the pattern and adjusting it to the 3D model simultaneously
Virtual Fashion	Free	Simple, patterns could not be modelled as tailors do. It is like Blender.
Optitex	Not specified but is not free	Similar to Marvellous Studio designer
V-Stitcher	Not specified but is not free	First create the pattern and later adjust it to a model, simple interface

Table 1.1. Alternative programs

1.2.1. Alternative Pattern Making Programs

The following programs were considered for the project but did not meet the requirement fully for it.

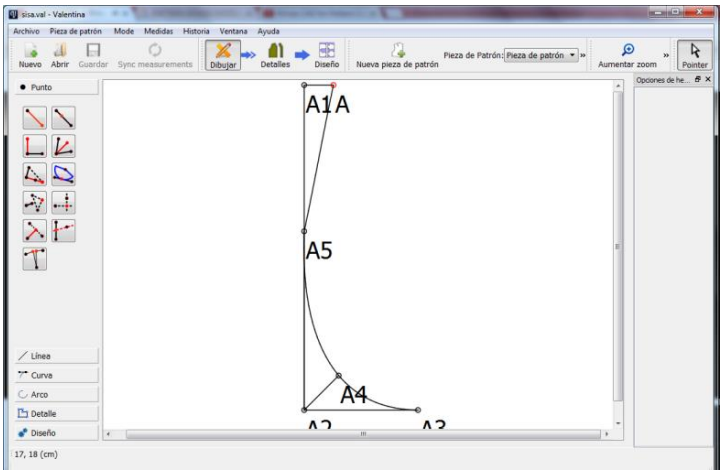


Figure 1.2.1.1 Valentina interface

- *Valentina*: is a Free Open Software used to produce clothing patterns. It allows importing the pattern to SVG so that is compatible with Blender. The problem with Valentina is that curves, like the neck or sleeve, have to be done manually. There was no way of simulating the tailor's ruler or to create a curve with the tools given. First you will have to create a Segment between points and later adjust that segment to a curve. This is of no interest as that will make it inaccurate and will, probably, have to be corrected once the pattern is sewed.

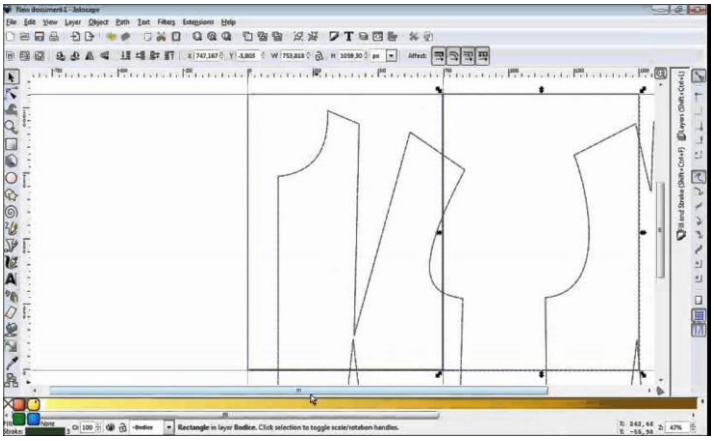


Figure 1.2.1.2 Inkscape interface

• *Inkscape*: Professional SVG editor and Open Software for drawing. It is usually used as a substitution to Adobe Illustrator as it is free. The problem with Inkscape was similar to Valentina, drawing segment and curves had to be done manually which would make the pattern inexact.

• *Clothed*: Program for converting clothing patterns into three-dimensional meshes. It allows you to create a database and to export the pattern into Blender as a mesh so that you don't have to do it manually, though you may have to edit it to fit the model. Also, in order to use it, you would first need to draw the pattern in another editor (such as Inkscape), import it to SVG so that you can edit it in Clothed. This software is still in development and contains many bugs; another drawback is the amount of time that it costs to create the three-dimensional mesh.

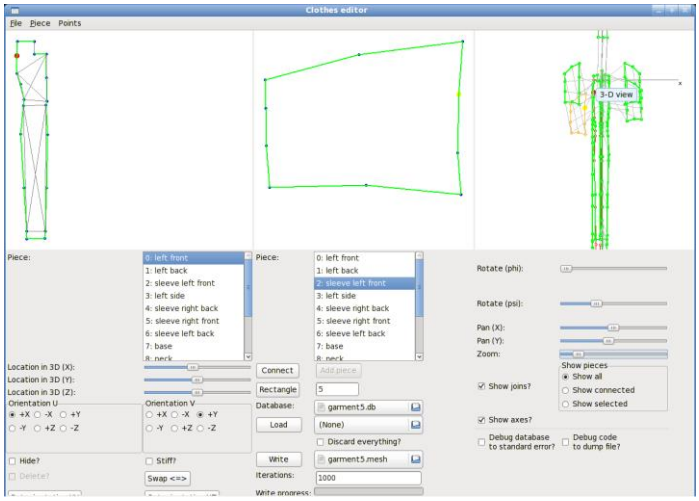


Figure 1.2.1.3 Clothed interface

1.2.2. Alternative Scanning Program

Textile Clothing Corp: The TC2-19 3D body scanner scans the body in 4D to detect movement. With a 360º camera allows you to inspect every centimetre of the body. It is the most accurate software scanning of the market and does not take too long to work. If packaged with ImageTwin (software of the same company) offers avatar creation, poses, animation, size matching, virtual fitting, personalization, and styling advice.

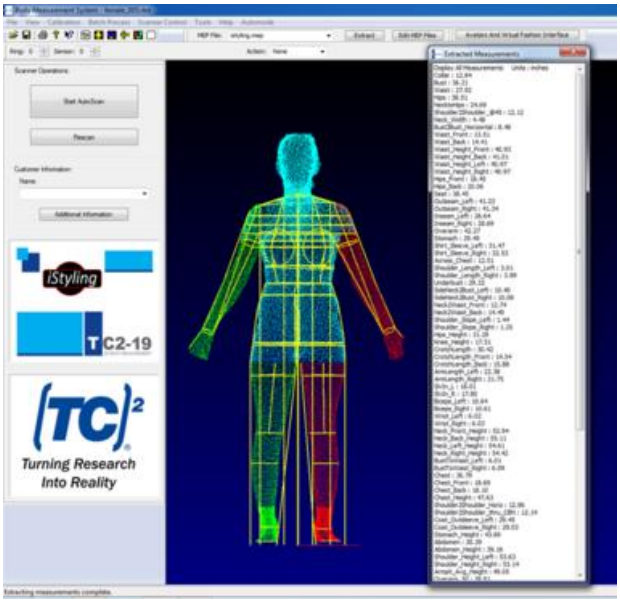


Figure 1.2.2.1 Textile Clothing Corp interface

1.3. Programs selected

This project is a combination of Matlab, Geogebra, Blender and MakeHuman. Matlab will connect with the Kinect and will retrieve the measurements. Geogebra will be the software used for creating the patterns as it can simulate the actions needed and counts with a web applet so that anyone can create their patterns automatically. MakeHuman and Blender will be in charge of modelling the clothes created in Geogebra and will export it into the webpage.

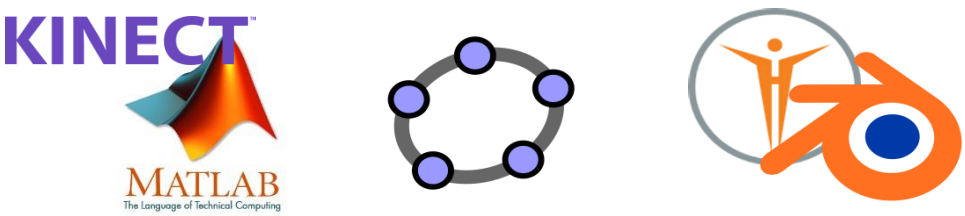


Figure 1.3.1. All the programs combined

2

Objectives, Methodology and Tools

The fashion clothing field is really competitive, very few designers achieve success and tailors are no longer easily to be found. Schools no longer teach kids to sew so clients have no experience in this world and depend on the trends of the market to get what they want.

Technology can make thing easier to all of them. Software on the market can help you see the final result on a mannequin with the exact measurements, not symmetric as the ones used on real life, which would save you time on fixing it. Unfortunately these solutions are usually too expensive, buggy or incomplete. They may lack of accuracy or the tools are too limited.

The project's intention is to create patterns automatically; later on a more complete free solution was created. It will first manage to scan the body in order to achieve the measurements needed to create the pattern. Creating some basic patterns that will be able to be adapted to the client's measures and later printed on PDF. Also, finding an easy and complete software so that companies can add more patterns in the future. And finally, bring that clothing to live so that the final result can be seen.

2.1 Scope of the project

The scope of the project is divided into two different phases:

1. The study of the market, which includes finding a suitable program for the pattern making production, and following the development of the Kinect software, Blender and MakeHuman since Ibon's project.
2. Developing and combining all the technologies.

The final result has to follow these aspects:

- The software used in it must be free.
- Contain the essential elements; the engine has to meet a minimum of quality in order to be acceptable. The patterns have to be made automatically and able to be printed on PDF as well as showing the clothes modelled and allowing the user to scan itself.

The development carried out follows these steps:

- Study the connection between Kinect and Matlab learning how to use the depth sensor to obtain the required measurements.
- Study the market technology and, based on what they offer, look for a free alternative that allows creating the patterns automatically.
- Research how to create clothes on Blender and export it to MakeHuman.
- Create the patterns using geometry and adapt them so that the only thing the client has to do is fill its measurements.
- Show the mannequin with the pattern previously created.
- Recreate Ibon Olabarria's project by scanning the user with the Kinect camera.

After studying and meeting some dead ends the project resulted on a two phase application, scanning and obtaining the pattern.

1. The client has to scan itself with the Kinect running a script already generated for Matlab. This was already achieved by Ibon Olabarria.

2. He will find a webpage containing Geogebra's applet (used to generate the patterns automatically) and three 3-D models, exported to ThreeJS from Blender, with the clothes that he can choose. The user has to decide whether he wants to generate a shirt or trouser already sized or customize it. In case he chooses the second one the page will ask for him to enter the measurements previously obtained from Matlab. Finally, the client can download the code of the patterns, open them on Geogebra and print it on a PDF in order to sew it.

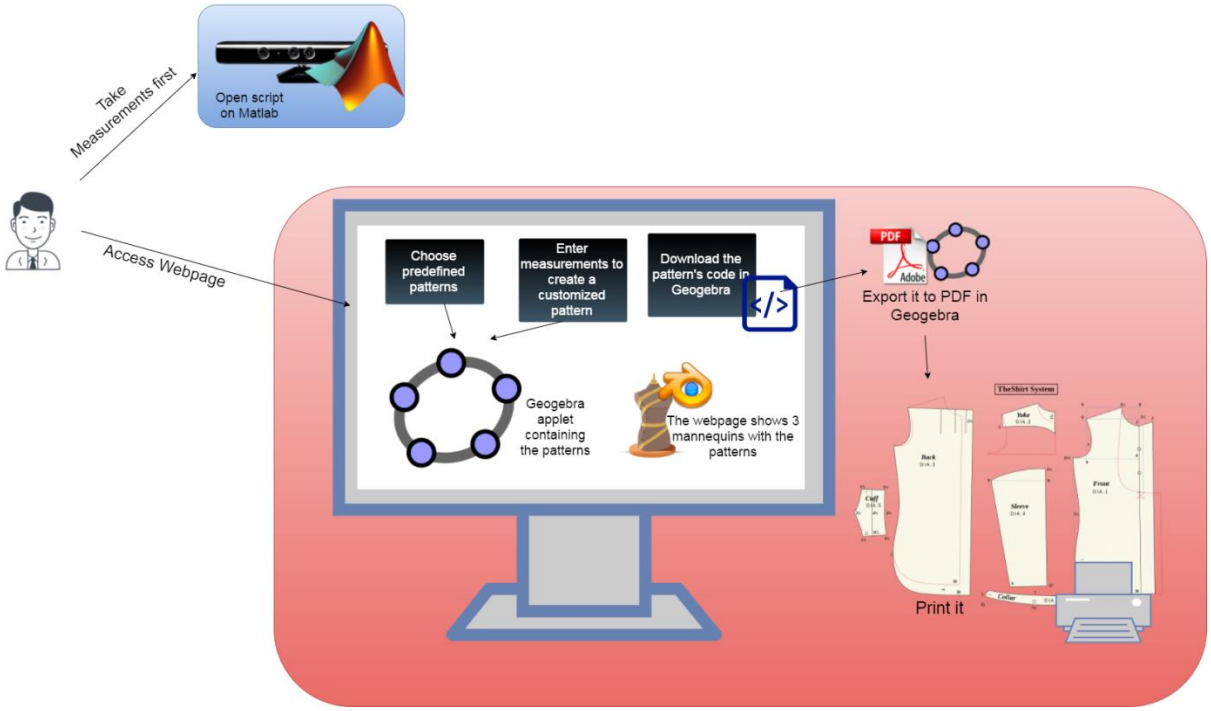


Figure 2.1.1. Project Overview

2.2 Exclusions of the project

The following list contains the tasks excluded from the project:

- Connect the Kinect with Matlab automatically; no script will start Matlab with the measurement script. The user will have to connect the Kinect, start Matlab and run the script manually in order to obtain the measurements.
- Fill in the measurements automatically on the webpage. Matlab returns a series of measurements that will not be then passed to the webpage, the client needs to copy them onto the webpage.
- A representation of the client with his or her measurements. Creating a MakeHuman mannequin and importing it to Blender to later export it to ThreeJS.
- PDF exportation from the webpage.

- Other Operative Systems compatibilities (Linux, Mac OS X).

2.3 Work Breakdown Structure

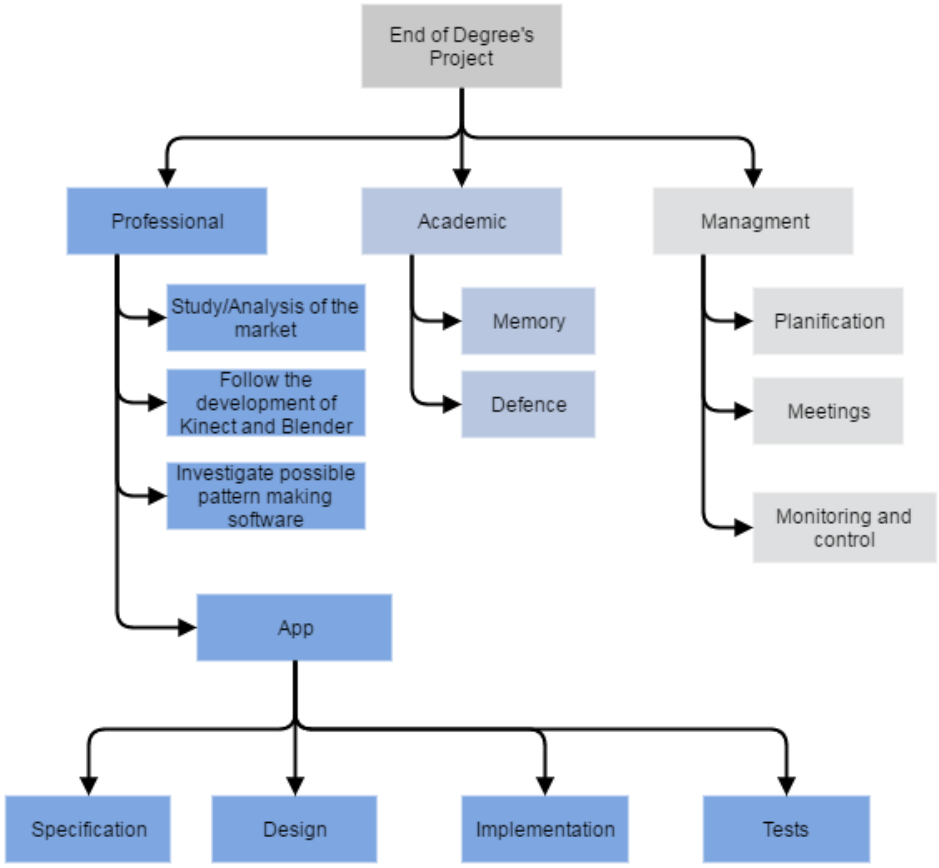


Figure 2.3.1. Work Breakdown Structure

2.4 Methodology

The methodology followed in this project is the Waterfall Model. It consists on a sequential (non-iterative) design process in which the progress flows steadily downwards (like a waterfall).The model is divided into sequential phases, with some overlap and splashback acceptable between phases. It is important to keep track of time, target dates and having a detailed planning. Control is maintained through the course of the project via meetings and approvals before entering a new phase. Written documentation is an explicit deliverable of each phase. The phases typically are: System and software Requirements (conception, initiation and analysis), design, implementation, testing/verification and maintenance.

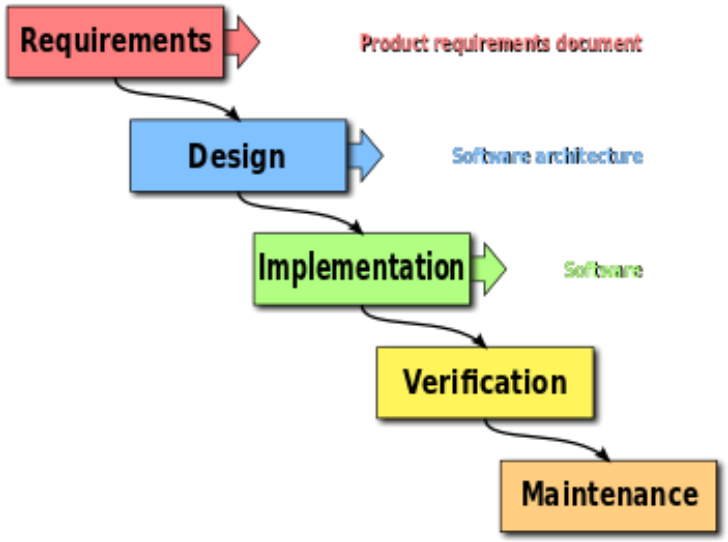


Figure 2.4.1. Waterfall Model Phases.

- In the first phase, System and software Requirements, the products requirement should be documented. In this project many programs are to be used so it is important that all of them were compatible for the same Operative System and that this one was the one I used. The only program limited to Operative Systems was Matlab, which is not available for Linux.

- The Design phase consists in the software architecture and use cases. In this project being the function of each program selected (Matlab, Geogebra, Blender and MakeHuman).

- When developing the Implementation/Coding phase the result has to be the development, proving, and integration of the software.

- Verification/Testing which consists on the systematic discovery and debugging of defects.

The work done in these phases will be better explained in section 6. Development.

2.5 Tools

It was decided to use Dropbox to keep track of the Project's development. Each program's name is a folder so that it is tidy and everything is easier to be found. Objects with similar properties were kept in the same place, like articles or documents for the report which saved on the same folder. Both the director and the author had access to the folder with the same privileges.

As for communication, first it was established to use emails for arranging meetings or clarifying any doubts that arose. Later it changed and the application Telegram was used, letting the communication work more fluently.

During the life of the project several meetings were arranged and documented by the author of the project. Later on, they were available for the director on Dropbox.

Finally, in order to create the Work Breakdown Structure and the application's flow functioning, draw.io was used. This is a free web diagram software application for making flowcharts, prices diagrams, network diagrams, etc… The diagrams created can be exported directly to Dropbox which allows us to keep every object used in the same place.

3

Kinect and Matlab

Kinect and Matlab are used to obtaining the measurements that are needed for creating sewing patterns.

3.1. Obtaining the measurements

Once the Kinect is connected to the computer and the electricity it has to be placed on a plain surface, at two meters from the person and one above the floor. The person has to fit in the camera completely and its background should not interfere with the image. Now in order for the measurement to be taken the script should be run.

The script mentioned on the following paragraphs was created by Ibon Olabarria on his End of Degree's Project. Here the most important characteristics will be mentioned but in Anex A: Kinect and Matlab the whole script appears if you want to go deeper into the code.

First it is important to add the path to the Kinect for Windows Sensor Support Package:

```
utilpath = fullfile(matlabroot, 'toolbox', 'imaq', 'imaqdemos', 'html',
'KinectForWindows');
addpath(utilpath);
```

Program 3.1.1. *Add path.*

Now the Kinect has to be initialized in Matlab in order to use it. As Kinect has two sensors it is important to enable independent acquisition from each of these devices. Both

of them are treated as two independent devices in the Image Acquisition Toolbox which means that two separate VideoInput object needs to be:

```
hwInfo = imaqhwinfo('kinect')
hwInfo.DeviceInfo(1)
hwInfo.DeviceInfo(2)
% Create the VIDEOINPUT objects for the two streams
colorVid = videoinput('kinect',1);
depthVid = videoinput('kinect',2);
```

Program 3.1.2. *Initialize the Kinect.*

```
hwInfo =

       AdaptorDllName: [1x79 char]
    AdaptorDllVersion: '4.5 (R2013a)'
          AdaptorName: 'kinect'
            DeviceIDs: {[1]  [2]}
           DeviceInfo: [1x2 struct]

ans =

             DefaultFormat: 'RGB_640x480'
       DeviceFileSupported: 0
                DeviceName: [1x19 char]
                  DeviceID: 1
     VideoInputConstructor: [1x23 char]
    VideoDeviceConstructor: [1x29 char]
          SupportedFormats: {1x4 cell}


ans =

             DefaultFormat: [1x13 char]
       DeviceFileSupported: 0
                DeviceName: [1x19 char]
                  DeviceID: 2
     VideoInputConstructor: [1x23 char]
    VideoDeviceConstructor: [1x29 char]
          SupportedFormats: {1x3 cell}
```

Figure 3.1.1 *Initialize the Kinect.*

Acquiring synchronized colour and depth data has to use manual triggering instead of immediate triggering. The default immediate triggering suffers from a lag between streams while performing synchronized acquisition.

```
% Set the triggering mode to 'manual'
triggerconfig([colorVid depthVid],'manual');
```

Program 3.1.3. *Triggering manually.*

In order to give enough time to the Windows sensor to rack the skeleton the FramesPerTrigger property of the VideoInput objects will be set to 100. This means that it will acquire 100 frames per trigger. After it the colour and depth devices have to start acquiring the data and logging it.

```
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
% Start the colour and depth device. This begins acquisition, but does not
% start logging of acquired data.
start([colorVid depthVid]);
% Trigger the devices to start logging of data.
trigger([colorVid depthVid]);
% Retrieve the acquired data
[colorFrameData,colorTimeData,colorMetaData] = getdata(colorVid);
[depthFrameData,depthTimeData,depthMetaData] = getdata(depthVid);
% Stop the devices
stop([colorVid depthVid]);
```

Program 3.1.4. *Retrieve data.*

The Kinect for Windows sensor provides different modes to track skeletons, TrackingMode, SkeletonToTrack and BodyPosture properties. These modes can be accessed and configured from the VideoSource object of the depth device.

```
% Get the VIDEOSOURCE object from the depth device's VIDEOINPUT object.
depthSrc = getselectedsource(depthVid)
```

Program 3.1.5. *Get depthVid source.*

TrackingMode controls whether or not skeletal tracking is enabled and, when enabled, whether all joints are tracked, Skeleton, or if just the hip position is tracked, 'Position'. This is the property used with the Skeleton mode.

BodyPosture property determines how many joints are tracked, when Standing it will track twenty but if it is set to Seated it will track ten.

The SkeletonToTrack property can be used to selectively track one or two skeletons using the 'SkeletonTrackingID'.

```
% Turn on skeletal tracking.
depthSrc.TrackingMode = 'Skeleton';
```

Program 3.4.6. *Initialize the Kinect.*

Now the data from the depth sensor has to be retrieved.

```
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
start([colorVid depthVid]);
trigger([colorVid depthVid]);
% Retrieve the frames and check if any Skeletons are tracked
[frameDataColor] = getdata(colorVid);
[frameDataDepth, timeDataDepth, metaDataDepth] = getdata(depthVid);
% View skeletal data from depth metadata
metaDataDepth
% Check for tracked skeletons from depth metadata
anyPositionsTracked = any(metaDataDepth(95).IsPositionTracked ~= 0)
anySkeletonsTracked = any(metaDataDepth(95).IsSkeletonTracked ~= 0)
```

Program 3.4.7. *Retrieve data 2.*

The program above should have shown that there is at least one skeleton being tracked. To see which skeleton is being track use the IsSkeletonTracked.

```
% See which skeletons were tracked.
trackedSkeletons = find(metaDataDepth(95).IsSkeletonTracked)
```

Program 3.4.8. T*racked skeletons.*

Display skeleton's joint coordinates.

```
jointCoordinates     =     metaDataDepth(95).JointWorldCoordinates(:,     :,
trackedSkeletons)
% Skeleton's joint indices with respect to the colour image
jointIndices = metaDataDepth(95).JointImageIndices(:, :, trackedSkeletons)
```

Program 3.4.9. *Find Joints.*

Draw the Skeleton over the corresponding colour image

```
% Pull out the 95th colour frame
image = frameDataColor(:, :, :, 95);

% Find number of Skeletons tracked
nSkeleton = length(trackedSkeletons);

% Plot the skeleton
util_skeletonViewer(jointIndices, image, nSkeleton);
```

Program 3.4.10. Draw the skeleton.

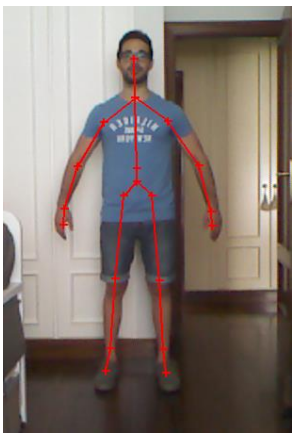


Figure 3.4.2. Skeleton drawn.

Now what is left is to calculate the length of the joints and their widths. First the length will be calculated. The joints are accessed through the jointCoordinates and each one is assigned one number (one to twenty) and the coordinates x and y (represented by one and two). These are the indexes:

Joints	Indexes
SpineBase	1
SpineMid	2
Neck	3
Head	4
ShoulderLeft	5
ElbowLeft	6
WristLeft	7
HandLeft	8
ShoulderRight	9
ElbowRight	10
WristRight	11
HandRight	12
HipLeft	13
KneeLeft	14
AnkleLeft	15
FootLeft	16
HipRight	17
KneeRight	18
AnkleRight	19
FootRight	20

Table 3.4.1. Joints Indexes.

As nobody is symmetric is better to calculate both extremities (example: shoulder right and left) and later calculate the mean:

```
%Calculate  the joint lengths
upperarm_left=sqrt(((jointCoordinates(5,1)-
(jointCoordinates(6,1)))^2)+(jointCoordinates(5,2)-
(jointCoordinates(6,2)))^2)*100;
upperarm_right=sqrt(((jointCoordinates(9,1)-
(jointCoordinates(10,1)))^2)+(jointCoordinates(9,2)-
(jointCoordinates(10,2)))^2)*100;
upperarm=(mean([upperarm_left,upperarm_right]))*100;
lowerarm_left=sqrt(((jointCoordinates(6,1)-
(jointCoordinates(7,1)))^2)+(jointCoordinates(6,2)-
(jointCoordinates(7,2)))^2)*100;
lowerarm_right=sqrt(((jointCoordinates(10,1)-
(jointCoordinates(11,1)))^2)+(jointCoordinates(10,2)-
(jointCoordinates(11,2)))^2)*100;
lowerarm=(mean([lowerarm_left,lowerarm_right]))*100;
frontcheast=sqrt(((jointCoordinates(5,1)-
(jointCoordinates(9,1)))^2)+(jointCoordinates(5,2)-
(jointCoordinates(9,2)))^2)*100;
necktowaist=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(2,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(2,2)))^2)*100;
waist_to_hip=sqrt(((jointCoordinates(2,1)-
(jointCoordinates(1,1)))^2)+(jointCoordinates(2,2)-
(jointCoordinates(1,2)))^2)*100;
shoulderright=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(5,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(5,2)))^2)*100;
shoulderleft=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(9,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(9,2)))^2)*100;
backdist=(mean([shoulderleft,shoulderright]))*100;
upperlegleft=sqrt(((jointCoordinates(13,1)-
(jointCoordinates(14,1)))^2)+(jointCoordinates(13,2)-
(jointCoordinates(14,2)))^2)*100;
upperlegright=sqrt(((jointCoordinates(17,1)-
(jointCoordinates(18,1)))^2)+(jointCoordinates(17,2)-
(jointCoordinates(18,2)))^2)*100;
upperleg=(mean([upperlegleft,upperlegright]))*100;
lowerlegleft=sqrt(((jointCoordinates(14,1)-
(jointCoordinates(15,1)))^2)+(jointCoordinates(14,2)-
(jointCoordinates(15,2)))^2)*100;
lowerlegright=sqrt(((jointCoordinates(18,1)-
(jointCoordinates(19,1)))^2)+(jointCoordinates(18,2)-
(jointCoordinates(19,2)))^2)*100;
lowerleg=(mean([lowerlegleft,lowerlegright]))*100;
neck=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(4,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(4,2)))^2)*100;
```

Program 3.4.11. Calculate joints.

Lastly, as we infer from the colour image we want to obtain the width of the following joints:

1. Arms superior width
2. Wrists width
3. Thighs width
4. Twin's width
5. Ankle's width
6. Neck's width.

For that, the silhouette of the body needs to be found, in other words, the pixels that limit the user's image. For that the colour image will have to be processed and turned into black and white.

```
prof=frameDataDepth(:,:,:,82);
util_skeletonViewer(jointIndices,prof,nSkeleton);


grayimage=rgb2gray(image);
g=histeq(grayimage,100);
[~,threshold]=edge(g,'sobel');
fudgeFactor=0.5;
BWs= edge(g,'sobel',threshold*fudgeFactor);
figure,imshow(BWs);
util_skeletonViewer(jointIndices,BWs,nSkeleton);
```

Program 3.4.12. Find the silhouette.

Finally, we will need to use the correct pair of indexes to get the width of each one. First, create a line between them in order to generate the perpendicular line that will go to the border of the silhouette.

```
grayimage=rgb2gray(image);
g=histeq(grayimage,100);
[~,threshold]=edge(g,'sobel');
fudgeFactor=0.5;
BWs= edge(g,'sobel',threshold*fudgeFactor);
figure,imshow(BWs);
util_skeletonViewer(jointIndices,BWs,nSkeleton);

joints=[5,6,9,10,13,14,17,18,14,15,18,19,3,4,6,7,10,11,14,15,18,1
9];
index=1;
medidas_pixel=[];
medidas_real=[];
while index<23,

num1=joints(index);num2=joints(index+1);joI=jointIndices;joC=join
tCoordinates;
    hline=imdistline(gca,[joI(num1,1), joI(num2,1)],[joI(num1,2),
joI(num2,2)] );
    api=iptgetapi(hline);
    medida_pixel= api.getDistance();
    medida_real=sqrt(((joC(num1,1)-joC(num2,1)))^2+((joC(num1,2)-
joC(num2,2)))^2)*100;

    angle=api.getAngleFromHorizontal();
    api.delete();
    m=tand(angle);
    m_per=-(1/m);
    lista=[];
    lista2=[];
    medidas_pixel=[medidas_pixel medida_pixel];
    medidas_real=[medidas_real medida_real];
    if(joI(num1,2)<joI(num2,2))
        min_x=joI(num1,1);
        min_y=joI(num1,2);
        max_x=joI(num2,1);
        max_y=joI(num2,2);
    else
        min_x=joI(num2,1);
        min_y=joI(num2,2);
        max_x=joI(num1,1);
        max_y=joI(num1,2);
    end
```

```
        distancia_per=0; distancia_per_izq=0;
distancia_per_der=0;
       nollegar=0; nollegar2=0;
    for j=min_y:4:max_y,
            i=round((-(j-min_y)/m)+min_x);

            while nollegar==0, %controlar que uno llega y otro
no&& nollegar2==0
                    i_per_izq=i-3;
                    j_per_izq=round(-m_per*i_per_izq+m_per*i+j);
                    distancia_per_izq=sqrt((i_per_izq-
i)^2+(j_per_izq-j)^2);

                if((sum(sum(image(j_per_izq-
1:j_per_izq+1,i_per_izq-1:i_per_izq+1)))~=0)||
(distancia_per_izq>(medida_pixel/2)))
%                hline=imdistline(gca,[i_per_izq,
j_per_izq],[i,j] );
                  nollegar=1;
                end
            end
          %same for the right
distancia_per=distancia_per_izq + distancia_per_der;
             lista=[lista distancia_per];

        lista=sort(lista);
        lista=lista(1:round(length(lista)/2)); %no big values
        radio=mean(lista);
        lista2=[lista2 radio];
    end

    index=index+2;
end
sol=[];
indice=1;
 while indice<length(medidas_real)+1,

aux=(lista2(indice)/medidas_pixel(indice))*medidas_real(indice);
     sol=[sol aux];
     indice=indice+1;
 end
```

Program 3.4.13. Calculate the widths.[1]

Once the nine lengths and six widths are obtained there will be left four more measurements:

1. Chest's circumference
2. Low chest's circumference
3. Waist's circumference
4. Hip's circumference

[1] This program may not be accurate as the author did not count with Ibon Olabarria's code

These data will be found thanks to the following table:

Size	44	46	48	50	52	54	56
Chest's circumference	88	92	96	100	104	108	112
Low chest's circumference	81	85	89	93	97	101	105
Waist's circumference	78	82	86	90	94	98	102
Hip's circumference	90	94	98	102	106	110	115
Back's length	42	43	43.5	44.5	45	45.5	46

Table 3.4.2. Burda's style men's measurement table

4

Geogebra

GeoGebra is an application used for statistics, geometry, algebra and calculus by either primaries schools or universities. The desktop version is available in multiple platforms: Windows, Mac OSX, Linux, Android and Ipad.

In this project GeoGebra is the component that creates the SVG patterns with the measurements that are taken with the Kinect. In other words, GeoGebra works as the tailor of the project.

First, the patterns are made on the desktop version, following a tutorial that will be later explained. Then, the existing GeoGebra applet is introduced on a JavaScript file and the steps taken in the program are implemented by using the Geogebra commands. The structure of the HTML page will be discussed later on this section. Finally, the pattern, after escalating it to fit the client, will be exported to the SVG format and imported to Blender.

4.1. Creating Patterns

Before beign able to create a pattern that fits some exact measures there are some steps to be taken. First, a basic block has to be done. Later on, based on it a master plan is made. And finally, the plan is escalated.

As mentioned on section Introduction the tailor's ruler is needed in order to create the sleeves and the neck. In Geogebra we will be using the commands Circumcircular Arc and Conic. However, the following paragraph explains how to create this ruler in Geogebra.

To achieve it first upload a PNG image of a tailor's ruler to Geogebra: click on the arrow down the Text tool and select Image, then, select the image to be uploaded.

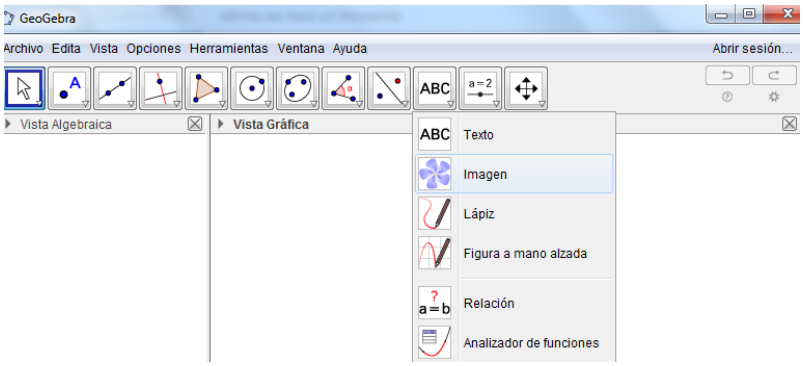


Figure 4.1.1. *Geogebra's image upload.*

Now it is on our interest to have the ruler escalated to centimeters, so by selecting the Point tool click on number zero on the ruler and then click on Segment's tool arrow to select Segment Given the Distance. Click on the point previously created and the write number one for distance, the segment will be facing the wrong direction, move the last point to the right direction and then scale the image to fit one centimeter.

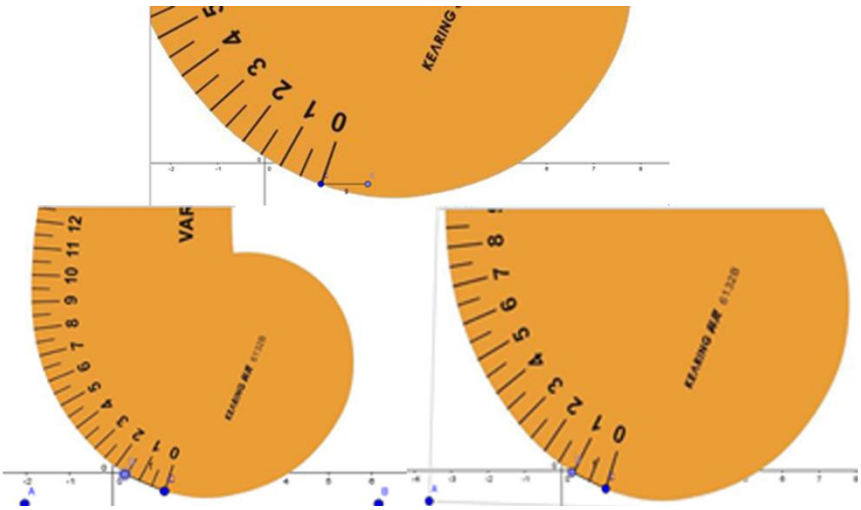


Figure 4.1.2. *Ruler uploaded.*

Continue doing it until the ruler is completed, then choose the Polygon tool and click on every point created. Close the polygon and now the ruler can be dragged around and its angle changed by using the Angle Given its Amplitude tool.

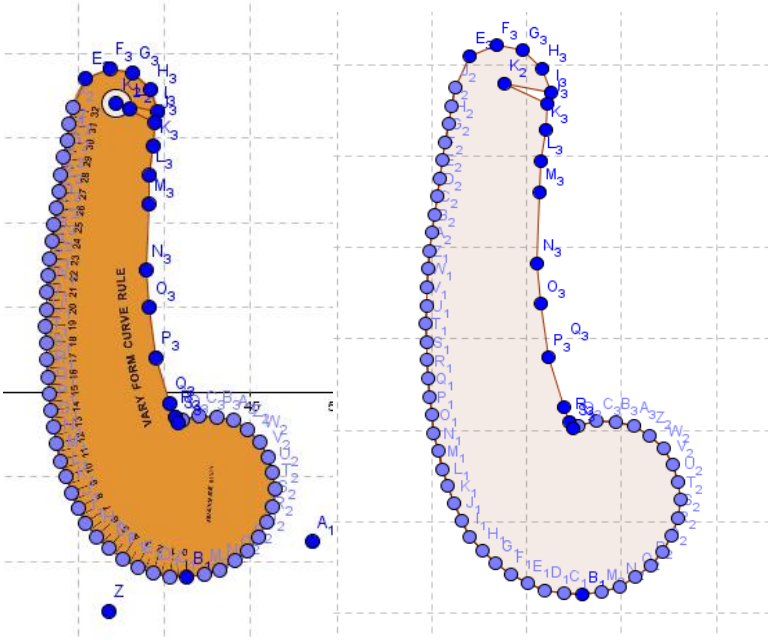


Figure 4.1.1 *Tailor's ruler created.*

Using this ruler can be quite complicated because sometimes the ruler needs to be rotate. There exits a command in Geogebra called Rotate, the problem is that this command will create a new ruler attached to the original one and can only be moved when the original is moved. This also brings as the problem of not knowing the exact angle that needs to be rotated to.

Now a tutorial that leads to the creation of a shirt and a trouser will follow. The shirt's basic block to be created is of a size 48 in Spain (38'' in Britain/USA) and the trouser's is size 40 is Spain(32'' in Britain/USA). The tutorial follows the instructions of the book 84Gareth Kershaw: *Patternmaking for Menswear,* 22nd October 2013. by Gareth Kershaw.

4.11. Basic block

4.1.1.1. Shirt

The following tables contain the measurements and calculations used for the shirt.

Measurements of the male body:

Chest Circumference	96 cm
Cloth fullness	12 cm
½ Chest Circumference	48+6.25(fullness) cm
½ Cloth fullness	6 cm
¼ Chest Circumference	24+6.25(fullness) cm
¼ Cloth fulness	3 cm

Table 4.1.1.1.1. *Male body measurements.*

Scye's depth:

Small	21 cm
Medium	22 cm
Big	23 cm
Extra Big	24 cm

Table 4.1.1.2. *Scye's depth.*

Needed Measurements:

¼ Chest Circumference	24+6.25(fullness) cm
Scye depth	22 cm
Neck's back centre to waist	57.1-10 cm
Waist to hip	20 cm
Cloth length	65.5 cm
½ Back's width	40/2 cm
Neck's base circumference	39 cm

Table 4.1.1.3. *Needed measurements.*

Once the mesaures are taken you can start creating the Cloth on Geogebra:

- First create a dot with the input bar at the bottom of the program. Write B=(0,0) and press enter, now a dot should appear on the View.
- Next step is to create point A=B+(0,67.5) and C=A-(0,2). The distance between C and B is 65.5 which corresponds to the cloth's length,calculated by adding the neck's back centre to waist and waist to hip. C will be the middle Back point of the neck.
- From C substract 22 cm to create point D (C-(0,22)), this measure depends on the cloth and arm's circunference.
- Dot E will be the Back point from neck to waist, write E=C-(0,45.5).
- Now, in order to have the chest's line, which will be F, add to D ¼ of the chest's width and ¼ of the cloth (F=D+(27,0)).
- To create the whole block add the same distance to A (creating point G), E (this will be the waist's line and dot H) and to B (beign the hip's line and point I)

- Unify these four dots to create the lateral seam.
- Write the input J=C+(8.5,0) (1/5 of the neck's base circunference plus 0.7) and create K by adding (0,2) to J.
- With the tailor's ruler created before unify K and C. In this case one command of Geogebra can also be used, create K' with the Reflect tool and then select the CircumcircularArc tool. Click on C, K and K' to create the curve.
- Subtract to C the difference between C and D divided by 2 (equals (0,11)) to create dot L.
- Create the Back point for the scye by adding (20,0), which is ½ of the back's width to L, this is point M.
- Create N by susbtracting to A 1/8 of the scyse depth minus 0.75 cm, N= A-(0,2).
- Point O is N plus (20,0)
- From O create a perpendicular line to the segment from D to F. The point that intersects is P.
- From O add (1.8,0) to create de Shoulder's back and front point, Q.
- This is now the most dificult part as Geogebra's tools are quite limited. From P we need a segment of 3 cm and in 45 degrees, in order to simulate that these steps should be followed:
    - First, create a circle from P of 3cm.
    - Now by using the Segment tool create one from O to P in order to then, using the Reflect command, create N'.
    - Finally using again the Segment command create one from N' to P, the intersection between this segment and the Circle (the one outside the pattern) is the point we were looking for.
- Create the Front dot Scye by adding to P (0,8).
- Now unify with the Conic tool dot Q with the Back and Front points of the Scye, with the Point at 3 cm and 45 degrees from and F.

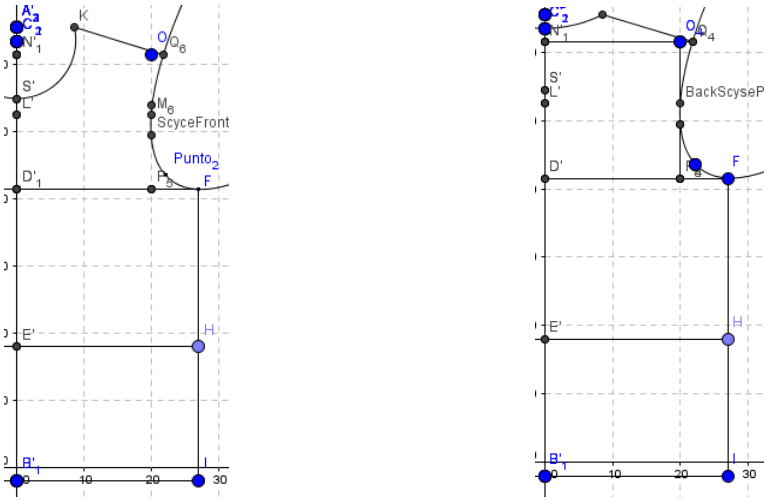


Figure 4.1.1.1 *Shirt's basic block.*

- Lastly to create the Front point of the neck subtract to C 1/5 of the neck's circumference minus 0.8 cm(R= C-(0,7.8)+(0,0.8)). And, like it was done with the Back point, unify K and K' with R.

4.1.1.2. Sleeve

Needed Measurements:

Scye	49+2.5 (fullness)=51.5 cm
Length of sleeve to wrist	70 cm
Length from crown (shoulder's top) to elbow	41 cm
Circumference of the top of the biceps	35.5 cm

Tablee 4.1.1.2.1 *Sleeves measurements.*

To create the sleeve that matches the previous shirt, follow this tutorial:

- Points A=(0,70) and B=(0,0) form the length of the sleeve.
- Add to A the circumference of the biceps, C=A+(35.5,0).
- Create a rectangle using points A, B and C.
- Divide the width of the rectangle by 4. The lines are, starting from the left: Scye's Back Seam, Back's Line, Middle Line, Front Line, Scye's Front Seam. The intersection between A-C's segment and the Middle Line is D.
- E is A minus 1/3 of the scye (0,17.1) and the back point of it.
- The front point of the scye, F, is calculated by adding (35.5,0) to E.
- Point G is the intersection of the Back's Line with the Segment from E to F. H is that Segment's intersection with the Forearm Line.
- HoleA will be the back's drape of the sleeve, which is placed (0,10) on top of G. The distance is 1/6 of the scye's size plus 1.5.
- HoleB, which is the front's drape of the sleeve, is 1/6 of the scye's sleeve (8.5) to the top of H.
- Unify HoleA and E with a line. With the Tool MidPoint create the line's middle point. Now create a Circle with the middle point as the centre and of 0.5 cm radius. Connect G with the middle Point and using the Intersect tool select the circle and the segment. Finally, unify this point with HoleA and E with the CircumcularArc tool.
- From D to HoleA do the same. In this case the circle is of 1.5 cm and the unification has to be done between the middle point and one of the Points that separates the patter in form (creating an angle of 90 degrees to the segment D-HoleA).
- These steps have to be repeated for F and HoleB. The radius of the circle is of 1 cm and the Point to unify the centre with is H.
- Lastly, follow the same steps for D and HoleB. This time the radius is 2 cm and the point to connect the middle point to is another one that divides the pattern in 4 and creates a 90 degree angle. Now unify D, HoleA, HoleB and both of the resulting points with the Conic command, this is the head of the sleeve.

- Divide by 2 the sleeve's length from the bottom of the sleeve's head. This is achieved by subtracting to the length of the sleeve the height of its head (70-17.1=52.9). This is also the difference between E and B. Then divide it by two (26.45 cm). Now create Aux by subtracting to E (0,26.45) and adding (0,2.5). Do the same to F in order to create Aux2. By connecting these two dots you will have the Elbow´s line.
- The Crown's point is place 0.5 cm to D's right.
- Give form to the seams. Create point R=B+(2,0), now with the Symmetry tool create R'. Join E with R and F with R'.
- Give form to the turn-up of the sleeve to incorporate the cuff. Create the middle point (AUXR) from R to R', now find the middle point (AUXMR) from AUXR to R and AUXR and R' (AUXM2R). From both middle points create circles of radius 0.5 cm and a perpendicular line from them to the segment they are on. Now, the intersections with the circles and the perpendicular lines create the new points to be connected through an arc with their respective AuxR and R (following the image below).

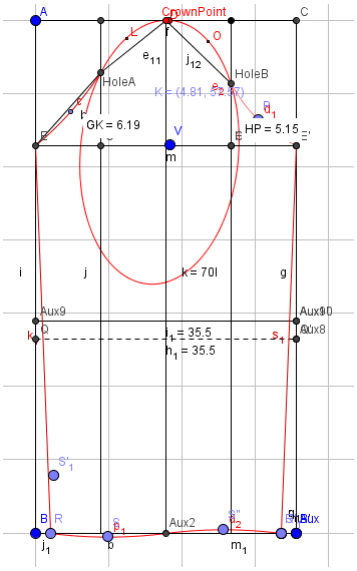


Figure 4.1.1.2.1 *Sleeve created.*

4.1.1.3. Trousers

Needed Measurements:

Waist Circunference	81+3 (fullness) cm
Hip Circunference	96+10 (fullness) cm
Front Hip	48 cm
Back Hip	58 cm
Inner Thigh Depth	26 cm
Exterior Leg Measurement	107 cm
Interior Leg Measurement	81 cm

Table 4.1.1.3.1 *measurements for the trousers.*

The frame for the basic block should be drawn first;

- Point A will be on coordinates (29,107) and B on A-(0,26), the inner thigh depth.
- C will be B minus the Interior Leg Measurement (0,81).
- Dot D will be at 17 cm from A (D=A-(0,17)).
- To create the Hip Line, ED, subtract to D ¼ of de Waist Circumference minus 2.5 cm (E=D-(24,0)).
- The Inner Thigh Line, FB, is calculated by subtracting to B the same number as before.
- Now subtract the same to A and unify A to the resulting point, G, to make the Centre Line.
- Connect G with E and F to have the Front Centre.
- Now H equals F-(5,0) which is ¼ of the difference between D and A.
- To finish the frame subtract to H (0,81), creating I. Now connect I to H, this will be the Interior Leg Seam, and to C, creating the turn-up.

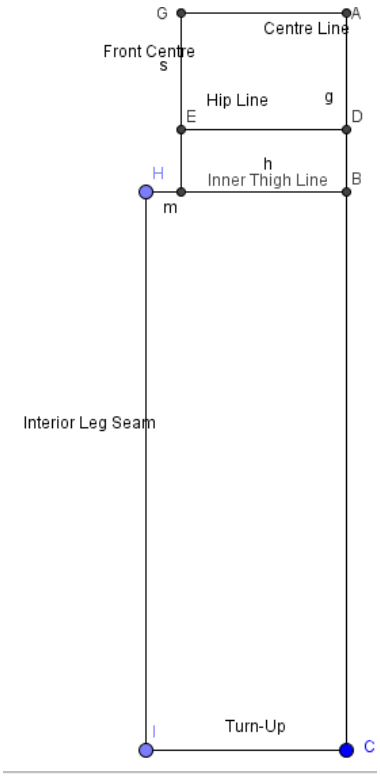


Figure 4.1.1.3.1 *Trousers' basic block.*

Front part:

- Create J=G+(9,0), then make a Segment from J with a given length of 107 cm. The point that intersects to Segment I-C is N.
- To create the fold we need to make point K=J+(4,0), the segment from J to K is the fold.
- Subtract to the rest of the Waist's Length 1.5 cm and add it to K (L=K+(9.5,0)). Connect L, D and B with the CircumcircularArc command.

- From F we need to create a 45 degree angle with H, then a segment of 3 cm whose dot, O, will have to be dragged until it forms the same angle. Now create the reflection of O and E, using a perpendicular line from H, and connect them with O, H and E with the Conic command to make the Front rise centre.
- The Intersection of F-B with J-N is M.
- To situate the Knee Line you will need to find dot P, which is equal to (M-N)/2 + (0,8), the distance between M and N is the length of the Interior Leg. Now create AuxL and AuxR. AuxL=P-(11,0), AuxR=P+(12,0). The line Between AuxL and AuxR is where the Knee line is.
- Finally create AuxL2 and Auxr2: AuxL2=N-(11,0) and AuxR2=N+(11,0). Unify H with AuxL and AuxL2 and connect B with AuxR and AuxR2. Both of those connections are done with the tool CircumcircularArc.

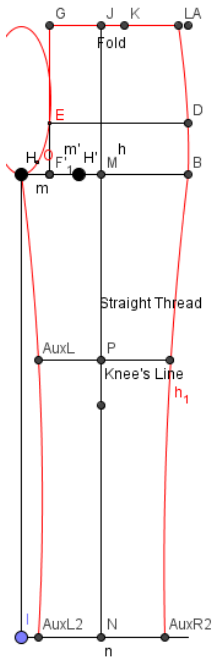


Figure 4.1.1.3.2 *Trousers' front part.*

Back part:

Made by copying the Front part and adding some modifications.

- Q is G plus (0,5).
- From A measure 5 cm to the right and call it R.
- Subtract 1.5 cm to H from the Y Axis and name it S
- Now add 29 cm to the right of S and label it as T
- Create the Back Spot of the Inner Thigh by subtracting 9 cm to the left of S, U=S-(9,0).
- To Q's right add 4.5 cm and connect it to R. This is the Back measure for the waist.
- Create a 14 cm radius circle with centre V and name AUX the intersection between it and segment VR. This is the centre of the clothespin, AUX=V+(14,0). Now create

W by creating a circle with centre AUX and radius 7.5 and a line perpendicular to VR from AUX. The intersection is W.

On the line between V and R create Aux_1 and Aux_2 by, as previously done, creating a circle of centre AUX and radius 1 that intersects on those points with VR. Then, create AUXR= Aux_1+(0,0.4) and AUXL= Aux_2+(0,0.4) to finally connect both with W and V.

- From F subtract 1.5 to the bottom and label it D_1.
- Add to E's top 3 cm, X=E+(0,3).
- Now from D_1 create a circle of radius 5.5 cm and a parallel line to OF. Finally connect dots V, U, the point created, the reflection of the created point and X with the conic command. This curve will be the Back rise centre.
-
- To end the pattern cerate AUXPL, AUXPR, AUXNL and AUXNR: AUXPL=P-(13.5,0), AUXPR=P+(13.8,0), AUXNL=N-(12.5,0) and AUXNR=N+(13.5,0). From U create an arc going from U to AUXPL and AUXNL. Now create G_1 by adding (2,0) to T, connect ,the same way as before, R, G_1 and AUXPR. Finally, connect with a Segment AUXPR and AUXNR.

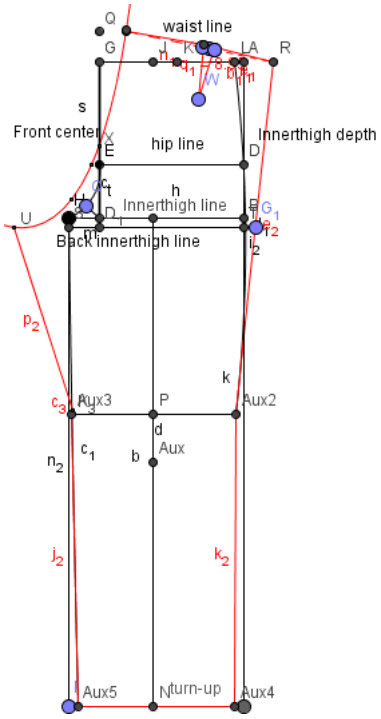


Figure 4.1.1.3.1 *Trousers' back part.*

4.1.2. Master Plan

The Master Plan consists on eliminating the lines that form a rectangle containing the blocks and leaving just the names of the lines.

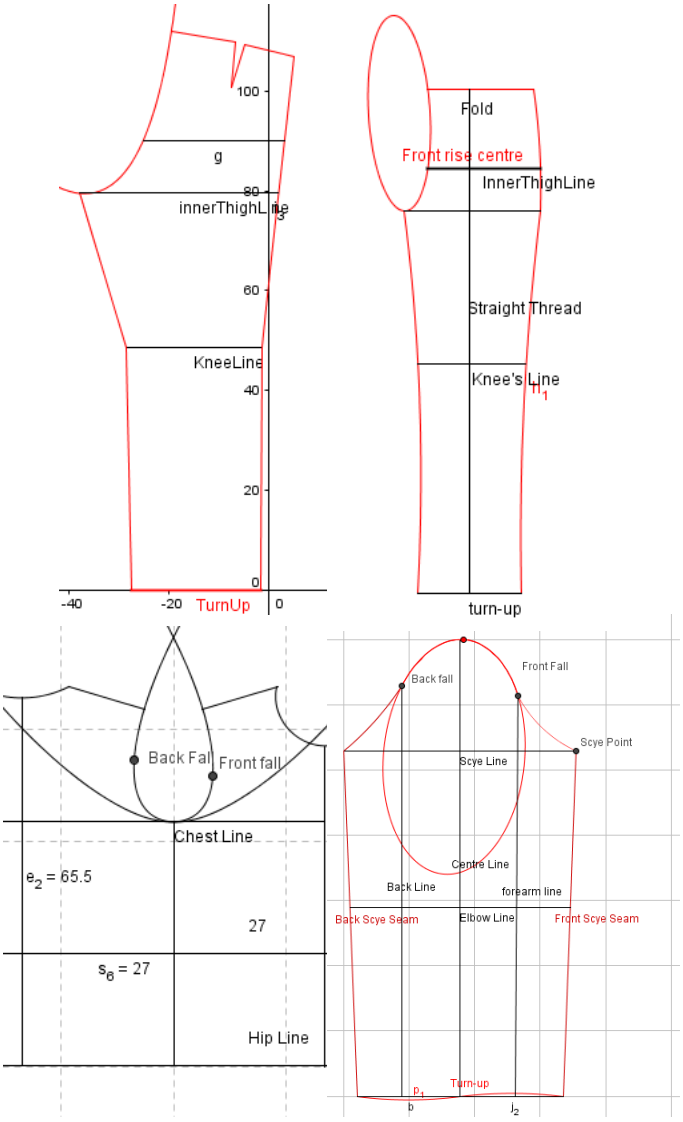


Figure 4.3.2.1. *Complete Master Plan.*

4.2. Geogebra-Web

This project's final result is a Web Page containing the above shirt and trousers, the user and several other options based on them. Those options include the customization of the patterns by filling a form and, also, being able to choose predefined sizes of patterns. These last patterns follow the Spanish sizes.

This webpage uses the Geogebra commands and for that Geogebra's JavaScript library was used. In order to do that the following lines have to be added to the code:

```
   <script                type="text/javascript"        language="javascript"
src="http://www.geogebra.org/web/5.0/web/web.nocache.js"></script>
```

Program 4.4.1. Geogebra's library.

Now that the commands can be accessed the drawing has to be shown somewhere in the webpage. First, open the Geogebra application and hide both the axis and the grid. Next, export it as an HTML, this will publish it in Geogebra's forum. There will be an option to embed it to a webpage; the code to be used is the one in the Media Wiki section after "ggBase64". This should be copied onto the following code:

```
   article class="geogebraweb" data-param-width="700" data-param-
height="700"
   data-param-showResetIcon="false" data-param-enableLabelDrags="false"
data-param-showMenuBar="false" allowRescaling="true"
    data-param-showAlgebraInput="false" enableLabelDrags="true"
   data-param-ggbbase64=" TO BE COMPLETED">
```

Program 4.4.2. Geogebra's applet.

After creating the container of Geogebra's applet the object needs to be initialized:

```
   var applet = document.ggbApplet;
```

Program 4.4.3. Geogebra's object.

The following code lines contain some of the commands used in the webpage to replicate de work done on the previous section:

```
   applet.setAxesVisible(false,false); //Show/Hide the axes
   applet.setGridVisible(false); //Show/hidethe Grid
   applet.reset();//delete previous objects
   applet.setCoordSystem(-120, 240, -120, 240)  //Show this View in the applet
xmin, xmax, ymin and ymax
   applet.evalCommand("AuxL2 = (0,120)"); //create a point
   Auxl2tX=applet.getXcoord("AuxL2'")+Auxl2tX;
   Auxl2tY=applet.getYcoord("AuxL2'")+Auxl2tY;
   applet.setCoords("AuxL2",Auxl2tX,Auxl2tY); //change a point coordinates
   applet.setVisible("c2",false); //Hide/show an object
   applet.setColor("seg9",255,0,0); //change the objects colour (RGB)
   applet.evalCommand("KneeLine=  Segment[AuxL,AuxR]");  //Create  a  segment
between points
   applet.evalCommand("a_1=CircumcircularArc[H,AuxL,AuxL2]");   //Create   an
arc that goes through 3 points
   applet.evalCommand("pf=PerpendicularLine[Hf,helperf]");//create         a
perpendicular line to helper that pases through Hf
   applet.evalCommand("Reflect[Ef,pf]");//create a reflection of Ef from pf
   applet.evalCommand("csf=Conic[Ef',Of',Hf,Of,  Ef]");  //create  an  arc  that
goes through 5 points
   applet.evalCommand("p4=Line[AUXB,BackWaist]");//parallel   line   to   the
line from the point
   applet.evalCommand("Waux=Intersect[p4,c8]");//  create  the  points  that
intersect both objects
   applet.evalCommand("AUXM2R=Midpoint[seg13]");//create the segment's midpoint
```

Program 4.4.4. Commands.

4.2.1 Predefined patterns

When creating the predefined patterns the basic block of each piece of clothing was replicated by using the command "applet.evalCommand("Step to be done")". These basic blocks, as previously stated, corresponds to size 48 and 40 of a shirt and a pair of trousers. What was left was to escalate them accordingly to the Spanish sizing.

4.2.1.1 Shirt scaling

Scaling a pattern is done by adding or subtracting the same numbers to the previous size. In order to escalate from size 48 to a 50 the following steps have to be done:

- Back of the shirt using as reference its left side:
    - Add to coordinate Y of the center back of the neck 0.8 cm.
    - Add to the point of the neck between the shoulders 0.8 cm in coordinate Y and subtract to its coordinate X 0.25 cm.
    - To the point on the shoulders add 0.6 cm vertically and subtract 0.5 cm horizontally.
    - In Back fall point add 0.3 cm to Y and subtract 0.5 cm to X.
    - For the scye's point subtract 0.9 horizontally.
    - To the point in the waist add 0.4 cm to its Y and subtract to its x 0.9 cm.
    - To the point in the turn-up subtract 0.4 cm to Y and 0.9 cm to X.
    - In the back center of the turn-up subtract 0.4 cm vertically.

    Do the same with the symmetric points but this time instead of subtracting to X add to it.

- Front of the shirt using as reference its left side:
    - Add to coordinate Y of the center front of the neck 0.6 cm.
    - Add to the point of the neck between the shoulders 0.8 cm in coordinate Y and subtract to its coordinate X 0.25 cm.
    - To the point on the shoulders add 0.7 cm vertically and subtract 0.5 cm horizontally.
    - In Front fall point add 0.35 cm to Y and subtract 0.5 cm to X.
    - For the scye's point subtract 1.1 horizontally.
    - To the point in the waist add 0.4 cm to its Y and subtract to its x 1.1 cm.
    - To the point in the turn-up subtract 0.4 cm to Y and 1.1 cm to X.
    - In the back center of the turn-up subtract 0.4 cm vertically.

    Do the same with the symmetric points but this time instead of subtracting to X add to it.

- Sleeve:
    - In the left's turn up point subtract 0.6 cm to X and 0.3 cm to Y.
    - In the right's turn up point add 0.6 cm to X and subtract 0.3 cm to Y.
    - In the left's scye's line point subtract 0.8 cm to X.
    - In the right's scye's line point add 0.8 cm to X.

- In the front fall add 0.4 cm to X.
- In the back fall add 0.3 cm to Y and to X subtract 0.2 cm.
- In the crown's point add 0.6 cm to Y.

4.2.1.2. Trouser scaling

- Front trouser:
  - In the left's fold point add to Y 0.4 cm and subtract to X 0.35 cm.
  - In the left's point were the hip line meets add 0.1 cm to Y and subtract 0.35 cm to X.
  - In the left's inner thigh line subtract 0.6 cm to X.
  - In the left's knee line subtract to X 0.25 cm and to Y 0.5 cm.
  - In the right's knee line add to X 0.25 cm and subtract to Y 0.5 cm.
  - In the left's turn-up point subtract to Y 1 cm and to X 0.25 cm.
  - In the right's turn-up point subtract to Y 1 cm and add to X 0.25 cm.
  - In the right's inner thigh line add 0.65 cm to X and 0.1 cm to Y.
  - In the right's point were the hip line meets add 0.1 cm to Y and subtract 0.65 cm to X.
  - In the right's fold point add to Y 0.4 cm and to X 0.65 cm.

  Point O on the basic block has to be recalculated too, add to Y 0,1 cm and to X subtract 0.35 cm.

- Back trouser
  - In the left's fold point add to Y 0.4 cm and subtract to X 0.3 cm.
  - In the left's point were the hip line meets add 0.1 cm to Y and subtract 0.3 cm to X.
  - In the left's inner thigh line subtract 0.7 cm to X.
  - In the left's knee line subtract to X 0.25 cm and to Y 0.5 cm.
  - In the right's knee line add to X 0.25 cm and subtract to Y 0.5 cm.
  - In the left's turn-up point subtract to Y 1 cm and to X 0.25 cm.
  - In the right's turn-up point subtract to Y 1 cm and add to X 0.25 cm.
  - In the right's inner thigh line add 0.65 cm to X and 0.1 cm to Y.
  - In the right's point were the hip line meets add 0.1 cm to Y and subtract 0.7 cm to X.
  - In the right's fold point add to Y 0.4 cm and to X 0.7 cm.
  - At the point of the clothespin add to Y 0.4 cm and to X 0.25 cm.
  - At the left's point of the clothespin add to Y 0.4 cm and to X 0.25 cm.
  - At the right's point of the clothespin add to Y 0.4 cm and to X 0.25 cm.

4.2.1.3. Javascript

In order to emulate the escalation in the webpage a button for each size was created. Clicking on the button will call to a function that contains auxiliary variables that will be added to the basic block depending on the size.

```
  function shirt46(){
    //front
    BauxY=0.4;
    IauxX=-1.1;
    HauxY=-0.4;
    HauxX=-1.1;
    FauxX=-1.1;
    …
    //Back those of the front that do not appear below are the same values
    I2auxX=-0.9;
    H2auxX=-0.9;
    F2auxX=-0.9;
    Q2auxY=-0.6;
    CauxY=-0.8;
    …
    Shirt();//calling the function of the basic block
  }
  …
  //show how they were applied
  applet.evalCommand("B = (0,0)"); //first create the point
  BauxY=applet.getYcoord("B")+BauxY; //escalate the basic pattern
  applet.setCoords("B",0,BauxY);
```

Program 4.4.1.1.1. Escalating the pattern.

4.2.2 Customized patterns

In order to make the patterns the needed measurements that were used in the tutorial to create the pattern in Geogebra were used as variables. The user fills in the form and the function that the submit button has recollects them so that later the pattern is calculated taking them into account.

```
<div id="forms" style="display:none;">
        Please submit the following measurements:<br>
          Chest Circumference: <input type="number" id="chest"><br>
          Neck's back centre to waist: <input type="number"
id="neckWaist"><br>
            …//get all the measurements
         <button type="button" onclick="validateForm()">Submit</button>
      </div>
```

Program 4.2.2.1. Form to be filled.

```
function validateForm2() {
    var Waistc = document.getElementById("Waistc").value;
.. //get all the measurements into variables
   if( Interiorleg=="" … )
    {// ask for it to fill it
    }
    else
trousersC(Hipc,Waistc,FrontHip,BackHip,Innerthighd,Exteriorleg,Interiorleg)
;
```

Program 4.2.2.2. Function form.

```
function
sleeveC(applet,chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,
scyesdepth,scyeM){
  applet.evalCommand("Bs = (0,140)");
  applet.evalCommand("As=(0,0)");
   applet.setCoords("As", applet.getXcoord("Bs")
 ,applet.getYcoord("Bs")+parseInt(arm));
…//calculate the rest
}
```

Program 4.2.2.3. Calculate the customized pattern.

4.2.3 PDF creation

Once the pattern is created the idea is to be able to print it in natural scale. First, generating the pattern in Javascript was considered but the units of Geogebra did not allow the scale to be the correct one. However, in the future it may be possible to create it with this method:

It does not exist a Geogebra function that converts the content in Geogebra's applet into a PDF; fortunately, it does have two functions that convert the whole screenshot to PNG:

1. applet.writePNGtoFile("myImage.png", 10, false, 72);

   This function saves the PNG on the computer. The parameters are: The name of the file to be saved, the conversion of the units or to how many centimetres will one unit be converted to, if it is going to be transparent and the DPI (which is currently ignored by html5).

2. String= applet.getPNGBase64(1, false,72);

   This one creates a base64-encoded string containing the PNG image. The parameters are the same as before, except for the name of the file.

In the program the second function was used. Now we need to be able to convert the PNG image to a PDF file. After some time we stumbled across PDFJs which is JavaScript library intended to render PDF files on HTML. First you will need to download and add the library to your code, along with the libraries that support PNG:

```
<script type="text/javascript" src="jsPDF-
master/libs/png_support/zlib.js"></script>
<script type="text/javascript" src="jsPDF-
master/libs/png_support/png.js"></script>
<script type="text/javascript" src="jsPDF-master/jspdf.js"></script>
<script type="text/javascript" src="jsPDF-
master/jspdf.plugin.png_support.js"></script>
<script type="text/javascript" src="jsPDF-
master/jspdf.plugin.addimage.js"></script>
<script type="text/javascript" src="jsPDF-
master/libs/FileSaver.js/FileSaver.js"></script>
```

Program 4.2.3.1. Adding PDF.js.

After doing it you will be able to use the functions implemented in the library. In this project a button was enabled so that the PDF would be created. This button called the function shown below:

```
function writePNG(){
   var applet = document.ggbApplet;
   var imgData =''+ applet.getPNGBase64(1, false,72);
   var pdf = new jsPDF("p", "cm", "a4");
   pdf.addImage(imgData, 'PNG', 0, 0);
   pdf.save("Pattern.pdf");
}
```

Program 4.2.3.2. Creating PDF.js.

First you need to transform the PNG image to a data image. For that add to the string created by Geogebra the following: “data:image/png;base64” Where you are stating that it contains a PNG image encoded in base64.

Secondly, create a new PDF and enter the value to the following parameters:

- Page orientation: “p” for portrait, which is the default option, and “l” for landscape.
- Unit: Tell PDF.js in which unit we want to work, centimeters “cm”, millimeters ”mm” (default value), inches “in” or points “pt”.
- Page format: The default is “a4” but it can also be “a3”,”a2”,”a5”, “letter” or “legal”.

As mentioned before, this was not the actual method. Now the client downloads either the customized code or the selected size code and opens it in Geogebra. Then exports it to PDF, with one unit to one centimetre, and prints it with the Poster option at 100%.

5

Blender & MakeHuman

Blender is used in this project to transform the SVG pattern, previously generated on Geogebra, to clothes and fit them on to the mannequin created on MakeHuman. This was possible thank to the Python Script generated with all the integrated necessary commands on Blender.

In this project MakeHuman will be used to import mannequins to Blender and help to create the clothes. In section Improvement proposal it is explained how makeHuman would be of great use in the future of the project. Ibon Olabarria's project goes more into details about how MakeHuman works.

5.1. Creating clothes with MakeClothes

Now I will explain how to create clothes on Blender in order to import it to MakeHuman. Open Blender and delete the default scene, then load the mesh: Average Female with Helpers situated in the tools panel at MakeClothes.

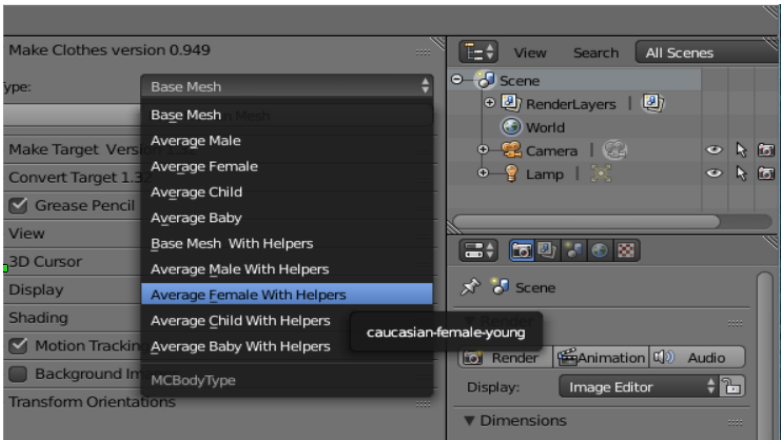


Figure 5.1.1. Load Average Female with helpers.

This mesh is a standard mannequin of a woman, the body's mesh and other objects. The objects on the human body are: Body, Eyes, Tongue, Teeth, Tights, Skirt and Hair.

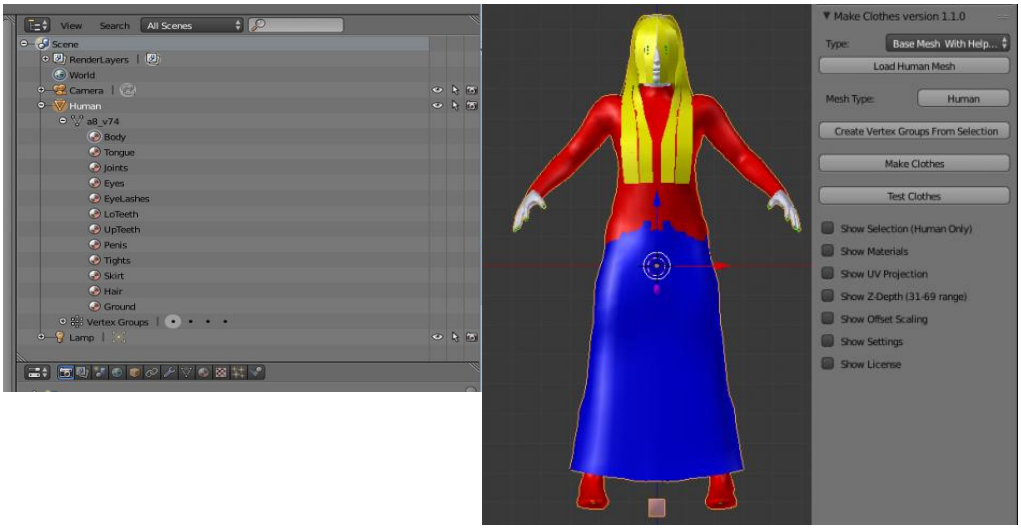


Figure 5.1.2. Average Female with helpers loaded.

The helpers on the mesh are invisible geometries over the mesh that helps to create an object that will adapt to the body.

Now on Edit mode, click on the bottom of the scene, the objects Pennis and Hair have to disappear. By selecting on material, at the right editor, all the objects can be found, each one with its own material.

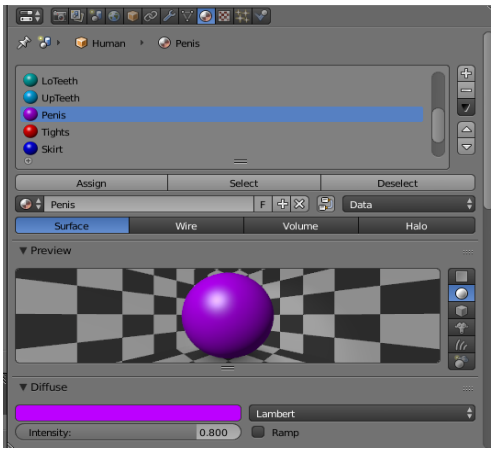


Figure 5.1.3. Penis material selected.

Selecting the objects previously mentioned (one by one) and by hitting the select button modifying them will be available. Once that is done scaling it will be possible by clicking on the scene with the mouse and pressing the S button (scale). After making it as smaller as possible on the materials window the object must be deselected before selecting the next one.

Afterwards the material Tights should be selected as this are indeed the mesh helpers. By hitting Shift and D at the same time the material will be duplicated, as it is of no interest to move the duplicate the right mouse button must be hit. Now both the copy and the original object are together, for separating it key P should be pressed and then the option selection chosen.

This copy will now be named Human.001, it is recommended to select it and change the name, in this case, to dress. It is also best to move it to another layer by pressing M and choosing the layer.

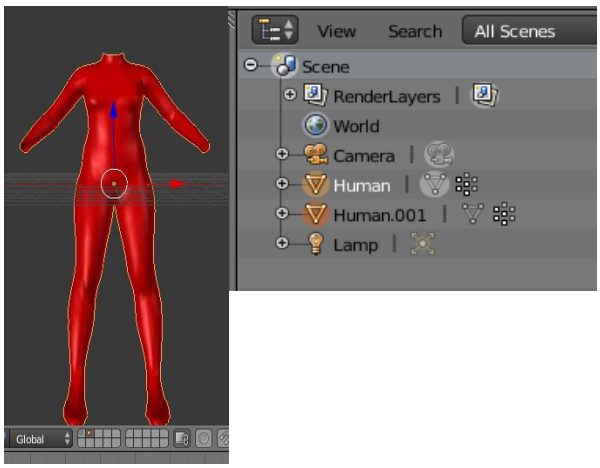


Figure 5.1.4. Thights copied.

Important: the mesh can be deformed or cut, but all the elements that coins must be four-sided (four vertex and four sites). If this does not happen it would not adjust to the body and it would be impossible to create the clothing.

Still on edit mode and on the layer where the Tights are located pressing A will select them. Now the editing starts choose a vertex by left clicking on it. Continue adding vertexes with Ctrl and '+' until the area to be cut is selected. Then by pressing Supr and selecting the option Vertex these will be cut. The vertex ca also be selected by holding Shift while selecting them but it is important to be careful as no vertex should be left alone.

Once the cutting part is done it is possible to shrink or fatten, for example, the sleeves. For that the circle at the end of the sleeve should be selected and expanded by pressing S (Shrink-fatten). To stop it right clicking on the mouse will leave the changes.

This is the result of transforming the tights into a blouse with this method:

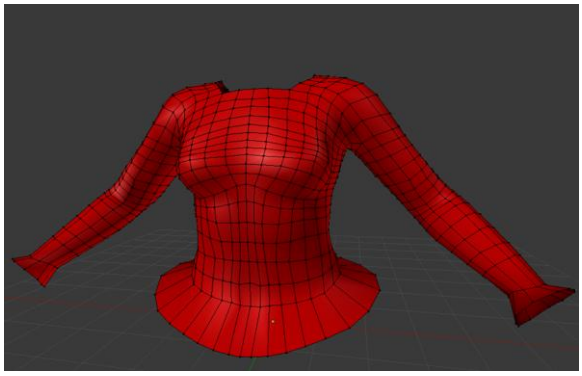


Figure 5.1.5. Thighsinto a blouse.

The next step would be to apply texture to the cloth and that is the reason for the UV editor. Once that editing is selected the view of the cloth (on the right side) should be adjusted.

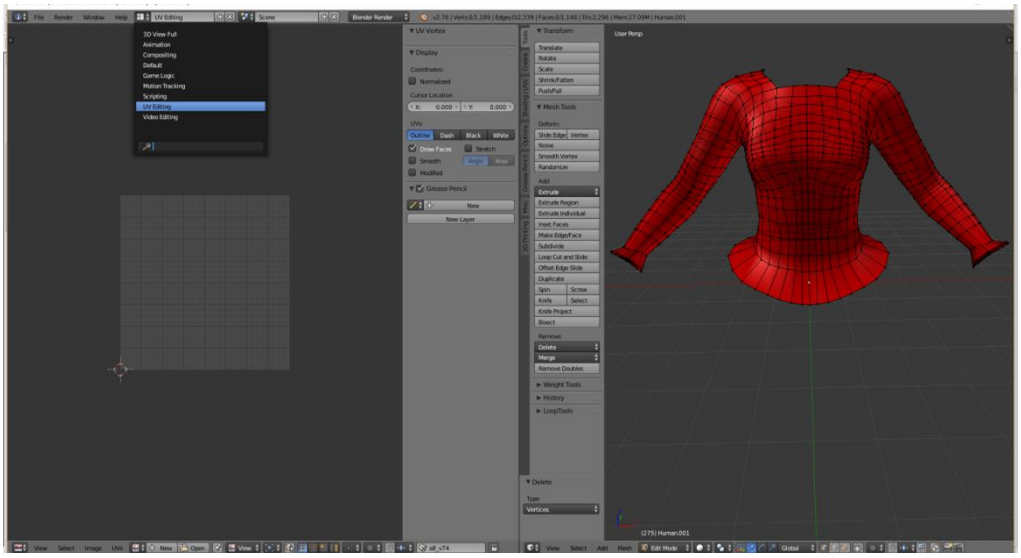


Figure 5.1.6. UV editor.

On the left screen is where the texture will be applied. Select new image and change the name of it.

Figure 5.1.7. Create texture 1.

Now the option of selecting an image will appear. Once the texture is selected it will be shown at the left screen.

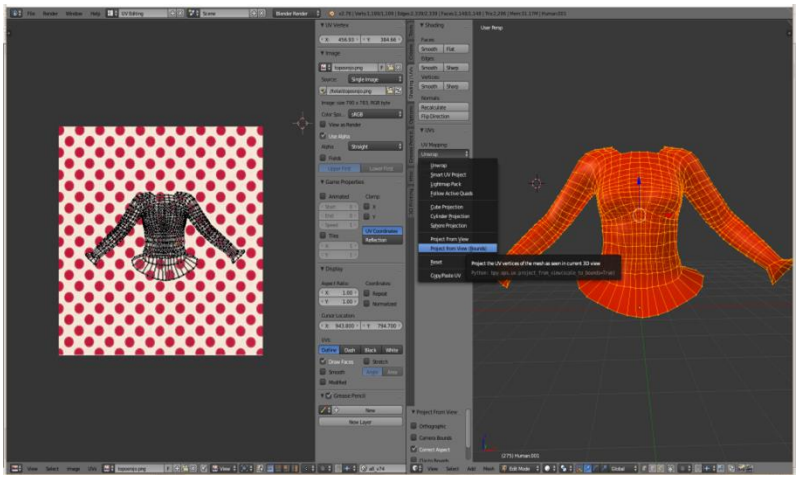


Figure 5.1.8. Create texture 2.

By selecting on the right editor the Shading/Uvs tab, and clicking A to select all the mesh, the options to see the result on the cloth will appear. On the option Unwraps→Project from view (Bounds) the mesh will appear on top of the texture at the left. Then, still on the right scene, click on '+' and select Texture solid from the Shading tab. This will apply the texture to the cloth.

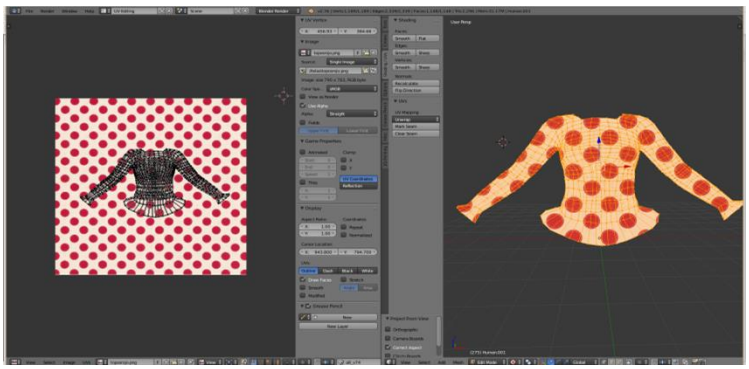


Figure 5.1.9. Create texture 3.

Go back to the Default view and to Object mode. The texture will disappear, to get it back select on the Shading texture at the bottom.

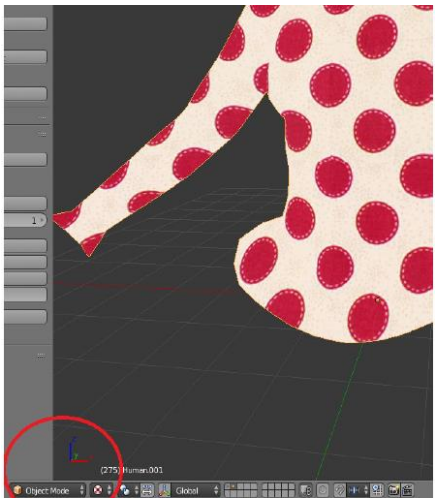


Figure 5.1.10. See texture.

To try the cloth the layer where the whole body is should be visible, for that both layers should be selected while holding down the Shift key.

When the Shading texture option is once again selected it can be seen that the blouse is smaller than the body. By pressing A and S the blouse can be fatten in order to fix it.

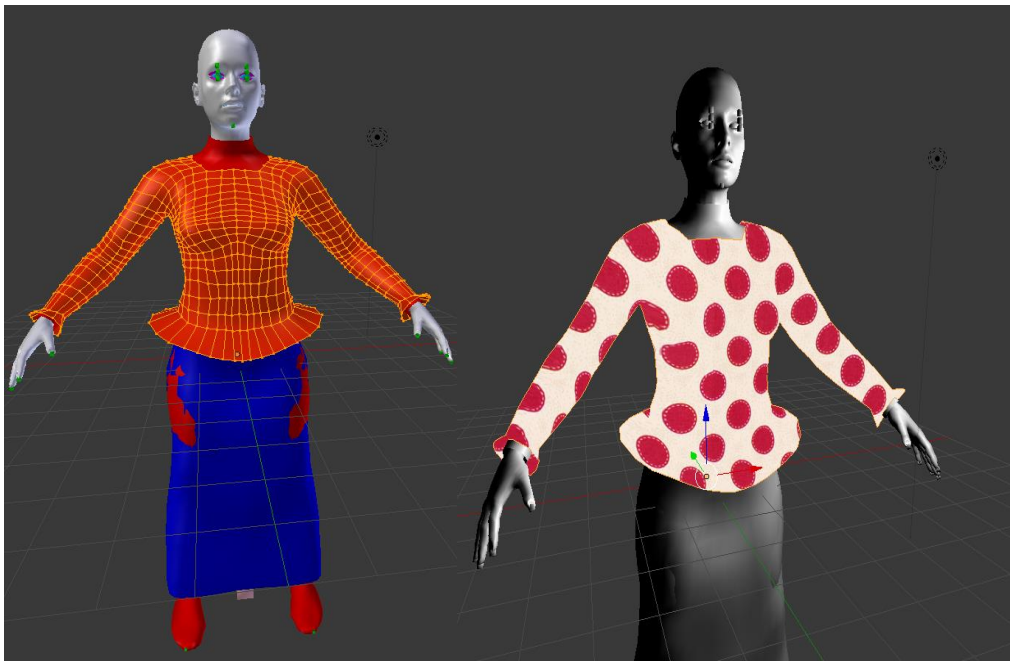


Figure 5.1.11. Try blouse.

The next problem to be solved is how the texture was applied. The pattern doesn't adjust to the blouse as it should. Going on Edit mode and UV editing again plus selecting the Cube Projection at the Shading/UVs tab will make the texture look plain.

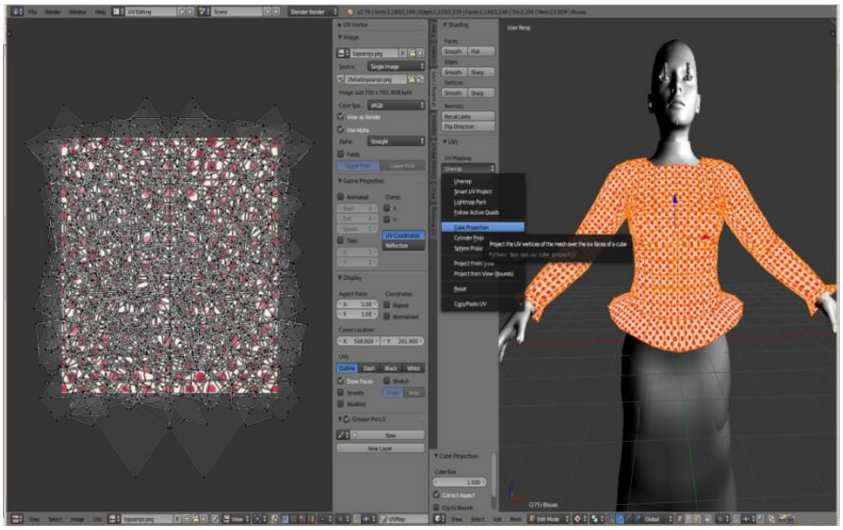


Figure 5.1.12. Cube projection.

This technique is called UV Mapping which is used to make the texture match the sides of an object. It is like cutting the object to lay flat. The texture may change on various parts of the objects body and needs to be mapped correctly.

Finally the last steps to create the cloth will be done:

a) Erase the helpers: The blouse contains several mesh elements that have to be erased. On the layers hide the mannequin so that we can only see the blouse. Now every material, except for the one that applies to the texture, should be deleted by pressing '-'.

b) Create the mannequin group vertex: Both the blouse and the mannequin have group of vertex that should coincide. For example, selecting the right part of the blouse as in the image. Deselect the right part and select the Mid, central line of the cloth. On Edit mode and solid method create the vertex groups which appear on the '+' panel under the MakeHuman tab.

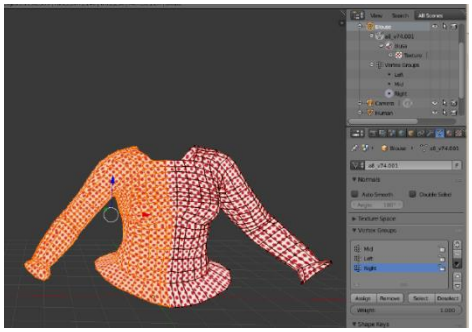


Figure 5.1.13. Select right group.

On the layers only show the mannequin and delete its three vertex groups. Once they are deleted, and with the mannequin selected, once again open the '+' panel and in this case select the button Create Vertex Groups From Selection. These groups are the same as the cloths.

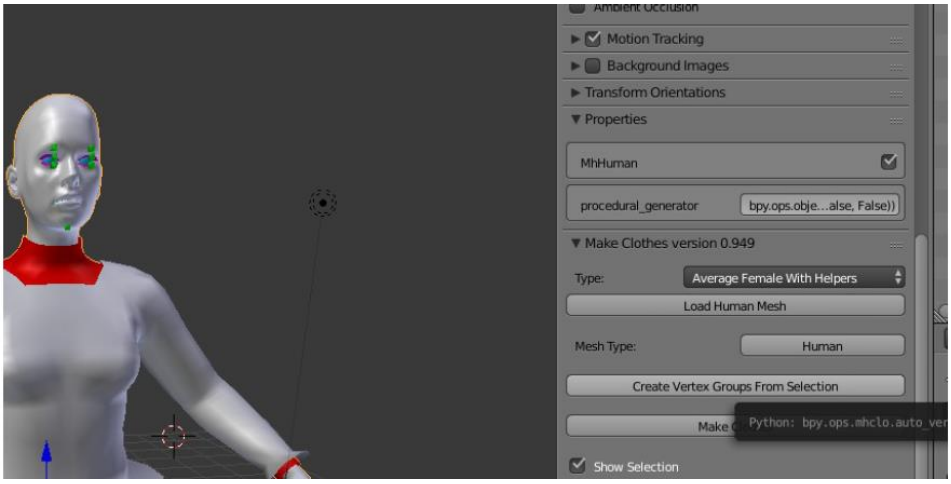


Figure 5.1.14. Create vertex Groups.

Later on the vertex group name can be changed to .blouse so that it is the same as the mannequin's. Now once both the blouse and the mannequin are selected the cloth will be created by pressing the Make Clothes button.

Figure 5.1.15. Make clothes.

The cloth is saved in the MAKEHUMAN folder, which appears on Documents once MakeHuman is installed, and can be opened on MakeHuman.

5.2. From SVG to Cloth

This was the first idea to create the cloth from the pattern; unfortunately, it did not work as Blender kept eliminating vertex needed to pin together the cloth.

Even though it did not work it will be explained as in the future it could come in handy. First select Import SVG to Blender in Edit and choose the pattern previously generated in GeoGebra. Once the pattern is in the program select the lines one by one, with the right mouse button and pressing shift, until the whole piece is completed. To join them press CTRL and J at the same time.

Now use ALT+C, in Object Mode, and select the option “convert from curve to mesh”. This creates the mesh from the piece and now allows us to give it the property of clothes, as seen in the image below.

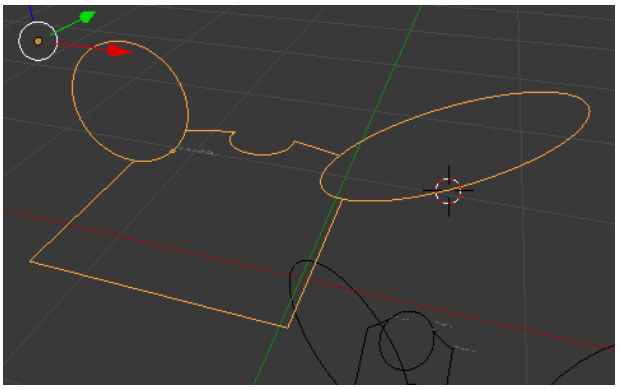


Figure 5.2.1. From SVG to mesh.

Now eliminate the vertexes from the conic forms the same way as it was done on the tutorial before.

Figure 5.2.2. Delete innecessary vertexes.

Once all the parts that create the shirt (front, back parts and two sleeves) are imported and the meshes are created all of them should be rotated 90 degrees in the X axis. Then rotate the Sleeve’s Z coordinate to 45 degrees. Afterwards they need to be aligned, press ALT, Shift, Ctrl and C and move the origin to the center of the geometry. Then, press the Spacebar, type Align objects and choose positive sides. Finally, align axis X, Y and Z.

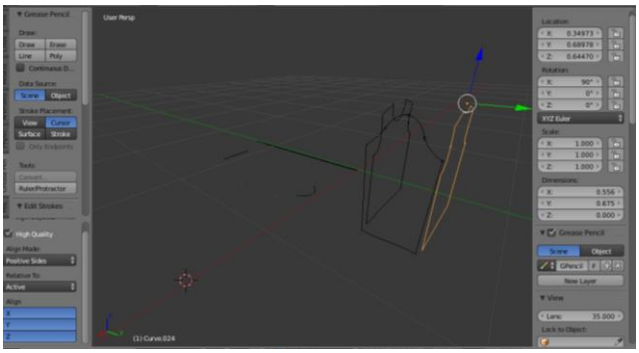


Figure 5.2.3. Pieces aligned.

Now here a problem arises, as the SVG was formed by lines and curves the mesh cannot be fully filled. When on edit mode and with the object selected press F repeatedly, in some moment its stops filling the faces and some are left empty. That si why the SVG has to be redone in Geogebra by using the Polygon tool and connecting all the dots. After exporting it as an SVG with the proper units (10 units to 1 cm), import it in Blender and redo the same steps as before. That would lead to this:

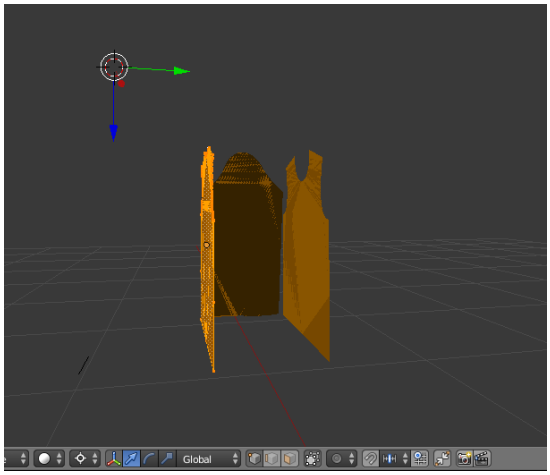


Figure 5.2.4. SVG edited.

Open MakeHuman and generate a male mannequin with the patterns measurements. Though, unfortunately, some of the measurements cannot be added as they exceed the maximum range for the joint. So instead we will try to approximate to them, in the first tab (Modelling→ Main), drag the gender slider completely to the right. Now still on modelling click the sub-tab Measure and fill the following measurements:

- Neck circumference: 43.08 cm.
- Upper arm circumference: 35.50 cm.
- Upper arm length: 36.03 cm.
- Lower arm length: 32.10 cm.
- Front chest distance: 38.04 cm.
- Bust circumference: 109.12 cm.
- Under bust circumference: 96.47 cm.
- Waist circumference: 84.05 cm.
- Nape to waist: 44.37 cm.
- Waist to hip: 19.76 cm.
- Hip circumference: 96.36 cm.
- Upper leg height: 37.82 cm.
- Lower leg height: 55.82 cm.

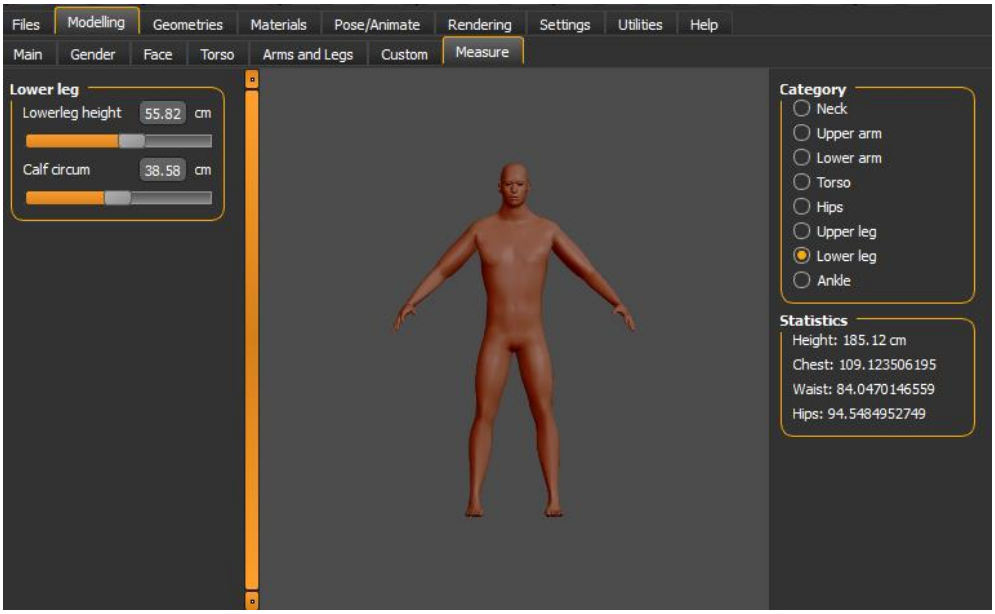


Figure 5.2.5. Mannequin created.

To finish with MakeHuman the model has to be exported to Blender. Go to the tab Files→ Export and choose the Blender option along with meters as the unit. Open it in Blender and align it with the clothes.

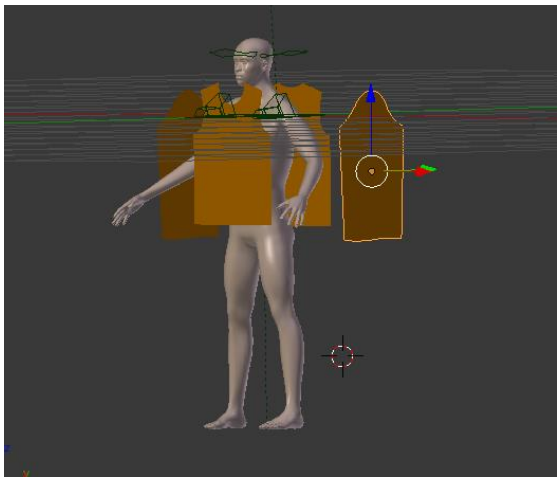


Figure 5.2.6. Mannequin alligned.

Onto the final steps. Select each piece of clothing seperately and on Edit Mode press W and click on Subdivide until there are a suficiente amount of new vertexes. Now, having selected oth the front and the back of the shirt, press CTRL and J to join them. Go to Edit Mode and select the vertexes from the shoulders to unify them. Press spacebar, once the vertextes of one shoulder (on both sides) are pressed, and type Bridge loops. Then hit SUpr and delete only the faces.

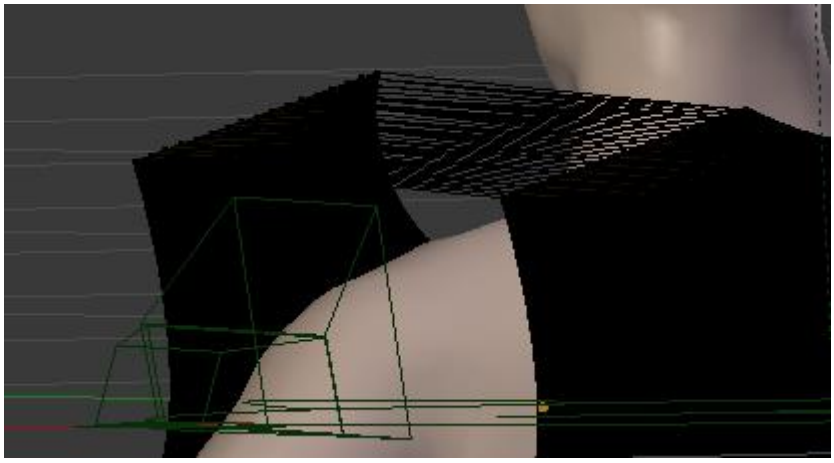


Figure 5.2.7. Shoulder unified.

Continue doing it until it looks like this:

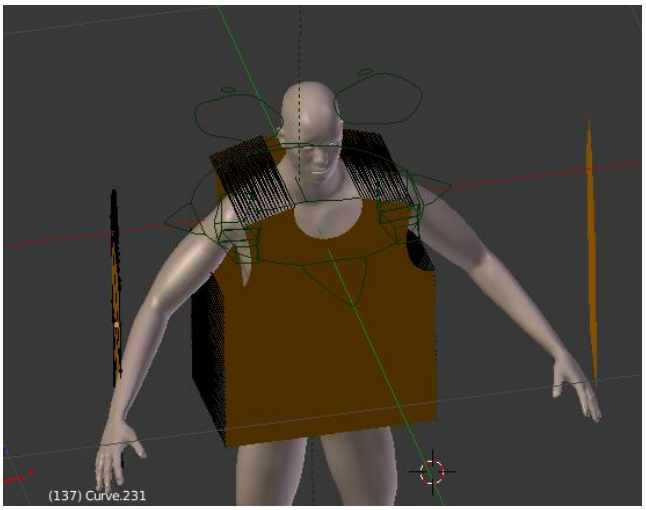


Figure 5.2.8. Back and front unified.

Select the mannequin and in context go to the modifiers (the adjustable wrench) and add the collision property. This will make the cloth stop from colliding when it hits the mannequin. Selecting the unified part of the shirt go to the same context but this time add the cloth modifier. Still with the cloth selected go to the last conext, physics, and change the values. Unfortunately this would not work and the result no matter what the values were it resulted into this because of the lack of vertexes:

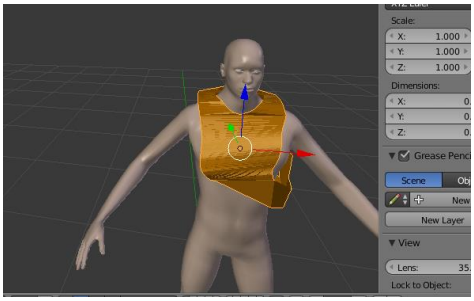


Figure 5.2.9. Sewed shirt.

Still I will like to explain what some of the properties in the physics conext mean:

- Cloth panel:
    - Presets: Contains a number of predefined Cloth material (Cotton, Denim, Leather, Rubber and Silk)
    - Steps: Set the number of simulation steps per frame. The higher they are the slower they are, but quality is.
    - Mass: The mass of the cloth material.
    - Structural: Overall stiffness of the cloth.
    - Bending: Wrinkle coefficient. Higher values create more large folds.
    - Spring: Damping of cloth velocity. Higher creates more smoothness and less jiggling.
    - Air: Slows falling things down.
    - Velocity: The velocity in which the cloth will reach the resting position (1.0 = no damping, 0.0 = fully dampened).
    - Pinning: Enable/disable pinning of cloth vertices to other objects/positions. You will need a Vertex Group. Once you have it set, all you have to do is select it and choose the stiffness you want it at.
    - Frames: Start simulation a number of frames earlier to let the cloth settle in.

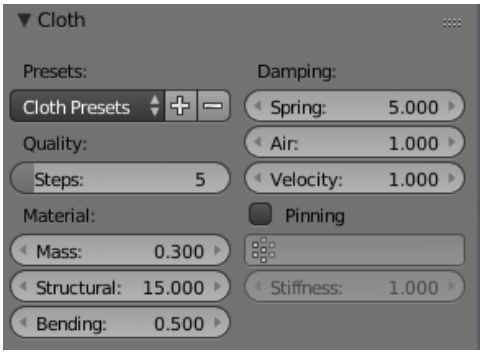


Figure 5.2.10. Cloth panel.

- Cloth Cache:
    - Start and End: When will the animation start and end.
    - Bake: Simulate the cloth with the settings in the context.

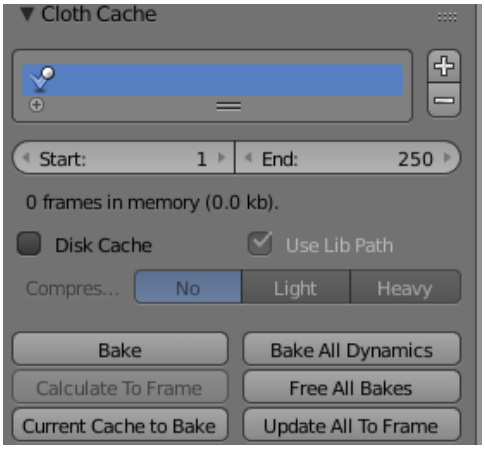


Figure 5.2.11. Cloth cache.

- Cloth Sewing Springs:
    - Sewing force: maximum sewing force.

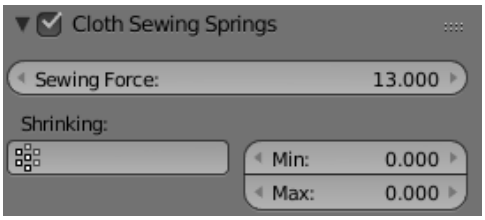


Figure 5.2.12. Cloth Sewing Springs.

5.3. Creating the clothes

In the end the patterns had to be simulated with the steps used in section Creating clothes with MakeClothes. Some changes were made, in this case an Average male with helpers was loaded and the thighs were duplicated twice in order to create the shirt and the trousers. The way the textures were added changed to and it will be explained later in this section.

First it had to be investigated how to show the clothes from Blender in the webpage. After some research the conclusion was to use ThreeJs, which is a cross-browser JavaScript library/API used to generate and display 3D models in a web browser using WebGl. There exists and Add-on for Blender that lets you export the mesh to a JSON.

Download the add-on and put the io_three folder under ~/.config/blender/2.7/scripts/addons/. Then, in the user preferences, enable it.

As mentioned before the textures had to be applied differently, if it was done as before the information was lost when exported. This new way is done by the Texture context, you create a new texture of type image or movie and select the image you want. In coordinates, under the mapping sub-context, you can change the projection but UV is already enabled by default. Create new textures for the shirt, trousers and the mannequin and then join them together before exporting it.

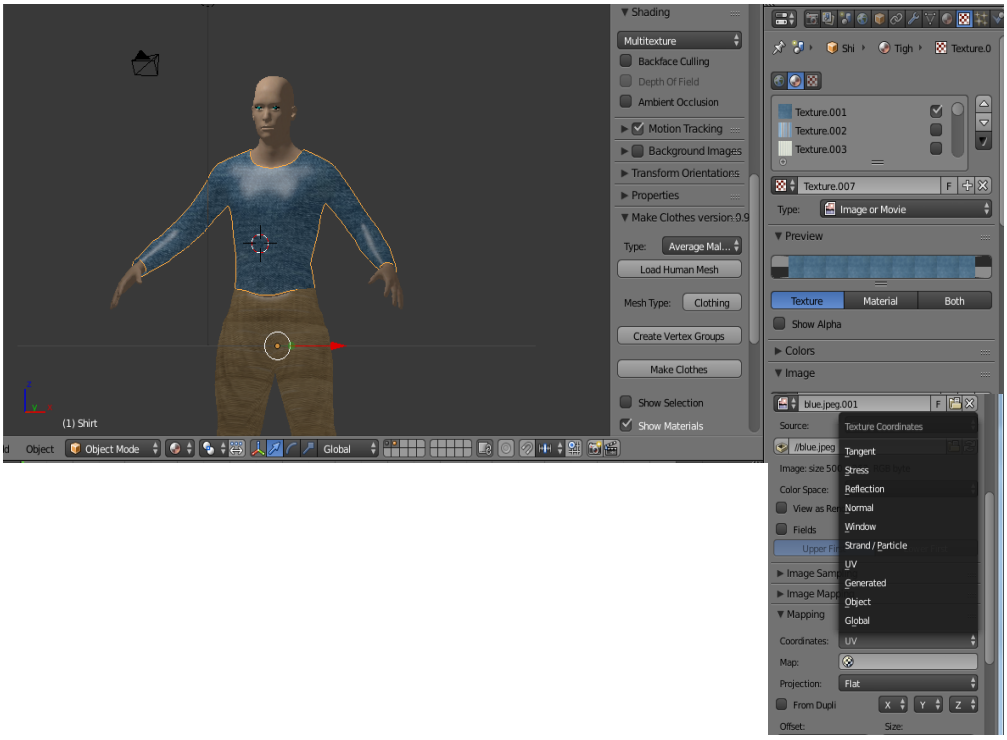


Figure 5.3.1. Model with the textures.

In the webpage add the ThreeJs library and create the scene. In order to create it there needs to be a container, a camera, a renderer and a scene. The container will be the place where the models will be seen. The camera will have to look at the scene and be kept within the dimensions of the container. The scene will held the draw the models and contain the lights. The renderer will be responsible of displaying the image and using the available renderer.

```
<script src="Three.js"></script> //add the library

     var container, camera, scene, renderer;
      init();//initialize the scene
      animate();//render the scene
```

Program 5.3.1. Threejs initialize.

```
      function init() {
        container = document.createElement( 'div' );//create and add the
container
        document.body.appendChild( container );
        camera = new THREE.PerspectiveCamera( 45, window.innerWidth /
window.innerHeight, 1, 2000 ); // vertical field of view,aspect ratio, near
plane, far plane.
        camera.position.y = 400;

        scene = new THREE.Scene();
        scene.add( new THREE.AmbientLight( 0xffffff )

  var loader = new THREE.JSONLoader();
   loader.load( "Model.json", function(geometry, materials) {
      var material = new THREE.MeshFaceMaterial( materials ); //use the
material exported
        mesh = new THREE.Mesh(geometry,material); //create a new mesh
      mesh.scale.set( 20, 20, 20 );
      mesh.position.set(0,0,0);
      scene.add(mesh);
    });
        renderer = new THREE.WebGLRenderer( { antialias: true } );
        renderer.setSize( window.innerWidth, window.innerHeight );
        container.appendChild( renderer.domElement );
      }

      function animate() {
        requestAnimationFrame( animate );
        render();
      }

      function render() {
        camera.lookAt( scene.position );
        renderer.render( scene, camera );
      }
```

Figure 5.3.2. Load the scene.

As the models have to be accessed locally ThreeJS forces us to use a server. NodeJs was used in this project; it does not have to interact with the html file but provides a place to locate it. It was implemented as shown below:[2]

```
 var static = require('node-static');
var http = require('http');
var fs = require('fs');
var file = new(static.Server)();
var line_history = [];
var clients = {};
var app = http.createServer(function (req, res) {
  file.serve(req, res);
}).listen(8080,'0.0.0.0');
```

Figure 5.3.3. Server.

NodeJs must have been installed and the node modules that are required too, these last ones are installed through the command line console with “npm init nameOfTheModule”. Later, throw the command npm init that will install the rest of the modules in the file where the server will be located. Once the installation is completed and the server’s code is written you only have to write the command “node nameOfTheServer.js”.

[2] The rest of the code appears in Anex C: Blender & MakeHuman

6

Development

This chapter gathers up all the documentation related to the process for the development of this project.

6.1. Specification and requirements

In this section the requirements and specifications for this Project are listed below:

This project is a combination of programs and so they must be able to work on the same Operating System. Matlab works on MAC OS X and Windows, whereas Blender, MakeHuman and Geogebra work on Windows, MAC OS X and GNU/Linux. Matlab is the one that restricts the use to MAC OS X and Windows but the drivers for Kinect, which allow the connection between Kinect and Matlab, are made by Microsoft which leaves us with Windows as the only option.

When Matlab and the drivers are installed and the Kinect is connected and recognized by the computer the user will run the script in Matlab. Currently the patterns generated are only for men so the client should be of that gender in order to have the pattern fitted

properly. The client must do it by positioning the camera at two meters from him and at 1 meter from the floor on a plain surface and with no object between them. The user must also be sure that all the body fits the camera and that the colour of the background does not blend with him.

After running the script and obtaining the results the client has to save them in order to use them later. Now a running server is needed, this will provide accommodation to the webpage and access to the ThreeJS elements in it. For it open the command line on the folder containing the server and run the command node server.js.

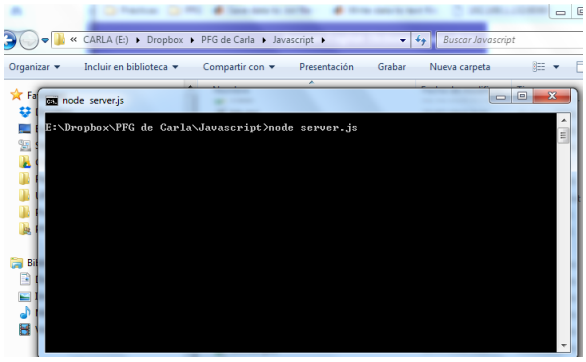


Figure 6.1.1 Running server

Once the server is running the user will access the webpage by writing the IP of network's computer and port 8080, for example: 192.168.1.1:8080. The webpage now has to load the three mannequins and the Geogebra applet, this may take a while. Once everything is loaded there a two main options: Select a sized cloth (a shirt size 38 or a trouser size 50) or creating a pattern with their own measurements.

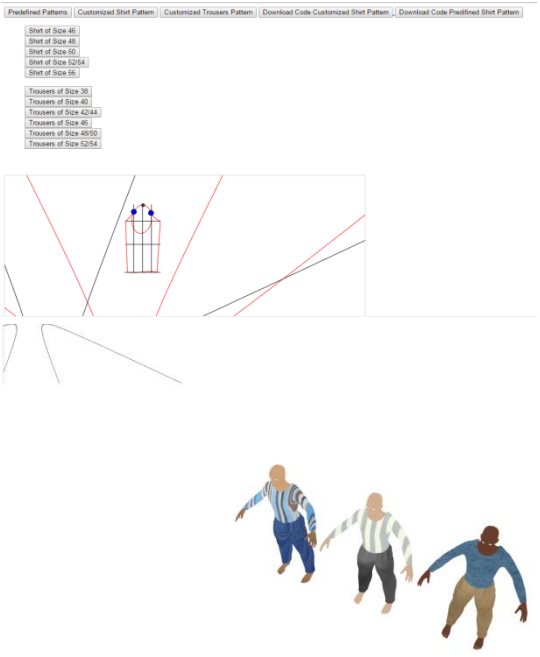


Figure 6.1.2 Webpage loaded and shirt selected.

On the image above the option selected was to generate a Shirt of Size 46 which is then shown on the Geogebra applet. The client can choose between a Shirt (sizes 46 to 56) and a pair of trousers (sizes 38 to 52/54).

In case that the user prefers to generate a pattern with his measurements he will have to fill in the following form and once it is submitted the pattern will appear as in the other cases.

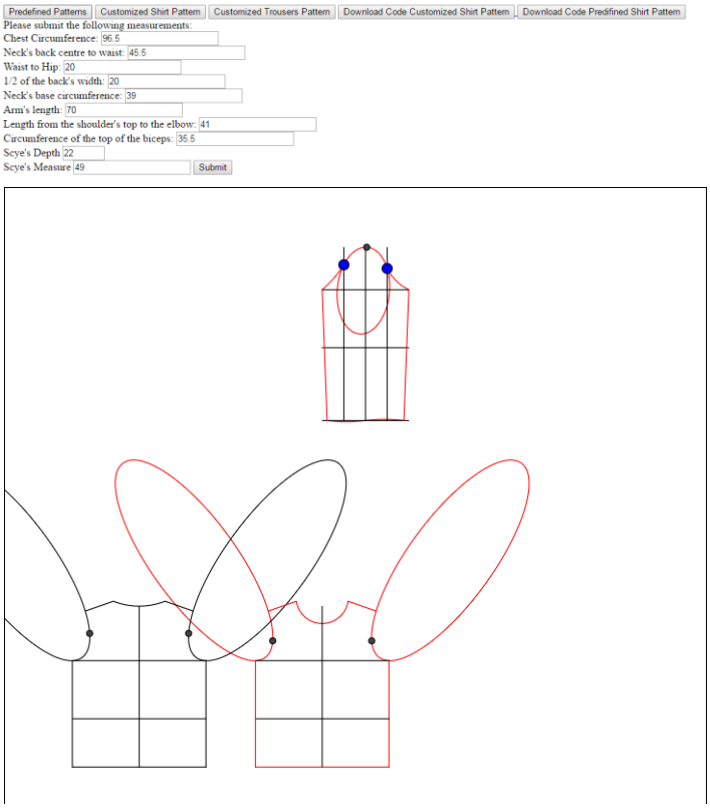


Figure 6.1.3 Customized pattern created.

The client can move the view of the pattern by selecting on the Geogebra's applet and moving the mouse around. He can also zoom in and out.

Finally, the user can print the pattern that appears on screen to a PDF so that he later can print it and sew it by clicking on Save to PDF. He should take into account what he can see in the Geogebra's applet because that is what the pdf will be generating. When the PDF is ready the user will see that it has been escalated so that it looks as it will do in real life.

6.2. Design

Once the requirements were identified and analysed the design of the use cases related to the analysis has to be done.

6.2.1. Use cases

Based on the analysis previously done the following use cases have been obtained:

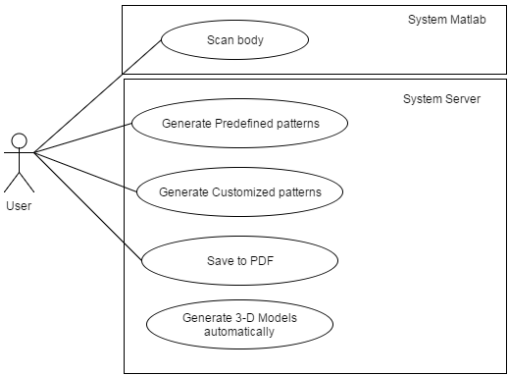


Figure 6.2.1.1 Use cases.

6.2.1.1 Use case: Generate predefined patterns

In this case the webpage is in charge of generating the patterns by sending commands to the Geogebra applet.

Name	Generate predefined patterns
Actors	The client/actor
Description	The actor wants to generate a pattern
Precondition	The actor must have entered the webpage and clicked the button
Normal Flow	1. The webpage will call the function related to the button pressed 2. The function will send to Geogebra the commands to evaluate and reflect on the applet
Alternative Flow	None
Post Condition	The pattern will be reflected in the element containing the applet

Table 6.2.1.1.1 generate predefined patterns use case

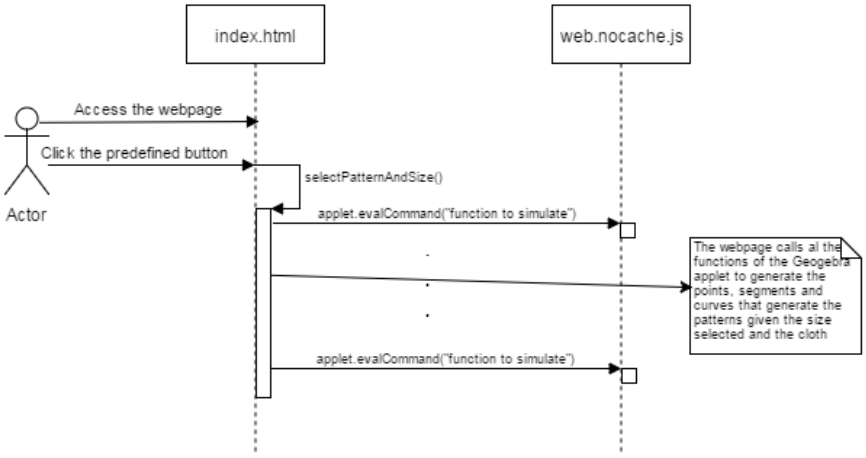


Figure 6.2.1.1.1 generate predefined patterns use case

6.2.1.2 Use case: Generate Customized patterns

In this case the webpage is in charge of generating the patterns by taking into account the measurement of the client and sending commands to the Geogebra applet.

Name	Generate customized patterns
Actors	The client/actor
Description	The actor wants to generate a customized pattern
Precondition	The actor must have entered the webpage, clicked the correct button and entered the measurements
Normal Flow	1. The webpage will call the function related to the button pressed with all the variables 2. The function will send to Geogebra the commands to evaluate and reflect on the applet
Alternative Flow	If no measurements were entered the page will wait until they are
Post Condition	The pattern will be reflected in the element containing the applet

Table 6.2.1.2.1 Generate predefined patterns use case

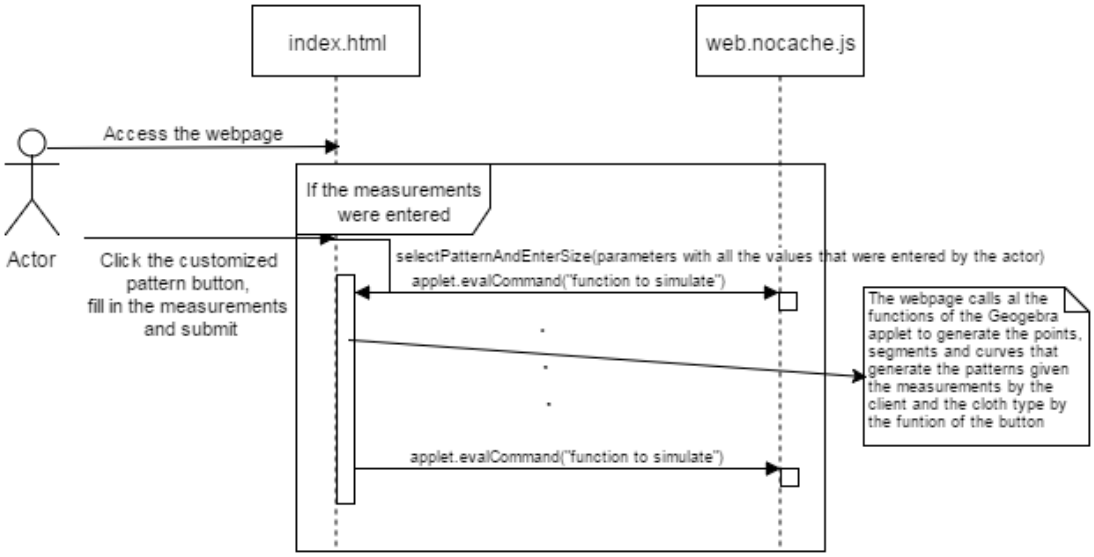


Figure 6.2.1.2.1 Generate customized patterns use case

6.2.1.3 Use case: Save to PDF

In this case the webpage generates a PDF with the content of the Geogebra applet.

Name	Save to PDF
Actors	The client/actor
Description	The actor wants to save the pattern into a PDF
Precondition	The actor must have entered the webpage and clicked the correct button
Normal Flow	1. The webpage will access the client's file and download the file 2. The client will open it on Geogebra and either enter the measurements or select which size he wants 3. Export it to PDF with the unit and centimetres to one but one piece at a time. Zooming in to the back, then to the front and finally to the sleeve of the shirt. Three PDFs will be created
Alternative Flow	None
Post Condition	The pattern will be saved onto a PDF

Table 6.2.1.3.1 Save to PDF.

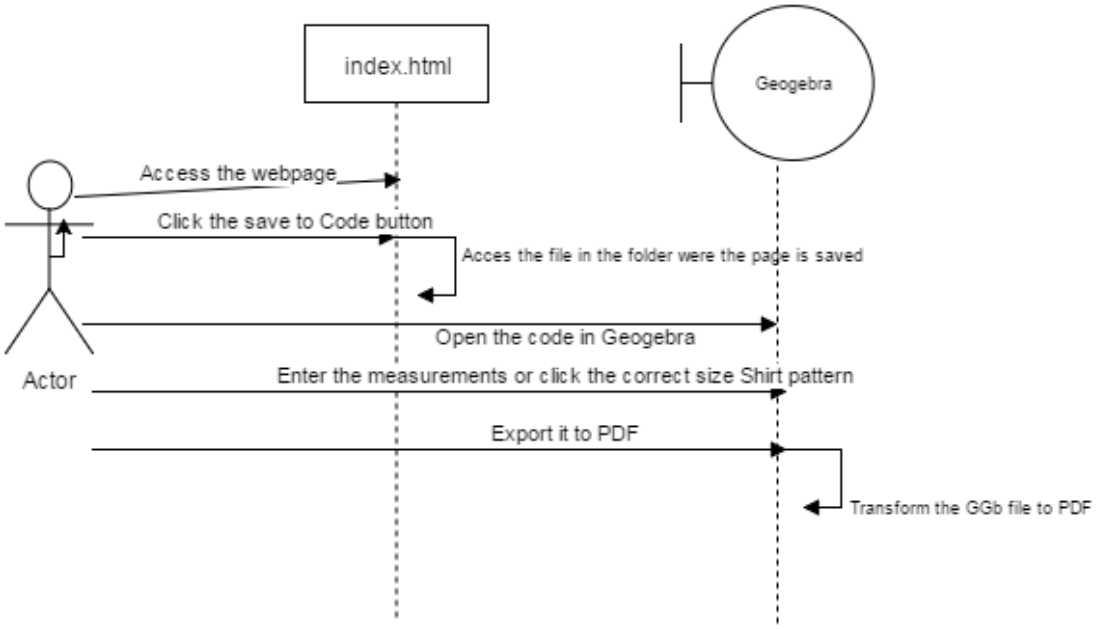


Figure 6.2.1.3.1 Save to PDF

6.2.1.4 Use case: Generate 3-D models automatically

In this case the webpage is in charge of generating the patterns by taking into account the measurement of the client and sending commands to the Geogebra applet.

Name	Generate 3-D models automatically
Actors	None
Description	The webpage automatically loads the three 3-D meshes
Precondition	The meshes must be on the same folder as the code
Normal Flow	1. The webpage will call the function init to initialize the scene and the JSON loaders for the mesh 2. The three meshes will be loaded with their materials 3. The web renderer will be initialised 4. The render function will keep the scene updated 5. The render function will also call the animation function to keep the meshes spinning
Alternative Flow	None
Post Condition	The meshes will appear on the webpage

Table 6.2.1.4.1 Generate 3-D models automatically.

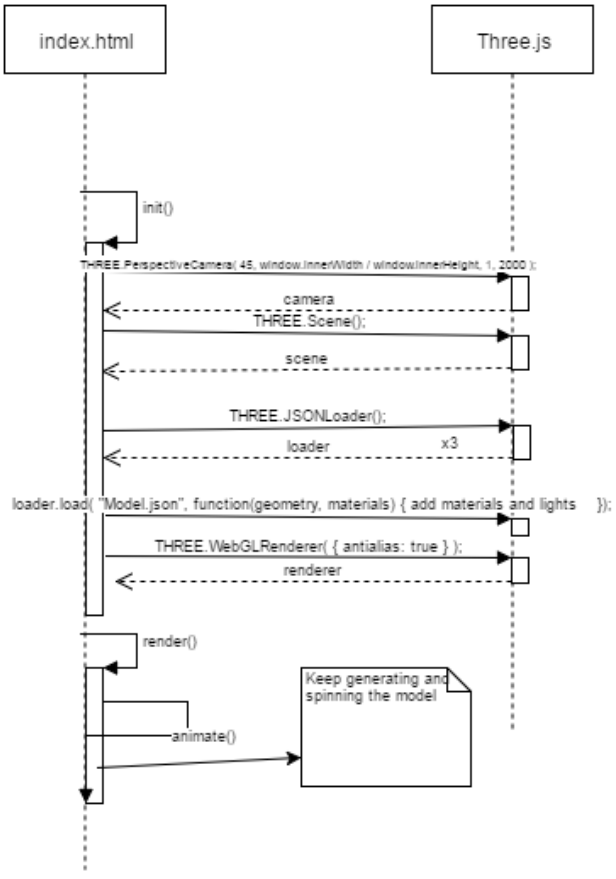


Figure 6.2.1.4.1 Generate 3-D models automatically.

6.2.1.5 Use case: Scan body

In this case the webpage is in charge of generating the patterns by taking into account the measurement of the client and sending commands to the Geogebra applet.

Name	Scan body
Actors	None
Description	The Kinect will scan the body and take the measurements
Precondition	The client must be a male and run the script on Matlab
Normal Flow	1. Matlab will find the path to the Kinect library 2. It will also initialize the Kinect and its camera (RGB and depth) 3. After starting the capture and retrieving the data it will then start the tracking in mode skeleton with the depth camera 4. Once again it will start retrieving the data and saved it 5. It will then select a frame and track its skeleton obtaining the image, the joint indexes and the joint coordinates of the skeleton 6. With all of that it will then combine the image and the joint Indexes to show the skeleton above the person 7. Then it will start to calculate each of the joints measurements 8. After it will create a grey image from the frame and show the histogram to detect the borders of the body and calculate the circumferences 9. Selecting each pair of joints it calculates the circumference with the mean of the distances between the perpendicular lines to the one unifying them.
Alternative Flow	None
Post Condition	The measurements will have been taken

Table 6.2.1.5.1 Scan body.

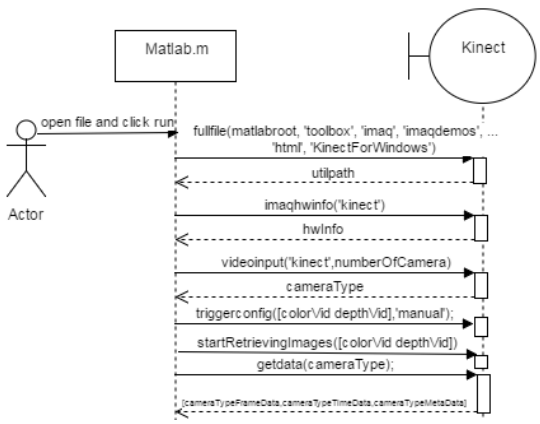


Figure 6.2.1.5.1 Scan body-initialize camera.

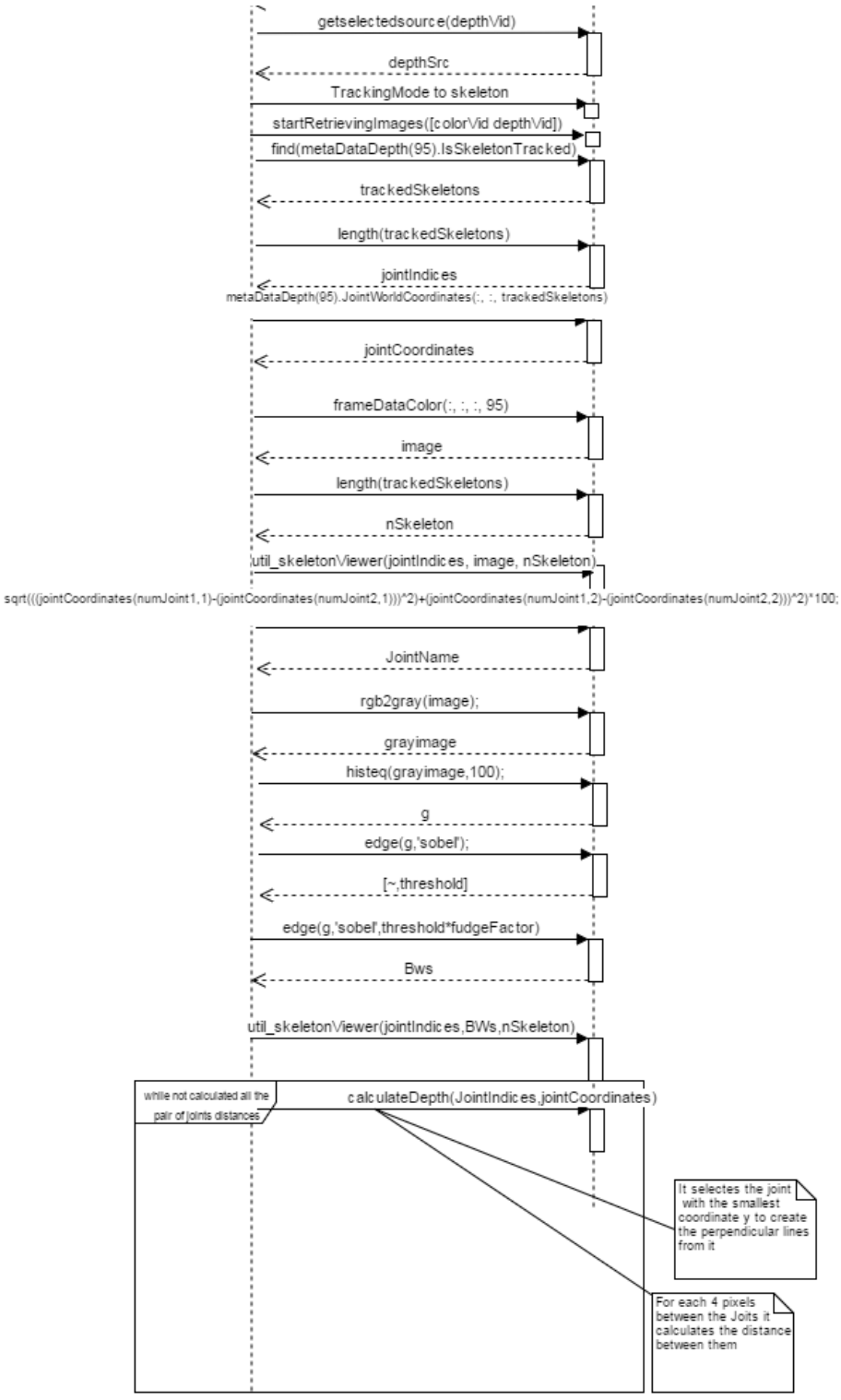


Figure 6.2.1.5.1 Scan body- calculations.

The image has been divided into two parts so that it is easier to difference when the Kinect is initializing the camera and when it is calculating the measurements.

6.2.2. Architecture

The architecture that meets the use cases and the requirements specified is the following:

- A local Server connecting the User with the Webpage.
- The Kinect camera that the user interacts with.

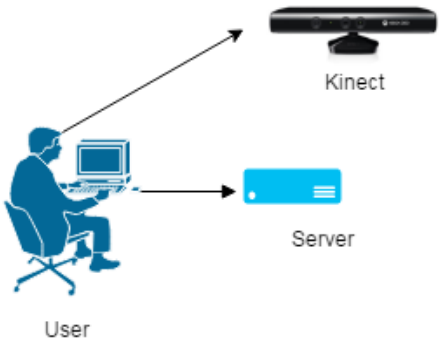


Figure 6.2.3.1 Architecture

6.3 Implementation

The implementation has been based on the needs of the webpage and the Kinect. They will be explained briefly below.

6.3.1. Implementation of the use cases regarding the webpage

The webpage is yet to be allocated on a not local server; you can see this in section Improvement proposal.

For now it works on a local server run inside the client's computer. He must first start de server with the command line "node NameOfTheServer.js" on the Windows command line console.

Now, the user can access the webpage by writing his IP followed by ":8080". There the HTML file containing all the JavaScript code will work as the user interacts with it; each button contains a different function. The functions will communicate with either the ThreeJs file (creating the 3-D models), the Geogebra applet through the Internet or the PDFJs file.

6.3.2. Implementation of the use case Scan body

For this implementation the correct drivers and version of Matlab have to be installed. The user must make sure that The Kinect is recognized as such in the computer and in Matlab. The combination of drivers' version 1.8 and Matlab 2013 worked on this project.

Then Image Acquisition Toolbox Support Package for Kinect for Windows Sensor Kinect has to be installed. Once the Kinect is place and the user is positioned in front of it, but not too near, the whole body has to be seen in camera. He may then run the script, which is written in Matlab, and wait a few minutes for the triggering to stop.

6.4. Verification of the use cases

The tests have been done in parallel to the creation of the code, making sure that each of then retrieved the wanted result. The following section will show these results.

6.4.1. Verification of the use case generating predefined patterns

While creating a pattern there may be two unwanted outcomes that need to be checked.

The first being the connection to Geogebra's applet, when on the page press F12, if no comment appears in regards to it and the applet has been charged it will work. If the applet is still loading then the buttons will not work, you should wait until it is loaded.

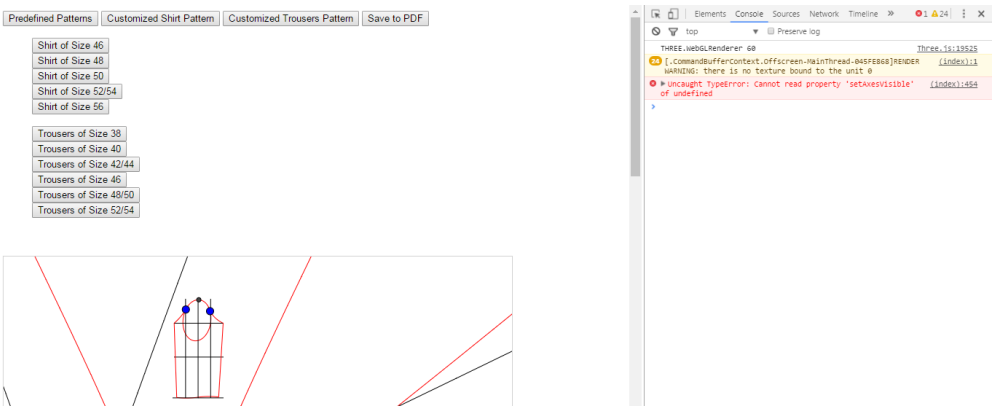


Figure 6.4.1.1 Error geogebra not uploaded

In the image above it is shown the error that will appear when the applet was not loaded and a button was clicked. It will try to use the Geogebra's commands but the connection has not been completely stablished. It can also be seen that the applet will still work afterwards, which means that it has connected to Geogebra correctly.

6.4.2. Verification of the use case generating customized patterns

While creating a customized pattern the same problems as before could arise but they are not the only ones. The Alternative Flow shows how if no measurements are entered then the function of creating the pattern will not start.

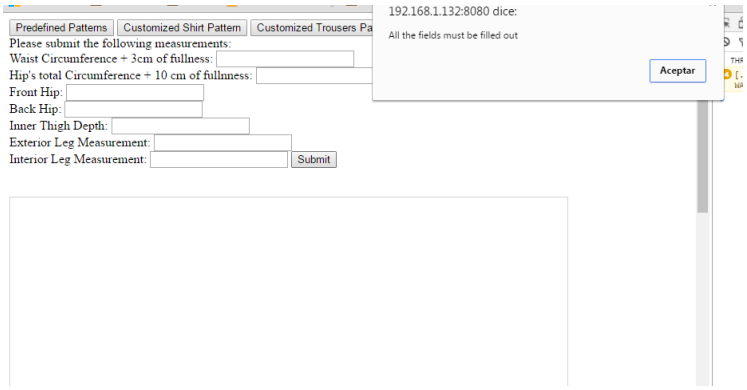


Figure 6.4.2.1 No measurements entered.

6.4.3. Verification of the use case Save to PDF

The PDF's outcome depends on the where the code is placed. If it is not in the same place where the code is then the code will not be able to be downloaded.

6.4.4. Verification of the use case Generate 3-D models automatically

In this use case we need to make sure three things:

1. The file ThreeJS is where the code looks for it. If this does not happen then the page will not be able to use the functions and neither create the models nor the scene.
2. The models are also in the place where the code looks for them. In this case the code would not load them and the following error would appear:

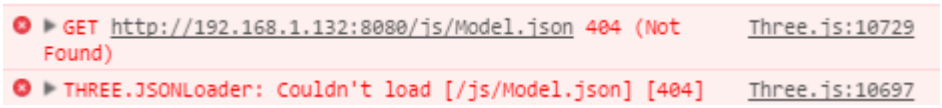


Figure 6.4.4.1. Meshes not found.

3. The materials of the models are neither on the same place where the JSON containing the models states they are. This will make an error appear in the console and will cause the model to be in black.

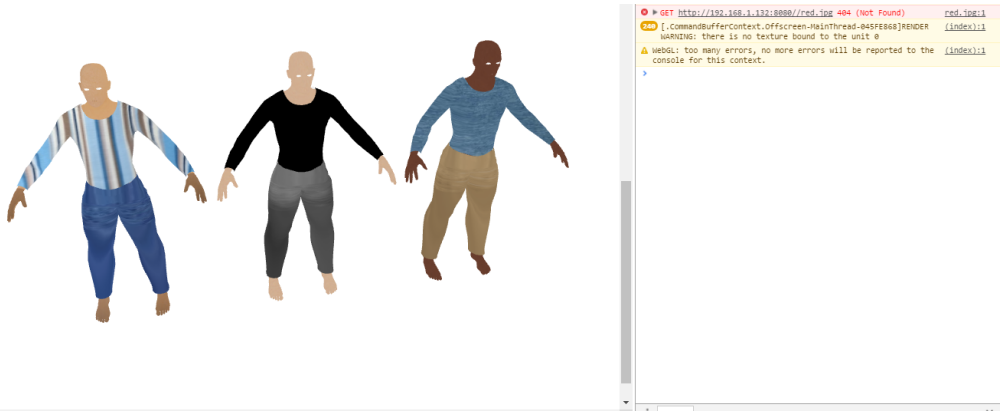


Figure 6.4.4.2. Material not found.

6.4.5. Verification of the use case Scan body

In regards to the Matlab script three errors could happen. The first one being that the Kinect would not be recognized, which can be because it is not connected to the computer or the Image acquisition toolbox has not been installed. The second one is that some function may not work properly and that can only be because the path to the image acquisition toolbox has not been added. The third, and last one, is that the Kinect does not recognize any skeleton. In order to avoid that the user must follow the instructions previously mentioned:

- Fit the whole body in the camera.
- The lighting should not interfere with the image and neither should the background or any other object.

7

Project Management

During the cycle of the project's life the scope had to be adapted because of the results obtained by the studies of the alternatives and development of some programs, previously described. Because of that new programs had to be studied and some tasks had to be redone, meaning that other tasks had to be excluded as there was no time for them.

7.1 Scope's Management

The scope has suffered three adaptations during the course of the project, one happening at the first phase and the rest happening at the implementation phase.

The first one was made during the installation of the Kinect software, the latest one could not be used as Microsoft did not support Kinect Xbox 360 any longer and neither did Matlab's later versions. The second one occurred when the adaptation of the pattern to the mannequin started, while creating the mesh from the SVG some vertexes where lost and the clothing plugin from Blender did not work as expected with that loss. The last one happened during the implementation of the Matlab Script, due to the amount of time

inverted on Blender and Geogebra the script that Ibon Olabarria had provided could not be changed, and so, Matlab could not be connected automatically to the Webpage.

At the beginning the scope of the Project covered the following points:

- Using the latest version of Matlab to reduce the error made by the Kinect's depth sensor in order to scan the body.
- Matlab would save the measurements that the Javascript of the webpage would later read and use to fill in the variables for the customized patterns.
- The webpage would transfer those measurements to Blender and execute a python code that would generate a mannequin with the measurements and the pattern adjusted. Then they would be displayed on the webpage along with the pattern automatically generated.
- The client would be able to save the pattern on a PDF document.

7.2 Viability study

Next the study in which the different aspects of the Project that could derive to the unfulfilment of the objectives and, as a consequence, the dissatisfaction of the waited results are identified will be detailed. Together with the study a plan is elaborated to avoid that possible events make the project fail or that prevents its development.

7.2.1. SWOT

	Helpful	Harmful
Internal origin	**Strengths** Family in the tailoring business Motivating Project Total dedication	**Weaknesses** New to the clothing field New to the technologies used
External origin	**Opportunities** Helping the family business Valued in the market Experience in modelling	**Threats** Software limited/still on development

Table 7.2.1.1 SWOT analysis

7.2.1.1 Strengths

- Family in the tailoring business: My dad owned a tailor's shop which is now my sister's business. Creating patterns is not an easy thing and requires both time and motivation to study it. Having someone near me in the market helped me through the creation of the patterns. If it weren't for my sister I would not have had the book "Pattern Cutting for Menswear" by Gareth Kershaw that helped me through all the

phases of pattern making and neither some expert telling me if the patterns that I created and modelled made sense.

- Motivating Project: This project consists in studying different approaches and learning how to use most of them. In the end it helped to develop new skills I had not learned during the Degree like modelling in Blender or the use of ThreeJS.

- Total dedication: There is no problem in dedicating more time to the project in order to achieve the wanted result.

7.2.1.2 Weaknesses

- New to the clothing field: I never had any experience in tailoring before which made the search for a suitable pattern making software more difficult. Not knowing the requirements beforehand made me had to change the program twice.

- New to the technologies used: From the technologies used the inexperience in Blender made it harder to work with it as I had to learn shortcuts and the tricky commands it uses.

7.2.1.2 Opportunities

- Helping the family business: The clothing sector is quite competitive and big companies are taking over it. A small business like my families could benefit from such software innovation.

- Valued in the market: This is related to the previous statement. This software is quite innovative and not many companies develop it which means that many companies will want to benefit from it.

- Experience in modelling: Throughout the Degree I had only created models in C and now I have had enough experience with Blender especially in clothing which is something very new and that not many people have experience with.

7.2.1.2 Threats

- Software limited/still on development: As I previously said, Blender's clothing plugin is still new and because of that is still quite difficult to use as many things have to be done manually. Another problem is how Windows is no longer developing software for the Xbox Kinect which limits the resources to use.

7.2.2. Risk identification

The risks that could affect negatively the development and the fulfilment of the objectives are the following:

- A very optimistic estimation: Not knowing the technologies to be used may cause an estimation that could affect the duration of the project.
- Academic failures: Having gone on Erasmus and not knowing if every subject was passed. In case that one is failed it will take time from the project.
- Problems with the software to be used: Not finding the correct pattern making program or finding out that the Kinect is no longer in use.

7.2.3. Risk quantification

The following graph evaluates the different risks defined previously. It values the probabilities of occurring and the impact that it would have on the project.

Risk	Probability	Impact
A very optimistic estimation	High	Medium
Academic failures	Low	High
Problems with the software	Medium	High

Table 7.2.3.1 risk quantification

7.2.4. Eventuality plan

- A very optimistic estimation: While developing the project the tasks will have to be replanned or reduced from the scope in order to achieve the objectives.
- Academic failures:
    - Many subjects failed: In this case the project will have to be postponed.
    - One or two subjects failed: Some tasks will have to be done in less time or the scope of the project will have to be reduced.
- Problems with the software to be used: The pattern making program is the core of the project, if this fails the project would be a failure.

7.3. Changes management

As in every project the scope of it may change during its life because of different factors that may affect positively or negatively on its execution. These changes may occur because of the result of an investigation or for satisfying new goals.

As I stated in section Project Management there were three main changes in the scope:

- Changes on the Matlab and Kinect software to be used: As I previously stated, Microsoft is no longer developing tools for the Xbox Kinect but it has developed one last driver for their Kinect. Unfortunately this one is not compatible with neither the Kinect 360 nor version 7 of Windows, which is the one used in this project. That also generated the problem of recognition between Matlab and the Kinect, versions from 2016 on forward do not recognize the Kinect for Xbox as such device. What was decided was to use the version of 2013 that Ibon Olabarria already used and the Version 1.8 of the Microsoft drivers so that no other problem could be encountered.

- Converting the SVG patterns to Blender: One of the reasons why Geogebra was chosen for this project was that it could export SVG images and Blender could import them. While importing it and creating the mesh everything worked fluently but when creating the clothing effect on Blender some vertex disappeared. What was first tried was to create a polygon in Geogebra with all the vertexes that would be needed later and then export it to SVG, but the same kept occurring. Blender lost the information when the SVG was converted to a mesh. The final solution was to recreate the clothes with the thigh material of the MakeHuman male mannequin with helpers as it was previously done as a tutorial for creating simple clothes with no effect.

- Changes in Matlab's script: Due to the amount of time dedicated to the transformation of the pattern in Blender there was less time to change Matlab's script and so it was decided that this could be done for the future instead.

- PDF creation from the webpage: One of the main objectives of the project was the possibility of creating a Life scale pattern. Unfortunately Geogebra's applet did not have the same function as the application so the pattern has to be opened on the application before exporting it.

7.4. Management costs

In this section the distribution of the time dedicated to each of the tasks that complement the project will be detailed. There are 6 tasks: Study the market, study the development of Blender, MakeHuman, Matlab and Kinect for Windows, study pattern design software, project management, project development, and academic work.

Task	Subtask	Time
Study the Clothing Market	Document	7h
Study the development of the Kinect, Matlab, Blender and MakeHuman	Kinect and Matlab	3h
	Blender and MakeHuman	10h
Study pattern design software	Valentina	3h
	Inkscape	3h
	Geogebra	6h
Project management	Scope's management	6h
	Cost's management	4h
	Meetings with the director	6x1h+10x2h
Project development	Specification	10h
	Analysis	10h
	Design	3h
	Implementation	90h
Academic work	Write report	67h
	Prepare the defence	30min
	TOTAL	249h 30min

Table 7.4.1 Management costs

7.5. Planning

This Project started the 27th of January of 2016 and it is expected to be finished for the 19th of August of 2016, date in which it will be uploaded to the digital platform ADDI.

The time that will be inverted during the course of the classes will be of around 2 hours. Once the classes and the internship are over will not be less to 4 hours. Unfortunately due to some personal problems I could not meet the hours and had to delay the internships. This was later solved by inverting more hours in the project.

7.5.1. Milestone diagram

In the table shown below the details of the important events or those that were relevant to the project's development will appear. Between them it appears the date that the project started and ended as well as some of the meetings with the director. The date of the projects defence appears also, though it is after the date of uploading it, because it is an important milestone. Some weeks will not appear as there were no important milestones those days.

	Project's Life Cycle 27/01/2016-6/09/2016												
Project's milestones	W1	W5	W11	W13	W17	W22	W26	W27	W28	W29	W31	W33	W34
Start of the project	♦												
Initial meeting	♦												
Start of the report				♦									
Finish Geogebra's part							♦						
Finish Blender's part								♦					
Finish Matlab's part									♦				
Meetings with the director		♦	♦	♦	♦	♦			♦			♦	
Register the project							♦						
Draft send to the director									♦				
Revise the report										♦			
Upload the project to ADDI											♦		
Start he presentation												♦	
Project's defence													♦

Table 7.5.1 Milestones diagram

Below the exact dates of each event is listed:

- 27th of January of 2016: Start of the project.
- 27th of January of 2016: First meeting with the director.
- 24th of February of 2016: Meeting with the director to monitor and control the studies development and select the patterning software.
- 6th of April of 2016: Meeting with the director to switch the software to Geogebra.
- 20th of April of 2016: Meeting with the director to monitor and control the project's development.
- 20th of April of 2016: Strat the documentation for the report.
- 18th of May of 2016: Meeting with the director to talk about the documentation.
- 22nd of June of 2016: Meeting with the director to monitor and control the project's development.
- 20th of July of 2016: Meeting with the director to revise the project.
- 21st of July of 2016: Finish Geogebra's part.
- 25th of July of 2016: Register the report for the defence.
- 27th of July of 2016: Finish Blender's part.

- 1st of August of 2016: Finish Matlab's part.
- 1st of August of 2016: Meeting with the director to settle the end point.
- 5th of August of 2016: Send a draft of the report to the director.
- 19th of August of 2016: Correct the report.
- 29th of August of 2016. Upload the report to ADDI.
- 7th of September of 2016: Start the presentation.
- 12th of September of 2016: Meeting with the director to talk about the presentation.
- 15th -19th of September of 2016: Present the project's defence.

Many other meetings were held with the director that were not included as they did not represent any important moment in the project.

8

Conclusions

In this chapter an analysis of the objectives, stated in section Objectives, will be done. Also, there will be a subsection dedicated to the conclusions obtained in the project's management and the decisions taken throughout the time dedicated to it. Lastly, it will conclude with a personal opinion of the author about the personal experience achieved through the live of the project.

8.1. Objectives' conclusions

The hardest objective to achieve was the main core of the project, generate patterns automatically. It took longer than expected and lots of revisions were made. Not having a virtual tailor's ruler caused a lot of problems; no mathematical function could emulate it and so, many pattern making programs forces the user to create the curves manually.

Fortunately Geogebra was found soon and the commands that come with it could emulate every necessary action in the pattern making production. It was a smart decision as it also included Geogebra-web that eliminated the need of interaction between client and tailor when creating the pattern, as a webpage is offered.

Though the main objective was satisfied and the research on the market was successfully thorough, connecting Matlab with the webpage and creating a mannequin with the client's

measurements was not been achieved. This will be discussed in section Improvement proposal.

8.2. Project's management conclusions

Thanks to the objectives that were clearly stated at the beginning of the project specific tasks could be defined. This led to an easy tracking and annotation of them. Every week there was a meeting with the director were doubts were solved and problems or achievements were discussed. Once half of the life's project passed specific end dates were stablished for the tasks that were left.

Because of the tracking and the stated tasks the project could carry on despite the problems encountered. Efficient solutions were found in time and the project concluded as a success.

Despite the personal problem that occurred to the author the management resulted to be the best one as it helped the reincorporation of her to the project.

8.3. Personal conclusions

In this kind of projects is really important to make a thorough research of the field. Knowing what people will expect and what does the market already offer is vital to the development of the software.

Geogebra has resulted to be a very simple program with lots of potential that no one seemed to have thought of for this job. Many other patterns could have been created if it were the only part of the project.

As for what it was learned during the project it has to be said that it will be valuable in the Computer Science field. These skills include modelling, not only in Blender but in web too and what was learned in Blender is not so common to find. Also the use of Matlab is very important; it is one of the most potential programs out there and been able to calculate the measurements of someone with an RGB image is quite impressive.

Throughout the degree I have found myself thinking if I could be able to accomplish anything with what I have learned in the courses. What I was not aware of is that apart from learning new languages o methods I was learning to be able to investigate on the matter, to be able to achieve things on my own.

9

Improvement proposal

This project has come to an end and unfortunately some things that could have improved the result could not be done. Here is a list containing the improvements that could be done in the future:

- The webpage's interface: Right now the webpage lacks of CSS and is not responsive. A simple style could be added so that the buttons appeared in order and the patterns would appear next to the mannequins instead of on top.

- Matlab's start: When accessing the webpage a script could be run in which it contained the script that was already done as well as Matlab's start.

- Save measurements: When matlab recollects all the measurements these could be saved on a simple file where the JavaScript code could find it and give the client the option of fill it automatically.

- More patterns: More patterns could be created and the option for woman clothing could appear too. This would mean adding more commands in the JavaScript code that access Geogebra.

- Client represented in a mannequin: When the client enters its measurements this could trigger a python code that would adapt a MakeHuman mannequin to it along with the clothes. This would mean using the macro-recorder to record what was previously done in Blender and then calculate the positions and sizes of the new cloth depending on the joints.

- PDF exportation: The client will not have to download the code and neither have the code already. He will be able to export it from the webpage.

- Texture selection: The client could have a button that allows him to change the texture on the cloth.

- Server: The server could not have to be run by the client when he wants to access the webpage.

Glossary

Base64: Group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation.

Blender: Professional free and open-source 3D computer graphics software product used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games.

Fullness: Extra fabric used across the width or sometimes height of a cloth.

JSON: (JavaScript Object Notation) Text format for the exchange of data.

C/C++/Java/Fortran/Python: Programming languages.

Clothed: Program for the conversion of patterns into meshes.

DPI: (Dots per inch) Measure unit used for the impression resolutions.

Driver: Is a computer program that operates or controls a particular type of device that is attached to a computer

Image acquisition toolbox: Package that provides functions and blocks that enable you to connect industrial and scientific cameras to MATLAB® and Simulink®

Infrared: Type of electromagnetic and thermic radiation whose wave is larger than the visible light, but smaller to the microwaves.

Inkscape: Vectors graphic editor, free and of open code. It can create and edit vectorial graphs.

Joints: Connection made between bones in the body.

Kinect: Is a line of motion sensing input devices by Microsoft for Xbox 360 and Xbox One video game consoles and Windows PCs.

MakeHuman: Program used to model mannequins.

Mesh: A collection of vertices, edges and faces that defines the shape of a polyhedral object in 3D computer graphics and solid modelling.

Motion sensing: The process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object.
Natural user interface: Interface in which the interaction is used without using any input device or remote controller.

Nodejs: Server-side JavaScript environment.

Pattern: A paper template created to copy into the fabric and generate a cloth, cutting, sewing and mounting the different pieces.

PNG: (Portable Network Graphics) Graphic format based in a compression algorithm without bitmaps loss.

Rendering: Process of generating an image from a 2D or 3D model.

Server: A system that responds to requests across a computer network to provide, or help to provide, a network or data service.

SDK: Set of software development tools that allow the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.

SVG: Scalable Vector Graphics is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation.

Tailor's ruler: Ruler used by tailors to create the scye's and necks curves.

Valentina: Open source pattern drafting software.

Webgl: Is a JavaScript API for rendering interactive 3D computer graphics and 2D graphics within any compatible web browser without the use of plug-ins.

Bibliography

[1] Gareth Kershaw: *Patternmaking for Menswear,* 22nd October 2013.

[2] Ibon Olabarria: *Sistema de comercio electrónico para el sector textil: generar maniquíes 3D a partir de imágenes del sensor Kinect*, June 4014. https://addi.ehu.es/handle/10810/13180

[3] *Official Geogebra's Manual 4.0.* https://www.dropbox.com/s/w5dybx62g6b0vy9/Official%20GeoGebra%20Manual.pdf?dl=0

[4] *GeoGebra Help: Official Manual 3.2*. https://app.geogebra.org/help/docuen.pdf

[5] *Geogebra's Javascript Manual.* https://www.geogebra.org/manual/en/Reference:JavaScript

[6] John M. Blaine: *An introduction to Blender 3D a book for beginners*. http://download.blender.org/documentation/pdf/John%20M%20Blain%20-%20An%20Introduction%20To%20Blender%203D%20-%20A%20Book%20For%20Beginners%20(2011).pdf

[7] *Blender Basics: Classroom tutorial book*, 4th Edition, 2011. http://www.cdschools.org/cms/lib04/PA09000075/Centricity/Domain/81/BlenderBasics_4thEdition2011.pdf

[8] How to sew clothes in Blender: *Sewing clothes in Blender [Available in >= 2.70]*. http://blenderartists.org/forum/showthread.php?258004-Sewing-Clothes-in-Blender-Available-in-gt-2-70

[9] Blender Tutorial*: How to make towels.* http://www.blenderguru.com/tutorials/make-towels/

[10] Blender's Manual: *Blender's reference Manual*. https://www.blender.org/manual/

[11] From SVG to cloth: *Making Clothes Marvelous Designer Style(Blender).* http://forum.maratis3d.com/viewtopic.php?id=848

[12] From SVG to mesh*: 2d to 3d Vector graphics for 3d Print (Illustrator to Blender)*. https://www.youtube.com/watch?v=0jNq7X4oXSU

[13] *Using the Kinect® for Windows® from Image Acquisition Toolbox™.* http://es.mathworks.com/help/imaq/examples/using-the-kinect-r-for-windows-r-from-image-acquisition-toolbox-tm.html

[14] *Acquiring image and skeletal data using Kinect.* http://es.mathworks.com/help/imaq/acquiring-image-and-skeletal-data-using-the-kinect.html

[15] Transform a mesh to JSON in Blender: *Importing a Modelled Mesh From Blender to Three.js.* http://www.jonathan-petitcolas.com/2015/07/27/importing-blender-modelized-mesh-in-threejs.html

[16] *ThreeJS documentation*. http://threejs.org/docs/

[17] *Top 10 Clothing Design Software for Amateur and Professional Designers.* http://www.vagueware.com/top-10-clothing-design-software-for-amateur-and-professional-designers

[18] *Valentina's Wiki*. https://wiki.valentinaproject.org/wiki/Main_Page

[19] Information about Valentina: *Valentina, Software Libre de diseño de moda.* http://www.escuelaslibres.org.ar/2014/01/valentina-software-libre-de-diseno-de-moda/

[20] *PDFjs documentation.* https://mozilla.github.io/pdf.js/getting_started/

[21] Nodejs documentation: *Node.js v6.3.1 Documentation.* https://nodejs.org/dist/latest-v6.x/docs/api/

Anex A: Kinect and Matlab

A.1. Kinect

Kinect is a line of motion sensing input developed by Microsoft for Xbox (360 and One) video game consoles and Windows PCs. Users are able to control and interact with the console/computers without needing a game controller, through a natural user interface using either spoken commands or gestures.

A.1.1 Hardware

The device has one infrared Sensor, one infrared camera, an RGB camera, a LED, a Motorized tilt and a Microphone array of four elements.

- Infrared Sensor: In order to obtain a 3D image the Kinect uses Three-dimensional sensor which tracks your body within the play space.
- Infrared Camera: This camera captures de light that the sensor projects so that it is able to calculate the depth of that pixel.

- RBG Camera: This one captures a color image. An image is a group of pixels, each pixel is composed by and RBG value of 24 bits. An image is composed by 3 components: Red, Green and Blue. As every image can save a value from 0 to 255 per pixel we can conclude that there are up to 2^{24} colors.

- LED: The LED light indicates the state of the device. If the LED is off then the Kinect is not plugged into a socket, if its red then the device is off but plugged and if its green then the device is on (plugged into a socket and to a computer or Xbox).

- Motorized tilt: It is a mechanical drive in the base of the sensor that automatically tilts its head up and down. This should not be done manually.

- Microphone Array: These microphones are used for voice recognition. As each microphone is situated in different places, sounds would not arrive to them at the same time that is why they are able to calculate the position of the sound and the source of it. They are also able to filter the human voice by eliminating any frequency that is not between 80 Hz and 1100Hz.

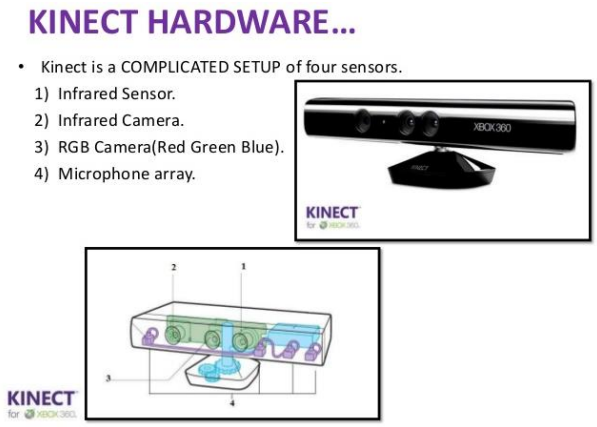


Figure A.1.1.1. Kinect Hardware.

A.1.2 Types

There are two categories of Kinect: Kinect for Xbox and Kinect for Windows.

- Kinect for Xbox: These types of devices are compatible with the Xbox and can be used for developing software. There are two types:
    - Kinect 360: This Kinect is the one used in this project.

Figure A.1.2.1. Xbox 360.

Characteristics	Values
Vertical Vision Field	43 degrees
Horizontal Vision Field	57 degrees
Motorized tilt rage	[-27,+27] degrees
Images per second	30 fps, frames per second

Table A.1.2.1. Xbox 360 characteristics.

- Kinect One: This device was released later and counts with a better depth sensor. It is able to distinguish someone's pulse.

Figure A.1.2.2. Xbox One.

- Kinect for Windows: Windows created these devices to be used on Windows PCs. There are two versions of this Kinect with much bigger differences; the second is able to track the state of the hand, has better microphones and the depth sensor is 3 times more accurate. Once the second version was out Windows stopped supporting the Xbox Kinect, which means that the Developer toolkit for Kinect would not detect the Kinect but that is no problem for the project as we will be using Matlab.

  - Kinect V1:

Figure A.1.2.3. Kinect V1.

  - Kinect V2:

Figure A.1.2.4. Kinect V2.

Properties	Kinect V1	Kinect V2
Video	640×480 @30 fps 1280×960 @12 fps	1920×1080 @30 fps High Definition
Depth	320×240, 640×480 0.8 to 4 meters in default mode 0.4 to 3 meters in near mode	512×424 0.5 to 4.5 meters
Body tracking	Able to detect 6 people, only 2 can be tracked 20 joints per skeleton (Skeletal stream)	Able to detect and track 6 people 25 joint detected per person (BodySource)
Inclination motor	Yes, between +27 degrees to –27	No. Bigger vision angle Can be manually tilted
USB	2.0	3.0
Operating System	Win7+	Win 7.1 or superior (only 64 bits)

Table A.1.2.1. Windows Kinect characteristics.

Big improvements were done in version 2 of Kinect for Windows which can be seen in the table above.

A.2 Microsoft Kinect for Windows SDK

An SDK (**software development kit**) is a set of software development tools used to create applications for certain platforms or operating systems. In this project we are talking about the SDK that Microsoft has developed for Kinect. It is compatible with both the Xbox and Microsoft versions. It features tools for creating applications in C++, C# or Visual Basic using Microsoft Visual Studio such as:

- Access data that is not processed: Access de depth sensor data, de RGB camera and two the four microphones.
- Track the skeleton: The capacity of following one or two skeletons
- Advanced audio capabilities: Supress the noise or echo, identify the source of the sound and the integration of an API which works similarly to the Windows voice recognition
- Sample codes and documentation.

A.2.1 Installation

In order to install the Kinect SDK we need to have Microsoft Visual Studio installed. We will now download the SDK from the official Microsoft Kinect for Windows page. Before installing anything make sure that the Kinect is not plugged into the computer.

There are several versions of the SDK, in this project I will use version 1.8, as the latest (2.0 is not supported by Windows 7).

After following the installation steps connect the device to the computer and verify that the drivers were installed. For that, go to Window's Device Administrator and under Kinect for Windows check that Kinect for Windows Audio Array Control, Kinect for Windows Camera and Kinect for Windows Security Control appear.

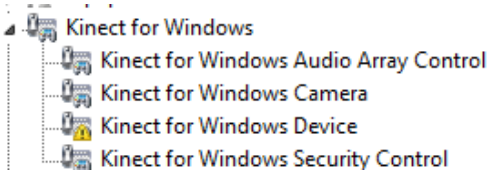


Figure A.2.1. Kinect Drivers.

Now the installer will suggest you to download the developer tools, this will not be necessary as we will use Matlab and because it does not recognize this type of Kinect as a Kinect for Windows Device.

A.2.2 Versions

A.2.2.1 Microsoft Kinect for Windows V1.7

Published in March of 2013 with the following features:

- Kinect Interactions: Programmers are now able to create applications that use the most common gestures of the people.
- Kinect Fusion: It gives the opportunity of rendering 3D images in real-time.

A.2.2.2 Microsoft Kinect for Windows V1.8

This one was published the September of 2013 and adds the following features:

- Kinect Background Removal: API that allows deleting or changing the background image of the user.
- Web Server for Kinect data flow: It is a component that brings the opportunity of accessing the data that the Kinect obtains to Webs made on HTML5. As a result you can control an HTML5 web with Kinect interactions.

This is the version used in this project.

4.2.2.3 Microsoft Kinect for Windows V2

This is the latest version which came out in 2014. Unfortunately it requires Windows 7.1 or higher so I cannot use it. Some of the new features are:

- Wider vision range: 70 horizontal degrees (against the 57 degrees of the predecessor) and 60 degrees vertically (before 43 degrees).
- It does not have a motorized tilt
- Higher Resolution: 1920 x 1080 pixels Full HD (before 640x480). Allows are more precise detection. It is capable of differentiate the body's orientation including hands and it can also distinguish the fingers.
- A more accurate depth sensor: instead of being around 1.2-35.5 meters of error now is between 0.5 to 4.5 meters.
- USB 3.0: Faster configuration and data flow.
- Capability of recognizing movements in the dark.

A.3. Matlab

Matlab[3] is a multi-paradigm numerical computing environment and a programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

For this project the version of 2013 will be used as the newest ones only work with Kinect for Windows.

A.3.1 Installing the Kinect for Windows Sensor Support package

Kinect is not recognized by Matlab unless the Kinect for Windows Sensor Support package is installed. For it to be it the following steps have to be followed:

1. In Matlab open the installer: Home > Environment > Add-Ons > Get Hardware Support Packages. On the Select an action screen, select Internet then click Next. Support Package Installer downloads and installs the support package and third-party software from the Internet.

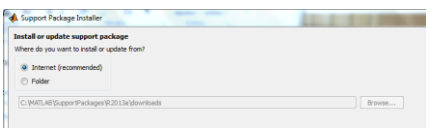


Figure A.3.1.1 Support Package Installer

[3] A Matlab license was achieved for free thanks to the KU Leuven, university where the author was on Erasmus and gave free licenses for their students.

2. On the Select support package to install screen, select Kinect for Windows Runtime from the list and click Next.

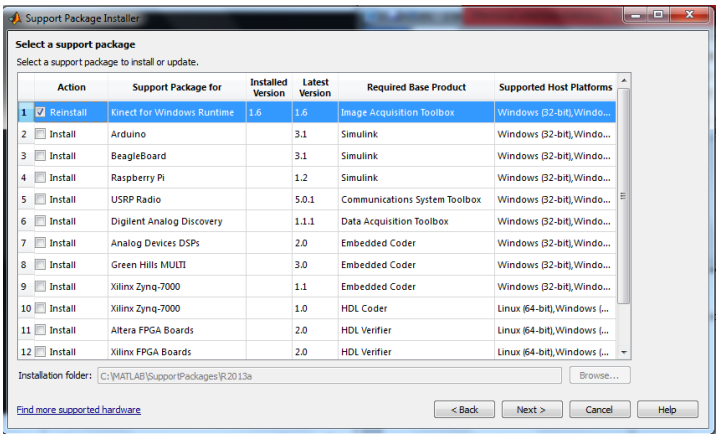


Figure A.3.1.1 Kinect for Windows Runtime

3. Now Matlab will ask you to log in with your account. Once that is done accept the agreement on the MATHWORKS AUXILIARY SOFTWARE LICENSE AGREEMENT screen and click Next.
4. The Third-party software licenses screen displays your choice of Image Acquisition Toolbox Support Package for Kinect for Windows Sensor. After reviewing the information hit Next.
5. On the Confirm installation screen, Support Package Installer confirms that you are installing the support package, and lists the installation location. Confirm your selection and click Install.

A.4. Code

```
utilpath = fullfile(matlabroot, 'toolbox', 'imaq', 'imaqdemos', 'html',
'KinectForWindows');
addpath(utilpath);

hwInfo = imaqhwinfo('kinect')
hwInfo.DeviceInfo(1)
hwInfo.DeviceInfo(2)
colorVid = videoinput('kinect',1)
depthVid = videoinput('kinect',2)
triggerconfig([colorVid depthVid],'manual');
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
preview(colorVid)
start([colorVid depthVid]);
pause(20)
trigger([colorVid depthVid]);
[colorFrameData,colorTimeData,colorMetaData] = getdata(colorVid);
[depthFrameData,depthTimeData,depthMetaData] = getdata(depthVid);
stop([colorVid depthVid]);
```

```
depthSrc = getselectedsource(depthVid)
depthSrc.TrackingMode = 'Skeleton';
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
start([colorVid depthVid]);
pause(20)
trigger([colorVid depthVid]);
[frameDataColor]    =    getdata(colorVid);        %    [frameDataColor]    =
getdata(colorVid,100,native);
[frameDataDepth,   timeDataDepth,   metaDataDepth]   =   getdata(depthVid);
stop([colorVid depthVid]);
metaDataDepth
anyPositionsTracked = any(metaDataDepth(95).IsPositionTracked ~= 0)
anySkeletonsTracked = any(metaDataDepth(95).IsSkeletonTracked ~= 0)
trackedSkeletons = find(metaDataDepth(95).IsSkeletonTracked)
jointCoordinates     =     metaDataDepth(95).JointWorldCoordinates(:,     :,
trackedSkeletons)
% Skeleton's joint indices with respect to the color image
jointIndices = metaDataDepth(95).JointImageIndices(:, :, trackedSkeletons)
image = frameDataColor(:, :, :, 95);%error
nSkeleton = length(trackedSkeletons);
util_skeletonViewer(jointIndices, image, nSkeleton);%error

upperarm_left=sqrt(((jointCoordinates(5,1)-
(jointCoordinates(6,1)))^2)+(jointCoordinates(5,2)-
(jointCoordinates(6,2)))^2)*100;
upperarm_right=sqrt(((jointCoordinates(9,1)-
(jointCoordinates(10,1)))^2)+(jointCoordinates(9,2)-
(jointCoordinates(10,2)))^2)*100;
upperarm=(mean([upperarm_left,upperarm_right]))*100; %%es la mitad?
lowerarm_left=sqrt(((jointCoordinates(6,1)-
(jointCoordinates(7,1)))^2)+(jointCoordinates(6,2)-
(jointCoordinates(7,2)))^2)*100;
lowerarm_right=sqrt(((jointCoordinates(10,1)-
(jointCoordinates(11,1)))^2)+(jointCoordinates(10,2)-
(jointCoordinates(11,2)))^2)*100;
lowerarm=(mean([lowerarm_left,lowerarm_right]))*100; %%es la mitad?
frontcheast=sqrt(((jointCoordinates(5,1)-
(jointCoordinates(9,1)))^2)+(jointCoordinates(5,2)-
(jointCoordinates(9,2)))^2)*100;
necktowaist=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(2,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(2,2)))^2)*100;
waist_to_hip=sqrt(((jointCoordinates(2,1)-
(jointCoordinates(1,1)))^2)+(jointCoordinates(2,2)-
(jointCoordinates(1,2)))^2)*100;
shoulderright=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(5,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(5,2)))^2)*100;
shoulderleft=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(9,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(9,2)))^2)*100;
backdist=(mean([shoulderleft,shoulderright]))*100; %%es la mitad?
upperlegleft=sqrt(((jointCoordinates(13,1)-
(jointCoordinates(14,1)))^2)+(jointCoordinates(13,2)-
(jointCoordinates(14,2)))^2)*100;
```

```
upperlegright=sqrt(((jointCoordinates(17,1)-
(jointCoordinates(18,1)))^2)+(jointCoordinates(17,2)-
(jointCoordinates(18,2)))^2)*100;
upperleg=(mean([upperlegleft,upperlegright]))*100; %%es la mitad?
lowerlegleft=sqrt(((jointCoordinates(14,1)-
(jointCoordinates(15,1)))^2)+(jointCoordinates(14,2)-
(jointCoordinates(15,2)))^2)*100;
lowerlegright=sqrt(((jointCoordinates(18,1)-
(jointCoordinates(19,1)))^2)+(jointCoordinates(18,2)-
(jointCoordinates(19,2)))^2)*100;
lowerleg=(mean([lowerlegleft,lowerlegright]))*100; %%es la mitad?
neck=sqrt(((jointCoordinates(3,1)-
(jointCoordinates(4,1)))^2)+(jointCoordinates(3,2)-
(jointCoordinates(4,2)))^2)*100;


prof=frameDataDepth(:,:,:,82);
util_skeletonViewer(jointIndices,prof,nSkeleton);
grayimage=rgb2gray(image);
g=histeq(grayimage,100);
[~,threshold]=edge(g,'sobel');
fudgeFactor=0.5;
BWs= edge(g,'sobel',threshold*fudgeFactor);
figure,imshow(BWs);
util_skeletonViewer(jointIndices,BWs,nSkeleton);
grayimage=rgb2gray(image);
g=histeq(grayimage,100);
[~,threshold]=edge(g,'sobel');
fudgeFactor=0.5;
BWs= edge(g,'sobel',threshold*fudgeFactor);
figure,imshow(BWs);
util_skeletonViewer(jointIndices,BWs,nSkeleton);

joints=[5,6,9,10,13,14,17,18,14,15,18,19,3,4,6,7,10,11,14,15,18,19];
index=1;
medidas_pixel=[];
medidas_real=[];
while index<23,

num1=joints(index);num2=joints(index+1);joI=jointIndices;joC=jointCoordinat
es;
    hline=imdistline(gca,[joI(num1,1), joI(num2,1)],[joI(num1,2),
joI(num2,2)] );
    api=iptgetapi(hline);
    medida_pixel= api.getDistance();
    medida_real=sqrt(((joC(num1,1)-joC(num2,1)))^2+((joC(num1,2)-
joC(num2,2)))^2)*100;

    angle=api.getAngleFromHorizontal();
    api.delete();
    m=tand(angle);
    m_per=-(1/m);
    lista=[];
    lista2=[];
    medidas_pixel=[medidas_pixel medida_pixel];
    medidas_real=[medidas_real medida_real];
    if(joI(num1,2)<joI(num2,2))
        min_x=joI(num1,1);
```

```
        min_y=joI(num1,2);
        max_x=joI(num2,1);
        max_y=joI(num2,2);
    else
        min_x=joI(num2,1);
        min_y=joI(num2,2);
        max_x=joI(num1,1);
        max_y=joI(num1,2);
    end
        distancia_per=0; distancia_per_izq=0; distancia_per_der=0;
       nollegar=0; nollegar2=0;
    for j=min_y:4:max_y,
             i=round((-(j-min_y)/m)+min_x);

             while nollegar==0, %controlar que uno llega y otro no&&
nollegar2==0
                      i_per_izq=i-3;
                      j_per_izq=round(-m_per*i_per_izq+m_per*i+j);
                      distancia_per_izq=sqrt((i_per_izq-i)^2+(j_per_izq-
j)^2);

                 if((sum(sum(image(j_per_izq-1:j_per_izq+1,i_per_izq-
1:i_per_izq+1)))~=0)|| (distancia_per_izq>(medida_pixel/2)))
%                  hline=imdistline(gca,[i_per_izq, j_per_izq],[i,j] );
                   nollegar=1;
                 end
             end
            while nollegar2==0, %controlar que uno llega y otro no&&
nollegar2==0

                      i_per_der=i+3;
                       j_per_der=round(-m_per*i_per_der+m_per*i+j);
                       distancia_per_der=sqrt((i_per_der-i)^2+(j_per_der-
j)^2);
                 if((sum(sum(image(j_per_der-1:j_per_der+1,i_per_der-
1:i_per_der+1)))~=0)|| (distancia_per_der>(medida_pixel/2)))
%                      hline=imdistline(gca,[i_per, j],[-i_per,- j_per] );
                       nollegar2=1;
                 end
            end
             distancia_per=distancia_per_izq + distancia_per_der;
              lista=[lista distancia_per];

        lista=sort(lista);
        lista=lista(1:round(length(lista)/2)); %no big values
        radio=mean(lista);
        lista2=[lista2 radio];
    end

    index=index+2;
end
sol=[];
indice=1;
 while indice<length(medidas_real)+1,
     aux=(lista2(indice)/medidas_pixel(indice))*medidas_real(indice);
     sol=[sol aux];
     indice=indice+1;
 end
```

```
%5h30'indice=indice+1;
 end
```

Anex B: Geogebra

GeoGebra is an application used for statistics, geometry, algebra and calculus by either primaries schools or universities. The desktop version is available in multiple platforms: Windows, Mac OSX, Linux, Android and Ipad.

Before showing the JavaScript code the concept of SVG will be introduced and GeoGebra's interface will be briefly explained.

B.1. SVG

Scalable Vector Graphics (SVG) is an XML based vector image format for two dimensional graphics with support for interactivity and animation. It is an open standard developed by the World Wide Web Consortium (W3C).

SVG images are defined in XML text files, which means, that they can be searched, indexed, compressed and scripted as one. They can be edited and created by any test editor but they are usually created with drawing software, like the ones I mention in section "1.3 Existing Programs".

B.2. Interface

GeoGebra provides different Views for mathematical objects, which are displayed in different representations (e.g. algebraic and graphical) and are linked dynamically. This means that if you modify an object in any of the Views, its representations in the other Views automatically adapt to these changes if possible.

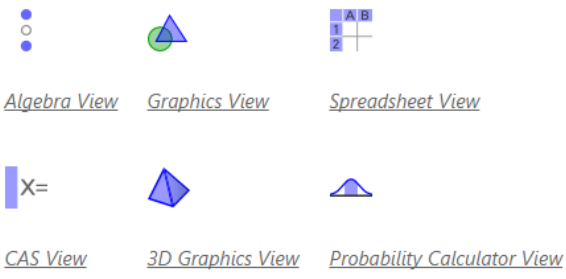


Figure B.2.1 *Geogebra's views.*

The program also comes with a JavaScript command line, accessed by clicking Ctrl and E, where it is possible to change the values and the behavior of the objects.

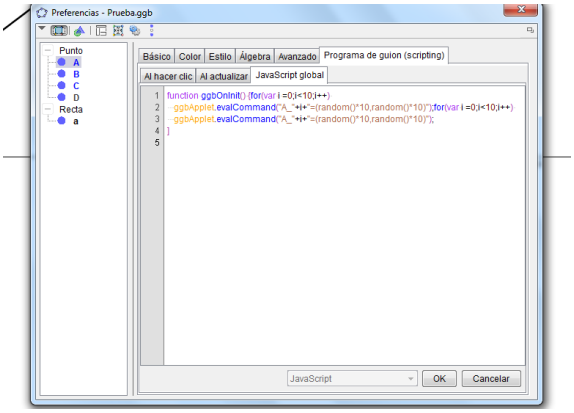


Figure B.2.2 *Geogebra's JavaScript command line.*

The functions to be written for JavaScript corresponds to the Geogebra commands, which can be consulted by using them on an object and later clicking on the object's properties. The following are the Geogebra's tools/commands that were used for the project: CircumcircularArc (creates an arc of three points), Segment, Segment Given the Distance, Intersect, Circle (creates a circle on the point selected with the given radius), Point on Object, Reflect, Conic, PerpendicularLine, Line (creates a parallel line).

Figure B.2.3 *Geogebra's tools.*

B.3. Webpage script

The page elements:

```
<html>
<head>

    <input type="button" value="Predefined Patterns"
onclick="ShowShirtTrouser();">
    <input type="button" value="Customized Shirt Pattern"
onclick="ShowFormShirt();">
    <input type="button" value="Customized Trousers Pattern"
onclick="ShowFormTrouser();">
    <input type="button" value="Save to PDF" onclick="writePNG();">

    <ul id="shirt" style="display: none; list-style-type:none;">
      <li><input type="button" value="Shirt of Size 46"
onclick="shirt46();"></li>
     <li><input type="button" value="Shirt of Size 48"
onclick="shirt48();"></li>
     <li><input type="button" value="Shirt of Size 50"
onclick="shirt50();"></li>
     <li><input type="button" value="Shirt of Size 52/54"
onclick="shirt52();"></li>
```

```
     <li><input type="button" value="Shirt of Size 56"
onclick="shirt56();"></li>
   </ul>
    <ul id="trouser" style="display: none; list-style-type:none;">
      <li><input type="button" value="Trousers of Size 38"
onclick="trousers38();"></li>
      <li><input type="button" value="Trousers of Size 40"
onclick="trousers40();"></li>
      <li><input type="button" value="Trousers of Size 42/44"
onclick="trousers42();"></li>
      <li><input type="button" value="Trousers of Size 46"
onclick="trousers46();"></li>
      <li><input type="button" value="Trousers of Size 48/50"
onclick="trousers48();"></li>
      <li><input type="button" value="Trousers of Size 52/54"
onclick="trousers52();"></li>
    </ul>

      <div id="forms" style="display:none;">
        Please submit the following measurements:<br>
          Chest Circumference: <input type="number" id="chest"><br>
          Neck's back centre to waist: <input type="number"
id="neckWaist"><br>
          Waist to Hip: <input type="number" id="waistHip"><br>
          1/2 of the back's width: <input type="number" id="back"><br>
          Neck's base circumference: <input type="number" id="neck"><br>
          Arm's length: <input type="number" id="arm"><br>
          Length from the shoulder's top to the elbow: <input type="number"
id="shoulderElbow"><br>
          Circumference of the top of the biceps: <input type="number"
id="biceps"><br>
          Scye's Depth <input type="number" value="21" min="21" max="24"
id="scyesdepth"><br>
          Scye's Measure <input type="number"  id="scyeM">
         <button type="button" onclick="validateForm()">Submit</button>
      </div>

       <div id="formt" style="display:none;">
          Please submit the following measurements:<br>
          Waist Circumference + 3cm of fullness: <input type="number"
id="Waistc"><br>
          Hip's total Circumference + 10 cm of fullnness: <input
type="number" id="Hipc"><br>
          Front Hip: <input type="number" id="FrontHip"><br>
          Back Hip: <input type="number" id="BackHip"> <br>
          Inner Thigh Depth: <input type="number" id="Innerthighd"><br>
          Exterior Leg Measurement: <input type="number"
id="Exteriorleg"><br>
          Interior Leg Measurement: <input type="number" id="InteriorLeg">
         <button type="button" onclick="validateForm2()">Submit</button>
       </div>
       <br>
       <br>
<!--<canvas id="canvas"></canvas>-->
</head>
<body >
<script  type="text/javascript" language="javascript"
src="http://www.geogebra.org/web/5.0/web/web.nocache.js"></script>
```

```
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></s
cript>
  <script type="text/javascript" src="jsPDF-
master/libs/png_support/zlib.js"></script>
<script type="text/javascript" src="jsPDF-
master/libs/png_support/png.js"></script>
<script type="text/javascript" src="jsPDF-master/jspdf.js"></script>
<script type="text/javascript" src="jsPDF-
master/jspdf.plugin.png_support.js"></script>
<script type="text/javascript" src="jsPDF-
master/jspdf.plugin.addimage.js"></script>
<script type="text/javascript" src="jsPDF-
master/libs/FileSaver.js/FileSaver.js"></script>
  <script src="Three.js"></script>
    <script src="jsPDF-master/OrbitControls.js"></script>
<article class="geogebraweb" data-param-width="700" data-param-height="700"
data-param-showResetIcon="false" data-param-enableLabelDrags="false" data-
param-showMenuBar="false" allowRescaling="true"
 data-param-showAlgebraInput="false" enableLabelDrags="true"
data-param-ggbbase64="CODE OF THE GEOGEBRA?S DOCUMENT”
  <script >
```

Creating a shirt size 52:

```
function shirt52(){
  //front
  BauxY=-0.4*2;
  IauxX=1.1*2;
  HauxY=0.4*2;
  HauxX=1.1*2;
  FauxX=1.1*2;
  FrontScyeauxX=0.5*2;
  FrontScyeauxY=0.35*2;
  BackScyeauxX=0.5*2;
  BackScyeauxY=0.3*2;
  QauxX=0.5*2;
  QauxY=0.7*2;
  KauxX=0.25*2;;
  KauxY=0.8*2;
  RauxY=0.6*2;
  //Back those of the front that do not appear below are the same values
  I2auxX=0.9*2;
  H2auxX=0.9*2;
  F2auxX=0.9*2;
  Q2auxY=0.6*2;
  CauxY=0.8*2;

  RsauxX=-0.6*2;
  RsauxY=-0.3*2;
  EsauxX=-0.8*2;
  FsauxX=0.8*2;
  HoleBauxX=0.4*2;
  HoleAauxY=0.3*2;
  HoleAauxX=-0.2*2;
  DsauxY=0.6*2;
```

```
  Shirt();
}
function Shirt() {

     var applet = document.ggbApplet;
     applet.setAxesVisible(false,false);
     applet.setGridVisible(false);
     applet.reset();//delete previous objects
     applet.setCoordSystem(-120, 240, -120, 240)
     front(applet);//Set the view
     Back(applet);

  ///Manga
  applet.evalCommand("Bs = (0,140)");
  applet.evalCommand("As=Bs+(0,70)");
  applet.evalCommand("Cs = As+(35.5,0)");
  applet.evalCommand("Left = As+(35.5,0)-(0,70)");
  applet.evalCommand("seg=Segment[Bs,Left]");
  applet.evalCommand("Mid=Midpoint[seg]");
  applet.evalCommand("seg2=Segment[Bs,Mid]");
  applet.evalCommand("Mid2=Midpoint[seg2]");
  applet.evalCommand("seg3=Segment[Left,Mid]");
  applet.evalCommand("Mid3=Midpoint[seg3]");
  applet.evalCommand("Mid4 =Mid2+ (0,70)");
  applet.evalCommand("Mid5=Mid3+(0,70)");
  applet.evalCommand("backLine=Segment[Mid2,Mid4]");
  applet.evalCommand("ForearemLine=Segment[Mid3,Mid5]");
  applet.evalCommand("Dsaux=Mid+(0,70)");
  applet.evalCommand("Ds = (0,0)");
  DsauxX=applet.getXcoord("Dsaux");
  DsauxY=applet.getYcoord("Dsaux")+DsauxY;
  applet.setCoords("Ds",DsauxX,DsauxY);

  applet.evalCommand("centreLine=Segment[Mid,Ds]");
  applet.evalCommand("Esaux=As-(0,51.5)/3");
  applet.evalCommand("Es = (0,0)");
  EsauxX=applet.getXcoord("Esaux")+EsauxX;
  EsauxY=applet.getYcoord("Esaux");
  applet.setCoords("Es",EsauxX,EsauxY);

  applet.evalCommand("Fsaux=Es+(35.5,0)");
  applet.evalCommand("Fs = (0,0)");
  FsauxX=applet.getXcoord("Fsaux")+FsauxX;
  FsauxY=applet.getYcoord("Fsaux");
  applet.setCoords("Fs",FsauxX,FsauxY);

  applet.evalCommand("seg4=Segment[Es,Fs]");
  applet.evalCommand("Gs=Intersect[backLine,seg4]");
  applet.evalCommand("Hs=Intersect[ForearemLine,seg4]");
  applet.evalCommand("HoleAux=Gs+(0,51.1)/6+(0,1.5)");
applet.evalCommand("HoleA = (0,0)");
  HoleAauxX=applet.getXcoord("HoleAux")+HoleAauxX;
  HoleAauxY=applet.getYcoord("HoleAux")+HoleAauxY;
  applet.setCoords("HoleA",HoleAauxX,HoleAauxY);

  applet.evalCommand("HoleBaux=Hs+(0,51.1)/6");
  applet.evalCommand("HoleB = (0,0)");
  HoleBauxX=applet.getXcoord("HoleBaux")+HoleBauxX;
```

```
HoleBauxY=applet.getYcoord("HoleBaux");
applet.setCoords("HoleB",HoleBauxX,HoleBauxY);
applet.evalCommand("seg5=Segment[Es,HoleA]");
 applet.evalCommand("MSeg5=Midpoint[seg5]");
 applet.evalCommand("c1=Circle[MSeg5, 0.5]");
 applet.evalCommand("Saux5=Segment[Gs,MSeg5]");
 applet.evalCommand("MidP=Intersect[Saux5,c1]");
 applet.evalCommand("a_5=CircumcircularArc[HoleA, MidP, Es]");
 applet.setVisible("seg5",false);
 applet.setVisible("MSeg5",false);
 applet.setVisible("Saux5",false);
 applet.setVisible("c1",false);
 applet.evalCommand("seg6=Segment[Ds,HoleA]");
 applet.evalCommand("MSeg6=Midpoint[seg6]");
 applet.evalCommand("c2=Circle[MSeg6, 1.5]");
 applet.evalCommand("Saux6=Segment[Mid4,MSeg6]");
 applet.evalCommand("MidP2=Intersect[Saux6,c2]");
 applet.setVisible("seg6",false);
 applet.setVisible("MSeg6",false);
 applet.setVisible("Saux6",false);
 applet.setVisible("c2",false);
 applet.evalCommand("seg7=Segment[Ds,HoleB]");
 applet.evalCommand("MSeg7=Midpoint[seg7]");
 applet.evalCommand("c3=Circle[MSeg7, 2]");
 applet.evalCommand("Saux7=Segment[Mid5,MSeg7]");
 applet.evalCommand("MidP3=Intersect[Saux7,c3]");
 applet.setVisible("seg7",false);
 applet.setVisible("MSeg7",false);
 applet.setVisible("Saux7",false);
 applet.setVisible("c3",false);
 applet.evalCommand("cs=Conic[HoleA,MidP2,Ds,MidP3, HoleB]");
 applet.evalCommand("seg8=Segment[Fs,HoleB]");
 applet.evalCommand("MSeg8=Midpoint[seg8]");
 applet.evalCommand("c4=Circle[MSeg8, 1]");
 applet.evalCommand("Saux8=Segment[Hs,MSeg8]");
 applet.evalCommand("MidP4=Intersect[Saux8,c4]");
 applet.evalCommand("a_6=CircumcircularArc[Fs, MidP4, HoleB]");
 applet.setVisible("seg8",false);
 applet.setVisible("MSeg8",false);
 applet.setVisible("Saux8",false);
 applet.setVisible("c4",false);

 applet.evalCommand("AuxS= Es-(0,26.45)+(0,2.5)");
 applet.evalCommand("Aux2S= Fs-(0,26.45)+(0,2.5)");
 applet.evalCommand("elbosLine=Segment[Aux2S,AuxS]");
 applet.evalCommand("CrownsPoint= Ds+(0.5,0)");
 applet.evalCommand("Rsaux= Bs+(2,0)");
  applet.evalCommand("Rs = (0,0)");
  Rs1auxX=applet.getXcoord("Rsaux")+RsauxX
  Rs1auxY=applet.getYcoord("Rsaux")+RsauxY;
  applet.setCoords("Rs",Rs1auxX,Rs1auxY);

 applet.evalCommand("Rsaux'=Reflect[Rs,centreLine]");
 applet.evalCommand("Rs' = (0,0)");
  Rs2auxX=applet.getXcoord("Rsaux'")-RsauxX
  Rs2auxY=applet.getYcoord("Rsaux'")+RsauxY;
  applet.setCoords("Rs'",Rs2auxX,Rs2auxY);
```

```
    applet.evalCommand("seg9= Segment[Es,Rs]");
    applet.evalCommand("seg10= Segment[Fs,Rs']");
    applet.evalCommand("seg11= Segment[Rs,Rs']");
    applet.evalCommand("AUXR=Midpoint[seg11]");
    applet.evalCommand("seg12= Segment[AUXR,Rs]");
    applet.evalCommand("seg13= Segment[AUXR,Rs']");
    applet.evalCommand("AUXMR=Midpoint[seg12]");
    applet.evalCommand("AUXM2R=Midpoint[seg13]");
    applet.evalCommand("c5=Circle[AUXMR, 0.5]");
    applet.evalCommand("p2=PerpendicularLine[AUXMR, seg12]");
    applet.evalCommand("c6=Circle[AUXM2R, 0.5]");
    applet.evalCommand("p3=PerpendicularLine[AUXM2R, seg13]");
    applet.evalCommand("MidP5=Intersect[p2,c5]");
    applet.evalCommand("MidP6=Intersect[p3,c6]");
    applet.evalCommand("a_7=CircumcircularArc[Rs, MidP5_2, AUXR]");
    applet.evalCommand("a_8=CircumcircularArc[Rs', MidP6_2, AUXR]");
}
  }
```

Creating a shirt given the measurements:

```
function validateForm() {
    var chest = document.getElementById("chest").value;
    var neckWaist = document.getElementById("neckWaist").value;
    var waistHip = document.getElementById("waistHip").value;
    var back = document.getElementById("back").value;
    var neck = document.getElementById("neck").value;
    var shoulderElbow = document.getElementById("shoulderElbow").value;
    var arm = document.getElementById("arm").value;
    var biceps = document.getElementById("biceps").value;
    var scyesdepth=  document.getElementById("scyesdepth").value;
    var scyeM= document.getElementById("scyeM").value;

   if (scyeM==""||biceps=="" || shoulderElbow=="" ||neck==""|| back==""||
waistHip==""||neckWaist==""|| chest=="" ||arm=="")
    {
        alert("All the fields must be filled out");
        return false;
    }
    else
ShirtC(chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,scyesdep
th,scyeM);
}

function
frontC(applet,chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,s
cyesdepth,scyeM){
      applet.evalCommand("B = (0,0)");
     applet.evalCommand("A=(0,0)");
     applet.setCoords("A",
applet.getXcoord("B"),applet.getYcoord("B")+parseInt(waistHip)+parseInt(nec
kWaist)+2);

     applet.evalCommand("C=A-(0,2)");
     //applet.evalCommand("Daux =C-(0,22)");
     applet.evalCommand("D =(0,0)");
     applet.setCoords("D", applet.getXcoord("C"),applet.getYcoord("C")-
parseInt(scyesdepth));//scyes depth

     applet.evalCommand("E=C-(0,45.5)");
     applet.evalCommand("F=(0,0)");
     applet.setCoords("F",
applet.getXcoord("D")+parseInt(chest)/4+3,applet.getYcoord("D")); //1/4
chest+12
     applet.evalCommand("G=(0,0)");
     applet.setCoords("G",
applet.getXcoord("A")+parseInt(chest)/4+3,applet.getYcoord("A")); //1/4
chest+12
     applet.evalCommand("H=(0,0)");
     applet.setCoords("H",
applet.getXcoord("E")+parseInt(chest)/4+3,applet.getYcoord("E")); //1/4
chest+12
     applet.evalCommand("I=(0,0)");
     applet.setCoords("I",
applet.getXcoord("B")+parseInt(chest)/4+3,applet.getYcoord("B")); //1/4
chest+12
```

```
     applet.evalCommand("s = Segment[C, B]");
     applet.evalCommand("s2 = Segment[B, I]");
     applet.evalCommand("s3 = Segment[I, F]");

     applet.evalCommand("s5 = Segment[D, F]");
     applet.evalCommand("J=(0,0)");
      applet.setCoords("J",
applet.getXcoord("C")+parseInt(neck)/4+0.7,applet.getYcoord("C"));//1/5
neck base +0.7
     applet.evalCommand("K=J+(0,2)");
     applet.evalCommand("Reflect[K,s]");
     applet.evalCommand("a_1=CircumcircularArc[K', C, K]"); ///cuello
trasero
     applet.setVisible("a_1",false);
     applet.evalCommand("L=C-(C-D)/2");
     applet.evalCommand("BackScye=(0,0)");
     applet.setCoords("BackScye",
applet.getXcoord("L")+parseInt(back),applet.getYcoord("L"));     //1/2 back


      applet.evalCommand("N=(0,0)");//A-1/8 of scyce depth +0.75
     applet.setCoords("N", applet.getXcoord("C") ,applet.getYcoord("C")-
parseInt(scyesdepth)/8+0.75);
     applet.evalCommand("O=(0,0)");
     applet.setCoords("O", applet.getXcoord("N")+parseInt(back)
,applet.getYcoord("N"));
      //P=interseccion per
      //P=interseccion perpen O a DF
     applet.evalCommand("p=PerpendicularLine[O,s5]");
     applet.evalCommand("P=Intersect[p,s5]");
     applet.evalCommand("Q=O+(1.8,0)");


     applet.evalCommand("circle=Circle[P, 3]");
     applet.evalCommand("sOP=Segment[O,P]");
     applet.evalCommand("auxOP=Reflect[N,sOP]");
     applet.evalCommand("sPN=Segment[auxOP,P]");
     applet.evalCommand("Point=Intersect[circle,sPN]");
     applet.setVisible("circle",false);
     applet.setVisible("sOP",false);
     applet.setVisible("auxOP",false);
     applet.setVisible("sPN",false);

     applet.evalCommand("FrontScye=P+(0,8)");
     applet.evalCommand("R=(0,0)");//1/5 of necks circumference+0.8
     applet.setCoords("R", applet.getXcoord("C") ,applet.getYcoord("C")-
parseInt(neck)/5+0.8);


     applet.evalCommand("a_3=CircumcircularArc[K', R, K]");
     applet.evalCommand("s6 = Segment[K,Q]");
     applet.evalCommand("s7 = Segment[L,BackScye]");
     applet.evalCommand("s8 = Segment[E,H]");
     applet.evalCommand("Reflect[s6,s]");
     applet.evalCommand("Reflect[s7,s]");
     applet.evalCommand("Reflect[s8,s]");
     applet.evalCommand("Reflect[FrontScye,s]");
     applet.evalCommand("c=Conic[Q,BackScye,FrontScye, Point, F]");
```

```
     applet.evalCommand("Reflect[I,s]");
     applet.evalCommand("Reflect[c,s]");
     applet.evalCommand("Reflect[s5,s]");
     applet.evalCommand("Reflect[s3,s]");
     applet.evalCommand("s10 = Segment[I',B]");}
function
ShirtC(chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,scyesdep
th,scyeM) {

     var applet = document.ggbApplet;
     applet.reset();//delete previous objects
     applet.setAxesVisible(false,false);
     applet.setGridVisible(false);
     applet.setCoordSystem(-120, 240, -120, 240)

frontC(applet,chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,s
cyesdepth,scyeM);
BackC(applet,chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,sc
yesdepth,scyeM);

sleeveC(applet,chest,neckWaist,waistHip,back,arm,neck,shoulderElbow,biceps,
scyesdepth,scyeM);
}
```

Anex C: Blender & MakeHuman

Blender is professional free and open-source 3D computer graphics software. It is used for creating video games, animated films, art, 3D printed models and visual effects. Blender's features include: Photorealistic rendering, UV unwrapping, sculpting, fast modeling, video editing, camera tracking and modelling. Further features can be integrated by the use of libraries.

MakeHuman is an open source 3D computer graphics software middleware designed for the prototyping of photo realistic humanoids. It is developed by a community of programmers, artists, and academics interested in 3D modeling of characters.

C.1 Installation

C.1.1. Program

The version used in this project is the 2.77a which was released the 6th of April. Blender is available for Windows, MAC OSX and GNU/Linux in their webpage; I will explain how to install it for Windows. Download either the ZIP or the installer (.msi) for the bit version of your Operating System. Later just unzipped or follow the instructions of the installer.

C.1.2. MakeHuman add-on

As stated before, it is possible to add new features/tools to Blender. In this case I will be adding the Makeclothes add-on. For that it is first need to copy the folder blendertools, situated in the MakeHuman folder which contains four subfolders: makeclothes, maketarget, makewalk and mhx_importer. Only copy the subfolders to the addons folder of Blender.

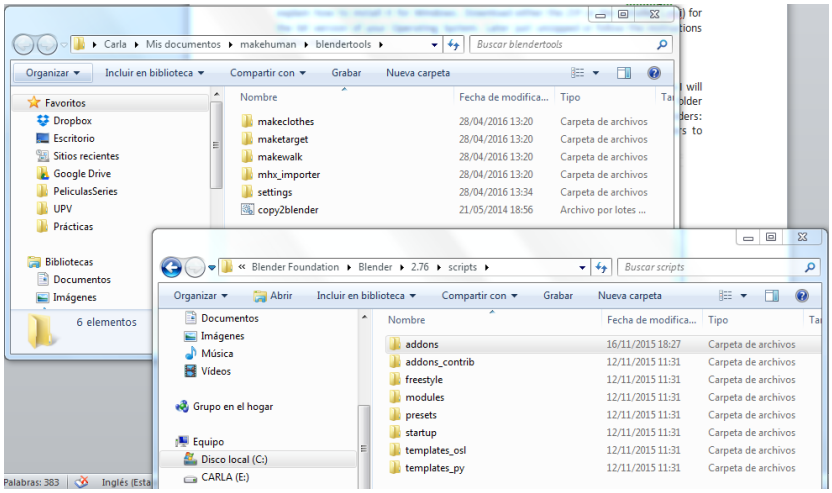


Figure C.1.2.1. Makehuman folder.

These Blendertools are Python scripts for Blender and have to be activated from the application. From the File menu click on User Preferences →Add-ons. Look for Makewalk, Makeclothes and Maketarget, activate them by checking the box and saving the settings.

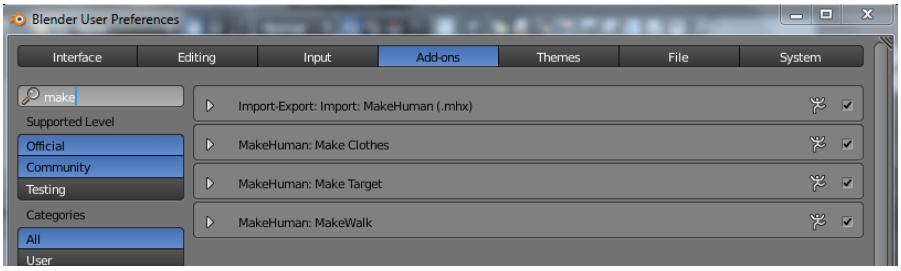


Figure C.1.2.2. Blender Add-ons

Now these tools should appear on Blender's tool panels. Situated at the right and opened by clicking on the '+' Symbol.

Figure C.1.2.3. MakeHuman tools on Blender

C.1.3. Macro Recorder add-on

The macro recorder addon is a python script that will record the work done on Blender as a pyhton script. It is very helpful when any problem arise while working on it.
The installation it is done from the User Preferences, click on Install from File and select the file. Then cick the Macro Recorder checkbox.

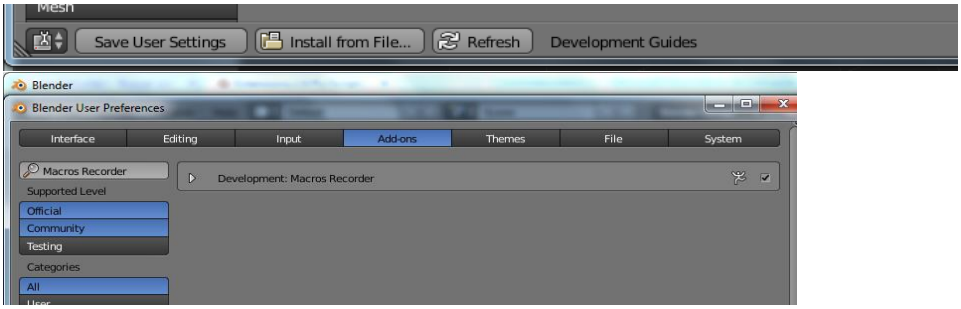


Figure C.1.3.1. Macro-recorder Add-on.

When enabled, it would add "Record Macro" menu entry to the Text menu in the Text Editor, and a panel in the Tool Shelf with the buttons to record/stop recording and to regenerate active procedural object.

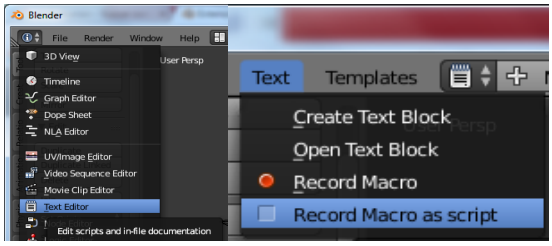


Figure C.1.3.2. Use Macro-recorder Add-on.

C.1.4. Exporting SVG

This add-on will allow us to import our Objects to SVGs. The installation works as the Macro Recorder one, from the User Preferences, click on Install from File and select the SVG importer file. Then click on the Import-Export:Scalable Vector Graphics (SVG) checkbox, once it is checked, the function will be available through File→ Import.

C.2. Interface

The Blender interface is organized into one or more Areas, within each region there is an Editor, this composition is called a Screen Layout. Editors (5 by default) are divided into two: one Header and one or more Regions. This last one may have smaller structuring elements like panels with buttons, controls and widgets placed within them. The previous description is shown in the following image:

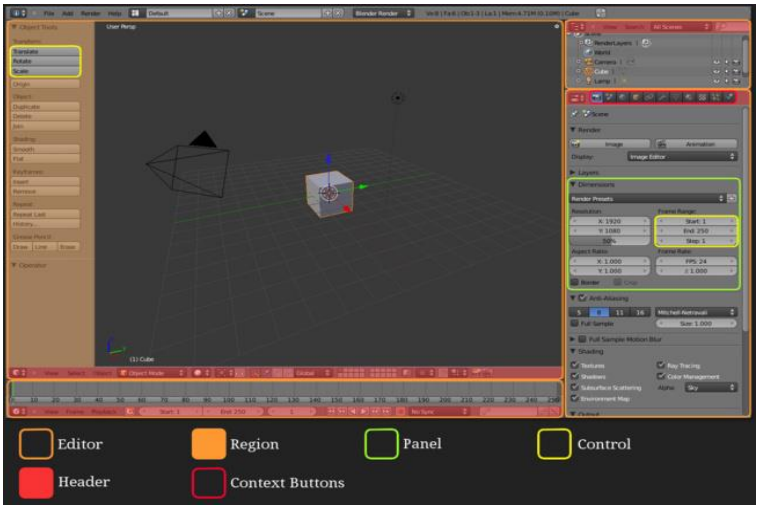


Figure C.2.1. Blender's Interface.

But the most important part will be the modes, the viewport shading and the context buttons.

C.2.1. Modes

Modes are a Blender-level object-oriented feature, which means that there cannot be two or more modes active at the same time and that the available modes vary depending on the selected active object's type. Each mode is designed to edit an aspect of the selected object and the default one is Object mode.

You can only select object in the Object mode, when the rest of the modes are activated the object selection is "locked" and can't be changed.

Modes can affect many things in Blender:

- They could modify the panels and/or controls available in some Buttons window's contexts.
- The behaviour of whole windows could change.
- They can modify the available header tools.

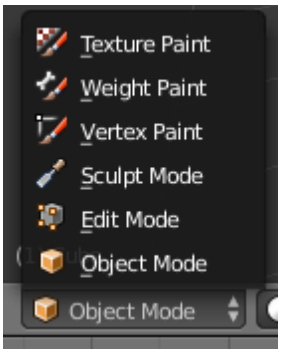


Figure C.2.1.2. Blender's Modes.

In this project only the Edit mode and the Object Mode were used. The Edit mode is available to all renderable objects and allows modifying textures and vertexes. As for the Object mode, this one is available for all objects and is dedicated to Object data block edition like resizing, rotating or positioning.

C.2.2. Viewport Shading

The viewport shading changes the way objects are drawn and lit in the 3D View.

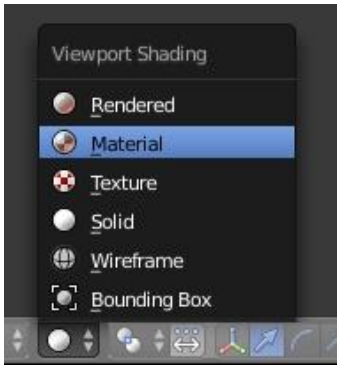


Figure C.2.1.2. Blender's Viewport shading.

Three of these options were use in the project: Texture, Material and Solid shading. All these shades depend on the lighting of the scene. The Material shading is an approximation of the applied material. Texture shading shows meshes with an image applied using the UV Mapping option. Finally, Solid shading is the default drawing mode using solid coloured surfaces and simple lighting.

C.2.4. Context buttons

The Buttons window shows six main contexts, each one of them might be subdivided into a variable number of sub-contexts.

The sub-context will be determined by the object selected and the mode. For example, when a mesh is in Object mode the materials in the material context will not be able to be selected. Whereas in Edit mode you can select and deselect as many material as you want.

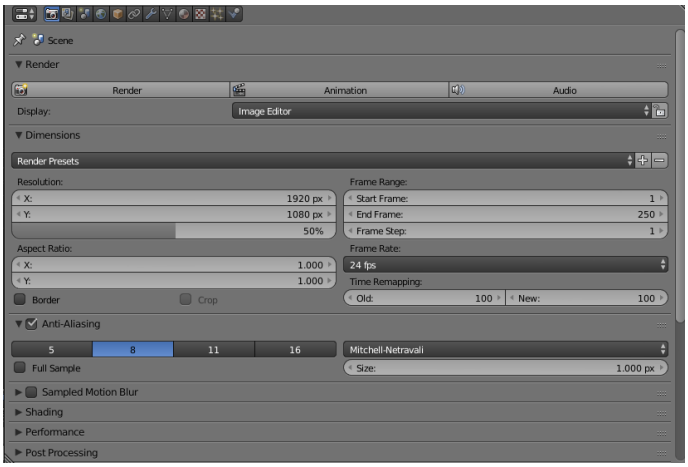


Figure C.2.4.1 Blender's render conext and sub-context.

C.3. Command's shortcuts

In order to use Blender properly it is necessary to be using a mouse and a keyboard. Most of the actions that are needed for this project can be done by using shortcut. The following are the most used ones:

- Alt and A: start animation.
- Right mouse click on object: select object.
- Right mouse click, Ctrl and '+' or '-': vertex selection of the selected object.
- Numbers 1 to 8 move the camera's angle.
- Mouse wheel or '+': move camera closer.
- Move wheel or '-': move camera backwards.
- Shift and mouse wheel: move camera up and down.
- Ctrl and mouse wheel: move camera left or right.
- S: scale, moving the mouse forward or backwards. Once you are done left click on the mouse to leave it.
- E: extrude selection (similar performance to scale).
- R: rotate object.
- A: select or deselect all the objects in the scene.
- M: move object to another layer.
- Ctrl and I: Inverse selection.
- P: separate object from its parent.
- Ctrl and Z: undo action.
- Ctrl and S: save file.
- Shift and D: duplicate object.
- Alt and C: convert object to mesh (used for creating clothes out of patterns).
- Crtl and J: join to objects together.
- W: special commands.

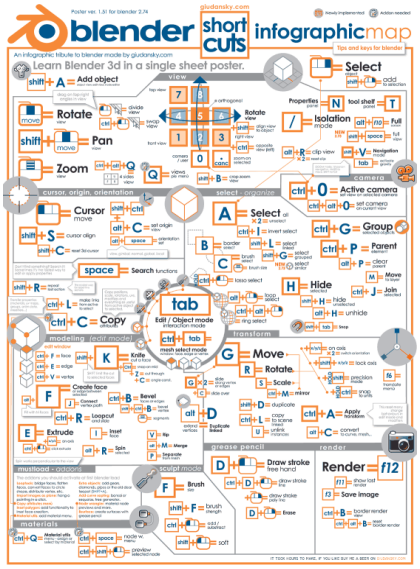


Figure C.3.1. Blender's shortcuts.