vMannequin: a Fashion Store Concept Design Tool

Paolo Cremonesi, Franca Garzotto,
Marco Gribaudo, Pietro Piazzolla
Dipartimento di Elettronica,
Informazione e Bioingegneria
Politecnico di Milano
via Ponzio 31/32, Milano, Italy
Email: {paolo.cremonesi,franca.garzotto,
marco.gribaudo,pietro.piazzolla}@polimi.it

Mauro Iacono
Dipartimento di Scienze Politiche
Seconda Università degli Studi di Napoli
viale Ellittico 31, Caserta, Italy
Email: mauro.iacono@unina2.it

KEYWORDS

Concept Design; End User Development; 3D Computer Graphics

ABSTRACT

The fashion industry is one of the most flourishing fields for visual applications of IT. Due to the importance of the concept of look in fashion, the most advanced applications of computer graphics and sensing may fruitfully be exploited. The existence of low cost solutions in similar fields, such as the ones that empower the domestic video games market, suggest that analogous low cost solutions are viable and can foster innovation even in small and medium enterprises. In this paper the current state of development of vMannequin, a dynamic, user mimicking, user enacted virtual mannequin software solution, is presented. In order to allow users designing dress concepts, the application simulate the creation and fitting of clothes on virtual models. The interaction is sensor based, in order to both simplify the user interface, and create a richer involvement inside the application.

INTRODUCTION

The fashion industry is one of the application fields in which proper IT applications may be a strong enabling factor for innovation. Despite the information content of its final products is low, due to the peculiarly physical nature of pieces of cloths, there is a significant margin to be exploited by means of IT solutions in the production, sales, after sales support and services areas, due to the high information content of the related processes.

The availability of low cost computers capable of complex, real time graphical manipulation of realistic, dynamic 3D models, together with the availability of low cost and low impact positioning and movement sensors enables a number of innovative applications to support the processes of fashion design, being it in the phase of conceptual shaping of haute couture or mass market pieces of clothes or in the phase of personal outfit style shaping, and sales, in presence or by means of e-commerce web stores or virtual reality applications. Such solutions potentially offer many advantages; they may reduce the development cost for products, as much as the piece of cloth is made of expensive materials or needs a complex manufacture, by reducing the need for prototyping; they may empower low cost tailor made production; they may meet the needs of customers with special needs by custom, unconventional, or even out of market pieces of clothes; they may allow small producers to emerge and acquire vertical market shares or reach a wider customer base; they may lower the barriers of the market for emerging stylists or firms; they can enable the creation virtual firms producing or assembling pieces of clothes or total looks by means of remote collaborative crowd production or crowd design. The benefits for a scattered (from the point of view of the size of the producers and from the point of view of price categories) market, like the Italian one is, are unpredictable, but possible developments may reasonably be considered to be interesting.

The availability of visual tools that allow the simulation of dressing up a customized mannequin and the effects of fabric, colour, cut on it while its movements mimic the movements of the user may improve the sales process, widening the potential market even for small firms or shops and lowering the TCO of show rooms by potentially minimizing the crowd and the waiting time for customers, and paves the way to more advanced technologies such as automatic shop assistants or sales advisors, personal shopper support systems or remote fashion advising.

In this paper we present the current state of development of vMannequin, a computer application designed to help fashion store customers to design the dress concept of their next purchase. The final goal of vMannequin is to be used in virtual fitting rooms, as an in-store solution, while the goal of the project is to obtain a flexible, computer based support for advanced virtual fashion applications. At the state, the focus is on enhancing the real time aspects of clothes dynamics, to increase the realism of the moving virtual mannequin.

The original contribution of this work is thus the description of the architecture of a customizable application that: i) allows the users to create, design and test new clothes on a virtual mannequin; ii) allows a data-driven configuration to customize it for different fashion contexts without requiring the writing of extra code; iii) provides a robust and innovative interaction model that can exploit available sensors and actuators to provide the final user with an immersive experience.

Proceedings 30th European Conference on Modelling and Simulation ©ECMS Thorsten Claus, Frank Herrmann, Michael Manitz, Oliver Rose (Editors)
ISBN: 978-0-9932440-2-5 / ISBN: 978-0-9932440-3-2 (CD)

RELATED WORKS

Research and enterprise both have focused on virtual fitting rooms since more than a decade[1]. Many application that implements this idea are currently on market, sometimes very different one another in terms of goals and technologies involved. Some are conceived as plug-ins for e-commerce web sites, like e.g.: Virtusize [2], others as full web services like Fitnect[3]. In other cases still, like e.g. Fit.me[4], these web application leverages on robots able to simulate the size of the users. Augmented reality is enabled in case of products like Swivel [5], while triMirror [6] resorts on virtual avatars. However, it is difficult to find any technical details of these systems. For space constraints, we limit the related work analysis to those academic works closely related to ours. Dress dynamics in real time is considered by some authors, like e.g.:[7], where a physical based approach is used to realize real-time virtual try-on of garments by user interaction. Their approach is different from ours since the intended use of their application is to test dresses as they are designed by professional, moreover they do not specifically address animation of the virtual bodies in use. The introduction of the Microsoft Kinect sensor introduced novel interaction strategies, that are currently under investigation. In [8], the authors use an high definition camera to record the movement of the user, while the Kinect sensor analyze it. The analysis is then used to compute dynamic dress fitting which is then composed on the camera recoding. Even if the basic installation setting of the application is closely related to ours, their approach is not in real-time like ours. Moreover they do not address customization of dresses. The same overlapping technique is also exploited in [9], differently from our approach that preferred a virtual mannequin for the fitting. More recently, in [10] authors exploit the use of the virtual avatars for the fitting. This work lacks the visual appeal that is one of the focus of our proposed system, but introduced the use of real-time virtual body animation. Differently from it, however, we preferred an approach based on the recognition of a movement that triggers the closely matching animation present in a database of animation, instead of matching the user on time. In this way we avoided the animation artifacts that influenced their work.

THE SIMULATOR

The vMannequin simulator is intended to provide support to end-users involved in fashion concept design. The application will provide them with a variety of 3D assets, ranging from dresses to props, from shoes to hair styles that can be easily but thoroughly customized. A virtual 3D model, male or female as chosen by the user, can be dressed with the selected assets to show how they fit. The 3d model is displayed on a big screen, animated in real-time and can mimic users movements. To enhance the realism of the simulation, a great care has been devoted in implementing the visual part of the application using state-of-the-art shading techniques. Dress dynamics is also considered for the same reason.

In this section we describe the vMannequin simulator high-level architecture, highlighting the elements that compose the application and the environment where it is run, as well as their interaction. Figure 1 presents the application architecture, that the next Sections will describe into details.

Sensors

Essentially, two types of interaction are required to be handled. The first is the *customization interaction,* that is the interaction to customize the dresses before sending them to the 3D character for fitting. The second type concerns *animation interaction*, that is the sequence of gestures made by the user that causes the character to animate. Different currently on-market interaction devices can be adopted to these ends, ranging from QR-code reader or sound and motion recognition sensors to smart displays. We group them all under the definition of 'sensors' to focus on their ability to capture inputs from users and decode them into parametrized triggers for the application. The presence of different sensors is important since the customization interaction may require a deeper involvement as well as more precision gestures by the user, compared to animation interaction.

System Configuration

The simulator relies on an asset manager, for example a database, to handle the elements involved in visualization and customization of the the 3D models. This part of the application is intended to be transparent to the final user, but requires careful planning from the point of view of application developers. Since in this context developers requires not only programmers and IT technicians, but also 3D and computer graphics artists, as well as fashion and design experts which rarely have sufficient coding skills, the application has been developed to be *data-driven*. The goal is to have a simulator that can be completely configured, in term of types of assets available, sensors used and interaction models, by only inserting proper information in the database. This is to allow the maximum deployment flexibility in different kinds of fashion retail shops. After all, an high-end fashion store has different requirements in terms of customization options, animations, gestures to be recognized than a sport store.

The elements, or assets, required by vMannequin can be divided into four broad categories: Characters configurations, Animations, Gestures and Dresses. Since assets belonging to a category may need information stored for another category or needs to be related with it, a specific category of data is required to handle this inter-category communication: the Orchestration.

- *Characters configurations*. The virtual models to be used to show the dress fitting are essentially one male and one female 3D virtual model that can be adapted to the user's needs. Features like weight, height, age, eye colors, skin tone, tattoos, nails colors can be customized as required and the database stores all the associated parameters. Depending on the degree of customization allowed, this category can require a larger or a smaller storage space.
- *Animations*. Virtual models are not static meshes but can be animated. Even if it is possible to let the 3D virtual models simply mimic the user's movements this is not advisable. Animation directly mapped from a sensor's body detection capability to the virtual model may result in visually awkward movements, that may break the simulation realism. Instead we propose an indirect mapping between the sensor readings and

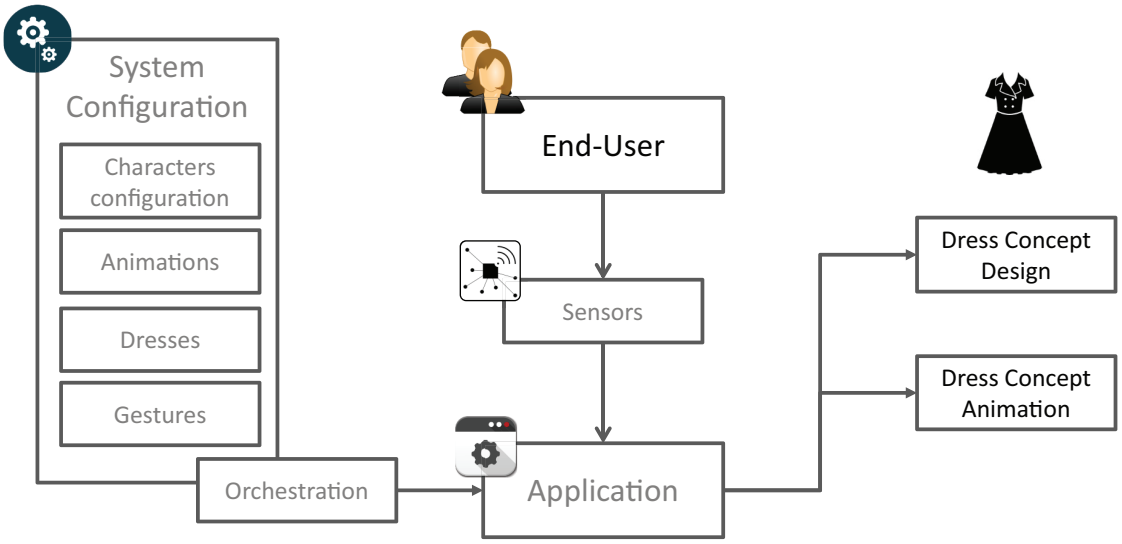


Fig. 1. The Simulator Structure

the animation played. The sensor recognizes an input pattern made by the user and the application select which of the database's available animations is the closest match. In this way, user's actions still affects the virtual mannequin in real-time but the realism of the movement is not compromised.

- *Dresses*. This category constitutes the bulk of the database, and again its size depends on the number of dresses, props, hairs the application is set to use. The meshes, i.e. the geometric definition of the 3D objects, can be further divided between *conforming* or *dynamic*. To the first group belong those dresses which follows skin-tightly the 3D virtual models (such as a pair of leggings). From a computational perspective they are the less expensive to handle because require the same animation techniques used for the virtual characters. Dynamic dresses, on the other hand, try to reproduce the majority of clothes physical characteristics and thus require specific management techniques, often resulting in an higher storage space requirement. In Figure 2 an example of the two kind of dresses is presented.

- *Gestures*. The interaction with the application is governed by gestures. Different type of sensors can allow different types of interactions. Full body sensors can recognize the position of the user, and return the orientations and the positions of the various joints that corresponds to the head, the torso, the arms and the legs. They can also recognize a sequence of movements as a specific action performed by the user. Speech recognition sensors might return identify words and sentences pronounced by the user and return. Pressure sensors or other haptic devices might return other interactions performed by the user. In this work, we will imagine that all the sensors will be able to identify a finite set of actions performed by the user. We will call *gestures* the data required to configure the sensors to identify the action performed by the user: for full body sensors, they correspond to sequence of

a. b.

Fig. 2. The difference between conforming (a.) and dynamic (b.) clothes.

positions assumed by the user; for speech recognition sensors, they correspond to the vocabulary that must be recognized; and so on. Since the application might be configured to be used in different contexts (i.e. a sport goodies store, a bride-dress manufacturer, a department store), different gestures might be required

(a sport store customer might want to run or dance, while a bride-to-be might want to throw a bucket). The gesture data-base holds the specific gestures for the considered configuration. These gesture might trigger animation and configuration steps.

- *Orchestration*. The orchestration category holds the information required to connect the gestures, to the dress selection and configuration, to the characters and to the animation. Since the goal is to simplify the customization of the application using a data-drive approach, it exploits a formal specification (that will be described in next Section) that allows the setup of the interaction model without the requirement of programming skills.

THE APPLICATION

The purpose of the application is to enable the two objectives of the simulator: the design of a dress concept and the animation in real time of the result. To this end, it handles the following tasks: render the image on the big screen device, load and unload the assets from the database, allow the customization of these assets, react to the inputs received from interaction devices, play the animations. The rendering task requires a good trade-off between performances vs realism. One of the key factors of the vMannequin simulator is its ability to attract and involve shopper, hence the need for a state-of-the-art visual quality coupled with an immediate reaction to user's gestures. Visual quality can be easily achieved nowadays but comes with an higher requirement cost in terms of per asset storage space, which in turn translates in a possibly higher loading times before an element is displayed on screen. To this end the integration with the database is a critical issue, especially when considering the loading time of precomputed dynamic dresses. This is a non-trivial challenge that has been successfully addressed by producing a proprietary binary file format, which allowed, along with other techniques described in Section -A to reduce it considerably. In particular we have used principal component analysis based techniques to compress the animations, as well as a near-exhaustive precomputation of secondary cloth effects [11].

A. The database architecture

The database (which in this case can also be seen as a *file system*) contains all the assets required by the application as shown in Fig. 3. Note that the figure is not a classical entity-relationship diagram, since the configuration DB is not a relational database. In fact the configuration DB is a NoSQL database whose description goes beyond the scope of this paper.

To present the configuration DB we start with the table of the characters configuration component, that is represented with blue boxes. The *Character* box represents the type of characters that are available in the specific simulation. It usually includes two items (one for the male and another for the female), but can be increased depending on the context (for example to include young teens or children). Each element of the Character table also includes a small image that the application can use to show the preview of her selection. The *Geometry* box includes the meshes used by the application. Each character must be connected to one and only one element of the geometry table. However the geometry table also holds data for other 3D objects that are used in the simulator such as dresses and add-ons which will be described later. Geometry data includes all the specifications required to properly draw the 3D objects: it includes a scene tree composed of several hierarchically interconnected nodes and mesh pointers, a set of index buffers, a set of vertex buffers. For characters and conforming clothes it also includes a bone hierarchy and a binding pose expressed using offset matrices [12]. Vertices have a variable format that always includes the positions, and accompany it with the directions of vertex normals, the UV coordinates. For characters and conforming clothes vertices includes also a set of up to four indices to influencing bones, and the weight using to blend final pose. The character configuration also includes the *Texture* box that, as the name suggests, includes all the texture sets associated to a character. Texture elements are collections of several images used by the shader to produce the final render: they usually includes diffuse color map, transparency map, normal and bump map and specular reflection map. Each character can be associated to more than one texture to allow simple customizations like changing the tone of the skin or the color of the eyes. Each texture set is thus also characterized by a preview image. In special circumstances characters might not have any texture involved: in this case the application will allow the user to select a fixed color.

The dresses components are represented in Fig. 3 with green boxes. The *Dress* table contains one element for each dress supported in the configuration. Again, each element of the dress table also includes a small image to preview its appearance. As for characters, each dress must have a pointer to an element of the geometry table and might have a pointer to one or more texture sets. As previously introduced, dresses are divided into two main types: conforming and dynamics. However, the dress table includes also a third type of element, called *body features*. The latter is used to add body features like hairs, piercing or tattoos. Body features can either be conforming or dynamic (to support dynamic hairs). In case of tattoos, the features might not have associated a geometry. Body features are implemented by the simulator as normal dresses: however the distinction allows the introduction of the elements in different locations of the user interface, and prevent them to be considered as clothe features to be exported to the real dress production plan. Not all dresses could be tailored for all characters (e.g. a kids clothe cannot be fit on an adult): for this reason the table *Can fit* represents a many-to-many relation that defines which clothes can be worn by a given character. Dresses are also grouped into different types, as defined by the table *Clothe Type*. In this way the application can present the clothes grouped into different categories, considering for example tops, trousers, leggings, socks, shoes and so on. The table *Clothe clash* contains instead couples of clothe types that cannot be worn together. It avoids for example to fit at the same time two evening dresses, one on top of the other. Beside changing the texture, clothes can also be configured as specified in the table *Add-on*. Each dress might have associated zero or more add-ons, which are divided into two types: *Decals* and *Props*. As for other customizable elements, they are both characterized by a preview image. Decals are simple texture overlays that can be superposed to the fabric to

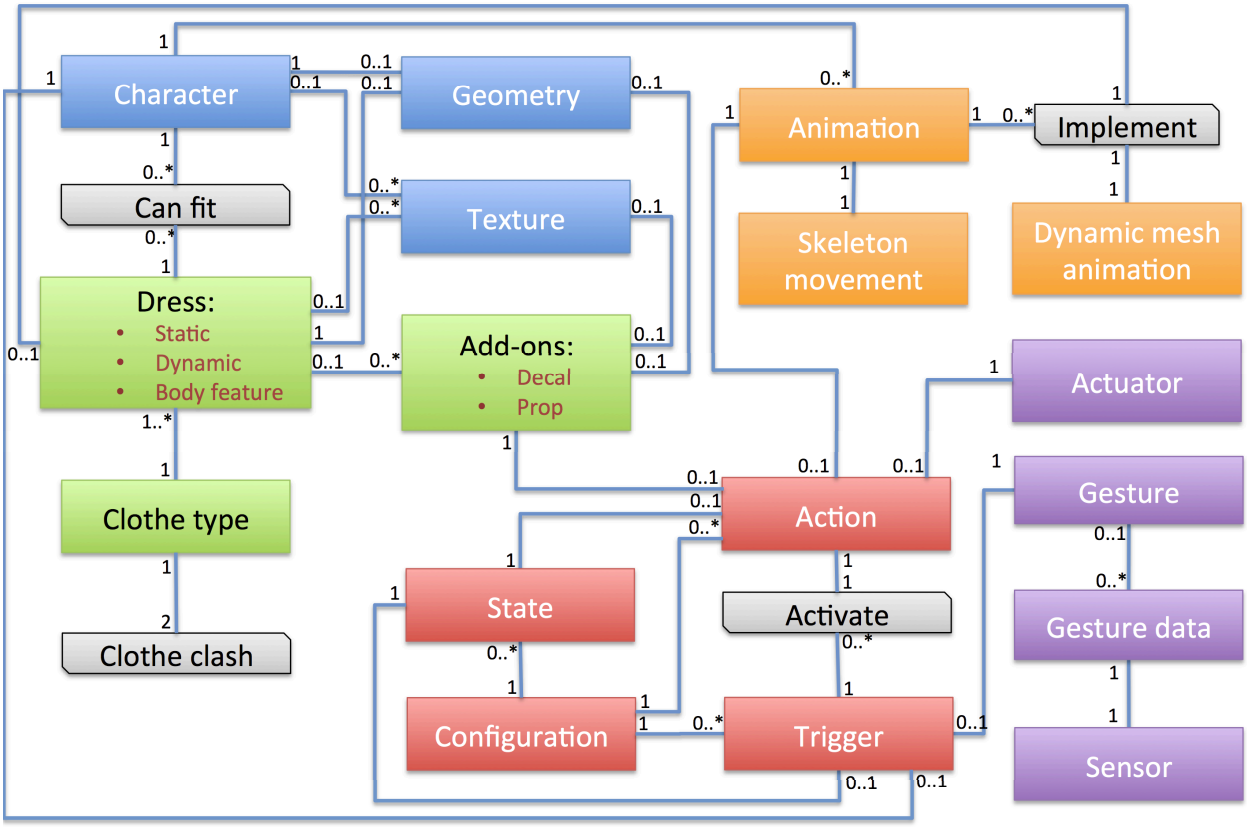


Fig. 3. The Application Database: character configurations (blue), animations (orange), dresses (green), gestures (purple) and orchestration (red).

create an high level, but controlled, degree of personalizations. The application allows to move, rotate and scale the decal on the surface of the dress. Since decals are basically extra textures, they are connected to one and only one element of the Texture table. Props are instead pieces of geometry that can be superposed to the dress (such for example an extra button or a pendant). The application allows to position and rotate them, keeping them anchored to the surface of the dress. Props requires a pointer to one and only one element of the geometry table, and can point to one or more elements of the texture table to allow the selection of different texture sets to further customize the add-on.

Elements of the animation component of the DB are represented with orange boxes in Fig. 3. In particular, each animation that a character can perform has a corresponding element in the *Animation* table. Each animation is characterized by its length in frames, and it must be associated to a skeletal animation that is stored into table *Skeleton movement*, which holds the positions, rotations and scaling of all the bones associated with a character. Dynamic clothes must also have associated an element of the *Dynamic mesh animation table* that stores the compressed deformations of the dress to produce a realistic effect. Elements of table *Implements*, associate the deformation to a given animation for a specified dynamic dress.

The gesture component of the DB is shown with purple boxes in Fig. 3. Table *Sensor* and table *Actuator* include respectively the description of all the sensors and actuators connected to the application. Each of the element of both tables select among a possible set of sensors (i.e. a Microsoft Kinect, a Midi device, a step-up motor, a LED based colored dynamic illumination system) preconfigured in the application, that can be used in the specific configuration. Elements of the *Gesture* table hold instead the possible user interactions that can be identified by the application: they can for example correspond to the detection of the wave of the arm, or the movement of a joypad, the interaction with central-left portion of a smart screen and so on. Each gesture must be then defined through a set of sensor-specific parameters contained in the *Gesture data* table. Note that the application defines a special type of sensor/actuator: the *timer*. Timers are special actuators that can be started, and produces a sensor reading when a given time elapses. This can be used to trigger special actions for example when the user does not interact with the application for a prolonged time.

In Fig. 3, red boxes represents the orchestration part of the DB. In particular, the application can run in one of several configurations, each one represented by an element of the *Configuration* table: this may allow the shop owners to run the installation in different operating modes, depending on the expected affluence in number and type of customers. Each configuration is characterized by a finite set of states stored in the *State* table. This allows to use state-machines as a theoretically proven effective mechanism to store the cause-effect interactions of the configuration in a data-driven way. In particular, the application can perform a set of actions that are defined in the *Action* table. An action can either activate

an actuator, play an animation or change the current state (as shown with the possible connections with the elements of the corresponding tables). Actions are started by an element of the *Trigger* table. Triggers are fired whenever the associated gesture is detected. The effect of the trigger can be confined to be effective only in a given state, or for a given character. Finally, each trigger can fire more than one action, as specified by the elements of the *Activate* table.

Users and their Experience

In Figure 4 it is possible to preview the expected installation of vMannequin in a fashion store. The user is detected by the application (In Figure 4-A) which exits its idle status and becomes active. A short set of instructions can be provided to the user at this point. If the user decides to interact with the simulator, it is prompted by the request for the specification of some characteristics, like gender, height, size, age, for the virtual 3D model used for the fitting. This step can be substituted by automatically detecting the user's features but this may introduce more problems than not. What if the dress to be purchased and that the user wants to virtually fit is not meant for her but as a present for someone else?

After this first stage, a catalog of the available customizable assets (mainly dresses, but shoes, hair styles are available too) is presented to the user (In Figure 4-B). The user is supposed to interact with the application for as long as required to be satisfied with the options selected. Each time an asset is completed, it can be sent to the 3D character and seen fitted. The user is not required to completely dress the character, and can see the real-time fitting and animation at any moment.

In the last stage, the 3D character is dressed up to the point the user needs (In Figure 4-c). At this point her experience is concluded and the results of its interaction, both in terms of pictures, both in terms of a checklist with the chosen dresses and their customization can be send to her mobile, mail account or other preferred communication methods.

PROOF OF CONCEPT

In this Section we present the proof-of-concept application that implements the most relevant features introduced in the previous Section. In Figure 5 the architecture of this system is shown. The goal of the application is to verify the technical feasibility of the proposed model.

The application has been developed using Microsoft Visual C# [13], while users can interact with the simulator by means of a Microsoft Kinect II for XBOX ONE sensor [14]. The sensor offers a wide range of detection features that can be used to implement both the customization and the animation interaction types. Other interaction devices are currently under study, since the customization procedure, because of its specific needs, can be demanded to a different device such as a smart screen or a mobile device connected to the application. The same big screen on which the simulation result is displayed can be also 'smart' to allow the customization of interactions.

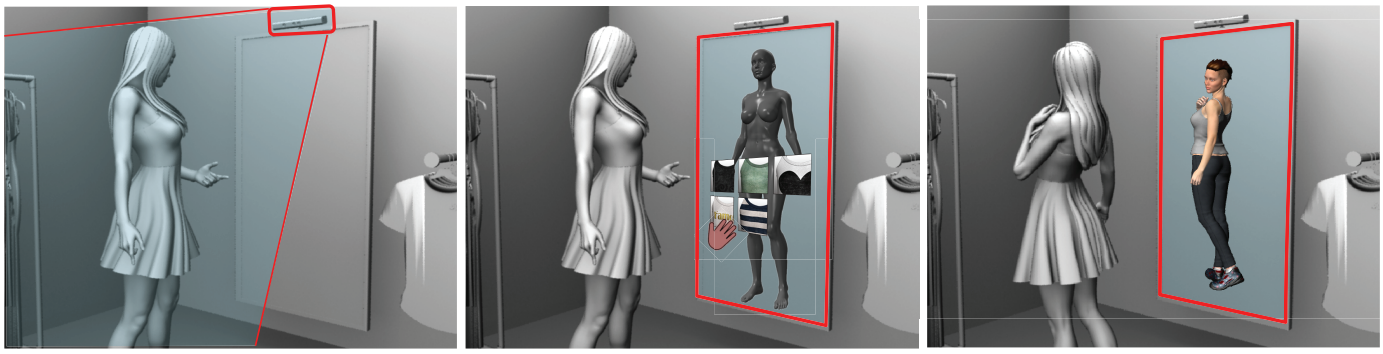
Rendering is performed using a specifically developed rendering engine tailored on the specific requirements of the simulator, in terms of performances vs realism, and uses simple but effective three points light model to produce believable images. To allow the maximum deployment flexibility on different hardware, the OpenGL graphic library has been chosen, accessed through the the OpenTK [15] wrapper. Since it is widely supported by different display adapters, the use of OpenGL could be of benefit especially in case of a future integration with mobile displays.

In order to have high quality models for the asset manager, we leveraged the repository of Daz 3D, a software dedicated to morphing, posing, animating virtual characters. The general idea is to use easy-to-find detailed objects that can near as much as possible the clothes that the retail store wants to show with the application. With a little extra effort by modeling professionals, those objects can be customized to exactly match the clothes in stock. Currently there are different similar programs, e.g. Poser, Bryce, MakeHuman, so there is a high number of available dress models, hair styles, props and other elements that can be reused.

Because of this, the origin of the used 3D models can be very different, introducing the challenge of having many different file formats to handle for the asset manager. This has been solved by using the Assimp[12] library, able to import in a uniform manner a vast number of standard formats. Since Assimp is currently unable to read dynamic dress information, stored as mesh *morphs targets* (or *blend shapes*) that only two file format are able to save (i.e.: Collada `*.dae` and autodesk `*.fbx`), we used Autodesk FBX SDK 2015.1 [16] to import them. 3D Model files decoded using these library are transformed in the proprietary format used by our application. Models are transformed off-line, before being used by vMannequin simulator. Since performances are an important issue for the concern of assets management (the user should experience immediate visualization of the chosen clothes and the applied customizations), the development of a proprietary binary file format allowed us to significantly shorten the loading time of dresses, especially of the precomputed dynamic ones. The animations used in the example application were baked at 30fps, and the simulator can run them while displaying one animating character in full attire, with at least one dynamic dress fit on it. The test was run on a machine equipped with an Intel Core i7 2.4GHz, 8GB of RAM and a NVIDIA GeForce GT670M at a 1920×1080 resolution.

CONCLUSIONS

In this paper we presented the current state of development of vMannequin, a computer application designed to help fashion store customers to design the dress concept of their next purchase. Future works goes in the direction of improving the performances of the application, also to adapt it to low end architectures. Moreover, we aim to enhance the realism of the simulated environment by means of techniques like spherical harmonics, that allows the illumination of the 3D models to match that of the retail store, to achieve an higher degree of realism and immersiveness. Next development stage will be directly tested by users to improve the interaction part, as well as its ability to improve retail store attractiveness.



a. b. c.

Fig. 4. The user experience. The Kinect sensor detects a user standing in front of the installation (a). The user interacts with the application, by Kinect or smart screen (b). The user can simulate the fitting of the dress concept created (c).

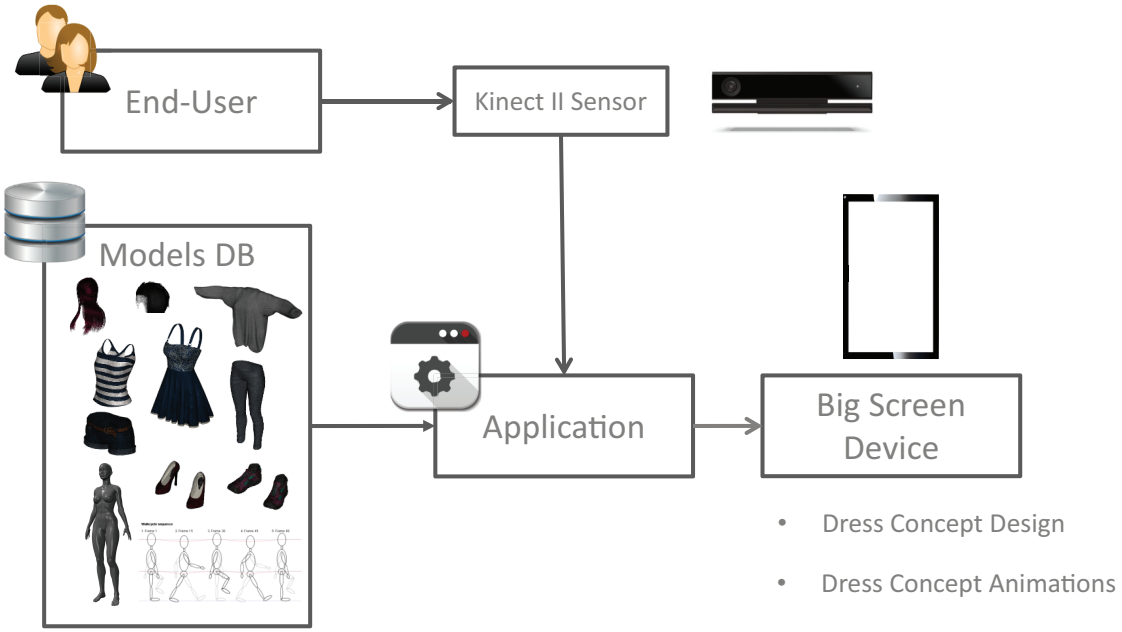


Fig. 5. The Simulator Proof-of-Concept.

REFERENCES

[1] D. Protopsaltou, C. Luible, M. Arevalo, and N. Magnenat-Thalmann, *Advances in Modelling, Animation and Rendering*. London: Springer London, 2002, ch. A body and Garment Creation Method for an Internet Based Virtual Fitting Room., pp. 105–122. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-0103-1_7

[2] Virtusize fitting solution. [Online]. Available: http://www.virtusize.com/site/

[3] Fitnect 3d fitting room system. [Online]. Available: http://www.fitnect.hu/

[4] Fit.me website. [Online]. Available: http://fits.me/

[5] Swivel virtual try-on system. [Online]. Available: http://www.facecake.com/swivel/

[6] trimirror virtual fitting room. [Online]. Available: http://www.trimirror.com/en/about/

[7] Y. Meng, P. Y. Mok, and X. Jin, "Interactive virtual try-on clothing design systems," *Comput. Aided Des.*, vol. 42, no. 4, pp. 310–321, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cad.2009.12.004

[8] S. Giovanni, Y. C. Choi, J. Huang, E. T. Khoo, and K. Yin, *Motion in Games: 5th International Conference, MIG 2012, Rennes, France, November 15-17, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Virtual Try-On Using Kinect and HD Camera, pp. 55–65. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34710-8_6

[9] S. Hauswiesner, M. Straka, and G. Reitmayr, "Virtual try-on through image-based rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 9, pp. 1552–1565, 2013.

[10] U. Gltepe and U. Gdkbay, "Real-time virtual fitting with body measurement and motion smoothing," *Computers & Graphics*, vol. 43, pp. 31 – 43, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849314000600

[11] D. Kim, W. Koh, R. Narain, K. Fatahalian, A. Treuille, and J. F. OBrien, "Near-exhaustive precomputation of secondary cloth effects," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 87:1–7, July 2013.

[12] A. Gessler, T. Schulze, and K. Kulling. The open asset import library. [Online]. Available: http://assimp.sourceforge.net/main_doc.html

[13] C# reference for visual studio 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/618ayhy6.aspx

[14] Kinect for windows software development kit (sdk) 2.0. [Online]. Available: https://dev.windows.com/en-us/kinect

[15] Opentk toolkit library. [Online]. Available: http://www.opentk.com/

[16] Fbx data exchange technology. [Online]. Available: http://www.autodesk.com/products/fbx/overview