A Sketch-Based Interface for Modeling Articulated Figures

Msc Thesis – Extended Abstract

João Pedro B. Pereira
joao.pedro.pereira@tecnico.ulisboa.pt
Departamento de Engenharia Informática
IST-Alameda, Universidade de Lisboa,
1049-001 Lisboa, Portugal

ABSTRACT

Building articulated models with contact surfaces is a common task in Computer Graphics and Multibody Dynamics settings. However, articulated models such as armatures, rigs or kinematic structures are usually built with tedious mouse-keyboard inputs or, even worse, text-based interfaces where the user numerically specifies body transformations and joint types. Programs like Blender, Maya or OpenSim use this kind of approach. Alternatively, sketching on an interactive surface provides a familiar way to define such articulated models as the user can draw directly the content. In this paper, a sketch-based interface which aids the design of articulated models is here presented. In order to evaluate modeling efficiency, we did a comparative study between the proposed interface and conventional WIMP software, as well as integrating superovoidal shapes as primitives in our piece of software due to its vast use in illustration. User tests were done on both lay and professional users that did 3D modeling, along with some others trying to determine how to calligraphically recognize superovoids. We prove that the interface enables speed sculpting that can be used not only in humanoid design but for an arbitrary character or mechanical system and determine a way to classify superovoids calligraphically. A type of interface that, even though it gained the reputation of being used for "quick and dirty" tasks, has increasingly surprised us of its viability in the completion of highly detailed tasks.

Keywords

Sketched Based Interface and Modeling; 3D articulated models; dynamic interaction; ovoidal shapes.

INTRODUCTION

The combination of geometric primitives with articulated skeletons can express the general features of an articulated object, even though they are composed by simple graphical elements (see **Fig. 1**). Additionally, algorithms such as collision detection, mesh generation, pattern recognition (Feng & Pavlidis, 1975), Minkowski sum computation (Agarwal et al., 2000), motion planning (Hert & Lumelsky, 1998), skeletonization (Lien & Amato, 2006), and origami folding (DeCarlo et al., 2003) perform more efficiently on convex objects. And it is because they are simple that they are easy to draw. This emerges some very interesting trade-offs not only between the geometric quality and the number of parameters (i.e., with few parameters it is possible to obtain a satisfiable shape) but as well as between the geometric quality and the amount of defined geometry (i.e., with few geometric shapes I can represent a big part of the original object). Conventional software tools used for this purpose like Blender (Blender Online Community, 2016), Maya (Autodesk, Inc., 2016b) or 3DS Max (Autodesk, Inc., 2016a) offer a Windows, Icons, Menus, Pointer (WIMP) interface which has very well-known limitations regarding its efficiency and expressiveness, i.e., the speed of the workflow and the quantity of diverse information it can relay to the user. Although interactive surfaces have begun to crawl into the everyday work life of a designer, certain features still lack better graphical interfaces, better ways to evoke calligraphically an ovoid as well as to quickly sketch the skeleton of the model or to define the type of joint adjusting its range of motion.

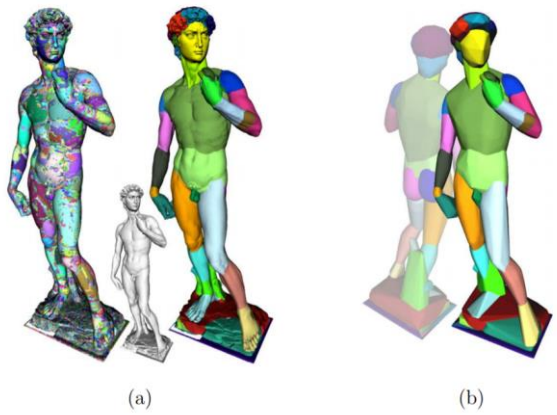

Figure 1. (a) An exact convex decomposition (ECD) (left) and an approximate convex decomposition (ACD) (right) with convexity less than 0.04 of the David model have 85,132 and 66 components, resp. (b) The convex hulls of the ACD components represent David's shape. Even with just the convex hulls, we can still perceive the general features of the model. (Lien, 2006)

Superovoids are by nature a geometric primitive for only having a handful of parameters defining it. In fact, they are often used for character design to define the different unique silhouettes or base shapes of the many characters of an animation or a game. It is a shape that despite its simplicity and its many uses in regular drawing and character design, has yet, as of today, to be implemented in

many software packages as a primitive (Unity 3D, Unreal Engine 4, Blender, Maya, etc.).

The main aim of this paper is to compare WIMP and SBIM interfaces in the case of 3D modeling of general articulated shapes composed of a skeleton structure and primitive shapes based mesh. If SBIM interfaces prove to be more efficient for this kind of task, then this type of software design can be considered as an appealing alternative to conventional WIMP ones. We also aim to see the viability of ovoidal shapes as primitives in 3D modeling applications.

RELATED WORK

Sketch based interfaces application are being actively researched and developed by the scientific community. This kind of interface mainly relies on the intuition of the user, offering, thus, a much simpler way of providing inputs, like hand-drawn sketches. To ease the user's experience, the system imposes constraints and automates repetitive and predictable tasks. As such, the production of a 3D articulated model becomes almost effortless and can be used in visual communication between teams for example (Olsen et al., 2009).

Teddy from (Igarashi et al., 1999) stands out as a very popular system for model sketching. Through 2D cursor strokes, the application is able to create freeform 3D shapes. The latter are created and filled in real time and intuitive editions such as extrusions, cutting and deformation are achievable by drawing on top of the 3D models.

(Zheng et al., 2010) made a system that, through sketches in a touch-screen made by the user; the latter can generate a skeleton structure for a preexisting model. The user simply draws freeform lines on top of the 3D articulated character and the bones are created automatically. The structure of the skeleton for a 3D humanoid should look something like a stick figure. If there is a region of the stroke where the stroke becomes too curved or bent, the user can segment that stroke and subdivide it into multiple bones.

(Yang & Wunsche, 2009) completely automatized the generation of a skeleton out of 2D strokes drawn by the user. Through the contours of the shape, the algorithm produces a set of bones and joints which are then rigged to the 3D mesh for animation. Since the algorithm has no knowledge of the nature of the shape or the physical context it is in, due to its automation, some joints might be mistakenly configured.

(Mao et al., 2009) developed a tool that first requests the user to draw human stick figure as a reference for the skeleton. The stickman, interpreted as a 3D pose, is then fleshed-out using hand drawn shapes. Body fat distribution models are then run to extract its natural shape. The user can then re-edit the 3D articulated figure by drawing on top of it. Once this is done, the 3D model can then be animated thanks to the stickman drawn in the first phase.

(Guay et al., 2013) made a software package based on the concept of the line of action (LOA), a drawing concept used by cartoonists and illustrators as a way to pose different articulated characters; a line that the whole body follows. The way the system interacts is very simple: you draw a line with a single stroke and the model "automagically" aligns according to that line of action. This means that in a few minutes you can produce poses that would have taken hours with other pieces of software.

In another work, (Guay et al., 2015) presented what they called space-time sketching of character animation. Here the authors used the concept of a dynamic line of action (DLOA), which is very similar to LOA concept, but a bit different. It still represents the body line of the character, but, in this case, it is its positioning it over a certain period of time. With this new concept, from a single stroke they can define a whole sequence of animation, allowing, thus, the system to do many types of animations like bouncing, rolling, twisting as well as altering the whole body line mid-action. However, this system is limited to simple character morphologies since multi-legged figures like a humanoid or a cat, is a more complex problem which was not explored in this paper.

SketchiMo, from (Choi et al., 2016), is a software package that relies on lines of the motion of the 3D character to do animation editing. They offer different types of visualization for maximum flexibility in terms of the possible motions the model can have. The user can alter the body line or temporal joints, edit the joint trajectory according to its parent and control the flow of the motion over a certain period of time using *global*, *local* and *dynamic* sketch space respectively. Despite the big flexibility of movement, SketchiMo does not support extensive manipulation of time and can edit the movements through strokes, but does not actually create them.

(Feng et al., 2017) built MagicToon, an interactive modeling application with mobile augmented reality (AR) that allows the user to capture a 2D cartoon drawing on a sheet of paper and build its corresponding 3D model that can then edit and transform in order to create your scene. It supports simple animations by interpolating positions, rotations and scales between two end points by default, as well as advanced character animations. However, shadowing and reflections, may sometimes interferes with the robustness of the segmentation algorithm. Additional objects sketched on top of the main drawing are flat, since the algorithm only inflates the main object; you do not see the bumps on the texture.

Playsketch is another application similar to the previous one. It is capable of capturing drawing made on a sheet of paper as well as other real items available on your desk and convert them into interactable objects inside the application. The user can then use these objects to create game assets, like making the drawing of a track for car racing game, or

little aliens out of plasticine to use them as the asset for the enemies in a *Space Invaders* game.

BlobMaker (Araújo & Jorge, 2003) is a 3D surface modeling software using variational implicit surfaces. This program, with a post-WIMP interface (i.e. regular WIMP interface that contains one or more additional interaction techniques that does not depend on 2D widget such as menus and icons), tackles the problem of the generation of free-form shapes using. From a 2D stroke, the system inflates the shape into a 3D figure and generates it inside the software. The user can then merge the different blobs by going into the corresponding mode and drawing a line connecting both shapes. It is also possible to deform/oversketch an existing blob, to make translation, rotation and cloning operations, to show the wireframe mesh as well as basic undo and redo operations.

METHODOLOGY

WIMP applications

When considering physics-based simulations, such as car accident like Madymo (TASS, 2016), game scene simulation like Mario from Super Smash Bros. Brawl (Nintendo Co., Ltd. 2008) or doll creation like Smart Doll from Danny Choo (Culture Japan, 2016), it is crucial, for their efficiency and accuracy, to have a 3D representation of it. Most modeling applications use Windows, Icons, Menus, Pointer (WIMP) interface well suited for precision work, but they can prove to be very complex to utilize and have a very steep learning curve for effective usage. Not only these pieces of software contain a big amount of information on the screen, but user might also not have all the knowledge in Computer Graphics to understand each and every setting available. Additionally, with a mouse and keyboard interaction, certain tasks can prove to be slow and tedious, such as creating and rigging body parts like the hand. Mapping 2D translations of the mouse into 3D transformations is not trivial and can sometimes be nerve-wracking for the user to achieve the desired results.

Blender (Blender Online Community, 2016) is a software package where the user can assemble highly detailed meshes and link them to equally complex skeletons for the creation animated films, visual effects, art, 3D printed models, interactive 3D applications and video games through mouse and keyboard interactions. For the tasks of bone construction and its association to a mesh, Blender mainly uses a keyboard shortcut system, making the user have to remember a lot of key combinations in order to have a good workflow. The system also possesses a plugin allowing to do bone sketching, but the interaction is indirect, through the mouse not providing, thus, the natural feel of drawing. The small size of the cursor however, allows highly precise work, according to Fitts' law (Fitts, 1954).

Maya (Autodesk, Inc., 2016b) and 3DS Max (Autodesk, Inc., 2016a) are software packages that resembles Blender immensely due to the nature of the functionality the programs and the fact that they use the same type of interface. While Maya uses a similar process of bone association as Blender, in 3DS Max, the bones don't need to be connected to be linked. The core of 3DS Max is interacting with objects and to make basic transformation operations like in Blender's *Object* and *Edit Mode* (translation, rotation, scale, extrude). In her article, (Tay, 2014) also describes 3DS Max as having a more forgiving learning curve than Maya, that can be learned in under two months. Given the nature of their interface and the fact they perform the same of work, their trade-offs are the same.

OpenSim (OpenSim Community, 2016) is another one of these WIMP systems. This software package lets the user produce highly accurate models of humans and animals in order to understand their motion. This has many applications in biomechanics like determining the length of a person's hamstrings that is affected by Cerebral Palsy, a group of permanent movement disorders that appear in early childhood where the most common symptoms are poor coordination, stiff muscles, weak muscles, and tremors. Determining the length of these muscles is useful to figure out if the patient can walk in a more upright posture by lengthening that muscle through surgery. It is, thus used for tasks that require high precision and even the interface is less crowded then Maya, it still presents some complexity due to all the physical knowledge needed to interact with the system.

Multibody Sketcher, a SBIM software package

Multibody Sketcher is a system with a SBIM interface, first started by (Gonçalves, 2015), where the user can, through sketches and touch interac-tions, easily assemble an articulated skeleton held together by joints, and flesh out each link with Smooth Convex Surfaces (SCS) shapes such as ellipsoids, superellipsoids, ovoids and superovoids. This tool was designed to be operable on touch-screens devices such as tablets or table-tops. Developed in Unity3D, this piece of software contains four main modes (**Fig. 2**).

In *Skeleton* mode, the draws, edits and splits bones through sketch and gesture interactions. By drawing in a blank canvas, in the case of a straight line, the system generates a single bone with two joint at both ends of the stroke. If the angle, however, between segments is superior to a defined threshold (35º), the software creates multiple bones, unless the stroke past that segment is too short, which in this case does not create one. These features were implemented by (Gonçalves, 2015), however we added the sub-mode *Symmetric*, which, if activated, spawns the same bone chain in the symmetric position according to the vertical axis that goes through the middle of the screen. Every feature supported in regular *Skeleton* mode is also supported in the sub-mode *Symmetrical* and will edit the symmetrical bone/joint if there is one.

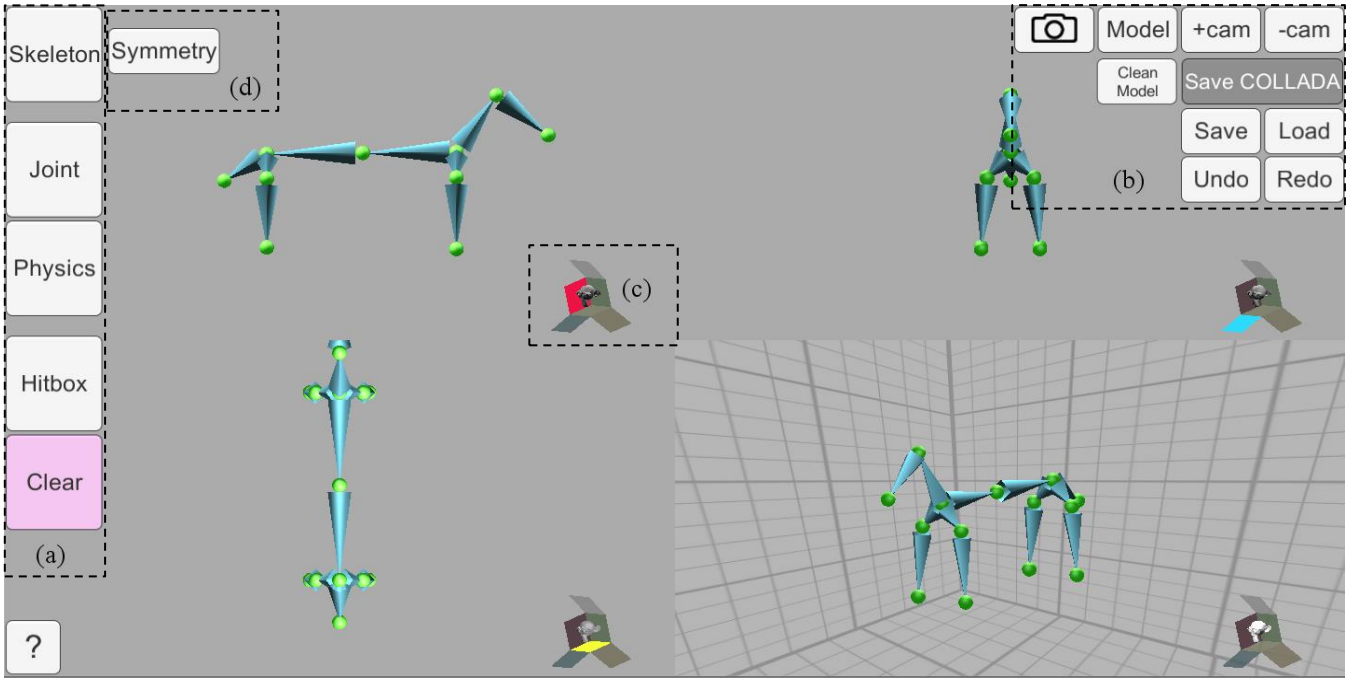

Figure 2. Main interface of the developed Multibody Sketcher prototype, showing 4 different views of the sketched articulated body example, a dog. (a) mode selection, (d) sub-mode selection, (b) addition or removal of perspectives, and general options, (c) open-box camera selection widget, one for each perspective.

In *Joint* mode, developed by (Gonçalves, 2015), you can select and edit a joint's range of motion. By tapping on top a bone, the system will open a pop-up window with joint's angular degrees of freedom in each one of the 3 axes. The user can manually edit these or use the presets available. The user can then test the model in *Physics* model, by tapping and dragging a joint; this will pull and twist the whole skeleton using inverse kinematics. Since the main purpose of this mode is to test the range of movement of the joints, all modifications done in this mode are temporary and will reset once the user changes mode.

In *Hitbox* mode, the user can directly associate a generated primitive shape to a bone by drawing its form on top of the latter. The recognition of these drawings is done using the CALI library (Fonseca et al., 2002), being, thus, limited to simple geometrical shapes (i.e., circles, rectangles, ellipses, diamond, etc.). Important to notice that since CALI is a 2D recognizer, while generating the shapes, Multibody Sketcher has to apply a default value to the third dimension for it to be 3D and not 2D. By dragging, making circular and pinch movement with a two-finger interaction, the user is able to move, rotate and scale the generated mesh respectively. The user can also invoke the superellipsoid widget, allowing him/her to model the mesh into a superellipsoid. This widget uses the following equation proposed in (Barr, 1981):

$$(1) \qquad F_{SE}(x,y,z) = \left(\left| \frac{x}{a_1} \right|^{\frac{2}{\epsilon_1}} + \left| \frac{y}{a_2} \right|^{\frac{2}{\epsilon_1}} \right)^{\frac{\epsilon_1}{\epsilon_2}} + \left| \frac{z}{a_3} \right|^{\frac{2}{\epsilon_2}}$$

Where a_1, a_2, $a_3 > 0$ represent the lengths of the shape in all 3 axes x, y and z, and ϵ_1, $\epsilon_2 \in]0,2[$ are the squareness factors for the $x0y$ plane and z axis, respectively. While dragging the finger through the widget, the user is changing the values of ϵ_1 and ϵ_2, modeling this way the shape into a superellipsoid. In addition to to previous version done by (Gonçalves, 2015), we developed a new one for superovoid modeling that instead of using the previous equation, uses this one also proposed in (Barr, 1981):

$$(2) \qquad F_{SO}(x,y,z) = \left(\left| \frac{x}{\left(T_x \frac{z}{a_3} + 1\right) a_1} \right|^{\frac{2}{\epsilon_1}} + \left| \frac{y}{\left(T_y \frac{z}{a_3} + 1\right) a_2} \right|^{\frac{2}{\epsilon_1}} \right)^{\frac{\epsilon_1}{\epsilon_2}} + \left| \frac{z}{a_3} \right|^{\frac{2}{\epsilon_2}}$$

Where a_1, a_2, a_3, ϵ_1, ϵ_2 have the same meaning as for superellispoids and T_x, $T_y \in [-0.5, 0.5]$ are the tapering (egg-shape) factors in x and y respectively. Instead of mapping the previous parameters, it maps T_x and T_y the same way it was done for ϵ_1 and ϵ_2, meaning that by dragging the finger, the user is altering the egg-factor of the shape, modeling it into a superovoid.

In addition to all of these models, (Gonçalves, 2015) developed a camera widget to move between perspectives, additional cameras to the screen so the user can see and edit the same time, *Undo* and *Redo* options to reverse the effects of a previous action done by the user and canceling the reverse respectively, a way to store and load a saved model from a permanent directory and a screenshot button. We thought it would useful as well to have a way to import an image model and displaying it on the screen to ease the

modeling process. We did so with the buttons *Model* and *Clean model* which creates a 2D plane texture associated to a camera view and removes the plane from that specific camera view respectively.

USER STUDY

In order to confirm or reject whether SBIM or WIMP interfaces are better for this kind of tasks and to determine a calligraphic recognition pattern for superovoids, user studies were conducted.

Superovoids calligraphic recognition

In order to perceive the reasons behind this enquiry, we must first understand what is the CALI library (Fonseca et al. 2002) and how it works. CALI is a hand-drawn classifier for general shapes; its main purpose is to recognize and determine the different features of simple sketches that are then used by other application to generate its corresponding 2D or 3D shape. The system captures the input provided by the gestures of the user and sends the sequence of points to its recognizer. The latter will calculate five different geometrical figures (**Fig. 3**): the convex shape with the smallest area around the scribble, called the convex hull (ch); the rectangle with the smallest area surrounding the scribble, called the enclosing rectangle (er); the quadrilateral with the biggest area inside the scribble, called the largest quadrilateral (lq); the triangle with the biggest area inside the scribble, called the largest triangle (lt); the rectangle with the smallest area surrounding the scribble and aligned with the axes, called axis aligned bounding box (AABB).

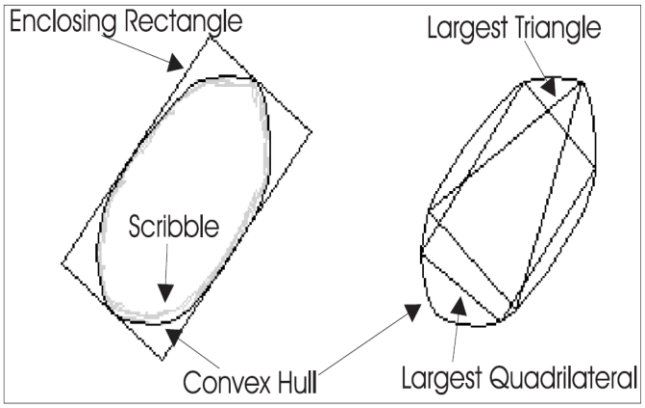

Figure 3. Polygons generated by CALI. (Fonseca et al. 2002)

The system will then retrieve different features of these shapes to then calculate relevant ratios to differentiate from the different geometrical figures CALI can recognize. For each kind of shape, we want to add to CALI, in order to be recognized by the latter, a study must be conducted to determine the smallest group of ratios that excludes every possible shape except one. To distinguish circles from other figures, (Fonseca et al. 2002) used the thinness ration (P_{ch}^2/A_{ch}), where A_{ch} is the area of the convex hull and P_{ch}^2 is its perimeter squared. Since the circle is the shape with the smallest perimeter, given a fixed area, the thinness is minimal with a value near 4π. To identify ellipses, (Fonseca et al. 2002) compared the area of the largest quadrilateral to that of the convex hull. The (A_{lq}/A_{ch}) will have the smallest value, nearing 0.7.

Using the same logic, in order to create the new type, Ovoid, we must determine what is the smallest combination of features which solely identifies this shape. Since the closest shapes to an ovoid are ellipses and circles, we used the ratios mentioned earlier as potential candidates. However, we also noticed by going through the list of features that if you take a random ellipse and simply change its tapering factor, you are not changing the area of the shape, but you are altering the area of the largest triangle, meaning that there is a high chance that the ratio A_{lt}/A_{ch} is a distinguishing factor between ovoids and its neighbor shapes.

Taking into account, we conducted a user study to create a database which would contain the list of values of the features we suspected would be relevant for each shape. Therefore, in a new Unity scene, we asked users to draw circles, ellipses and ovoids given a model we would present to them. The user first draws three differently sized circles, then four ellipses and finally three ovoids. The system would then automatically collect the data and store it into a csv file. Later, this is csv file is processed by R code to create box plot and visualize the results.

For this experiment, we managed to round up twenty-nine users with a range of age from 18 to 26 and where 38% were female and 62% of them were male. Most were in engineering, but we managed to find people from architecture, nutrition, medicine and design as well.

WIMP vs. SBIM

In order to fairly compare both types of interfaces, we needed at least two software packages to represent each side. Since Blender is a free and opensource tool with a large community, we decided to take this program as representative of WIMP applications, whereas for SBIM we took our own, Multibody Sketcher. For this study, we felt it to be unfair to have a person who had no prior knowledge with Blender, to suddenly start making armatures and meshes, modeling them, binding the two together and testing it in *Pose Mode* with five to ten minutes to practice. As such we conducted the typical laypeople and professional users to test our software, but we also rounded up what we call semi-lay users which are people that have already interacted with Blender before but do not use it in a regular basis.

For professional and lay user, we asked them to only interact with our system, to see how newcomers react with the system and to get as much feedback as possible from all of the available features. After explaining the interface and given the time to play around with the application, we ask the user to replicate three image models we had prepared as quickly as possible. The order in which they do these is random to avoid bias and some of them require additional

tasks, not timed, to interact with all of the features the program has to offer (e.g., creating a running man pose; turning a tree more organic). The robot arm is composed of a simple skeleton structure, two spheres, two cylinders and two ovoids. The main purpose of this model was for the user to manipulate with the *Hitbox* mode. The tree is composed of a simple skeleton structure and four superellipsoids. The main purpose for this one was to make the user edit the bones and joints in many perspectives. Finally, the humanoid is composed of skeleton structure and users had to draw additionally a sphere on its head. The purpose of this model was to use and test the sub-mode *Symmetry*, as well as the *Joints* and *Physics* mode. We record them doing the task and take screenshots of the finished model. In the end of the experiment the users answer two forms, one about the ease of use of each feature of the application, as well as of the result of the three tasks; the other is about the user's general opinion about Multibody Sketcher. The professional users have small recorded interview after this about the everyday life of 3D modeling professional as well as general comments on our prototype.

For semi-lay users, the experiment was similar, but we had them do models in Blender and in Multibody Sketcher, this means that due to time restrictions, we could only ask them to do two models for each piece of software and did not interact with as many features of Multibody Sketcher as did the others. Once again, the order in which application the tasks are preformed on a certain program is random to avoid bias. These users did the robot arm and human model without interacting with the *Joint* mode.

We managed to round up seventeen lay users, four semi-lay users and one professional users. The number of males and females people is almost 50% each and most of the lay and semi-lay users were engineering students.

RESULTS

Superovoids calligraphic recognition

In the end of our user study, we ended up with three-hundred and seven entries in our cvs file. For this experiment, we checked for four features: Hollowness (calculated by taking a triangle which is 60% of largest triangle that is inside the convex with the same barycenter and then counting the number of scribble points inside the smaller triangle), P_{ch}^2/A_{ch}, A_{lq}/A_{ch} and A_{lt}/A_{ch}.

The hollowness, was the less interest interesting one, given the fact that for either ellipses, circles or ovoids, you draw the contour of the shape and not its inside. This means that for all these three cases, unless the user slips while sketching, the hollowness should be zero and this indeed the case.

For the A_{lq}/A_{ch} ratio, although ovoid shapes have a very small fuzzy interval with circles, this is not the case for ellipses. Take an ellipse and without changing its area, morph it into an egg by altering the tapering factor. The largest quadrilateral for both shapes will likely be very similar since the area lost on one end when changing it to an ovoid is gained on the other. Therefore, this ratio is average to distinguish ovoids from circles, but it is even less robust to tell apart ellipses from ovoids (**Fig. 5**).

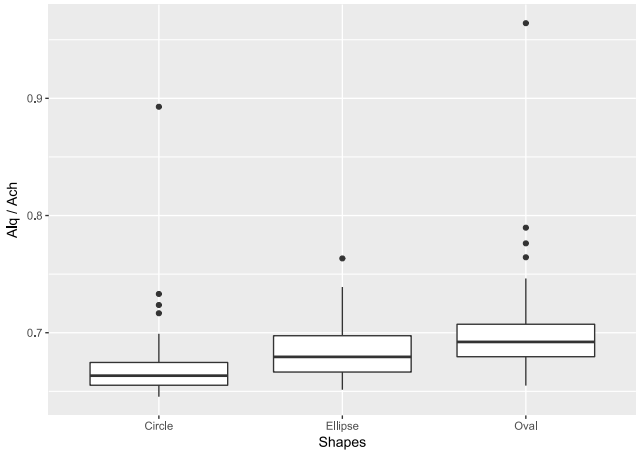

Figure 5. Boxplot of the A_{lq}/A_{ch} feature. As you can see most of the values of all kinds of shapes overlap each other, making this ratio a not very good option.

The P_{ch}^2/A_{ch} ratio, presented a very interesting and unexpected result. As stated in the previous section, this feature is the key one to differentiate circles from other shapes; this is because given a fixed area, the perimeter is minimal, nearing a value of 4π. The results for circles and ellipses seem consistent with (Fonseca et al. 2002)'s research and the values are very different from one another, however, if you observe the boxplot further (**Fig. 6**), you will see that the value of this ratio for ovoids is extremely similar the circle ones, but slightly bigger. We suspect that this is due to the fact that almost three quarters of an egg's shape is a circle; the only difference is the tip created by the tapering factor, which slightly increases the perimeter of the shape given a certain area.

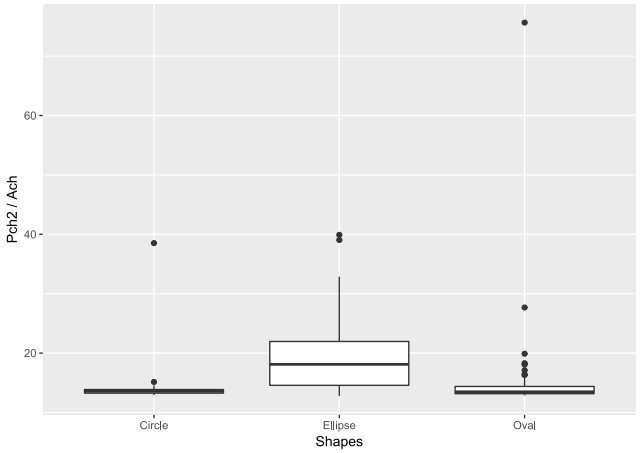

Figure 6. Boxplot of the P_{ch}^2/A_{ch} feature.

Finally, the A_{lt}/A_{ch} ratio was the one we were more hopeful for and it did not disappoint. As you can see from

the boxplot, most of the value for circles and ellipses are around 0.45, while for ovoids they are around 0.5. There are, however, a few overlapping values between ellipses and ovoids, but fewer between ovoids and circles (**Fig. 7**). This confirms our theory that by fixing an area and morphing an ellipse to an ovoid, you increase the area of the largest area triangle inside the shape.

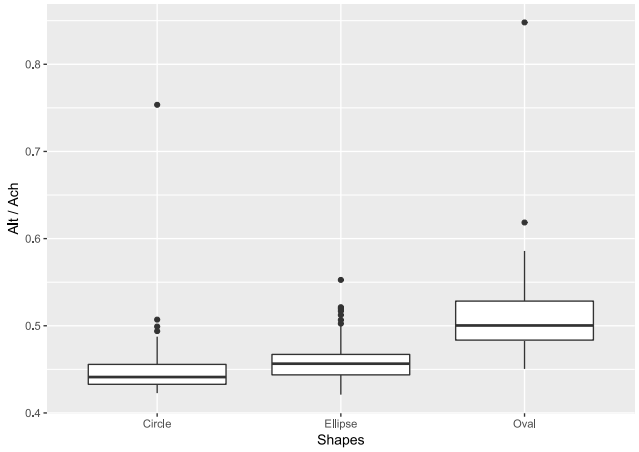

Figure 7. Boxplot of the A_{lt}/A_{ch} feature. Here you can notice that the main box of values of oval shapes is not overlapping any other one.

Given these results, the family of features we implemented for the new egg shape was hollowness that had to have the value of 0; P_{ch}^2/A_{ch} to distinguish ovoids from ellipses with values between 12.5 and 14 and with one fuzzy intervals (i.e., interval of values that might be confused with other shapes),]14 ; 16]; A_{lt}/A_{ch} to differentiate egg figures from circles with values between 0.48 and 0.54 and with two fuzzy intervals, [0.45 ; 0.48[and]0.54 ; 0.59]. After testing it, the system now indeed recognizes oval shaped sketches and classifies them as so, with a few errors of course, which need to be look upon in more detail in the future.

WIMP vs. SBIM

For laypeople, the average time for laypeople for the robot arm, tree and humanoid test is 3min. 18s., 1min. 51s. and 2min. 22s. respectively. There is however a huge gap between users in the robot arm test since some of them completed the task in a minute, while others, got the 5min. mark without completing the task, not to mention that the standard deviation has a value of 1min. 04s. Users mainly struggled with understanding in which mode they should be to do the proper modifications, some tried to model a mesh without even creating one, some tried to edit the skeleton in *Hitbox* mode, some tried to change skeleton properties in Physics mode, etc.

As for semi-laypeople, every one of them managed to complete every model either in Blender or Multibody Sketcher and the results are what we expected. On average, the robot arm was completed in 5min. 32s in Blender, whereas in Multibody Sketcher it was 1min. 55s, less than half of the time. For the humanoid model it took, on average, 4min. 28s. in Blender and 2min. 20s. in Multibody Sketcher, again, almost half of the time. We can clearly see from these results and from the boxplot, that Multibody Sketcher is more efficient for these kinds of tasks.

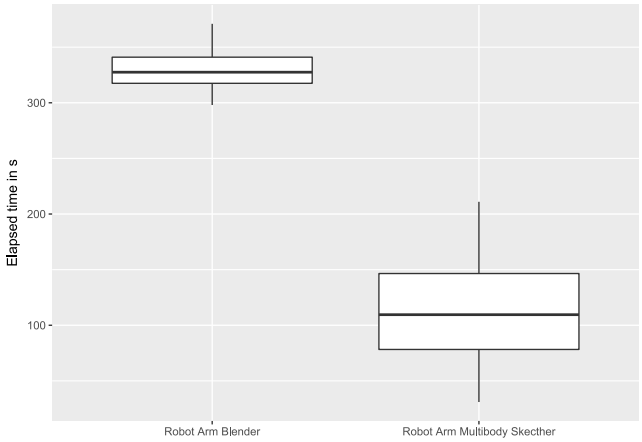

Figure 8. Boxplots of the time semi-lay users took to complete the Robot Arm task. The elapsed time value on the vertical axis is in seconds. You can clearly see that the best time for Blender took still longer the worst time of Multibody Sketcher.

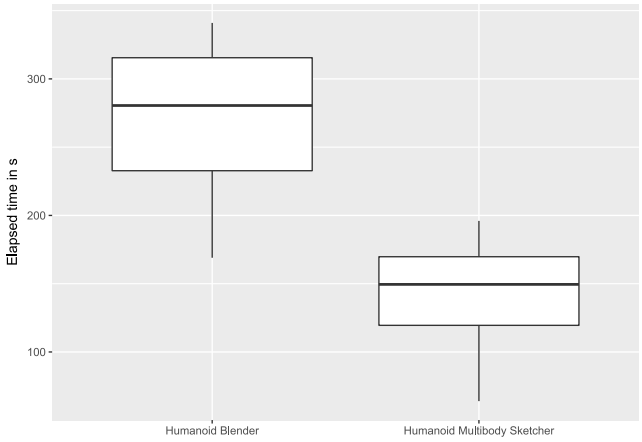

Figure 9. Boxplots of the time semi-lay users took to complete the Humanoid task. The elapsed time value on the vertical axis is in seconds.

Last, but not least, the professional user completed the robot arm, tree and humanoid in 2min. 15s., 2min. 02s., 1min. 36s. respectively, a reasonable time. The user described the application as “quick and very cool”; he complimented on the fluidity of the creation of the skeleton and direct association of a mesh to the bone and found the camera navigation to be very intuitive. He also stated that this application is more suited in drawing organic figures than robotic ones, which contrary to conventional 3D modeling programs and so, being, thus, a big plus. The user also did not feel the need to use a pen to draw on the tablet, as well as a bigger screen, since we are building only simple models, there is no need for highly detailed editing, meaning that a small surface area is enough to do the task. He mentioned as well that instead of having meshes that

associate to individual bones, that we should create, for example, a sausage man mesh around 3D humanoid skeleton and have the system automatically calculate the weights and apply them to the model. This obviously has its limitation due to occlusion (e.g., the legs of a table), but it would allow a quick and simple rigging.

Users found in general that Multibody Sketcher was intuitive, interesting, interactive, flexible, fast and easy. All of them loved the bone sketching feature and a vast majority like the inverse kinematics as well, due to the organic movement of the model. The professional user mentioned that trying to pose the humanoid was much easier than with conventional tools, because when you would move the model's elbow, the shoulder needed almost no further editing to be in right place.

One of the main complaints, however, especially from semi-lay and professional users, was that with Multibody Sketcher you cannot edit with the free camera and this could cause occlusion problems while altering a model; the model they were struggling with was the humanoid since you can grab the pelvis without taking one arm out of the way. Another complaint was that in symmetry mode, a line on screen showing the user where was the line of symmetry would have been helpful to help to guide the drawing.

CONCLUSION AND FUTURE WORK

As new researches are being done, we get closer to an accurate 3D representation of the physical world; and the better the simulation, the higher the complexity. As such, the learning curve naturally becomes steeper and steeper. What we intend to do is to smooth that curve as much as possible by providing an interface that encapsulates some processes and provide a better workflow than the conventional tools currently available. We created such a tool, with the features that a typical 3D modeling software would have and added a few others (constrained sketching, image model background, etc.), notably the creation of superovoids as primitives since they are shapes that are vastly used by cartoonist when defining the silhouette of a character or in the first steps of drawing it. We determined a way to calligraphically recognize these shapes so that in the future we could spawn them the same way we do with ellipsoids and spheres in Multibody Sketcher. We conducted a user study to scientifically analyze the via-bility of SBIM interfaces on doing what conventional WIMP do and, as it stands, for the kind of work Multibody Sketcher proposes (i.e., designing the silhouette and general features of a character), our hypothesis, i.e, *If tasks like modeling and curves and surfaces drawing become natural to accomplish with SBIM, then the latter will improve the workflow of designing skeleton structures and simple contact geometries when compared to conventional WIMP interfaces*, confirms itself and additionally, the fact that these applications can be transferred to touch devices, makes them more portable.

However, there are still a few features that could be looked into and added to the application. As it was mentioned in the results section of WIMP vs. SBIM, Multibody Sketcher, would become much better if the user could edit with the free camera, solving, thus, the occlusion problems the professional user was describing. Another future work to consider is to do simple automatic rigging of a big mesh to the 3D model instead of having a small primitive associated to a single bone. This would allow, not only to have deformation of the mesh while testing it in *Physics* mode, but also make the silhouette of the character look more natural. Once the latter is implemented animation could also be considered as an additional feature to the software; similar to the one of (Guay et al., 2015) or SketchiMo (Choi et al., 2016). Since Multibody Skecther can only handle at the moment simple organic articulated skeletons with SCS meshes, one idea would be to create a touch interaction in Hitbox/Volume mode for the user to model the mesh similarly to the way it is creation (e.g., drawing a line on top of an existing mesh could create depth like when you are carving a stone statue, drawing a cross on top of an existing mesh could delete it, etc.). Relatively the calligraphic recognition, the next step of this work would be to take CALI with new the class Oval and ask Multibody Sketcher to generate the shape making the required geometrical adjustments. Another future work would be to replace CALI with another gesture recognizer library that is both quicker and more precise (e.g., 1$ Recognizer (Wobbrock et. al., 2007)) and implement it inside Multibody Sketcher. In another note, it would also be interesting to compare our current SBIM interface with post-WIMP ones like BlobMaker (Araújo & Jorge, 2003).

To sum up, SBIM interfaces have a recognized potential in "quick and dirty" drawing and modeling, however efforts have been made by the scientific community that this type of interface can be used for highly detailed works and we are still deciding on which interfaces are better for which kinds of tasks. As (Nealen et al., 2005) argue, though "our capability to derive a mental model from everyday shapes around us is well developed, we fail to properly communicate this to a machine. This is why we have to model in a loop, constantly correcting the improper interpretation of our intentions.".

REFERENCES

1. Araújo, B. R. and Jorge, J.A. (Oct. 2003), 'Blobmaker: Free-form modelling with variational implicit surfaces', Proceedings of "12° Encontro Português de Computação Grafica" (12° EPCG), pp. 17-26, Porto, Portugal.
2. Agarwal, P. K., Flato, E. and Halperin D. (Jan. 2000), 'Polygon decomposition for efficient construction of minkowsk sums', in European Symposium on Algorithms, pp. 20–31.

3. Autodesk, Inc. (2016a), 3DS MAX, <http://www.autodesk.com/products/3ds-max/overview>.
4. Autodesk, Inc. (2016b), *Autodesk Maya*, <http://www.autodesk.com/products/maya/overview>.
5. Barr, AH. (1981), 'Superquadrics and Angle-Preserving Transformations', *Computer Graphics and Applications, IEEE*, vol 1, no. 1, pp. 11-23.
6. Blender Online Community (2016), Blender 2.77 - a 3D modelling and rendering package, <http://www.blender.org>.
7. Choi, B., Ribera, RB., Lewis, JP., Seol, Y., Hong, S., Eom, H., Jung, S. and Noh, J. (2016), 'SketchiMo: Sketch-based Motion Editing for Articulated Characters'. KAIST, Weta Digital. SIGGRAPH 2016 Technical Paper, July 24-28, 2016, Anaheim, CA.
8. Culture Japan (2016), Smart Doll, <http://www.dannychoo.com/en/post/27350/Smart+Doll.html>
9. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S. and Santella, A. (2003), 'Suggestive contours for conveying shape,' ACM Trans. Graph., vol. 22, no. 3, pp. 848–855.
10. Feng H. Y. F. and Pavlidis T. (1975), 'Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition', IEEE Trans. Comput., vol. C-24, pp. 636–650.
11. Feng, L., Yang, X. and Xiao, S. (March 2017), 'MagicToon: A 2D-to-3D Creative Cartoon Modeling System with Mobile AR', Digital ART Lab, School of Software Shanghai Jiao Tong University, China
12. Fitts, P. M. (June 1954), 'The information capacity of the human motor system in controlling the amplitude of movement'. Journal of Experimental Psychology. pp. 381–391. doi:10.1037/h0055392.
13. Fonseca, MJ., Pimentel, C. and Jorge, JA. (2002), 'CALI: An online scribble recognizer for calligraphic interfaces', AAAI spring symposium on sketch understanding, pp. 51-58
14. Gonçalves, A. (2015), 'Efficient Contact Detection for Game Engines and Robotics', Lisbon. Master Thesis, Instituto Superior Técnico. <https://fenix.tecnico.ulisboa.pt/cursos/meec/dissertacao/565303595500513>
15. Guay, M., Cani, MP. and Ronfard, R. (2013), 'The Line of Action: an Intuitive Interface for Expressive Character Posing'. Laboratoire Jean Kuntzmann - Université de Grenoble – Inria, Grenoble, France.
16. Guay, M., Cani, MP., Ronfard, R. and Gleicher, M. (2015), 'Space-time sketching of character animation'. Laboratoire Jean Kuntzmann - Université de Grenoble – Inria, Grenoble, France. University of Wisconsin, Madison, Wisconsin, USA. SIGGRAPH 2015, Los Angeles, California.
17. Hert, S. & Lumelsky, V. J. (1998), 'Polygon area decomposition for multiplerobot workspace division', International Journal of Computational Geometry and Applications, vol. 8, no. 4, pp. 437–466. [Online]. Available: citeseer.nj.nec.com/hert98polygon.html
18. Igarashi, T., Matsuoka, S. and Tanaka, H. (1999), 'Teddy: A Sketching Interface for 3D Freeform Design', *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., Los Angeles, California.
19. Lien, J.-M. & Amato, N. M. (June 2006), 'Simultaneous shape decomposition and skeletonization', in Proceedings of ACM Solid and Physical Modeling Symposium (SPM'06).
20. Lien, J.-M. (Dec. 2006), 'Approximate convex decomposition and its applications', Texas A&M University, Dissertation to fulfill the degree of Doctor of Philosophy.
21. Mao, C., Qin, SF. and Wright, D. (2009), 'A sketch-based approach to human body modelling', *Computers & Graphics*, vol 33, no. 4, pp. 521-541.
22. Nealen, A., Sorkine, O., Alexa, M. and Cohen-Or, D. A., (2005), 'Sketch-based interface for detail-preserving mesh editing.', In: Proceedings of SIGGRAPH '05.
23. Nintendo Co., Ltd. (2008), Super Smash Bros. Brawl, <http://www.smashbros.com/wii/en_us/>.
24. Olsen, L., Samavati, FF., Sousa, MC. and Jorge, JA. (2009), 'Sketch-based modeling: A survey', Computers & Graphics, vol 33, no. 1, pp. 85-103.
25. OpenSim Community (2016), OpenSim, Stanford University, <http://opensim.stanford.edu/>
26. TASS International Software and Services (2016), Madymo, <https://www.tassinternational.com/madymo>.
27. Tay, T. (Jan. 8th, 2014), '3DS Max vs Maya: A Friendly Comparison', Udemy Blog, <https://blog.udemy.com/3ds-max-vs-maya/>
28. Wobbrock, J.O., Wilson, A.D. and Li, Y. (2007). 'Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes.', Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '07). Newport, Rhode Island (October 7-10, 2007). New York: ACM Press, pp. 159-168.
29. Yang, R. & Wunsche, BC. (2009), 'Automatic joint and skeleton computation for the animation of sketch-based 3D objects', *Image and Vision Computing New*

Zealand, 2009. IVCNZ '09. 24th International Conference, IEEE, Wellington.

30. Zheng, Q., Li, FWB. and Lau, RWH. (2010), 'Sketching-Based Skeleton Generation', *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on*, IEEE, Jinhua.