# Design of Novel Pseudo-Haptic Techniques for Tablets

Marco Freire

August 23, 2018

**Abstract**

Haptic feedback grants a more realistic and immersive way of exploring visual media content despite usually being limited by the lack of fidelity and the bulkiness of the apparatus used for this purpose. Pseudo-haptic techniques allow for a simpler and lower-cost experience on a more portable device such as a tablet while retaining most of the advantages of previous techniques.

**Keywords:** pseudo-haptics; haptic sensation; touch device

**Reference:** Intership done from May 22$^{nd}$, 2018 to August 3$^{rd}$, 2018 at the HYBRID team of Inria Rennes under the supervision of Antoine Costes, Ferran Argelaguet and Anatole Lécuyer.

## Contents

# Introduction

Data representation mostly relies on visual means nowadays. People are used to perceiving and remembering their environment through visual stimuli, so it is only natural that visual information is the most used means of communication. But there is many more dimensions to an object than its physical appearance: people can perceive sound, smell, taste or texture. In order to recreate faithfully the sensations experienced when in contact with the real world, which can be very important in the virtual reality domain to enhance user immersion, all of these channels must be explored.

Among all of these means of transmitting sensory information, haptic feedback is starting to get more and more attention, on phones and gamepads integrating vibration features for example. The word *haptic* designates everything related to the sense of touch and the perception of forces : when a person runs their finger on a surface, a wooden table for example, they experience haptic feedback that gives them information about the table's texture, shape, roughness and many other physical properties.

A variety of devices have been designed to apply forces on the user's body, such as force-feedback arms, or to create realistic simulations, such as motion platforms, but this kind of equipment is often either very expensive or not easily usable or ergonomic and therefore not accessible to the main consumer. If haptic sensations could be reproduced on simpler and more widespread devices such as personal computers, tablets or smartphones, it could render the technology available to everybody, but haptic feedback techniques are barely advanced or available enough to accurately replicate the feeling of touch on this kind of device.

When perceiving spatial properties, people tend to give more credit to what they see than to what they feel, so by introducing a discrepancy between these two stimuli, a new coherent representation of the environment is forged, thus possibly creating a haptic illusion, which establishes the basis of pseudo-haptics. Pseudo-haptic feedback can then be used to convey haptic sensations without cumbersome equipment and on simpler devices relying only on the sensory dominance of vision over touch and the creation of a conflicting visuo-haptic feedback.

The goal of this internship is to design and implement a system based on pseudo-haptic interaction enabling the haptic exploration of a virtual environment, enhancing the sensations experienced by the user. This system will rely on a handful of haptic effects per scene conveyed to the user through the deformation of an on-screen cursor.

In the remainder of this paper, related work in the field of pseudo-haptics and touchscreen interaction will be presented. Then the different methods and algorithms behind the alterations of the cursor's state, used to convey haptic sensations, will be explained in detail, leading to the construction of an application showcasing these methods. Finally the paper will conclude with a discussion about the work done during the internship and the different research leads it provides.

## 1 Haptic sensations and interaction

### 1.1 Pseudo-haptics

Many ways of recreating haptic feedback have been devised such as force-feedback arms that can accept input movement from the user or apply a force on the user's hand, motion platforms used in advanced flight or driving simulators or much more simple devices such as vibration-inducing mechanisms in game pads. These devices ease the immersion of the user in either a simulation or a game, but they can also help enhancing everyday experiences. These kinds of advanced equipment can be very expensive or cumbersome, making them not very widespread nor available to everybody.

Pseudo-haptic feedback tries to address these issues by recreating haptic sensations in virtual environments using visual feedback and properties of human visuo-haptic perception. This technique strongly relies on the sensory dominance of vision over haptic sensations when dealing with spatial interaction tasks. In order to work, pseudo-haptic feedback introduces the user to one or more sensory conflicts between visual and haptic information. Faced with this conflict, the user creates a new and coherent representation of their environment based on visuo-haptic feedback, potentially different from their actual environment. [4].

Different haptic properties such as friction [6] or stiffness [3] have been simulated successfully with this kind of techniques through haptic interfaces.

While the simulations were proven to be effective, external haptic devices are still necessary. In order to get rid of this constraint, other techniques have been developed using only the computer mouse as an interface. For example, the feeling of a texture was succesfully recreated by modifying the mouse cursor speed and size on certain regions of the surface [5]. Also, material elasticity was simulated by applying a deformation to the texture on the surface under the cursor to create the illusion of a force being exerted on it [1].

## 1.2 Touchscreen interaction

When applying pseudo-haptics to touch devices three issues must be taken into account. Firstly, cursors and applications have to be designed around hand and finger occlusion: most of the space under the finger and a part of the screen are obstructed by the hand of the user and are not visible. Cursors have to be designed around this and be larger or be controlled remotely from a specific part of the screen, breaking co-localization. Secondly, decoupling happens when the cursor is not directly tied to the finger and can travel at a different speed. At the beginning of the movement, the finger and the cursor start moving from the same point, but they stop at different places at the end of the movement, which can make the illusion fall apart. Ujitoko et al. circumvented this problem by having a moving background image to simulate a walk on a snowy terrain [7], but less research has been conducted in the co-localized scenario. Finally, input latency can be noticed on touch devices and can be disturbing to the user in some applications. Input latency compensation algorithms usually try to predict where the user will be touching the screen based on previous touch data [8].

## 1.3 Touchy

In order to address these first two issues Costes et al. [2] proposed the use of a circular deformable cursor following the user's finger. By means of the alteration of the motion or the shape of a co-localized cursor, a variety of haptic features such as the shape, roughness or friction of a texture can be expressed. In the demonstration, the user can see a single texture on the screen and can feel an individual haptic effect. The effect can be chosen independently of the displayed texture, but they should remain closely related in order for the illusion to be effective.

## 2 Contribution: the *Encase* effect

### 2.1 Concept

**Internship goal** The aim of the internship was to extend the approach of Costes et al. to virtual environments. The main idea is to enable the user to feel a surface or an object on a touchscreen through pseudo-haptic techniques. The aim of this technology is to enhance user experience and immersion while remaining portable and inexpensive. This proof of concept takes the shape of an application for touchscreen devices showcasing different haptic effects, mainly the *encase* effect which is the main contribution of this internship.

**Encase effect structure** The aim of the *encase* effect is to create an illusion of depth and transmit information about the surface geometry and relief of the zone of the object the user wants to inspect. This inspection is done through a cursor under the user's finger. This cursor must then be able to deform and adapt to the objects' surface which can be accomplished by using a cursor mesh. Because the 3D scenes featured in the application consist of 3D models also represented by meshes, the relief information will depend highly on the meshes' level of detail, which must be sufficient to extract accurate data. After surface geometry data is extracted, it must be transmitted to the cursor which has to be deformed to match the object and placed accordingly in the scene, where the user wants it to be. All of this has to be done taking into account the touchscreen device hardware and specifics.

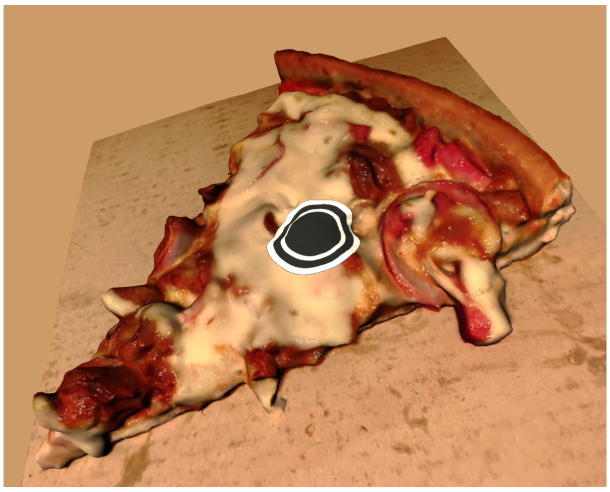
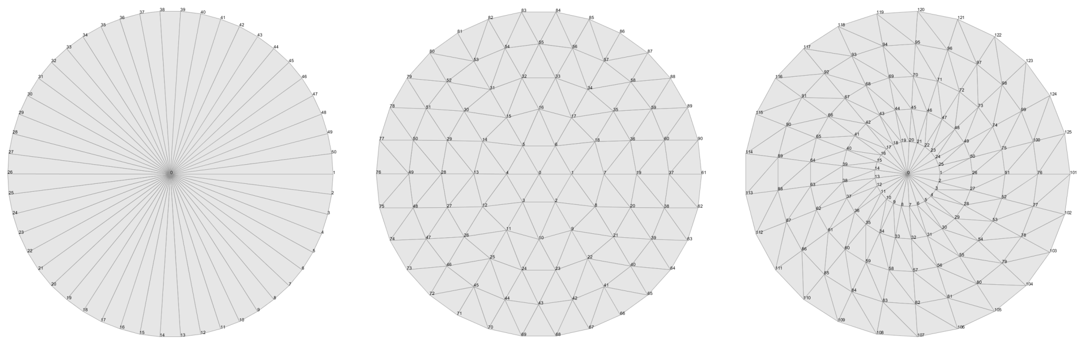


Figure 1: *Encase* effect

### 2.2 Implementation

#### 2.2.1 Mesh structure

Three cursor meshes have been implemented, each one having its advantages and its drawbacks: a simple radial cursor, a hexagon-based cursor and finally an extended radial cursor. Their detailed structures can be seen in figure 2.



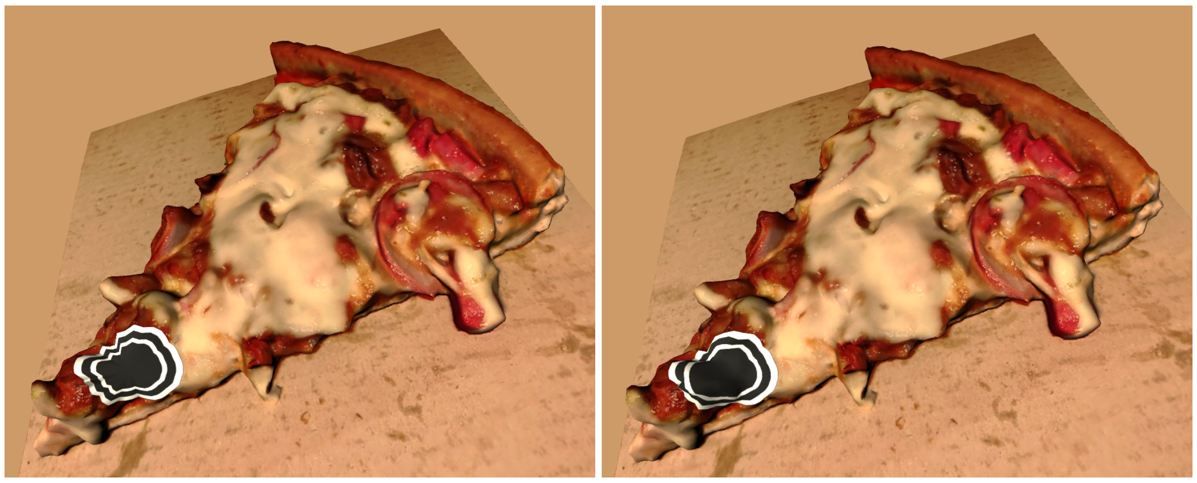
(a) Simple radial cursor
50 *outer vertices*

(b) Hexagonal cursor
5 *rings*

(c) Extended radial cursor
5 *rings,* 25 *vertices per ring*

Figure 2: Different cursor mesh templates

**Simple radial cursor** The simple radial cursor is the most basic shape possible for the cursor, with a vertex at its center and every other vertex on a circle with a given radius. The number of outer vertices can be adjusted to balance precision and performance. While this cursor performs well due to its elementary structure, the lack of vertices in the inside of the cursor can lead to loss of visual detail making the surface look flat as can be seen in figure 3.



(a) Simple radial cursor

(b) Hexagon-based cursor

Figure 3: Comparison of detail inside the cursor for two different structures

**Hexagon-based cursor** In order to have a uniform vertex distribution and minimise their number, the mesh triangles should be as regular as possible. This leads to the creation of a hexagon-shape structure, where vertices are placed regularly on the contour of concentric hexagons. Actually the vertices are projected on the circumscribed circles of the hexagonal rings so that every vertex on a particular ring is at a constant distance from the center of the cursor. The number of concentric rings is adjustable.

**Extended radial cursor** The third cursor structure came as a solution to problems found at a later stage that will be discussed at length in section 3.1. In this cursor, a fixed number of vertices are equally distributed on the circumference of a series of concentric rings. These vertices are all placed along radii of the cursor. Because of this, the angular density of vertices in the cursor is adjustable, but the regular density is much higher near the center than on the border. Both the number of vertices on each ring —which is constant across all rings— and the number of rings are adjustable.

**Cursor appearance** Two different cursor appearances were tested: in the first one, the cursor has a black background to increase contrast with the scene and white rings on the border increase deformation perception and in the second one the black parts become transparent, which blends better with the rest of the scene. The influence of the cursor color and texture on the impact of haptic sensations should be further investigated. The shape is fully symmetrical to avoid favouring a particular direction and the size takes into account the problem of finger occlusion presented in section 2.2.4. The comparison between the two cursors can be seen in figure 4.

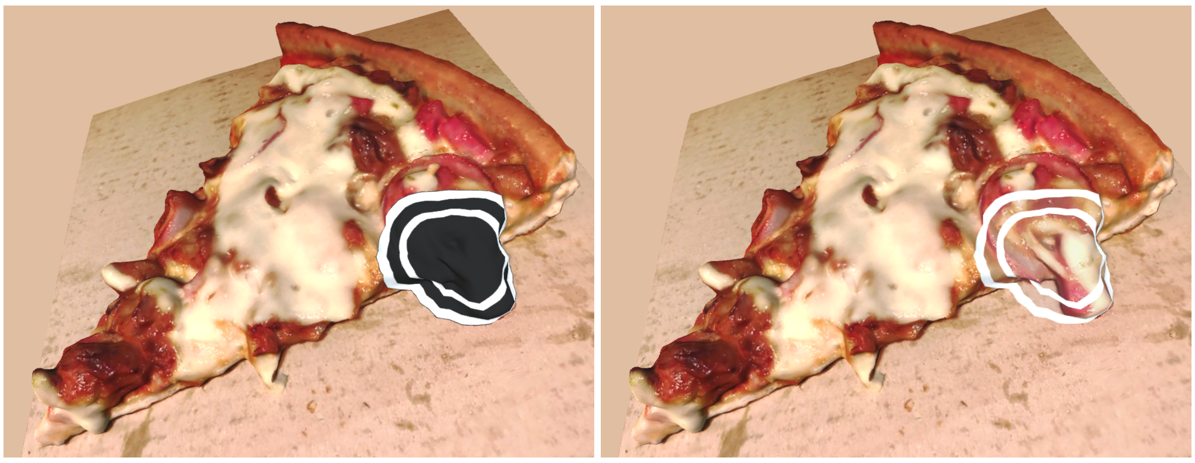

(a) Black cursor (b) Transparent cursor

Figure 4: Comparison between two different cursor appearances

### 2.2.2 Projection methods

The main idea behind the implemented techniques is to project the cursor shape on the scene surface below it. Two main types of projection were used: a raycast-based projection and a distance-based projection.

**Raycast-based projection** This projection relies on placing a reproduction of the cursor template on the world point corresponding to the finger position of the screen, with its normal aligned with the surface normal at this point. Then each cursor vertex is raycasted from the viewer camera position onto the object surface, the viewer camera being always fixed. This approach is computationally expensive since a raycast is done for each vertex of the cursor and each object in the scene requires a complicated mesh collider, the models used consisting of many vertices and triangles. Moreover the proportions of the cursor are very distorted using this method, which leads to unrealistic deformations on surfaces parallel to the viewing perspective.

**Distance-based projection** In order to avoid unbelievable deformations, this projection preserves cursor dimensions in the 3D space. It heavily relies on transformations between screen coordinates and world coordinates.

This transformation is based on the depth information of the scene, which can be accessed by rendering the depth buffer of the scene the camera is seeing, on a texture. From this texture, the distance between the camera and the content of a pixel can be obtained. Then *Unity* premade functions allow us to convert pixel screen coordinates into world coordinates.

The depth texture of the whole scene is captured at a very high resolution from the user's perspective as can be seen in figure 5. This texture encodes depth in a 32-bit floating point number, the usual 8-bit integer not being precise enough for the screen to world space position conversion.

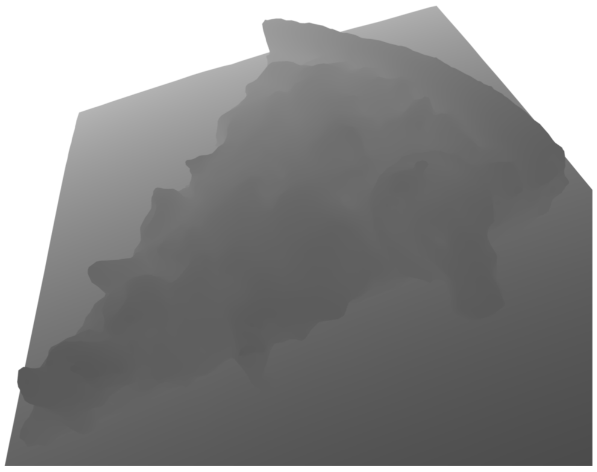


Figure 5: Depth texture example

First, the cursor center is placed in the scene. Then, for each vertex in the cursor, the distance and the normalized direction vector from the center to the vertex are calculated on the cursor template (see figure 2), the farthest vertices being at a distance of 1 from the center.

Then, starting at the center, steps of a given length are made on the template in the direction of the current vertex and the distance on the cursor template is translated into world distance. Once the traveled distance in the scene is greater than the distance between the center and the considered vertex, the last position before this condition is met, is transferred into world coordinates and set to be the position of the vertex.

An illustration of this principle can be seen in figure 6.

**Comparison** The main difference between these approaches is the way information about the scene geometry is gathered. The former uses the mesh structure of the scene to obtain this information, since raycasts return a collision point with the mesh collider, while the latter obtains depth data about the scene used in the rendering process, from which information on the geometry is extracted by the means of the transformation between screen and world coordinates.

The maintenance of a mesh collider and the constant raycasting being very demanding compared to the fetching of the depth texture of the scene, which is only done once at the start

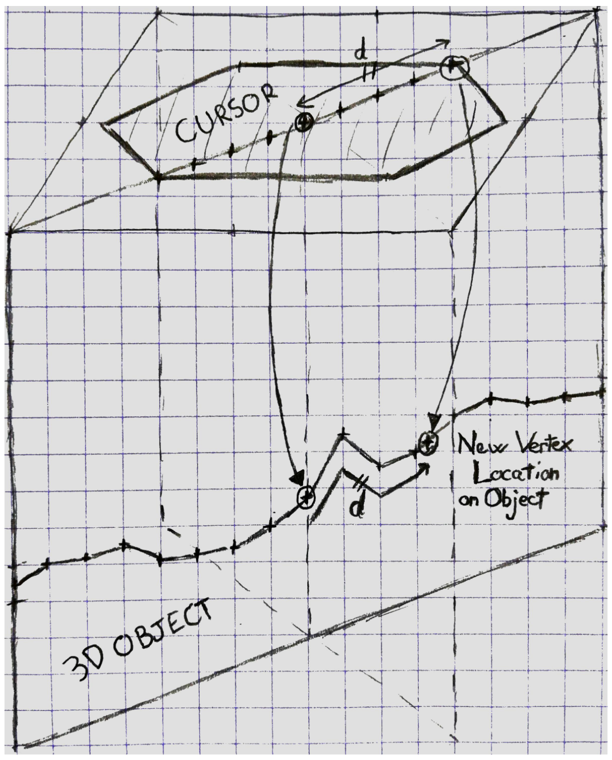


Figure 6: Encase principle

of the application, the latter approach is used.

### 2.2.3 Influence of cursor detail on visual appearance

Both the cursor detail and the step size can be adjusted in order to balance performance and visual quality. Depending on the cursor template used, detail will increase differently when changing the cursor parameters as stated in section 2.2.1.

For the simple radial cursor, detail only increases on the circumference of the cursor, effectively making its border look smoother the higher the number of vertices is.

As for the other two cursors, the more vertices there are, the more details of the surface geometry can be grasped. Figure 7 shows the influence of the step size on the extended radial cursor with a fixed number of rings and of vertices per ring on a flat surface. The larger the step size is, the worse the distance approximation made by the projection algorithm is, the object surface sampling being sparser and the distance increments per step of the algorithm larger.

In a similar vein, if the number of vertices is not high enough, however small the step size is, less surface details will be grasped and the cursor will appear irregular, as seen in figure 8.

A balance must then be found between the number of rings and the step size. As the number of rings increases, the step size should decrease, or else vertices will collapse around the same points, as can be seen in figure 7 for the two highest step size values.

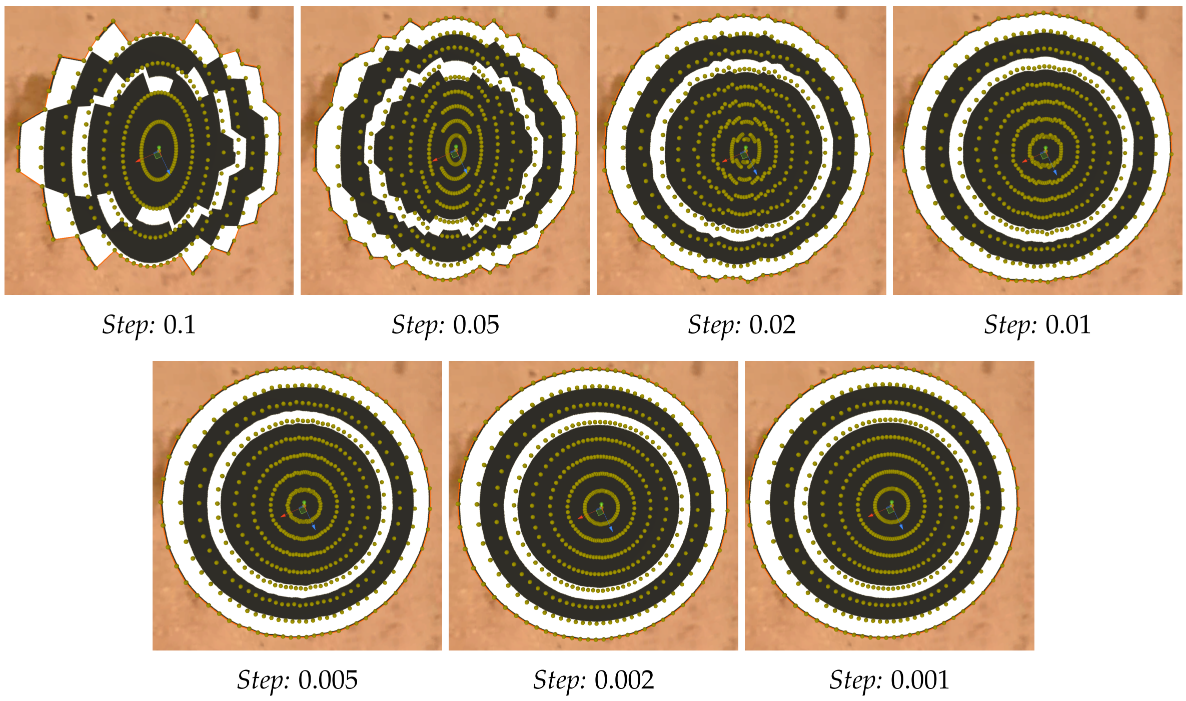


Figure 7: Variations of the precision of vertex position depending on texture step size
Extended radial cursor: 8 *rings*, 64 *vertices per ring*

### 2.2.4 Touchscreen specifics

**Touchscreen device issues** Interacting with a device through an indirect means such as a computer mouse or a direct means such as a touchscreen is radically different. As stated above, finger and hand occlusion or decoupling are problems that must be taken into account in the design of this application.

When the user interacts with the scene, most of the time the cursor will be occluded by their finger, which requires the cursor to have a minimum size or else the user will not be able to feel the pseudo-haptic effects that are being transmitted through the cursor.

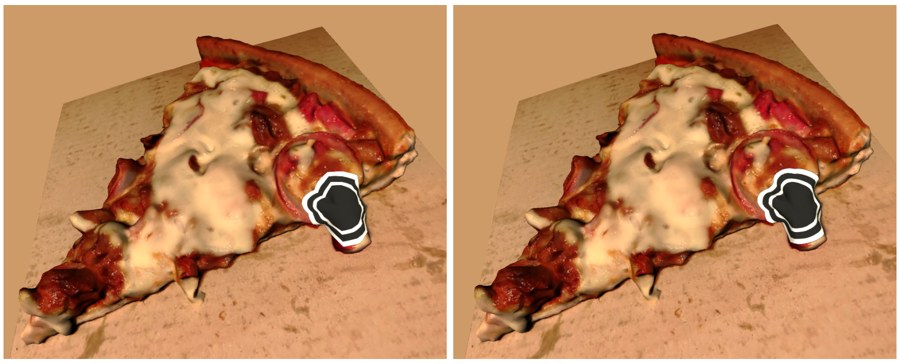
Another issue to take into account is that most of the time, touchscreen devices have less computational power than personal computers and in consequence cannot run some applications, particularly GPU-intensive ones.

The last issue is input latency: all touchscreen devices take a perceivable amount of time to respond to user input. Since the user should be able to feel what is under their finger at all times, the cursor should also be under their finger at all times and input latency should be reduced as much as possible.

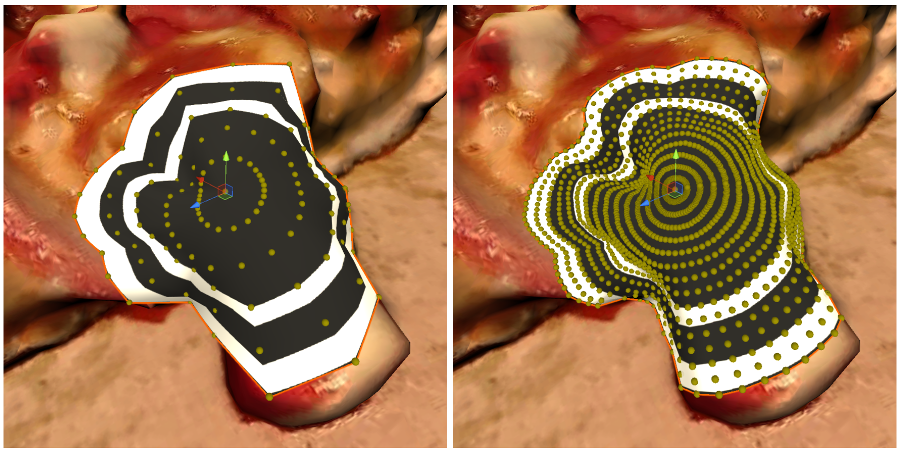
**Input latency compensation** Input latency correction is based on the past positions of the cursor, which are used to correct its new position. The prediction algorithm was supplied to me by one of my internship supervisors, Antoine Costes and is inspired by the work of Ushirobira et al. [8].

A simple linear prediction is applied to the measured touch position $f_{meas}$:

$$f_{pred} = f_{meas} + k_{pred} \times (f_{meas} - f_{meas}^{prev})$$



Cursor visual appearance



Cursor structure

Figure 8: Influence of the number of vertices on cursor visual appearance and structure Extended radial cursor: *left:* 5 *rings*, 25 *vertices per ring; right:* 15 *rings*, 100 *vertices per ring*

where $f_{meas}^{prev}$ is the measured touch position at the last frame and $k_{pred}$ the prediction coefficient. Then an exponential smoothing filter is applied to get the corrected position:

$$f = \alpha \times f_{pred}(1 - \alpha) \times f_{pred}^{prev}$$

where $f_{pred}^{prev}$ is the predicted position at the last frame and $\alpha$ the filter parameter. The parameters were set to $k_{pred} = 8$ and $\alpha = 0.3$ after testing.

This algorithm predicts accurately the future position of the cursor when traveling in a straight line, the prediction being much less accurate when the cursor travels along a curve, which can be disturbing. These inaccuracies can interfere with some haptic effects, thereby reducing their effectiveness and impact on the user.

### 2.2.5 Demo application

**Application structure** The main structure of the application is the following: at first, user input is read and touch coordinates are corrected to compensate for input latency and the cursor is placed according to these coordinates. Then modifications are applied to the cursor to account for the haptic properties of the surface below the cursor. Lastly, the cursor is deformed to match the surface geometry to give an illusion of shape and depth.

The application is made in *Unity* which allows an easy manipulation of 3D scenes, comes with a multitude of useful pre-made utilities and is widely used in virtual reality research or in the film industry. The scripting is done in C#.

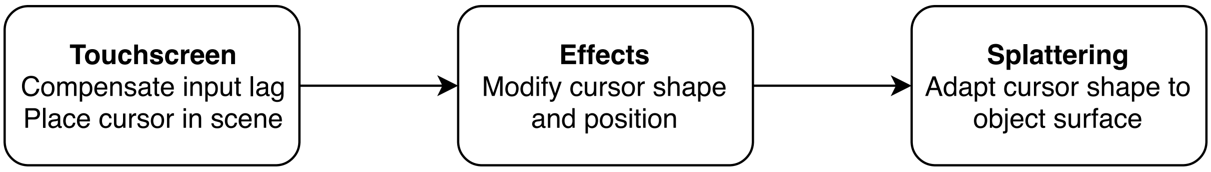


Figure 9: Structure of the application

**Effects** The goal of the application is to allow the user to feel the different surfaces and objects in the scenes featured in the application. In order to convey haptic sensations, different haptic effects were implemented:

- a vibration effect;
- an elastic effect;
- a slippery effect.

To each effect is associated a mask texture that defines in which zone of the scene the effect will be triggered, an example of a mask texture can be seen in figure 10.

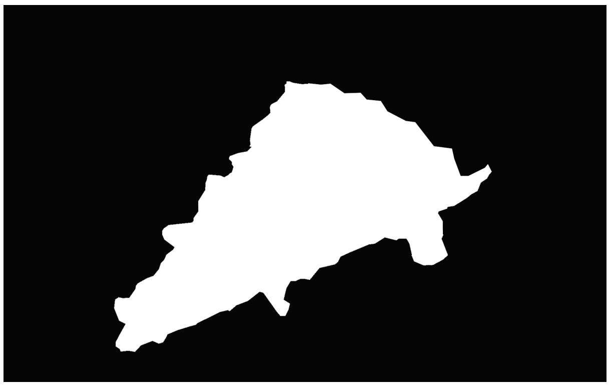


Figure 10: Elastic effect mask texture

**Vibration effect** This effect tries to recreate the sensations caused by the micro-structure of a surface. In order to represent visually these vibrations on a given surface, a random offset of adjustable magnitude is added to the position of the cursor on the screen. This effect triggers when the cursor speed goes beyond a certain threshold, indeed, if the cursor is too slow and the effect is applied, its movement seems erratic and unrealistic.

**Elastic effect** This effect tries to recreate the sensations experienced when feeling an elastic surface by acting on the size of the cursor. When the user touches the screen for the first time, the cursor shrinks depending on the elasticity of the surface and then grows back to its normal size when the finger is lifted. The size variation is linear with time. The shrinkage and growing speeds are adjustable, as well as the final, full shrunk size.

**Slippery effect** This effect tries to recreate slippery surfaces by making the cursor slide in certain surfaces of the scene: when the cursor is faster than a certain speed, the user loses control of it and it keeps on moving on a straight line either until it stops or reaches the boundary of the slippery zone.

**Scenes** The choice of the scenes featured in the application was driven by the idea of offering to the user the possibility to explore an object of a lesser or higher than human scale, providing a novel experience both through innovative pseudo-haptic techniques and through a scene standpoint.

Every scene contains at least the model of the object the user will be able to explore, a main camera from which the image displayed on the device is captured and the cursor traveling on the surface of the object, subject to different effects.

Each scene was supposed to showcase one or two different haptic effects and the *encase* effect. Because of this, the concepts, algorithms and techniques explained in this document will be illustrated on this scene. An overview of the different scenes can be seen in figure 11.

Due to the lack of time, only the pizza part scene ended up in the final application.

All of the different 3D scenes and models were taken from Sketchfab and include:

- a pizza part;
- a car in an arid landscape;
- a tree stump;
- a fruit and vegetable basket;
- a low-poly earth model;

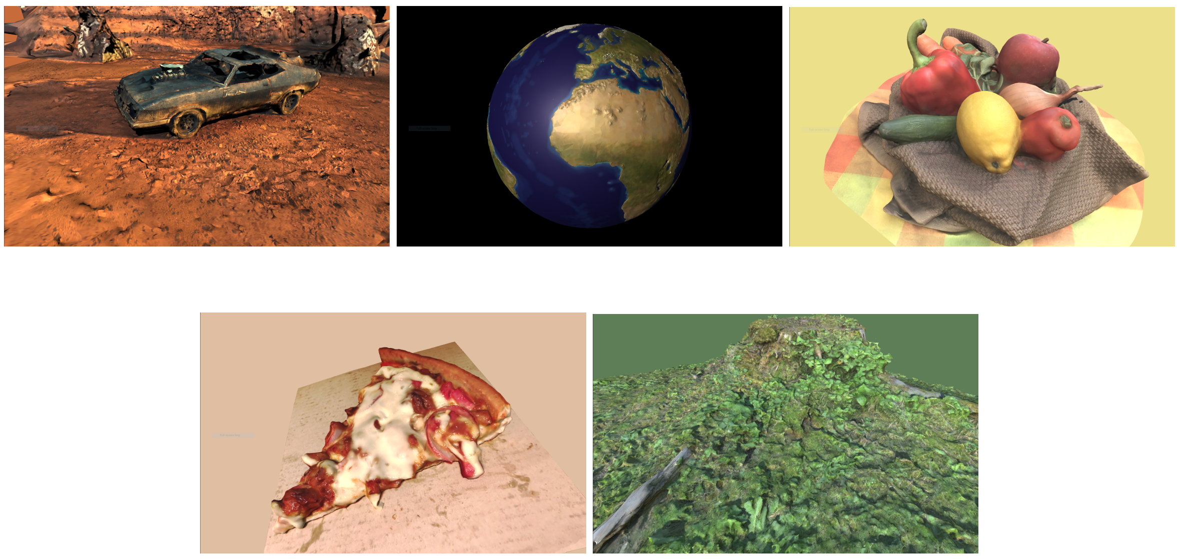


Figure 11: Different scenes

## 3 Discussion and future work

### 3.1 Current issues

**Perspective deformation** It can be seen in figure 12 that even if the cursor is on a flat surface, radii in some regions of the cursor are farther apart than in other regions. This is explained by the angle deformation caused by the perspective projection. When steps are taken on the cursor template on a given direction, the angle defined by this direction should be corrected depending on the perspective of the camera. Without this correction, angles on the closest and most distant sides of the cursor appear larger than they should, and those on the other sides appear smaller than they really are.

**Discontinuity of the cursor border** Depending on the placement of the cursor in the scene, one can see artifacts on the border of the cursor, such as the ones in figure 13. This happens when the cursor is over a region with substantial depth variations because the distance on the surface is calculated from the center of the cursor.

In order to fix this, it would be possible to add constraints on the length of the edges of the cursor, which would be computationally expensive on every cursor template except for the simplest one, the other ones having too many edges. When using the simple radial cursor, it is possible to average the magnitude of the vector connecting a given vertex to the center, by averaging it with the magnitudes of the vectors of its neighbouring vertices on the circumference of the cursor.

This results in a much smoother border at the cost of some extra computation. Unfortunately, some new artifacts appear as can be seen in figure 14, in which a vertex appears right above another.

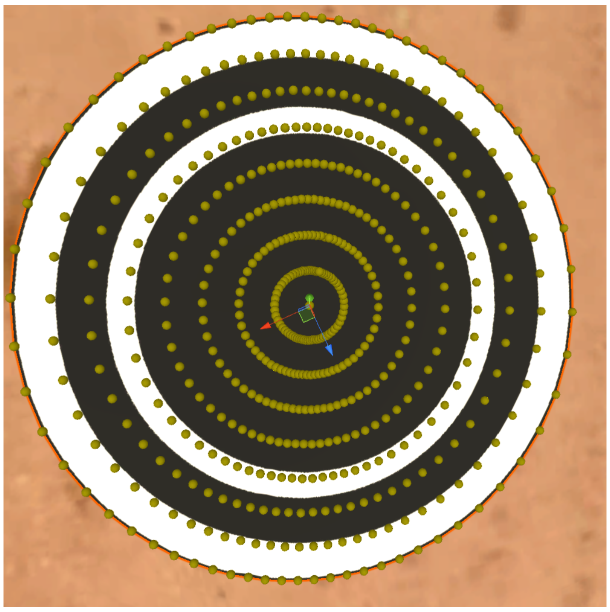


Figure 12: Angle deformation

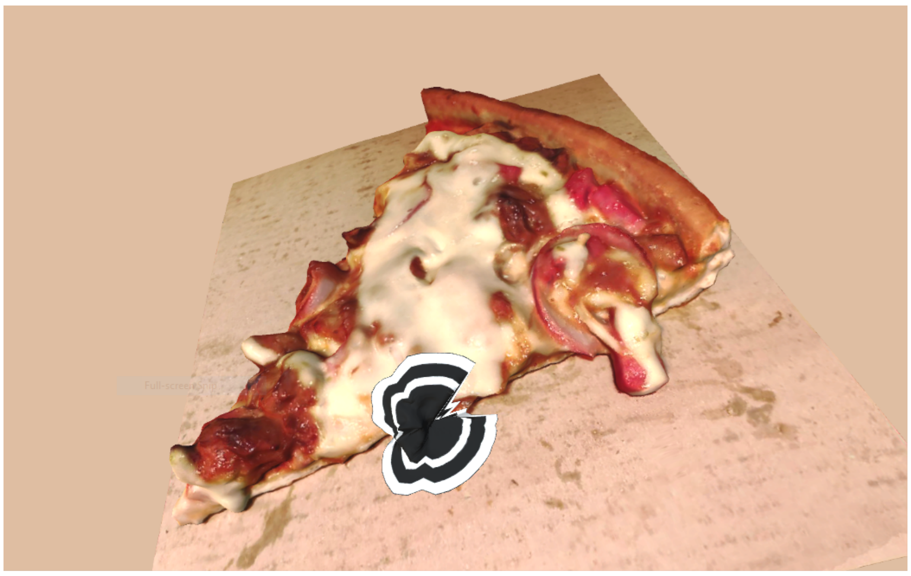


Figure 13: Cursor artifacts on height discontinuities

**Lack of surface detail with simple radial cursor** As we have seen, the simple radial cursor hardly conveys any detail of the surface underneath, not having any vertex in the inside aside from the center, as it was shown in figure 3. In order to circumvent this problem, when the scene is loaded, its normal map is captured and stored. Then the normal map of the scene is used for lighting the cursor, which gives the illusion that the cursor is actually taking the shape of the object underneath.

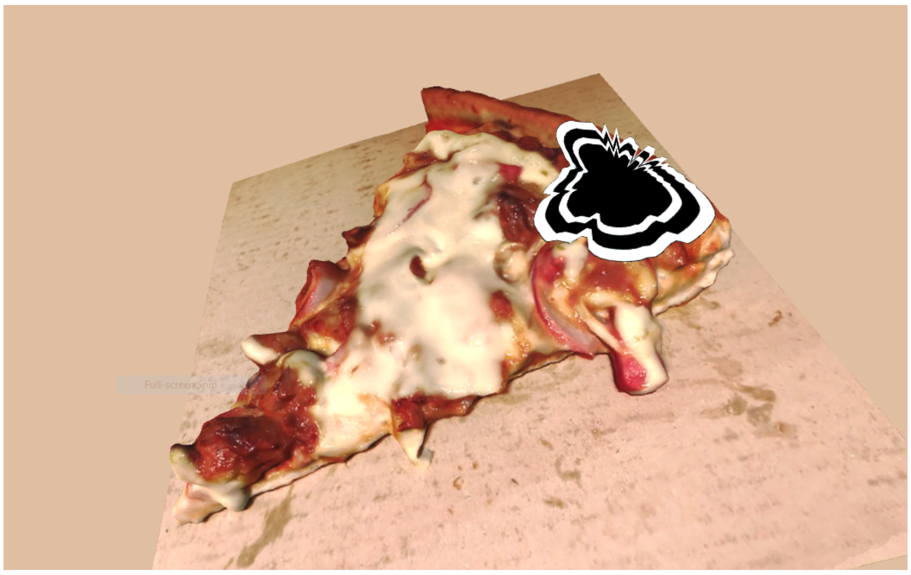


Figure 14: Cursor artifacts with neighbour averaging

## 3.2 Current optimisations

**Optimised *encase* algorithm** When the extended radial cursor was not yet implemented and the application used the hexagon-based one, we realized that the structure was not optimal for the projection algorithm. Indeed, when $n$ vertices are aligned along a radius of the cursor, the distance between the center and each vertex is traveled multiple times. Instead, the distance from the center to the farthest vertex on a radii could be traveled only once, storing for each vertex the point where the respective world distance was passed. This approach was impossible for the hexagon-based cursor since vertices are not aligned on the radii, but the extended radial cursor was created with this optimisation in mind as can be seen in figure 2. Performance is much better with this optimisation: for a radius on the extended radial cursor, the total number of steps needed goes from quadratic to linear in the inverse of the step size with the new method.

**Constant step size** It can be noticed that the step size is a constant, a vertex close to the center of the cursor will require a small number of steps in the projection algorithm. As a consequence, the variability of its position will be greater relative to the distance from the center than for a farther vertex. To fix this, the step magnitude could be scaled by the distance on the cursor template from the center to each vertex. The precision would be then the same for each vertex, independently of its distance to the origin, but it would highly increase the total number of steps needed by the algorithm and the performance would plummet. Furthermore, since the display resolution and the depth texture resolution are finite and limited this detail would mostly go unnoticed.

**Working with an image of the scene** Tablets and touchscreen devices have a hard time running the application, even when using the optimised *encase* algorithm. This is due to the high vertex and triangle count of the models used in our scenes, which were rendered on each frame. Since the camera is static, there is no need to keep rendering the models, so at the start of the application, a screenshot of the scene is taken and then displayed for the rest of the runtime, which compensates for the lack of GPU power stated in section 2.2.4.

### 3.3 Future work

Due to the short length of the internship, no time was left to evaluate user experience. This kind of applications being centered around user experience and the main goal being to improve the expressiveness of the visual medium by adding haptic components to it; most decisions regarding the haptic effects should be reevaluated after user feedback is gathered. What was accomplished during the internship should act as a basis for a more complete application, comprising additional effects and taking into account this feedback.

Despite the *encase* effect being a complete and functional effect, more time should be spent on improving it visually and finding a solution for the artifacts visible on discontinuous surfaces, which remain the main problem of this feature.

Additional effects such as stickyness, temperature or friction can be added to this application to improve the range of haptic sensations the user can feel, which is quite limited at the present moment.

Finally, in order to make the creation of scenes easier and more accessible, the link between the mask textures and the scenes should be strengthened, these textures being currently crafted by hand based on the scene they represent.

## 4 Conclusion

In this paper we present the research process behind the extension of the approach of Costes et al. to virtual environments, taking the shape of an application allowing the user to feel objects in a 3D scene displayed on a touchscreen device through pseudo-haptic techniques, developed during an internship in the HYBRID team at Inria Rennes.

In this application, the user interacts with the scene through a cursor controlled by their finger which, they can use to explore it. The cursor is affected by different pseudo-haptic effects depending on the surface it is on. These effects try to convey a sensation as close as possible as the one felt when touching a real surface of that kind. The implemented effects can transmit information on the roughness, the slipperiness and the elasticity of the surface, but also on the global geometry of the object through the *encase* effect, which was the main focus during the internship.

The work done during the internship establishes a basis for a future implementation of more pseudo-haptic effects such as stickiness, temperature or friction and a more diverse collection of scenes. Unfortunately user experience was not properly evaluated, despite being crucial for making sure that the application is effective. Gathering user feedback is the next logical step for making this proof of concept into a fleshed-out application. Also, the system to add effects to each scene should be improved, currently having to build by hand all of the mask textures for each effect. In order to have a variety of scenes, this should be automated, fully using the capacities of the *Unity* development platform.

## Acknowledgements

This internship would not have been possible without the supervision of Antoine Costes who was open to discussion all along and guided me in numerous occasions. I would also like to thank Ferran Arguelaguet who provided expertise for the more technical concepts and algorithms and Anatole Lécuyer for taking time off his busy schedule for our meetings. Finally, but not less important, thanks to the whole HYBRID team for their warm welcome and hospitality everybody showed to me.

## References

[1] Ferran Argelaguet Sanz et al. "Elastic Images: Perceiving Local Elasticity of Images Through a Novel Pseudo-Haptic Deformation Effect". In: *ACM Transactions on Applied Perception* 10.3 (Aug. 2013), 17:1–17:14. URL: `https://hal.archives-ouvertes.fr/hal-00907775`.

[2] Antoine Costes et al. "Touchy: Tactile Sensations on Touchscreens using a Cursor and Visual Effects". In: *Proc. Haptics Symposium 2018 Companion*. IEEE, 2018, pp. 132–132.

[3] A. Lecuyer et al. "Pseudo-haptic feedback: can isometric input devices simulate force feedback?" In: *Proceedings IEEE Virtual Reality 2000 (Cat. No.00CB37048)*. 2000, pp. 83–90. DOI: `10.1109/VR.2000.840369`.

[4] Anatole Lécuyer. "Simulating Haptic Feedback Using Vision: A Survey of Research and Applications of Pseudo-haptic Feedback". In: *Presence: Teleoper. Virtual Environ.* 18.1 (Jan. 2009), pp. 39–53. ISSN: 1054-7460. DOI: `10.1162/pres.18.1.39`. URL: `http://dx.doi.org/10.1162/pres.18.1.39`.

[5] Anatole Lécuyer, Jean-Marie Burkhardt, and Laurent Etienne. "Feeling Bumps and Holes Without a Haptic Interface: The Perception of Pseudo-haptic Textures". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. Vienna, Austria: ACM, 2004, pp. 239–246. ISBN: 1-58113-702-8. DOI: `10.1145/985692.985723`. URL: `http://doi.acm.org/10.1145/985692.985723`.

[6] Anatole Lécuyer, Sabine Coquillart, and Philippe Coiffet. "Simulating Haptic Information with Haptic Illusions in Virtual Environments". In: (Jan. 2000).

[7] Yusuke Ujitoko et al. "Yubi-Toko: finger walking in snowy scene using pseudo-haptic technique on touchpad". In: *SIGGRAPH Asia Emerging Technologies*. 2015.

[8] Rosane Ushirobira et al. "A forecasting algorithm for latency compensation in indirect human-computer interactions". In: *In proceedings of ECC'16, the 15th annual European Control Conference*. Aalborg, Denmark, June 2016, p. 6. URL: `https://hal.inria.fr/hal-01420653`.