

1 **HaptiEditor: Haptics Integrated Virtual Terrain Editing Tool**

2  
3 CÉLESTE MARUEJOL\*, École de technologie supérieure, Canada

4  
5 SEAN BOCIRNEA\*, University of British Columbia, Vancouver, Canada

6 RISHAV BANERJEE\*, University of British Columbia, Okanagan, Canada

7  
8 The contemporary landscape of virtual world design is characterized by the ubiquity of diverse tools and terrain design engines,  
9 which have significantly reduced the barriers to entry in this domain. Despite this progress, the predominant input modalities of  
10 mice and keyboards fail to provide users with haptic feedback during the processes of designing, testing, and experiencing virtual  
11 environments. Addressing this limitation, we introduce HaptiEditor, a virtual terrain editing tool that integrates haptic feedback  
12 through the utilization of force-feedback capabilities offered by the Haply 2Diy device, implemented within the Unity game engine  
13 framework. Our primary objective is to enhance both the design process and the evaluative capacity of designers, as well as to enrich  
14 the immersive engagement of users or players navigating these virtual worlds.

15  
16 Additional Key Words and Phrases: Haptics, Tool Design, Unity, Haply, ForceFeedback, Haptic Texture Rendering

17  
18 **1 INTRODUCTION**

19  
20 Haptics interfaces are often used for enhancing sculpting experiences. However, no significant implementation of force-  
21 feedback interfaces in the case of terrain editing and terrain painting has been created. Many existing haptic-augmented  
22 sculpting applications might be applicable to this task, but use haptic devices with three, or even six degrees of freedom,  
23 which are often both large and expensive. With HaptiEditor, we intend to use the target domain of terrain editing to  
24 our advantage, developing a compelling user interface with a relatively inexpensive two degree of freedom device.  
25 HaptiEditor is a project intending to explore this application of haptics, in the hope of evaluating the promise of our  
26 approach.

27  
28 HaptiEditor is a Unity application which has for objective to edit maps and terrain by painting textures and objects  
29 on different scales. By using the Haply 2DIY we hope to create an interesting approach to terrain creation and edition  
30 through haptic feedback. The goal is to allow real time editing of a virtual space, and simultaneously feel the effect of  
31 the changes right away.

32  
33 The development period span over a period 3 months during the Winter session of 2024, in the course entitled  
34 CanHap501. CanHap501 is a program which covers multiple Canadian universities in the hope of introducing graduate  
35 MSc students to haptic interfaces. This course teaches us how to conceptualize, prototype, develop and do user evaluation  
36 with multimodal human-computer interfaces and haptic experiences. This project is being developed in a team of  
37 three, each of us in different location (i.e. Montreal, Okanagan and Vancouver) with the management issues it entails.  
38 For instance, timezone issues as Okanagan and Vancouver are on a different timezone than Montreal, or versions of  
39 hardware given to each student were different depending on location.

40  
41 Since the development timeline of this project is very short we decided to go with a rapid prototyping approach  
42 as learned in parallel during this course. Moreover, since we didn't have enough time to create a fully fledged terrain  
43 editor software, we agreed to use Unity to simplify a lot of the designing and implementation to get to experiment with

44  
45 \*All authors contributed equally to this project

46  
47  
48 Authors' addresses: Céleste Maruejol, École de technologie supérieure, 1100 R. Notre Dame O, Montréal, Canada, celeste.maruejol.1@ens.etsmtl.ca; Sean  
49 Bocirnea, University of British Columbia, Vancouver, 2329 West Mall, Vancouver, Canada, seanboc@student.ubc.ca; Rishav Banerjee, University of British  
50 Columbia, Okanagan, 3333 University Way, Kelowna, Canada, rishav.banerjee@ubc.ca.

53 the haptic side of the project quicker. Especially since Unity, as a game engine, already implements some solid systems  
 54 for collision, forces and texturing.  
 55

56 HaptiEditor allows exploration of terrain at different scales via a novel zooming system. Using a system of sampling  
 57 existing textures from the surface's material in order to create haptic feedback and Unity's collision system, HaptiEditor  
 58 is able to provide different haptic feedback depending on the scale of the end effector scale in the scene. This allows a  
 59 haptic continuum to enable the user to feel the terrain they are editing at any scale they would potentially need.  
 60

61 The present report is a statement of the advancements and findings made during the development of HaptiEditor.  
 62 We will go over the different avenues tried in order to create HaptiEditor, what was prototyped and how it shaped  
 63 the current software. We will then examine the results gathered through semi-formal user testing and the overall  
 64 appreciation the project received. The results of the user testing and our implementation will thoroughly be discussed  
 65 in the Discussions section.  
 66

## 67 2 RELATED WORK

68 *Image-based Haptic Texture Rendering.* Li et al. [3] produce a method for extracting normal forces from 2D images,  
 69 which may then be rendered by a 3D haptic interface. The paper distinguishes between normal forces, acting in the  
 70 vertical axis, and tangential forces, acting in the surface plane. In our work, we render these tangential forces and forego  
 71 normal forces due to our use of a 2 DOF haptic device. We do not introduce a method for *creating* normal maps for our  
 72 application, but instead reference the work of Li et al. [3] as an example of how one may extract normal information  
 73 from image textures. Li et al. [3] display high rates of differentiability between represented textures with their method,  
 74 though textures used in their evaluation are contrived and not based on real-world images.  
 75

76 *Haptic Perception of Material Properties and Implications for Applications.* Klatzky et al. [1] offer an overview of  
 77 state-of-the-art approaches to haptic rendering of material properties. We bring attention to the discussion on texture  
 78 representation, and note we use a normal mapping approach which allows for both direction and magnitude control  
 79 of surface normals, forming a modified single-point probe model. Klatzky et al. [1] also note roughness and texture  
 80 discrimination as a common evaluation metric for applications targeting textural rendering.  
 81

82 *Hand Movements: A Window into Haptic Object Recognition.* Lederman and Klatzky [2] catalogs exploratory techniques  
 83 used when exploring physical objects. Our interface affords the patterns of lateral motion, static contact, pressure and  
 84 contour folding, mediated through the single-point probe interface offered by our 2 DOF device. We note that these  
 85 affordances were deemed by Lederman and Klatzky [2] to be sufficient (in that they allow performance better than  
 86 chance) for texture and exact shape differentiation.  
 87

## 88 3 APPROACH

89 We attempt to deliver on a haptics focused terrain editing experience first and foremost. We first establish a reliable  
 90 coupling via a virtual proxy to Unity's physics system, and then create a proof of concept terrain editing tool with force  
 91 feedback driven by said proxy.  
 92

93 The three fundamental questions we had were as follows:  
 94

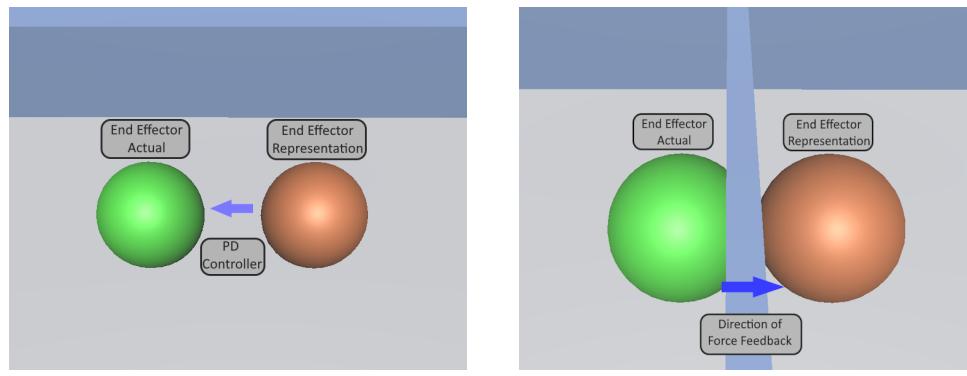
- 100 (1) How should we design a generic virtual coupling between Unity's physics engine and the Haply's force feedback  
 101 mechanisms?  
 102
- 103 (2) How should we detect and render textures in real-time?  
 104

105 (3) How should we design the tool itself, with the main mode of interaction being through the Haply?  
 106

### 107 3.1 Designing a generic virtual coupling between the Unity physics engine and Haply device

109 We built off the Unity template obtained from the Haply GitLab repository as the foundation for our implementation.  
 110 Upon closer examination, it became evident that the forces applied were hard-coded, prompting us to adopt a PD  
 111 controller model [4] facilitated by a virtual proxy (*See 1*).  
 112

113 In our implementation, we utilize a Unity game object, "**End Effector Actual**" to track the ideal positional data of the  
 114 Haply in the absence of obstacles in our terrain, proxied by another game object "**End Effector Representation**" which  
 115 respects collisions with scene objects thanks to its built-in sphere collider, allowing it to interact with Unity's physics  
 116 engine. Subsequently, we establish a PD controller relationship between these two entities. The underlying operational  
 117 logic mandates the "**End Effector Representation**" to consistently attempt to minimize the euclidean distance between  
 118 itself and the "**End Effector Actual**". This behavior is governed primarily by the proportional component of the  
 119 PD controller, supplemented by the derivative component to offer additional smoothing (*See 1a*). In instances where  
 120 the "**Representation**" detects any physical collisions, it directs the Haply to exert a force in the direction of the  
 121 "**Representation**" from the "**Actual**" (*See 1b*). This happens in parallel to the distance minimization attempts of the  
 122 "**Representation**". Consequently, this establishment facilitates an adaptable virtual coupling mechanism, subject to the  
 123 influence of Unity's physics engine via the intermediary proxy.  
 124



140 (a) PD controller moves Representation to Actual if  
 141 there is no obstruction

142 Fig. 1. Virtual Coupling between Representation and Actual End Effector. Note that the "**End Effector Actual**" is never visualized in  
 143 the tool itself.

144 Notably, while the direction vector is three-dimensional, only the X and Z components of this vector are translated  
 145 to the Haply to render force feedback. The selected axes are simply an artifact of our design decision for editing on a  
 146 terrain lying in the X-Z plane.  
 147

### 151 3.2 Generating and rendering textures in real-time

152 Tangentially influenced by the research conducted by Li et al. (2010) [3], our initial approach to texture rendering  
 153 was by sampling a three by three pixel window beneath the end effector representation, subsequently extracting the  
 154 brightness values of each pixel. Each pixel then exerted a force on the end effector away from itself proportional to its  
 155

brightness. This mechanism imparts the perceptual impression of being coerced towards regions of lower luminosity, which can intuitively be mapped to "lower points" in the texture.

However, our development environment affords us a different option. Unity, being a game engine, supports normal-mapping for textures, used in game development for more realistic rendering of surface features. Normal maps contain for each pixel a normal vector representing the direction of the surface of the material, allowing us to compute from a single pixel the direction in which a probe (our end effector) should be pushed by a surface interaction from a single pixel sampled from the normal map. We thus save both memory accesses and computation time, improving simulation speed. Normal maps for in-game materials not only commonly available, but are usually generated programmatically from a sculpted surface texture, and thus also offer improved accuracy without incurring significant overhead to potential users.

Critically, texture-driven force modulation only manifests during end effector movement, thereby avoiding any undesirable tremors during static user positioning. By recalculating this force on a per-frame basis, an appreciable frequency modulation is introduced to the end effector, directly mirroring the texture's visual attributes (See 2).

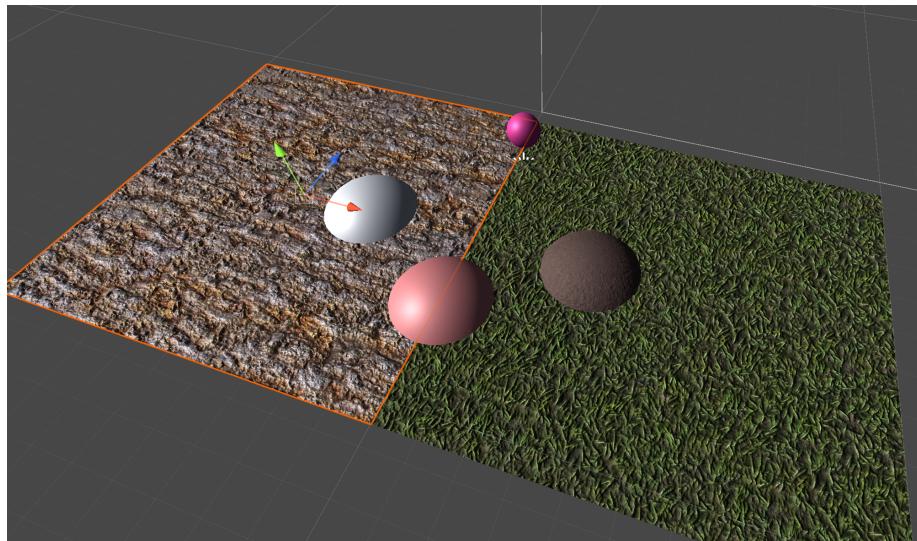


Fig. 2. Two different textures providing different jittery sensations

### 3.3 Tool design and Haply interaction

Game engines typically offer a range of tools for terrain editing, which are primarily oriented towards editing within the editor environment rather than functioning during runtime. Consequently, it became necessary for us to reconstruct the fundamental components of a terrain editing tool within Unity, essentially embedding them as "game mechanics". This endeavor primarily involved leveraging Unity's Terrain game object [5], which is specially built to optimally encode localized texture and static object placement data. A straightforward user interface facilitates the selection among textures, objects, and an eraser tool, further categorized into sub-menus delineating texture types (e.g., grass, sand) and object variants (e.g., trees, rocks) (refer to Figure 3). The user can then designate their preferred object or texture through mouse interaction. Upon selection, the Haply device acts akin to a virtual paintbrush, allowing object

placement or texture painting at the location of the end effector. Activation of the painting action is achieved either through the stylus button integrated with Gen3 DIY devices or, alternatively, by utilizing the space-bar in cases where the stylus button is unavailable.

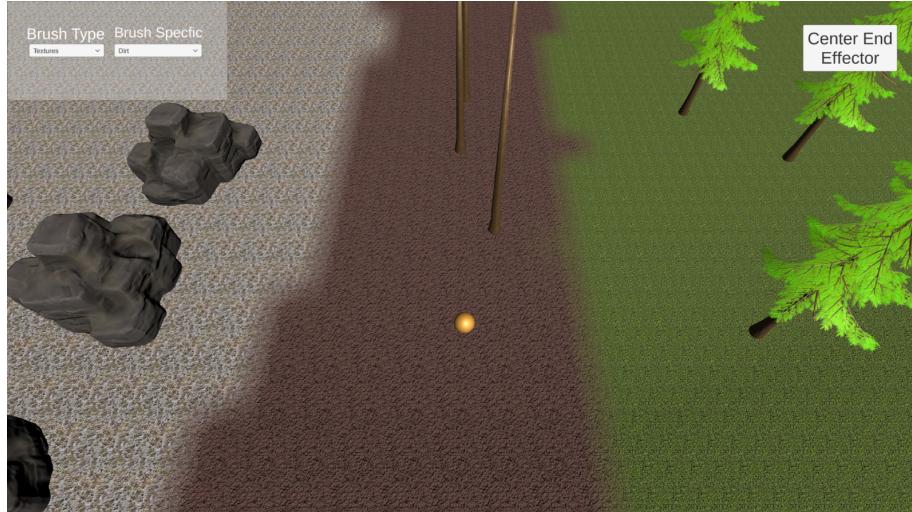


Fig. 3. The final terrain painting interface

The central utility of this approach lies in its capacity to provide users with tactile feedback with their interactions with the terrain in real-time. Objects impart a rigid collider-based resistance as a consequence of the virtual coupling (see Subsection 3.1), while textures provide haptic modulation based on their visual characteristics and behavior in accordance with haptic texture rendering principles (see Subsection 3.2).

## 4 PROTOTYPING

### 4.1 Developing for Unity

There were two main aspects to developing for Unity which we rehaul from the sample Unity Haply Gitlab repository:

*4.1.1 Board Configurations.* Since we have differently configured 2DIY Gen 3 Haply boards (and in some cases, Gen 2 boards), we need a way to switch easily between these configurations without spending time changing code whenever we push code to each other. Given that all the team members in this project worked remotely and in different time zones, this was imperative to a smoother development experience.

To facilitate this, we change the flow of logic for initialising the board to use configuration files, referred to as scriptable objects in Unity. By storing our different configurations in these files, we can easily swap in the appropriate configuration during development time without affecting any of the actual code.

*4.1.2 Utilising Unity's Physics.* The main benefit to utilising Unity was to leverage its physics engine for automated haptic experiences. The process of connecting the Haply to Unity physics has been described in detail in 3.1. There are minor nuances in the implementation itself, specific to Unity. These include making sure the physics engine is running at a higher framerate than normal since the physics is separated from the graphics, enabling continuous collisions

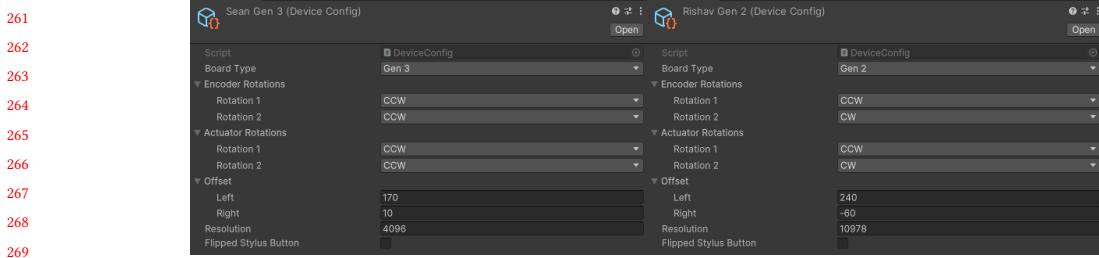


Fig. 4. Different board configurations for the Haply

for a smoother experience, and interpolating all collisions. While these changes require higher compute, our current application runs at 500+ frames per second, and generally improve the experience without any significant detriment to the performance of the program.

#### 4.2 Zooming

In order to allow the designer and user to explore the world at different scales, we incorporate the ability to zoom in and out of the world. From a design perspective, we wished to convey the sense of scale of the world. This was achieved by letting the user explore the ground and the side surfaces of objects in the world when small, and allowing the end effector to "climb" on top of clumps of objects when large, thereby providing force feedback on the basis of the top surface geometry of the objects.

Implementing this practically was rather trivial, since we had worked extensively on abstracting the texture and object force feedback information. By changing the scale of the representation and moving the camera a proportional amount to said scale, the correct forces and texture data was automatically conveyed. Note that the relationship between the camera's movement and the scaling of the end effector is a cubic one. This is in order to maintain an approximate size of the end effector on screen, such that it gives the appearance of the world growing and shrinking around the designer or the user.

#### 4.3 Painting

The incorporation of painting capabilities for objects and textures within a terrain editor is imperative for a multitude of reasons. Firstly, such functionality facilitates the creation of intricate and visually captivating landscapes by allowing users to precisely apply diverse textures and objects onto terrain surfaces, thereby enhancing realism and aesthetic appeal. More specifically, with the ability to add their own textures and objects, this feature enables users to customize their environments with precision, enabling the realization of their artistic visions.

In line with the comprehensive nature of this project, the process of painting underwent multiple iterations of prototypes throughout the development phase. In the forthcoming discussion, we shall examine each prototype and elucidate the insights gleaned from them. The creation of our initial prototype, referred to as "the sprinkler," transpired towards the end of the first iteration. Its primary objective is to validate the efficacy of the PD controller implementation and the integration of Haply's API for the third iteration of Haply's 2DIY. This prototype functioned as a testament to the feasibility of dynamically generating objects during runtime and eliciting force-feedback from interactions between the end-effector and these aforementioned objects. As illustrated in Figure 2. (1), the sprinkler is placing gray circles as

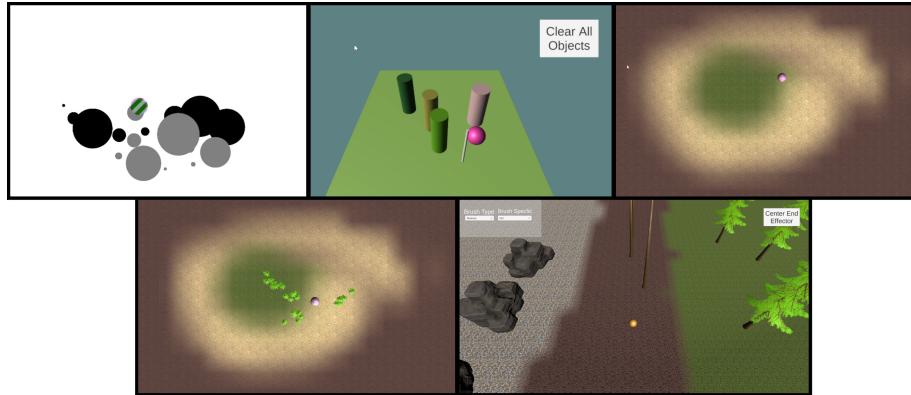


Fig. 5. Representation of the evolution of the prototypes throughout the development of the project (1) shows screenshot of the first prototype made to validate our first iteration (2) represents a screenshot of the first prototype after some code clean up and transferred to 3D space (3) displays a screenshot of the texture painting working on Unity's terrain game object (4) illustrates painting objects on the the terrain previously textured (5) is a capture of the final version which can be seen in Fig. 3

long as we were pressing the stylus button or the spacebar<sup>1</sup>. These gray circles serve as visual indicators devoid of collision detection, providing a prelude to the subsequent placement of black circles, complete with colliders, upon the eventual release of the stylus button or the spacebar key.

Just after finishing the transition from 2D to 3D, we repurposed the code of "the sprinkler" to be usable in 3D space since it is a quick way to, once again, determine whether or not everything is working as intended. We are reusing the same logic behind the placement of object as it has proven effective for the first prototype. In the second image of Figure 2., we can see that the scene is now in 3D and what was previously circles are now cylinders with random colors. The force-feedback is kept the same, if we move the End-Effector into a cylinder, we will be pushed out as if it is a wall. We concluded from this prototype that handling the object placement using a similar script is aligned with our goals for object placement, especially since it is easy to use and learn how to use it, thanks to the stylus affordance.

The subsequent prototype aims to enhance information retention across multiple executions. Our approach pivots towards leveraging Unity's terrain game object, which inherently retains terrain deformation, applied textures, and placed objects. This strategic alignment enables seamless integration with Unity's terrain editing system, obviating the need for extensive bespoke implementation. This not only yields temporal efficiencies but also enhances usability, capitalizing on user familiarity with the platform. Our primary focus lies on object and texture painting, thereby excluding terrain elevation adjustments.

In the prototype corresponding to Figure 2. (3), we introduced the capability to paint textures via mouse input. Employing mouse-based prototyping expedites debugging processes, preempting potential issues arising from haptic interfaces. This prototype proved its importance in facilitating rapid parameter experimentation, refining aspects such as brush radius, fall-off characteristics, and curve adjustments to ensure smoother edge rendering during texture painting. From this prototype as a base, we implemented object creation for terrain and object deletion, still utilizing the mouse as an input. Similarly, it permitted us experiment with different parameters to find what feels best, user experience-wise. The results are displayed in Figure 2. (4).

<sup>1</sup>the spacebar is used as backup throughout the project if the stylus button doesn't work for diverse reasons

365 Subsequently, the amalgamation of prototypes culminated in a singular definitive outcome, as depicted in Figure 2. (5).  
 366 Firstly, we transitioned the painting process to be contingent upon the positional data of the end-effector representation.  
 367 Secondly, we repurposed the coroutine mechanism utilized for object painting from the second prototype, integrating  
 368 it seamlessly with newly devised functionalities for object and texture painting, as well as object erasure. Lastly, we  
 369 devised a concise user interface, affording runtime modifications of tools and their respective painting attributes.  
 370

#### 372 4.4 Texture

373 Our initial approach to texture rendering involved firing a raycast to detect the material underneath the end effector.  
 374 This would then return the corresponding texture information, and the 3x3 window of pixels specifically underneath  
 375 the end effector representation. Using this we could extrapolate the ideal direction the end effector should be moving.  
 376 However using raycasts were computationally expensive, and required us to rethink our approach. The texture rendering  
 377 process in it's current state has been described in detail in 3.2, and overall performs better while retaining accuracy in  
 378 its representation.

## 382 5 EVALUATION AND RESULTS

383 Evaluation took the form of a user study, in which we provided a directed walkthrough of our work to novice users,  
 384 after which we asked a nine linear scale questions about their experience. We walked users through as follows:

- 385 (1) The user was briefed on the purpose of the project, and that the end effector would be their point of interaction  
     with the terrain, both for painting and for feedback.
- 386 (2) The user was directed through the menu and setup screens.
- 387 (3) The user was shown the texture selection menu and told to paint different textures.
- 388 (4) The user was allowed to explore this functionality to their satisfaction.
- 389 (5) The user was shown the object selection menu and told to paint different objects.
- 390 (6) The user was allowed to explore this functionality to their satisfaction.
- 391 (7) The user was shown the object deletion menu and told to delete existing objects.
- 392 (8) The user was allowed to explore this functionality to their satisfaction.

393 After this experience, the user was asked the following set of nine 5-point linear scale questions, and told to answer  
 394 between 5 meaning "most" and 1 meaning "least":

- 395 (1) How easy was it to tell the difference between feedback from objects and feedback from surface textures?
- 396 (2) How easy was it to tell the difference between feedback from different surface textures?
- 397 (3) How easy was it to tell the difference between feedback from different objects?
- 398 (4) How easy was it to tell the difference between feedback from all sources at different zoom levels?
- 399 (5) How helpful was feedback from objects in understanding and navigating the current state of the terrain?
- 400 (6) How helpful was feedback from texture in understanding and navigating the current state of the terrain?
- 401 (7) How synchronized was haptic feedback with the terrain you saw on screen?
- 402 (8) How physically fatiguing did you find the haptic feedback?
- 403 (9) How would you rate your overall enjoyment of the experience?

404 The responses to these questions is charted in Figure 6; we provide data on four respondents, grouped by color. Overall  
 405 sentiment was positive with respect to force feedback, with all respondents rating both the utility and differentiability

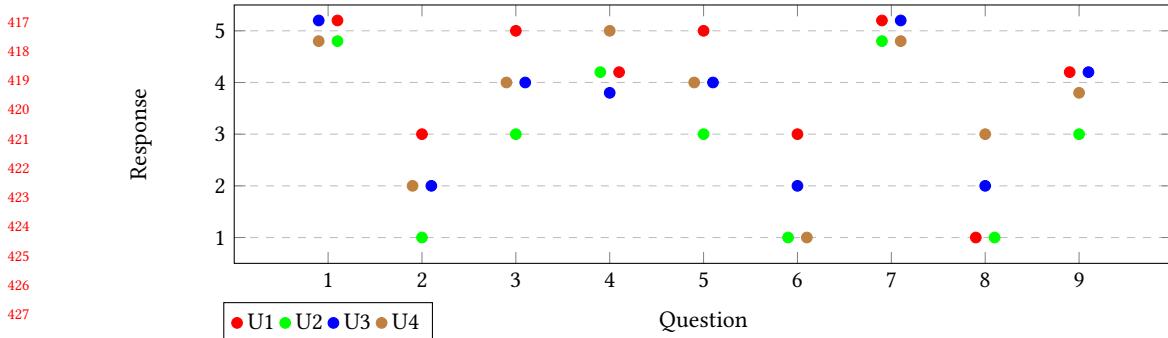


Fig. 6. User Study Responses

of shape-driven feedback highly. Users generally found it difficult to differentiate between textures, and did not find them helpful for terrain navigation. We conjecture improving texture differentiability would improve the usefulness of this feedback, though in its current state textural feedback still provides users feedback on scale and rate of motion. Users did however find it easy to discriminate scale on the basis of haptic feedback. Users generally did not find the experience fatiguing, and found it both well-synchronized with the visual terrain rendering and enjoyable overall.

We conclude that users generally felt that feedback from scene objects was both salient and useful, but that textures fell short due to difficulty in differentiation. However, the positive responses for both sentiment, fatigue, and usefulness of shape-driven feedback supports the viability of the project.

## 6 DISCUSSIONS AND LIMITATIONS

### 6.1 Discussions

*The Haptic-Editing process.* Through our evaluation, we discovered that users generally appreciated shape-driven haptic feedback, as it is salient, easily discriminated, offers practical uses like density estimation, and prevents users from overpopulating terrain regions. We also learned that texture feedback as we implement it in this work is of limited usefulness, as many users complained of poor differentiability. Use of a 2D pantograph interface to populate a 2D terrain object was grasped intuitively, and none of our participants had difficulty understanding the connection between the physical end effector and the representation of the end effector in our editor. We claim our interface would be well-suited as a plugin to Unity, allowing game developers to continue using tools they're familiar with, but with the added benefit of haptic feedback for terrain editing and similar applications. Indeed, force and texture feedback are both driven by parts of the engine, colliders and normal maps respectively, that will necessarily be used by a game developer in this context, and thus our interface requires no additional burden to use.

*Unity as a tool for haptics design.* Throughout history, various art forms, such as music, drawing, photography, film, video editing, game development, and XR development, have experienced significant advancements whenever developers have embraced user-friendly tools. These advancements have been facilitated by the availability of accessible cameras, freely available software for music composition and video editing, as well as widely supported game engines like Unity and Unreal. However, when the primary means of entry into these fields is limited to Java and Processing code, the potential for designing experiences becomes constrained by the technical skills of developers, creating a bottleneck effect. It is imperative to encourage the haptics community to explore beyond mere code frameworks and instead adopt

469 a singular, user-friendly engine (such as Unity) that can empower designers to focus on the user experience aspect of  
 470 haptics, rather than being bogged down by technical jargon. Such an approach would enable individuals to specialize in  
 471 various aspects of haptics, be it hardware, software, or design, akin to the specialization seen in the game development  
 472 industry, thereby enhancing the overall quality of output.  
 473

## 474 6.2 Limitations

475 *Movement in an infinite space.* With our current implementation, the end effector can only move in the virtual space  
 476 a distance proportional to the Haply device's arm length. While we can change the movement scale in the engine,  
 477 the end effector will eventually get stuck due to the haply's arm pantograph constraint. The primary difficulty lies  
 478 in changing the relative positioning of the proxy with respect to the world, and what the underlying user experience  
 479 design philosophy should be for the same without disorienting the user.  
 480

481 *Fine grain control of placed objects.* While we have the ability to place objects around a specific space, we do not have  
 482 the ability to move or rotate a placed object in any degree of freedom, or scale the object up and down. This was an  
 483 initial consideration our project had but had to be discarded in the interest of time and producing a working prototype.  
 484 The main issue with this comes in tackling the user interaction model to edit the transform data of an object. Since the  
 485 haply is the main mode of interaction, we could consider using it similar to a mouse, and designing movement, rotation  
 486 and scale gizmos that can subsequently be used for the editing process.  
 487

## 488 7 ACKNOWLEDGMENTS

489 We would like to thank all the professors of the CH501 course for providing us this unique opportunity of learning  
 490 and developing haptic experiences from scratch. This includes Dr. Oliver Schneider, Dr. Karon MacLean, Dr. Jeremy  
 491 Cooperstock, Dr. Pascal Fortin, Dr. Vincent Levesque and Dr. Pourang Irani. We would also like to thank Dr. Antoine  
 492 Weill-Duflos from the R&D department of Haply Inc. for providing technical assistance during the developing for the  
 493 Haply. Finally we would like to thank the Teaching Assistants Bereket Guta, Anuradha Herath, Sabrina Knappe and  
 494 Juliette Regimbal for guiding us through the various challenges in the course and our project.  
 495

## 496 8 CONCLUSION

500 Throughout the duration of this project, our conceptualization has undergone iterative refinement, integrating insights  
 501 gleaned from collaborative discussions and the outcomes of prototyping endeavors. Initially, our project envisioned  
 502 the development of a comprehensive terrain editor operating within three-dimensional space. However, subsequent  
 503 deliberations with our instructors and internal team discussions led to the realization that the Haply 2DIY platform  
 504 lacked the requisite capacity to accurately perceive depth within a three-dimensional environment. Consequently, we  
 505 resolved to focus primarily on surface-level terrain features, directing our attention towards the painting of objects and  
 506 textures while retaining the innovative capability to manipulate scale.  
 507

508 As a result of these findings, we concluded that the most judicious course of action entailed positioning our project  
 509 as a complementary plugin within the Unity ecosystem, specifically tailored to augment the functionality of Unity's  
 510 terrain game object. In essence, our endeavor reframes the editor as an extension of Unity's game engine, enhancing  
 511 the user experience by facilitating the intuitive painting of objects and textures via the Haply 2DIY.  
 512

513 Amidst our prototyping endeavors, we encountered the unexpected ease of transitioning from haptic texture to  
 514 haptic force feedback. Leveraging Unity's foundational concepts of textures and colliders, we observed that altering  
 515

516

517

518

519

520

521 the scale of the End-Effector representation sphere significantly influenced its collision behavior with surrounding  
 522 objects. At certain scales, the End-Effector exhibited the ability to navigate on top of obstacles that previously blocked  
 523 its progress.

524 Despite the current rudimentary state of our project, we believe it effectively showcases the future possibilities  
 525 granted by such integration. Especially providing the encouraging results we received from our user evaluation and  
 526 testing. Moreover, the extensibility of the current codebase utilizing Unity's robust scripting systems, which lay a solid  
 527 foundation for future expansion. For instance, there are multiple ways to bring enhancements to the projects, one of  
 528 which would come in form of improvements and additional functionalities to the painting UI such as being able to  
 529 change the brush radius. Additionally, experimentation is required to counteract our current limitations and reinforce  
 530 our texture haptic feedback.

## 534 REFERENCES

- 535
- [1] Roberta L. Klatzky, Dianne Pawluk, and Angelika Peer. 2013. Haptic Perception of Material Properties and Implications for Applications. *Proc. IEEE* 101, 9 (2013), 2081–2092. <https://doi.org/10.1109/JPROC.2013.2248691>
  - [2] Susan J Lederman and Roberta L Klatzky. 1987. Hand movements: A window into haptic object recognition. *Cognitive Psychology* 19, 3 (1987), 342–368. [https://doi.org/10.1016/0010-0285\(87\)90008-9](https://doi.org/10.1016/0010-0285(87)90008-9)
  - [3] Jialu Li, Aiguo Song, and Xiaorui Zhang. 2010. Image-based haptic texture rendering. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry* (Seoul, South Korea) (VRCAI '10). Association for Computing Machinery, New York, NY, USA, 237–242. <https://doi.org/10.1145/1900179.1900230>
  - [4] MathWorks. 2011. What is PID Control? – mathworks.com. <https://www.mathworks.com/discovery/pid-control.html>. [Accessed 18-04-2024].
  - [5] Unity Technologies. 2014. Unity - Manual: Terrain – docs.unity3d.com. <https://docs.unity3d.com/Manual/script-Terrain.html>. [Accessed 18-04-2024].

## 545 A VIDEO FIGURE

546 Please find a 2-minute overview of the functionality of our project here:

547 <https://www.youtube.com/watch?v=PPx2JuhQ9Us>

## 550 B INDIVIDUAL CONTRIBUTIONS

### 552 B.1 Rishav's Contributions

554 For the final iteration, we wanted to build an executable that anyone with a haply 2diy board can plug and play.  
 555 Expecting everyone to install Unity would be quite an ask, so we needed to build a live board configurator.

556 The live board configurator would need to modify the following parameters:

- 558 • Board Presets
- 559 • Gen2 or Gen3 (for arm distance offset)
- 560 • Encoder rotation direction
- 561 • Actuator rotation direction
- 562 • Arm offsets for base position
- 563 • Encoder resolution
- 564 • Flipped Stylus Button (Some Gen3 boards have flipped sensor data for the stylus port)

566 Actually passing the appropriate data and reloading the board was significantly more difficult. In a nutshell, I had to  
 567 do the following:

- 570 • Cancel the worker thread simulation task gracefully
- 571 • Flush all forces

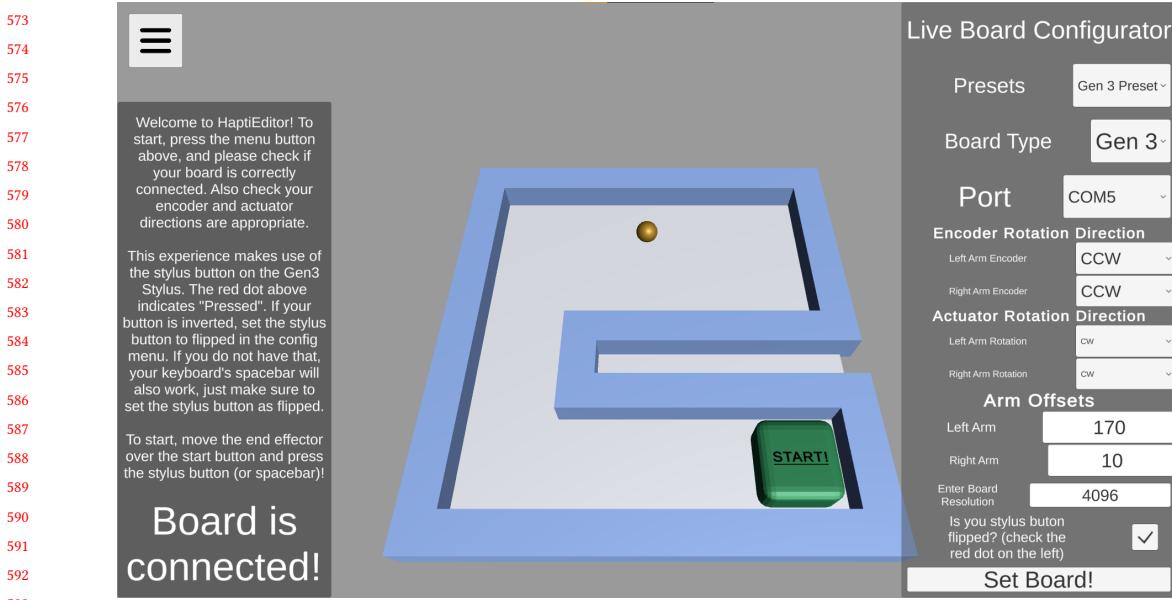


Fig. 7. HaptiEditor configuration menu

- Delete the existing instance of the board (along with the encoder, actuator and sensor parameters)
- Create a new board instance with the new parameters
- Attempt a connection to the new specified port based on the user's selection from current active ports
- Launch a new worker thread.
- Potentially connect to a Button Handler if the scene had one present.

This required a significant amount of refactoring of the core hAPI, but in the end I was successfully able to load and reload new user specified board configurations. The main chunk of this was happening in the `EndEffectorManager.cs` as follows:

```

1  public void ReloadBoard(DeviceConfig customConfig, string targetPort)
2  {
3      // Destroying existing setup
4      haplyBoard.DestroyBoard(); // added function to close port and destroy this board
5      CancelSimulation();
6      SetForces(0f, 0f);
7      Destroy(pantograph.gameObject.GetComponent<Device>());
8      // New Setup
9      device = pantograph.gameObject.AddComponent<Device>();
10     device.Init(); // re-establishes basic connections with pantograph and board
11     LoadBoard(customConfig, targetPort);
12     // Checking for button handler
13     ButtonHandler buttonHandler = gameObject.GetComponent<ButtonHandler>();
14     if (buttonHandler == null) return;
15     buttonHandler.SetButtonState(customConfig.FlippedStylusButton);
16 }
17
18 private void LoadBoard(DeviceConfig customConfig = null, string targetPort = null)
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
807
808
809
809
810
811
812
813
813
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1
```

```

62519 {
62620     device.LoadConfig(customConfig); // loads new config if custom config is not null
62721     haplyBoard.Initialize(targetPort); // attempts connection with new port
62822     device.DeviceSetParameters();
62923     // ...
63024     simulationLoopTask = new Task( SimulationLoop );
63125     simulationLoopTask.Start();
63226 }

```

After this, I decided to improve the visual experience of the project. Users will be expected to understand that they have to configure their board, and that their board might be set up different to our dev setups, so the menu scene should ideally be different from the actual terrain editing scene. Additionally, the stylus button (or space bar) is critical to the experience, so the users should be aware of how to do that as well.

To let the user learn this separately, I worked on a simple menu scene with a bit of flair, and added scene transitions. Users will now be expected to first configure their board, be able to move over a physical button in the world, and then click on the stylus button to enter the terrain painter tool.

I fished out a basic scene manager and transition handler from an older project, and after some tweaking, blender modelling and building an executable for Windows, I produced the menu scene in [Figure 7](#).

## B.2 Sean's Contributions

This iteration, I finalized work on the texture pipeline and integrated it with Celeste's work from iteration 2 on texture and object painting on the terrain object. Celeste wrote some logic to get world position into a corresponding position on the surface of the terrain object, removing the need for our expensive physics ray-cast. The rest of the process involved the following changes:

- Detecting which terrain texture was currently painted onto the surface of the terrain underneath the end effector. This required fetching the blend ratios of each material at the EE location and determining which was present:

```

1 float[,] swatch = terrain.terrainData.GetAlphamaps((int)pixelUV.x, (int)pixelUV.y, 1,1);
2
3 forces.x += normalPixel.r - 0.5f;
4 forces.y += normalPixel.g - 0.5f;
5
6 forces *= intensity;
7 previousPosition = eeTransform.position;

```

- Once I have the ID of the texture to sample, we fetch color from its normal texture, giving us both a direction and intensity of force we immediately apply to our end effector. This greatly simplified the sampling code:
- I introduced a variable `swatchscale` to allow us to control the ratio of world movement relative to movement on the normal map, controlling the linear density of our texture. Normal sampling also greatly improved the accuracy of forces, as we're no longer estimating heightmaps from surface color but now have geometry-accurate normal maps.
- Selected texturally distinct normal maps for our materials for a clearer distinction between painted materials.
- Added back space-bar support for painting, so a user has an alternative to the stylus button.

I then tuned the scaling code Rishav worked on in iteration 2 and added it to our scene. The following changed:

- I added a scale factor for the movement range of the end effector, so it expanded as the scene zoomed out, covering the new area.
- I then tuned the ratios for movement range, end effector size and camera zooming so that they scaled in tandem.
- I changed the painter code so that the painted objects had their appropriate colliders, allowing the large end effector to skate over rocks as we'd intended it to, rather than getting stuck like before.

Lastly, I tuned the texture sampling code again so that it would play nice with the zooming feature we'd introduced. I chose a `swatchscale` such that at high zoom levels, individual surface features like pebbles were quite large and discernible, but with a wider camera, surface features became higher frequency noise and force feedback from geometry became the primary force on the end effector. Textures representations were different depending on the painted texture on the terrain and could be rendered in tandem with terrain objects placed during use. We learned that these pipelines could all coexist in a cohesive way and were pleased to see our parallel approach to developing them paid off.

Post iteration 3, I fixed some outstanding bugs, worked on my section of the report, and prepared the video figure.

### B.3 Céleste's Contributions

As stated in our blog posts for iteration 2, the goal for the third was to merge every concepts and prototypes into one single, preferably enjoyable, experience. In this iteration we ended up working together way more and in closer collaboration by helping each others a lot more. This can be easily explained because we didn't prototype on a different aspect of the experience and were actually making something unique together. This is why sometimes the lines of contributions of what I did and what a team member did will blur into what we did this feature together.

While we knew that we would merge all of our progress into the Terrain scene because the end goal is to use Unity's Terrain game object has our editor, the `TerrainScript.cs` from iteration 2 wasn't usable as is. Firstly, we need to fix the lack optimizations. Secondly, the user should have a way to change the brush type and their specifics without using Unity editor menus (it is necessary if the user interacts with our software through an executable instead of the Unity Project). Thirdly, Lastly, it is unlikely the code we used for texture sampling for haptic feedback would work on the Terrain game object, because it has the particularity to not have a `MeshRenderer` (a component that allows a game object to render a mesh). Finally, there needs to be a way paint using the stylus button instead of the mouse.

During this iteration Rishav did an amazing work at going over the code that has been made and telling us what should be avoided in the future. Following those practices we started a refactoring on `TerrainScript.cs`. During this time, I found a way to optimize the management of the `TreeInstances` what were giving us issues during the second iteration, basically, deleted `TreeInstances` would never really be deleted but replaced with empty version of themselves.

This is problematic because with the core logic of the problem they would be given another collider the next time the software is running even though there is nothing to delete.

```
719 1 private void OnApplicationQuit()
720 2 {
721 3     //test
722 4     List<TreeInstance> trees = new List<TreeInstance>(terrain.terrainData.treeInstances);
723 5     List<TreeInstance> trees_cleaned = new List<TreeInstance>();
724 6     TreeInstance empty_tree = new TreeInstance();
725 7     for (int i = 0; i < trees.Count; i++)
726 8     {
727 9         if (!trees[i].Equals(empty_tree))
728 10             trees_cleaned.Add(trees[i]);
```

```

729 11     }
730 12     terrain.terrainData.SetTreeInstances(trees_cleaned.ToArray(), true);
731 13 }
732
    Using this code when the application is closed we remove all of the TreeInstances that we could consider empty.
733
    The way it works is by going through all the objects in the Terrain TreeInstances and comparing it with an empty
734
    object. If they are different we add them to the new list that will contain all the non-empty TreeInstance.
735

```

Then using the ObjectPlacer.cs as a reference I created two co-routines one for painting object on the terrain and the other for painting texture. We implemented an enumerator containing all the brushes types (i.e. Texture, Object and Object Eraser) and depending on this brush type one of the co-routine or the object deletion will be enabled.

From then on Rishav worked with me on a basic UI so we can change the brush types and which texture/object is being painted.

## C PREVIOUS ITERATIONS

For further reading into the development of this project, links to the previous iterations can be found on the Project Winnipeg website.

- (1) Iteration 1:
  - Celeste's Blog: Iteration 1
  - Sean's Blog: Iteration 1
  - Rishav's Blog: Iteration 1
- (2) Iteration 2:
  - Celeste's Blog: Iteration 2
  - Sean's Blog: Iteration 2
  - Rishav's Blog: Iteration 2

```

758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780

```