3D-Layers: Bringing Layer-Based Color Editing to VR Painting

Emilie Yu, Fanny Chevalier, Karan Singh, Adrien Bousseau

3D-Layers: Bringing Layer-Based Color Editing to VR Painting

EMILIE YU, Inria Centre Université Côte d'Azur, France
FANNY CHEVALIER, University of Toronto, Canada
KARAN SINGH, University of Toronto, Canada
ADRIEN BOUSSEAU, Inria Centre Université Côte d'Azur, France

Fig. 1. Our VR painting system allows artists to achieve rich, editable coloring effects using *3D-Layers*. Starting with *substrate layers* (a) that define the geometry and basic colors of the scene, users can stack multiple *appearance* layers (b) that are composited onto the substrate to produce the final 3D scene (c). Importantly, strokes painted in appearance layers only recolor the substrate strokes they intersect (b, intersections highlighted with a yellow boundary), which avoids the need to position the appearance strokes precisely on the surface of the substrate. In this example, we used appearance strokes to add texture details (white bands on the lighthouse, dark lines on the house and rocks), to paint shadows (lighthouse, rocks), to depict translucency (semi-transparent water painted on the rocks and seabed, subject to a vertical gradient in opacity over the rocks).

The ability to represent artworks as stacks of layers is fundamental to modern graphics design, as it allows artists to easily separate visual elements, edit them in isolation, and blend them to achieve rich visual effects. Despite their ubiquity in 2D painting software, layers have not yet made their way to VR painting, where users paint strokes directly in 3D space by gesturing a 6-degrees-of-freedom controller. But while the concept of a stack of 2D layers was inspired by real-world layers in cell animation, what should 3D layers be? We propose to define *3D-Layers* as groups of 3D strokes, and we distinguish the ones that represent 3D geometry from the ones that represent color modifications of the geometry. We call the former *substrate layers* and the latter *appearance layers*. Strokes in appearance layers modify the color of the substrate strokes they intersect. Thanks to this distinction, artists can define sequences of color modifications as stacks of appearance layers, and edit each layer independently to finely control the final color of the substrate. We have integrated *3D-Layers* into a VR painting application and we evaluate its flexibility and expressiveness by conducting a usability study with experienced VR artists.

Authors' addresses: Emilie Yu, Inria Centre Université Côte d'Azur, 2004 Route des Lucioles, Sophia-Antopolis, France, emilie.yu@inria.fr; Fanny Chevalier, University of Toronto, 40 St. George Street, Toronto, Ontario, Canada, fanny@dgp.toronto.edu; Karan Singh, University of Toronto, 40 St. George Street, Toronto, Ontario, Canada, karan@dgp.toronto.edu; Adrien Bousseau, Inria Centre Université Côte d'Azur, 2004 Route des Lucioles, Sophia-Antopolis, France, adrien.bousseau@inria.fr.

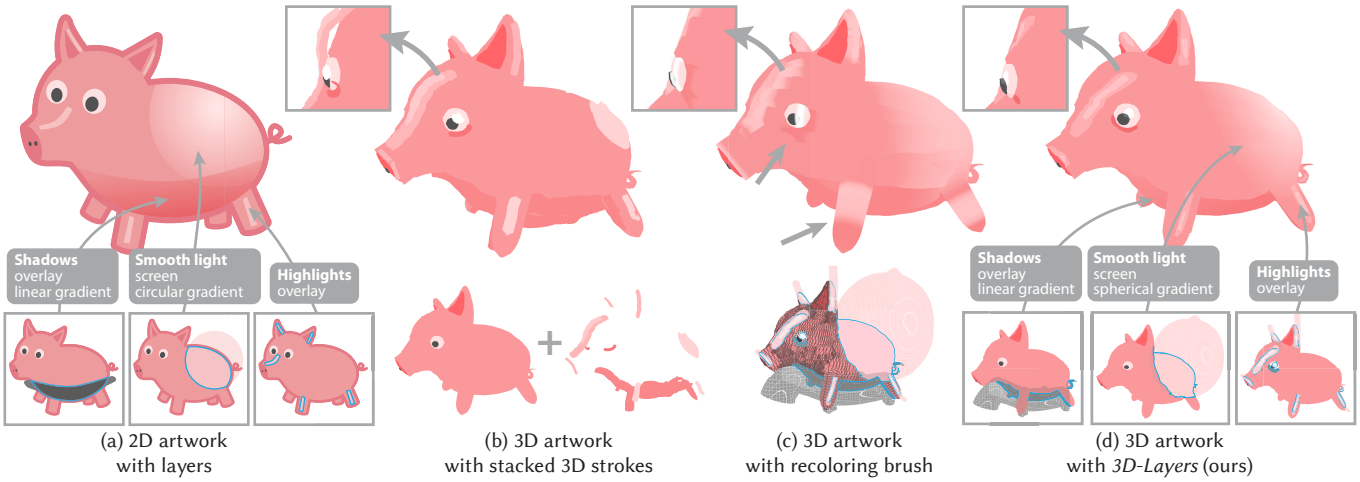
CCS Concepts: • **Human-centered computing** → **Interaction design**; • **Computing methodologies** → **Virtual reality**; Shape modeling; Non-photorealistic rendering.

Additional Key Words and Phrases: 3D painting, virtual reality, digital painting

ACM Reference Format:
Emilie Yu, Fanny Chevalier, Karan Singh, and Adrien Bousseau. 2024. 3D-Layers: Bringing Layer-Based Color Editing to VR Painting. *ACM Trans. Graph.* 43, 4, Article 101 (July 2024), 15 pages. https://doi.org/10.1145/3658183

1 INTRODUCTION

Painting techniques and styles have evolved through centuries, often in response to the invention of new painting technology. For example, impressionism benefited from the introduction of premixed paints in tin tubes, which allowed artists to work more spontaneously away from their studios [Hurt 2013]. More recently, vector graphics and digital painting software supported the emergence of various forms of graphic design by offering *non-destructive workflows* that allow artists to edit the shape, color and opacity of visual elements at will [Adobe 2023b]. We focus on VR painting [Gravity Sketch 2017; Icosa 2020; NVRMIND 2018; Smoothstep 2021], an emerging medium that offers the unprecedented ability to paint 3D brush strokes by gesturing in mid-air, allowing artists to depict 3D scenes in a more direct, expressive, and immersive setting than traditional 3D modeling.



(a) 2D artwork with layers

(b) 3D artwork with stacked 3D strokes

(c) 3D artwork with recoloring brush

(d) 3D artwork with *3D-Layers* (ours)

Fig. 2. **Conceptual Approaches to Layering.** In 2D graphics design, artists commonly use layers to modify object colors, for instance to add shading, shadows and highlights without overriding the base colors of the artwork (a). In VR painting, artists can achieve similar visual effects by stacking strokes close to object surfaces (b). However, since it is difficult to paint at a precise depth, strokes often inter-penetrate or leave visible gaps (b, inset). Alternatively, artists can use the recoloring brush (c), a tool that recolors any stroke vertex it intersects. For visualization purposes, we display the volumetric extent of the brush and we delineate its intersection with the strokes as blue curves (c - bottom). Recoloring vertices reveals the low resolution of the stroke mesh (c - arrows). In our approach, users paint color edits in separate layers (d), offering the same flexibility as in 2D graphics design. Our rendering algorithm computes a precise intersection between the layers and the strokes to be recolored, avoiding the visual artifacts of existing 3D solutions. We strongly encourage readers to watch the accompanying video for animated versions of these three VR workflows.

Despite decades of research [Deering 1995; Schkolne et al. 2001], VR painting applications do not yet offer the same level of editability as 2D painting software. Specifically, 2D digital artists make extensive use of *layers* to structure their artwork such that different components can be edited independently [Aksoy et al. 2017; Tan et al. 2015, 2016]. For example, artists commonly separate the color of an object from its shading by painting these two components in different layers that are stacked and blended in "multiply" mode, and occasionally paint bright highlights in a third layer blended in "screen" mode [Lopez-Moreno et al. 2013; Robertson and Bertling 2014]. But generalizing the notion of layers to 3D raises both technical and conceptual challenges: what should 3D layers contain? How should they affect the 3D painting? How should they be represented, manipulated, rendered? We address this gap by formulating the foundational design concepts for 3D layered VR painting, along with the accompanying data structure and rendering algorithm.

In 2D digital painting, layers offer users a powerful way of defining the final color of pixels as successive transformations of a background layer through blending with other layers. The order in which those transformations are applied is defined by the order in which the layers are stacked (Figure 2a). In contrast, strokes in VR painting are represented as coarse triangular meshes with interpolated vertex colors. While VR artists can paint several semi-transparent strokes in proximity to form a stack in 3D space, painting strokes close together is difficult due to the inherent imprecision of VR painting [Arora et al. 2017; Machuca et al. 2019], and often suffers from rendering artifacts such as visible gaps at grazing angle and *z-fighting* (Figure 2b). An alternative would consist in defining stroke colors through texture mapping [Hanrahan and Haeberli 1990], and implementing a layer stack in texture space. But this approach is impractical because VR paintings are often composed of hundreds of unstructured brush strokes for which it would be difficult to define a globally-consistent texture space. As a consequence, state-of-the-art VR painting software only allows users to define a single color per vertex, which is much less flexible than a layer stack, and must be handled with care to not reveal the coarse triangulation through color interpolation (Figure 2c).

We address the above limitations by introducing *3D-Layers* – a new data structure and companion rendering algorithm to perform layered painting in VR. We first conducted a series of formative interviews with artists who have extensive experience in using VR painting to create immersive illustrations and animations. We distilled a list of challenges unique to this medium, from the interviews, to guide our solution. Notably, the challenges all stem from the fact that in current VR applications, brush strokes define *both* the geometry and color of the object to be rendered. Our key idea is to decouple geometry and color, such that these two components can be edited independently with stroke-based interactions. On the one hand, we define *substrate layers* to contain brush strokes representing 3D geometry. On the other hand, we define *appearance layers* that are stacked over a substrate layer, and that contain brush strokes that modify the color of the substrate strokes they intersect (Figure 2d). Users can define the order and blending mode of the appearance layers arbitrarily, and we describe how to composite the resulting layer stacks efficiently using real-time rendering techniques. Finally, we conducted a usability study to observe how professional VR artists adapt their workflow to the new capabilities offered by 3D-Layers.

Software logs and user feedback highlight how our layering system enables precise and flexible color editing in VR painting.

In summary, we introduce the following contributions:

- The first study on how people paint in VR, which informed our subsequent design,
- The concept of *3D-Layers* that explicitly decouples geometry from colors in VR painting,
- An efficient rendering algorithm to display 3D layers within established VR game engines,
- An interactive system that builds on 3D layers to allow non-destructive color editing in VR painting,
- The results of an expert study demonstrating the expressiveness and usability of our approach.

2 RELATED WORK

2D layering. Layers are a staple feature of 2D digital painting software (e.g. [Adobe 1990; Interactive 2011]), and artists have adopted them in their workflow for a variety of tasks, such as using blending modes to paint shadows and highlights, using masks to delineate precise region boundaries, using parametric gradients or image textures to fill-in such regions [Robertson and Bertling 2014]. Organizing an artwork in multiple layers also facilitates selective and non-destructive editing operations [Myers et al. 2015], as the content of one layer can be modified while other layers are preserved. Motivated by all these benefits, multiple methods have been proposed to generate layered images [Lopez-Moreno et al. 2013], to extract and vectorize layers from photographs, videos, and 3D models [Aksoy et al. 2017; Du et al. 2023; Eisemann et al. 2009, 2008; Favreau et al. 2017; Richardt et al. 2014; Tan et al. 2015, 2016], or to ease navigation in large layer stacks [Shimizu et al. 2019].

Color editing in VR painting. Different aspects of content creation have been studied in VR and XR: planning and storyboarding [Henrikson et al. 2016a,b]; 3D modeling [Drey et al. 2020; Jackson and Keefe 2016]; prototyping full-fledged interactive experiences [Nebeling et al. 2020]. We focus on creating static 3D content by painting colored 3D strokes. While others have studied authoring the *shape* of strokes [Arora et al. 2018; Elsayed et al. 2020; Jiang et al. 2021; Rosales et al. 2021; Yu et al. 2021], we study the complementary challenge of editing their *color* using a layer-based metaphor.

While commercial tools for VR painting provide some forms of layers [Gravity Sketch 2017; Icosa 2020; NVRMIND 2018; Smoothstep 2021], these layers only serve to organize strokes into groups. Artists can then modify all strokes of a group at once, for instance to recolor or animate them. However, such tools do not offer the concepts of *layer stacks* and *layer blending*, which are key to achieve complex, non-destructive color edits.

Another closely related feature in VR painting software is the "recoloring brush", which allows users to recolor existing strokes [Icosa 2020; Smoothstep 2021]. The recoloring brush can be thought of as generating a transient 3D stroke that applies color wherever it intersects with existing strokes. Different blending modes are available to control how the color of the brush mixes with the color of the strokes it intersects. However, since this brush acts on the color of stroke vertices, the quality of the color edits depends on the resolution of the stroke meshes. Moreover, the edits are destructive: once the brush updates a vertex color, the previous color is lost. Our appearance strokes extend the concept of the recoloring brush to support multiple layers, and our rendering algorithm composites these layers independently of mesh resolution.

In contrast to most VR painting systems that define strokes as triangular meshes, Kim et al. [2018] propose a volumetric representation supported by a dynamic octree data structure. The color of each voxel is expressed as a mixture of the colors that have been brushed over that voxel. However, the octree only stores the final mixture, and as such shares the same limitations as vertex colors in terms of non-destructive editing.

Painting over 3D surfaces. Our work also relates to past research on painting over 3D objects, as pioneered by Hanrahan and Haeberli [1990]. In such systems, users trace brush strokes over the surface of the object, and the stroke colors are stored at the vertices of the surface [Agrawala et al. 1995], within a texture map [Hanrahan and Haeberli 1990], or as a textured triangle strip whose visibility is deduced from the visibility of the surface on which the stroke is painted [Kalnins et al. 2002; Katanics and Lappas 2003]. While strokes could be projected to the surface and laid on layers defined over vertex colors or in texture space, such solutions would require high-resolution meshes or a globally-consistent texture map respectively, which is difficult to achieve for VR paintings that are typically made of hundreds of disconnected strokes modeled as low-resolution meshes. Rosales et al. [2019] propose a method to consolidate VR paintings into manifold surfaces, though their method is restricted to ribbon-like strokes that roughly depict such a surface – an assumption that does not always hold (e.g. a single tubular stroke depicts the pig's tail in Figure 2). Furthermore, while strokes drawn in 2D from a single viewpoint can be trivially projected to a 3D surface by ray-casting [Bae et al. 2008; Dorsey et al. 2007; Lee et al. 2022; Sketchsoft 2022], 3D strokes drawn in VR around 3D surfaces require defining a usable and practical projection operation [Arora and Singh 2021].

Alternatively, 3D strokes can be layered at different offsets around the object, as done in *Overcoat* [Schmid et al. 2011]. But the distance between offsets needs to be large enough to achieve proper occlusions between strokes in successive offsets, resulting in visible shells. Baran et al. [2011] describe an algorithm to composite strokes in an order that depends both on their depth and on their painting order to avoid z-fighting. We share a similar goal, as we want our appearance strokes to be composited according to layer order, yet account for the visibility of the substrate strokes they are applied onto. However, in contrast to methods that accumulate flat strokes positioned at the vicinity of a surface [Baran et al. 2011; Kalnins et al. 2002; Katanics and Lappas 2003; Schmid et al. 2011], we model the effect of appearance strokes by their 3D *intersection* with substrate strokes. This formulation is better suited to VR painting as it allows artists to recolor many strokes at once by intersecting them with a single large appearance stroke, approximately placed near the surface.

Finally, octrees have also been used for parameterization-free texture mapping [Benson and Davis 2002; DeBry et al. 2002], where colors over a 3D surface are defined by its intersection with a volumetric texture. While such a representation could be extended to

form a stack of volumetric layers, octrees suffer from limited resolution compared to triangular meshes. In our system, we represent all strokes as triangular meshes to offer VR artists a familiar representation — being to paint the shape substrate or its appearance — and we visualize the intersection between appearance and substrate strokes using techniques borrowed from real-time CSG rendering [Goldfeather et al. 1986].

3 HOW DO PEOPLE DEPICT SHAPE AND APPEARANCE IN VR PAINTING?

Compared to 2D digital painting and 3D modeling, VR painting is a relatively new media with unique capabilities. While a few studies exist on VR *sketching* — mostly about accuracy [Arora et al. 2017; Machuca et al. 2019] and creativity support [Herman and Hutka 2019; Israel et al. 2009] — we are not aware of any study about how people depict shape and appearance in VR *painting*.

As a preliminary step to the development of our system, we surveyed tutorials about existing VR painting tools (T1-10, see complete list in supplemental material) and interviewed 4 experienced VR artists. We focused on Quill [Smoothstep 2021] because this particular VR application has a very active and open community of users[1]. We invited participants that are active members in user communities (Discord channel, Facebook group), or on art sharing platforms (Sketchfab, Artstation, personal blogs).

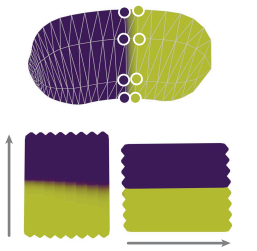
Table 1 details the expertise of the participants we recruited, several of which also participated in the usability study of our system (§ 5.3). All have extensive personal and professional experience working with 2D and VR painting software. They used VR painting to create assets and animations for animated shorts, games, immersive storytelling, physical prints. Several have documented their workflow as blog posts and video tutorials (references will be included upon publication).

We interviewed each participant for one hour over video conference call, after instructing them to be prepared to discuss a recent project that they worked on. Following the *story interview* methodology from user-centered design [Mackay 2023], we encouraged participants to give a detailed account of their workflow by walking interviewers through their process in creating a particular artwork, and we asked for clarifications or more context when participants would describe breakdowns and how they overcame those. These interviews helped us understand typical workflows as well as enduring challenges. Below we focus our discussion on color editing; we provide additional discussion on other aspects of VR painting in supplemental material.

VR painting workflow. Most participants described their workflow as analogous to traditional painting, as summarized by P2: *"you can just kind of draw in the world, right?"* They create shapes by tracing colored strokes in mid-air, and optionally deform or recolor these strokes to refine the result. Quill renders all strokes without shading, meaning that the color that is displayed is the color of the stroke itself.

[1]Facebook group: https://www.facebook.com/groups/virtual.animation/ ; Youtube channel: https://www.youtube.com/@VirtualAnimation

Attempting to render Quill strokes with computer-generated shading reveals that they are made of low-resolution triangle strips or tubes (see inset), resulting in what some participants referred to as "sausages". How then do artists create the illusion of light and shade? P1 explains: *"For oil painting or traditional painting you have to paint light, and that's what you're doing in Quill."* Participants all had techniques to paint shading and shadows in their artwork, either by editing the stroke vertex colors with the recoloring brush or by painting strokes of a different shade just above other strokes. This ability to paint shading by hand is precisely what participants enjoy in their workflow: *"I like to have as much control up front rather than relying on fidelity of what the computer can render. I'd rather have something that works because it's well art directed rather than because the computer renders it." (P3)* P2 explained how painting shadows in an indoor scene allowed him to convey *"the style that we're going for in the game"* , by choosing deliberately to simplify the shape of shadows (e.g. representing the shadow of a stool with an ellipse, despite the physically-accurate shape of the projected shadow being much more intricate).



Challenge 1: Spatial resolution of colors. In existing VR painting tools, strokes are 3D meshes with colors encoded as per-vertex attributes (see inset, top). The finite and typically low resolution of this discretization limits how finely appearance can be defined: *"All the painted lighting has to be bound to the strokes, which means the quality of your lighting is directly proportional to the resolution of the geometry you're using to paint the lighting." (P3)* In addition to placing strokes to define 3D surfaces, artists need to carefully plan the direction of the strokes along these surfaces to achieve the intended color variations (T8, see inset bottom, where the sharpness of a color transition varies with stroke direction). Alternatively, artists artificially increase stroke resolution (T5,T9), although this option should be used with care *"otherwise Quill drawings get really heavy" (P4)* .

Challenge 2: Lack of editability. Artists often need to create assets that adapt to different lighting conditions, or wish to try several color schemes for their artworks. But because the intrinsic color of objects is mixed with shading within vertex colors, performing global color edits over a VR painting requires many local operations, which discourages exploration and even hinders downstream applications, such as VR animations with dynamic lighting. As a workaround, P4 recounts saving duplicates of her assets before recoloring them, for instance by creating a character *"with just a very basic color"* and only painting shading *"as the very last step"* so that she can reuse the asset in different lighting environments.

Challenge 3: Accuracy. Precise alignment of a stroke along a curved surface is hard to achieve when sketching freehand in VR [Arora et al. 2017]: *"It's hard to do something precisely in Quill, because I'm using my hand, I'm using my controller" (P1)* . This lack of precision makes it difficult to paint texture patterns, shadows, and highlights over objects, without carefully repositioning and deforming strokes after the fact. Participants developed strategies to

Table 1. Participants self-reported demographics for our formative (F) and usability (U) studies.

ID	Age	Gender	Profession	Experience (VR painting)	Experience (2D painting)	Study
P1	30-39	W	Product Designer, Artist	Intermediate, 2 years Quill, Tiltbrush	no data	F
P2 – Nick Ladd	18-30	M	App developer, Artist	Expert (5), 6 years Quill, AnimVR, Tiltbrush	Expert (5) Photoshop	F+U
P3 – Edward Madojemu	18-30	M	Art Director	Expert (5), 6 years Quill, Gravity Sketch, Tiltbrush	Expert (5) Photoshop, Procreate	F+U
P4	18-30	W	VR Illustrator, Animator	Expert (5), 6 years Quill, Tiltbrush	Expert (5) Photoshop, Procreate	F+U
P5 – Shaool Levy	30-39	M	Animator	Expert (5), 2 years Quill	Proficient (4) Photoshop, Illustrator	U
P6 – Rene Rebora	30-39	M	Actor, VR-Artist, Wine-Seller	Expert (5), 6 years Quill, Gravity Sketch, Tiltbrush	Expert (5) Photoshop, GIMP	U

alleviate the need for accurate stroke placement, as is the case for P2 who leverages the snap-to-axis feature of Quill's "line" tool to model human-made environments with large quads that align precisely. However, this strategy does not apply to freeform, non-axis-aligned shapes.

Design goals. Based on our findings from the formative study, we define the following design goals:

- **G1. Decouple colors from shapes**: make color edition independent of shape resolution and of stroke arrangement.
- **G2. Non-destructive color mixing**: enable easy backtracking and global editing of colors.
- **G3. Permissive editing**: relax the requirement for precise 3D positioning of coloring strokes.

4 COLORING WITH 3D-LAYERS

To fulfill the above design goals, we introduce *3D-Layers*, a layering system for 3D painting. We first present the overall concept of 3D layers in the context of prior work on layering and compositing, followed by our fast algorithm to render 3D layers in real-time within a VR application.

4.1 Background on layer compositing

In 2D digital painting, an image $\mathbf{I}_n$ composed of a stack of n layers $\mathbf{L}_{i=1..n}$ is formed by applying a compositing operator $\circ$ recursively at each pixel $\mathbf{x}$, in layer order:

$$\mathbf{I}_n(\mathbf{x}) = \begin{cases} \mathbf{L}_n(\mathbf{x}) \circ \mathbf{I}_{n-1}(\mathbf{x}) & n > 1 \\ \mathbf{L}_1(\mathbf{x}) & n = 1. \end{cases} \tag{1}$$

The *over* operator [Porter and Duff 1984] is typically used, in combination with various *blending modes* f_{blend}. Assuming an opaque base layer $\mathbf{L}_1$, we simplify the general formula for compositing and blending ([W3C 2023], §6) to:

$$\mathbf{L}_n(\mathbf{x}) \circ \mathbf{I}_{n-1}(\mathbf{x}) = \alpha_n(\mathbf{x}) \cdot f_{\text{blend}}(\mathbf{L}_n(\mathbf{x}), \mathbf{I}_{n-1}(\mathbf{x})) + (1 - \alpha_n(\mathbf{x})) \cdot \mathbf{I}_{n-1}(\mathbf{x}). \tag{2}$$

Popular blending modes include the *normal* mode $f_{\text{normal}}(\mathbf{L}, \mathbf{I}) = \mathbf{L}$, as well as more advanced modes like *multiply*, *screen* and *overlay* [W3C 2023].

Before presenting our generalization of layer compositing to 3D, let us first revisit existing solutions in light of Equation 1.

The recoloring brush of existing VR painting systems can be formulated as a compositing operation applied to the stroke vertices. Denoting $\mathbf{S}$ the stroke to be recolored, and $\mathbf{B}$ the stroke traced with the recoloring brush, the recolored stroke $\mathbf{S}'$ is obtained by updating the color of the vertices $\mathbf{v} \in \mathbf{S}$ that are intersected by the brush[2]:

$$\mathbf{S}'(\mathbf{v}) = \begin{cases} \mathbf{B}(\mathbf{v}) \circ \mathbf{S}(\mathbf{v}) & \mathbf{v} \in \{\mathbf{S} \cap \mathbf{B}\} \\ \mathbf{S}(\mathbf{v}) & \text{otherwise.} \end{cases} \tag{3}$$

By restricting its effect to the strokes it intersects, the recoloring brush does not require precise positioning (**G2** – see §3). However, by acting on vertex colors, it suffers from limited resolution (**G1**, Fig. 2c) and does not keep track of successive recoloring operations since the new color $\mathbf{S}'(\mathbf{v})$ overrides the previous color $\mathbf{S}(\mathbf{v})$ (**G2**).

Prior work on 3D painting [Baran et al. 2011; Katanics and Lappas 2003; Schmid et al. 2011] uses compositing to mix brush strokes painted over 3D surfaces, either using depth ordering [Schmid et al. 2011], or switching to painting ordering for strokes that lie at a similar depth [Baran et al. 2011]. Formally, the color of a pixel $\mathbf{x}$ covered by n ordered strokes is given by a similar recursion as for 2D layer compositing:

$$\mathbf{I}_n(\mathbf{x}) = \begin{cases} \mathbf{S}_n(\mathbf{x}) \circ \mathbf{I}_{n-1}(\mathbf{x}) & n > 1 \\ \mathbf{S}_1(\mathbf{x}) & n = 1. \end{cases} \tag{4}$$

While this approach is well suited to rendering 3D paintings composed of small, overlapping strokes, users need to place these strokes accurately in depth to achieve a desired ordering (**G3**). Accuracy is especially difficult to reach when painting large, long strokes, as the VR controller trajectory easily penetrates existing strokes, or moves too far from them (Fig. 2b). Yet, large long strokes are handy to paint shading and shadows, as is the case in many of our results.

4.2 The 3D-Layers representation

Inspired by the recoloring brush, we differentiate strokes that affect shape from strokes that affect color, and we tolerate imprecision in coloring strokes by restricting their effect to their *intersection* with shape strokes (Fig. 2d). However, we introduce two key differences to the recoloring brush. First, we organize recoloring strokes in **stacks**

[2]In Quill [Smoothstep 2021], applying the recoloring brush on one vertex of a tubular stroke changes the color of all vertices belonging to the same transverse triangle strip as the edited vertex

of 3D layers, making their effect non-destructive and editable (**G2**). Second, we perform compositing on any point of a stroke rather than on its vertices, allowing us to achieve precise coloring independently of mesh resolution (**G1**).

Denoting $\mathbf{L}_{i=1..n}$ a stack of 3D layers, we compute the color $\mathbf{I}_n$ displayed at a 3D point $\mathbf{p}$ as:

$$\mathbf{I}_n(\mathbf{p}) = \begin{cases} \mathbf{L}_n(\mathbf{p}) \circ \mathbf{I}_{n-1}(\mathbf{p}) & n > 1 \text{ and } \mathbf{p} \in \{\mathbf{L_n} \cap \mathbf{L_1}\} \\ \mathbf{L}_1(\mathbf{p}) & n = 1. \end{cases} \quad (5)$$

Intuitively, layer $\mathbf{L}_1$ contains strokes that define the geometry of the scene, while layers $\mathbf{L}_{i=2..n}$ contain strokes that affect the color of this geometry. We call $\mathbf{L}_1$ the **substrate** layer, and $\mathbf{L}_{i=2..n}$ **appearance** layers.

To further relax the need for tracing precise 3D strokes on curved surfaces (**G3**), we let users adjust a tolerance threshold d to make the intersection $\{\mathbf{L_n} \cap \mathbf{L_1}\}$ more permissive if need be. Figure 3 illustrates how this feature helps drawing fine details on surfaces.

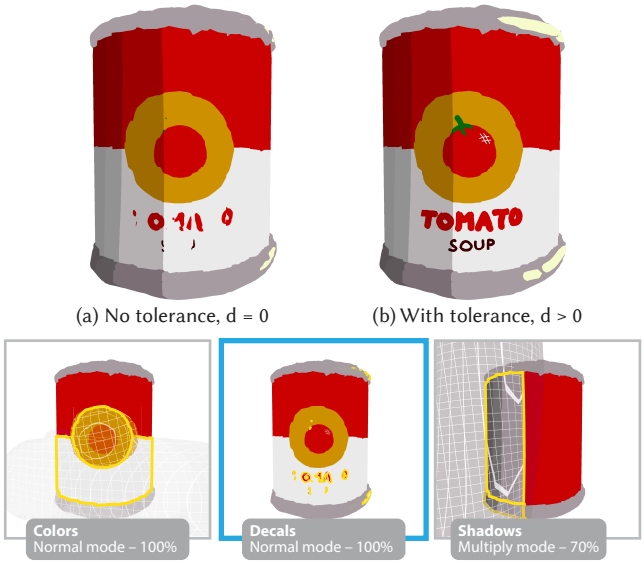
Similarly to layers in 2D painting software, each 3D layer can be edited independently of other layers (**G2**), either by painting new strokes in that layer, by modifying existing strokes (e.g., changing their color or moving them around), by adjusting the layer opacity, the blending mode, or the position of the layer in the stack. We also support a per-layer spatially-varying opacity $\alpha_i(\mathbf{p})$ by extending the concept of linear and radial gradients to 3D.

Importantly, despite our separation of layers into those that define shape and those that affect color, all layers are comprised of 3D strokes. This design makes it possible to reuse strokes in different layers: a substrate stroke can be duplicated and reused as an appearance stroke, alleviating the need to carefully redraw complex shapes when a duplicate can suffice (**G3**). Several of the results shown in this paper were created by duplicating strokes across layers, as discussed in §5.

4.3 The 3D-Layers rendering algorithm

At the heart of our approach is the concept of appearance strokes that modify the color of the substrate strokes they intersect. Our key technical challenge is to compute this intersection efficiently to enable a real-time painting experience, where the user sees the result of an appearance stroke as soon as they start painting that stroke. Since each stroke is represented as a triangle mesh, stroke intersection could be computed using boolean operations on meshes. Yet, while fast and robust algorithms exist for mesh booleans [Cherchi et al. 2022], such algorithms do not yet reach real-time performances for large scenes (more than 100K triangles). Our solution to this computational challenge is to resort to an image-based algorithm inspired by techniques for real-time display of CSG models [Goldfeather et al. 1986]. In addition to being extremely fast (above 90 FPS for scenes composed of hundreds of strokes, corresponding to more than 1000K triangles), our algorithm relies on widely available features in graphics hardware, and is easy to implement within the constraints of typical game engines used for VR development.

Rendering a layer stack. For each stack, we first perform a standard rendering pass of the substrate layer, which writes the color and depth of the visible substrate strokes to the main framebuffer. Additionally, we write an identifier of the stack in the stencil buffer – allocating the 6 lowest bits to that effect, which we will use later on as a mask to ensure that only appearance layers from that stack can act on the color of its substrate. We then loop through all appearance layers of the stack to render them. For each appearance layer, we first composite all strokes to an auxiliary framebuffer, before compositing the result to the main framebuffer. This separation in two framebuffers is necessary and sufficient to achieve the same behavior as 2D painting software, which distinguishes how stroke colors are blended within a layer, from how they are blended between layers.



(a) No tolerance, d = 0 (b) With tolerance, d > 0

Fig. 3. Appearance strokes only recolor the substrate strokes they intersect. But thin strokes need to be placed very close to the substrate to intersect it (a). We allow users to increase the intersection tolerance d of an appearance layer to alleviate the need for precise stroke placement (b).

Rendering strokes in an appearance layer. Within a layer, appearance strokes are alpha-composited in their drawing order if they intersect with the substrate. Concretely, a fragment of an appearance stroke colors the pixel it covers if and only if three conditions are met: (i) the pixel falls within the mask of the substrate ; (ii) the substrate fragment covering that pixel lies inside the appearance stroke ; and (iii) that pixel has not been colored by another fragment of the stroke. We formulate conditions (i) and (iii) with stencil tests, and condition (ii) with depth and stencil tests that detect substrate fragments that lie between the front-facing and back-facing fragments of the stroke, in the spirit of CSG rendering [Goldfeather et al. 1986].

We implement these conditions in 3 rendering passes. The first pass resets the stencil's two highest bits from previous operations. The second pass renders back faces of the stroke and sets the two highest bits of the stencil buffer to 1 if the back-facing appearance fragment is behind the substrate fragment (i.e., if $\delta_A^b > \delta_S$, where $\delta_A^{\{f,b\}}$ is the depth of the front/back appearance fragment and δ_S is the depth of the substrate fragment). The last pass renders front

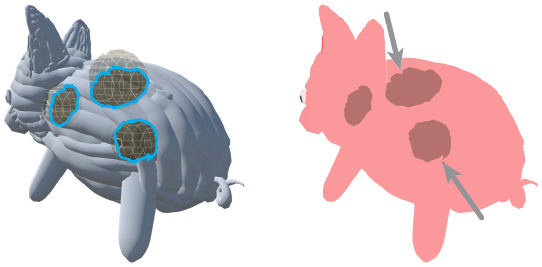
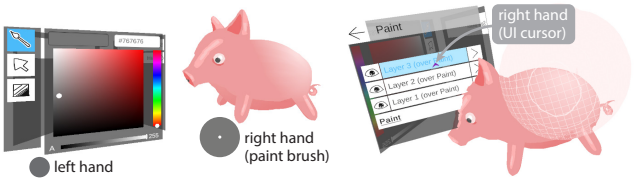


Fig. 4. Substrates that are made of coarse tubular strokes form a bumpy surface (left). The intersection of this surface with the coloring strokes can produce jagged boundaries from some viewing angles (right).



(a) VR UI with hand-held 3D menu (b) Layer panel and visualisation

Fig. 5. We implement 3D-Layers in a VR painting system (a) that exposes the layers' structure via menu panels (b). To help users navigate in layers, we implement a transient highlighting visualisation upon hovering the layer's name (b). Here we show the right-handed user interface.

faces of the stroke if both the usual depth test ($\delta_S > \delta_A^f$) and the stencil test (stack ID == current stack, two highest bits of stencil == 1) pass. This pass writes to the color buffer and sets the second highest bit of the stencil to 0 so that subsequent fragments of the stroke are ignored (condition (iii)). Using a single stencil bit to record the depth test can result in false positives for non-convex strokes that do not intersect the substrate, but intersect the viewing ray more than twice. This is rare in practice, but can be solved by allocating more bits to a depth test counter [Crow 1977], or using a generic buffer to handle stack IDs (at a performance hit).

Coloring with tolerance. Since it is difficult to paint strokes precisely near a surface (Fig. 3a), we achieve tolerance to depth imprecision by implementing an additional render pass to the three passes described above, with a custom depth test in the fragment shader. This pass colors all appearance fragments within a depth threshold d (i.e., $|\delta_A - \delta_S| < d$), if both conditions (i) and (iii) are met (Fig. 3b).

Color blending and layer opacity. Once all strokes of an appearance layer have been rendered to its framebuffer, we composite this framebuffer with the main framebuffer using the specified color blending mode — normal, multiply, screen, overlay — subject to a layer-wise opacity function, which can be a constant, a linear, or a spherical gradient.

Technical limitations. While, our appearance layers could apply to semi-transparent substrates for richer visual effects, we only implemented opaque substrates because rendering transparency by rasterization is challenging in itself. Our formative study revealed that while Quill supports semi-transparent strokes, artists avoid using it due to unpredictable results on different headsets. Finally, the visual quality of our color edits inherently depends on the underlying shape of the substrate. Specifically, irregularities of the substrate surface can result in jagged color boundaries (see Fig. 4).

4.4 Integration within a VR painting system

We integrated 3D-Layers within a prototype VR painting system implemented in C# in Unity [2023]. We leverage Unity's general purpose rendering engine to implement our compositing algorithm[3]. Our prototype offers features similar to commercial VR painting software, including stroke creation via mid-air controller gesture, selections of strokes, rigid transformation, duplication, recoloring or deletion of a selection, navigation in the canvas, undo/redo (see Fig. 5a). We opted to follow design conventions from Quill — e.g. button mappings — since we aim to evaluate our prototype among expert Quill users. Leveraging participants' "interaction knowledge" allows the evaluation to focus on the discovery, learning, and utilization of the 3D layering mechanism and its integration within users' workflows [Renom et al. 2023]. Our prototype also provides basic layer organization and navigation features, such as 2D panels listing all layers in a stack with the possibility to reorder layers; visualization of the content of a layer in-situ with a transient highlighting upon hovering a layer button (Fig. 5b); the possibility to move selected content to another layer. We support importing Quill paintings into our system, enabling users to benefit from the larger feature set of Quill for bootstrapping paintings, then authoring colors with 3D-Layers.

Finally, we instrumented the prototype to log all actions performed by the user during a painting session, and allow users to save and reload their session at any time.

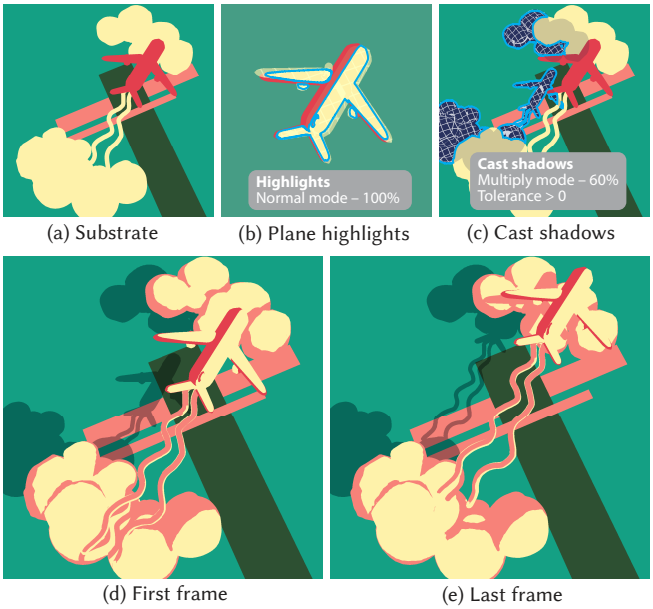
[3]We schedule rendering commands with command buffers (https://docs.unity3d.com/Manual/GraphicsCommandBuffers.html) and write custom shaders with multiple passes (https://docs.unity3d.com/Manual/shader-objects.html)

5 RESULTS AND EVALUATION

We proposed a design for *3D-Layers*, a new primitive in the VR painting workflow that enables artists to use *3D strokes* as representations for both 3D shape and 3D appearance. We evaluate our approach in three complementary ways: (i) we demonstrate the *expressivity* of 3D-Layers, through the reproduction of visual effects inspired by stylized illustrations (see §5.1); (ii) we assess the *performance* of our approach in a technical evaluation (§5.2); and (iii) we evaluate the *usability* of 3D-Layers interaction workflows, through a user study with expert VR painting artists (§5.3).

5.1 Results: 3D-Layers support rich visual effects

We demonstrate the expressive power of 3D-Layers by creating two scenes that showcase a variety of rich visual effects (texture, shading, shadows, highlights). We focus on the creation of these effects, and skip the details of the creation workflow of the base 3D scene – a necessary step regardless of the coloring approach. In practice, we created the geometry of the scenes in Quill to benefit from its larger set of 3D stroke creation features. We organized these strokes into groups, which we imported as substrate layers in our prototype to be enriched and edited using appearance layers. Below, we refer to



(a) Substrate (b) Plane highlights (c) Cast shadows

(d) First frame (e) Last frame

Fig. 6. In this example, we duplicated the substrate strokes of the airplane and shifted them toward the sun to create a large highlight (a,b). We also used duplication to create shadows of the airplane on the clouds and ground (c). Finally, we created a simple animation of the airplane and its shadow by translating the airplane strokes along with their duplicates (d,e). Please refer to the accompanying video to view the animation.

a stack of layers by naming the object painted in the substrate layer of that stack.

Flying over the clouds. We created a VR painting of a plane flying over clouds and fields (Fig. 6), inspired by a 2D marker illustration by Priya Mistry [Mistry 2022]. We reproduce the stylized shading from the illustration by using appearance layers with fully opaque strokes. For the plane, we quickly create a cartoon shading effect by copying strokes from the plane substrate layer to an appearance layer, and slightly translating them toward the sun to create an intersection (see Fig. 6b, intersection outlined in blue). For the shadows cast by the plane and trail, we copy substrate strokes of those objects and paste them in appearance layers stacked over the ground and cloud substrates respectively (Fig. 6c). Since both the plane and its shadow are formed by 3D strokes, we easily animate them by applying the same forward translation (Fig. 6d and e).

Lighthouse island. Figure 1 details the painting of an island with a lighthouse, inspired by a 2D illustration by Owen D. Pomery [Pomery 2021]. We first give an illustrative look to the scene by painting thin black lines on the house and rocks, increasing the tolerance threshold d of the intersection test to make sure that we reach the substrate surface as we paint. Next, we stack two layers over the lighthouse, first to paint the white bands with coarse, doughnut-shaped strokes, and then to create a black shadow on the side by duplicating the lighthouse shape and offsetting it. This layer is composited in multiply mode to obtain plausible shadow colors on both the red and white bands. Finally, we depict a translucent slab of water by painting blue strokes over the seabed, as well as blue strokes over half of the immersed rocks. We set the opacity of the latter to follow a vertical linear gradient, such that water appears to be more transparent near the surface.

Table 2. **Rendering performance.** We provide the number of strokes, triangles and layers for the substrate (S) and appearance (A) for the scenes shown in this paper, along with the rendering speed (in frames-per-second) for the substrate layers only and for all layers composited. For the scenes used in the user study (Forest and Room), we report the numbers averaged over all participants. Our rendering algorithm exceeds 90 FPS on all scenes, even for Fig. 1 that contains more than 1000K triangles.

	# strokes		# triangles		# layers		FPS	
Scene	**S**	**A**	**S**	**A**	**S**	**A**	**S**	**all**
Soup can	54	44	31K	50K	3	7	535	337
Island	298	372	179K	994K	6	18	396	100
Airplane	37	113	35K	428K	6	7	529	271
Forest	255	166	101K	504K	6	11.6	402	128
Room	174	94	51K	285K	7	12.2	458	184

5.2 Technical evaluation: efficiency of 3D-Layers

The usability of 3D-Layers hinges on our ability to update them as soon as users interact with them, tantamount to rendering dynamic intersections between hundreds of 3D meshes in real-time. Table 2 details the rendering performance that our algorithm achieves for representative scenes shown in this paper. Our algorithm performs 3 rendering passes per stroke (4 if $d > 0$, due to the custom depth test) and 2 rendering passes per layer, so both the number of strokes and the number of layers impact performances. Our algorithm exceeds 90 frames per second (FPS) for all of the scenes, as measured by rendering the scenes from the same viewpoints as shown in their respective figures at 1920×1080 stereo resolution on a Nvidia GeForce RTX 2080 GPU.

In terms of memory usage, the fact that we maintain appearance strokes in separate layers inevitably requires additional storage compared to existing solutions based on vertex coloring. For the *Forest* scene that participants of our user study painted in both our system and Quill (see §5.3 below), the Quill version contains 162.5K triangles on average, while our multi-layer version contains 604.8K triangles on average.

5.3 User Evaluation

5.3.1 Study Design. We conducted a study with expert users of Quill to evaluate the utility and usability of our 3D layering solution. The study employs two tasks designed to gain qualitative insights into the following:

(i) Conceptual understanding of 3D-Layers. Given the conceptual similarities to traditional 2D layering, and to vertex re-coloring tools currently available in VR painting, we expect digital artists to easily comprehend the integration of 3D-Layers for visual effects into their painting workflow. We thus include both reproduction and free-form tasks, with a semi-structured interview, to analyze overall user understanding of 3D-Layers.

Fig. 7. **Reproduction task.** Given the forest scene (left), we ask participants to approximately match the look of a target (middle). Once this is done, we instruct them to change the lighting such that the scene appears to be at sunset (right, not shown to the participants).

(ii) Usability of 3D-Layers. Decoupling colors from shapes (**G1** – see §3), enabling non-destructive color mixing (**G2**), and relaxing the requirement for precise 3D positioning of coloring strokes (**G3**), all aim to achieve high usability. We evaluate this by comparing 3D-Layers to the baseline vertex re-coloring method, on a reproduction task.

(iii) Creativity support. A usable, non-destructive workflow should invite creative exploration and experimentation with visual effects. We assess this through a free-form task and follow-up interview.

Guided Painting Task (1): Focused on *(i)* and *(ii)* above, participants were given a starter painting of a forest scene (Fig. 7 left). They were instructed to approximately **reproduce** a target prompt with a variety of visual effects to augment the scene (Fig. 7 middle). Once happy with their result, participants were given a **new objective** as a prompt from a hypothetical client: "I would like the scene to look more like it is happening at sunset." This guided task allows us to observe the process participants adopt to achieve a specific outcome, and modify the outcome given new constraints. Participants were encouraged to complete the task within 30 minutes.

Open-ended Creation Task (2): Aimed at studying *(iii)* above, participants were asked to work on a given painting (see Fig. 9, initial room scene) to **improve or edit it as desired**, within 30 minutes.

We restricted our study to experienced VR painting artists to ensure that our observations pertain to the use of 3D-Layers, uncontaminated by the VR painting learning curve. We also provided a starter painting for both tasks, to keep the study focused on layer use for coloring effects. Additional details on our study design and rationale can be found in the supplemental material.

5.3.2 Protocol. We conducted our study through a video conferencing system, in two sessions. The first session focused on using 3D-Layers. After filling out a brief demographics questionnaire, participants were invited to watch a tutorial video explaining the 3D-Layers features (see supplemental material). Once ready, participants were asked to complete Task 1; after which they filled out the NASA Task Load Index (NASA-TLX) usability questionnaire [Hart 2006]. Then, they were prompted to complete Task 2, followed by the Creativity Support Index (CSI) questionnaire [Cherry and Latulipe 2014] and a semi-structured interview. We invited participants to come back for a second session, to complete Task 1 with Quill only this time, followed by the NASA-TLX questionnaire and a semi-structured interview. Participants were instructed to think aloud as they painted, and were encouraged to take breaks when needed. The study lasted for about 2h over the 2 sessions; participants were compensated for their time. Our prototype was instrumented to log operations, sessions were screen and audio recorded, with semi-automatic transcription. The study was approved by the board of ethics at our institution.

We stress that our approach with 3D-Layers is complementary to usual VR painting tools such as Quill, thus the goal of our study is not to measure our system against an equivalent one (there is none), but rather to observe how experts adopt the new features we propose, and the effects they create with them. The intent in asking users to repeat Task 1 in Quill is to help them reflect on both the expressive power and workflow differences that layers bring. We also note that using the status-quo tool as a baseline can induce human biases, since participants can easily infer the novel system is ours. Participants might favor our system because they perceive it as innovative, or in an attempt to please the evaluators. Conversely, expert participants have deep familiarity with Quill and might feel adversely towards changing their workflow [Remy et al. 2020]. While a longitudinal study might alleviate these effects, such studies are much more demanding to conduct.

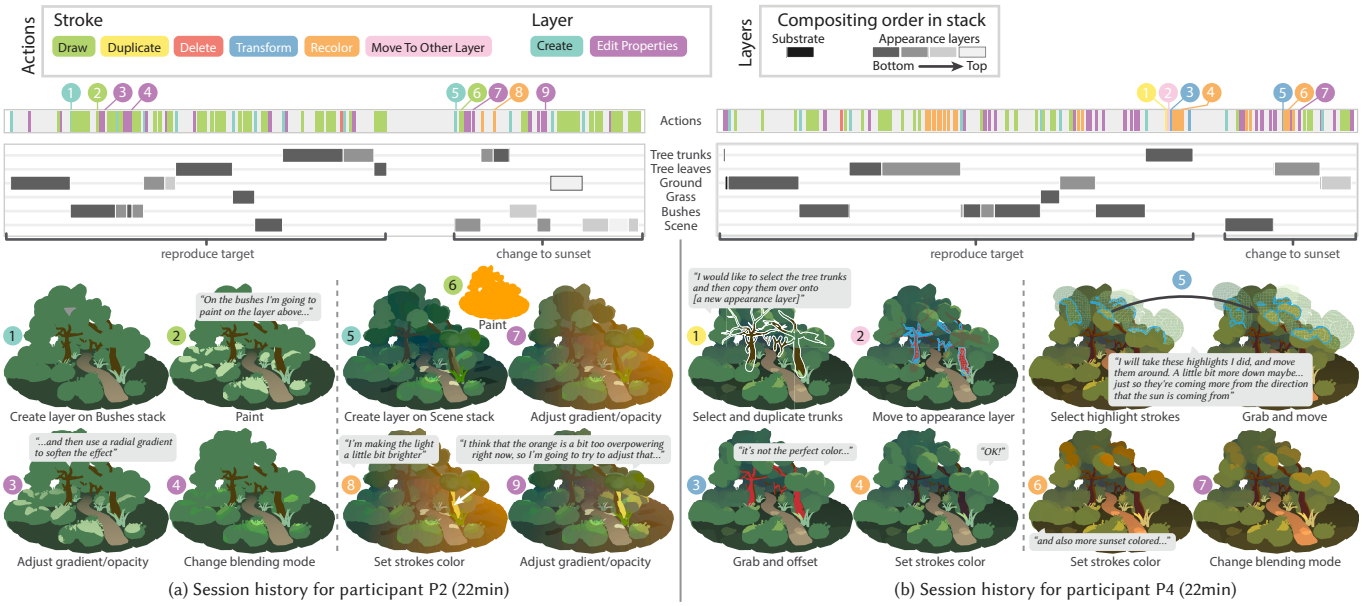
5.3.3 Analysis and Discussion. Five Quill experts participated in our study (P2-P6 in Table 1). We performed a qualitative analysis of the logs, audio transcripts and participant creations, and we organize our findings along our three study objectives *i-iii.* Additional details can be found in the supplemental material.

(i) Conceptual understanding of 3D-Layers. All participants successfully completed Task 1 with little to no guidance, indicating a good grasp of using 3D layers to create visual effects. Fig. 9 features the resulting scenes for the replication component of the task (top row, insets) and after instruction to turn it to a sunset (top row). Participants realized diverse effects to match the reference, including toon-like shading (e.g. all participants on the tree tops and on the bushes), sharp light spots hitting the ground (e.g. P5, reproduction task), atmospheric fog (e.g., P4 with the scene becoming more blue-tinted in the distance) and cast shadows (P2, trees and bushes cast shadows on the ground).

Across both components of Task 1, all participants created multiple layers (avg: 12.4 ; min: 4 ; max: 24), and used them to organize their paint strokes. They also experimented with changing layer parameters, such as blend modes, opacity, and gradients. Participants understood the parameters and their ability to create complex effects. For example, P5 enjoyed using the gradient tool because *"it gives a little bit of a feeling of depth, which has also been missing generally in [flat colored] VR,"* and P6 recounted adjusting layer opacity while adding colors to the room scene (Task 2) because *"you add pure white, that's not how it works in real life because that white, all the colors [...] in the room will reflect on it, that's what I liked about adding a new layer and then reducing the opacity of the gradient"* .

The study suggested a smooth learning curve of the concept, with evidence that participants effectively transferred knowledge from prior practice: *" I was really reminded of what I do a lot when I paint in 2D in Photoshop. There I'm just putting another layer on as overlay or multiply and then adjusting that and so on." (P4)*

In general, participants rated 3D-Layers as slightly more mentally demanding (avg: 6.4 / 10), compared to working in Quill (avg: 4.8)



(a) Session history for participant P2 (22min)

(b) Session history for participant P4 (22min)

Fig. 8. **Participant workflows.** We visualize the series of actions performed by two participants on the guided task. For each participant, we plot the different actions along a timeline using color coding (top). We also visualize which layer stack is edited as additional tracks below the timeline, using different shades of gray to depict layer order. We illustrate representative actions (bottom) and include the comments that participants made while performing those actions. We provide similar timeline visualizations for other participants as supplemental material.

– see Fig. 10 – attributable to the time and cognitive effort needed to shift habits, for example *"It took me a moment to wrap my head around how it works and how the nested layer hierarchy works. But once I understood how it operates, it slowly became second nature." (P3)* P3 also conflated 3D-Layer strokes, with vertex re-coloring strokes they were used to in Quill: *"My first instinct, I think it's because I'm used to using Quill, is I'm trying to colorize instead of using them as strokes. I keep assuming that it is going to work the same way, even though it's... it's definitely very different" (P3)* .

Our visualization of participant workflows (Fig. 8) shows usage across different features of our system to reach their goal, although some participants used a greater diversity of features than others (see additional visualizations in the supplemental material).

The ability to re-use strokes to create different intersections with the substrate, an advantage of 3D-Layers over vertex re-coloring, was particularly well utilized by P4. When prompted to create a sunset effect in Task 1, P4 opted to simply shift the existing highlight strokes down *"just so they're coming more from the direction that the sun is coming from"* (see Fig. 8-b (5)). Similarly, P4 selected and duplicated trunks into an appearance layer to temporarily modify their color as a whole (see Fig. 8-b, (1)-(2)).

(ii) Usability of 3D-Layers. Participants did not face any notable usability challenges in executing their artistic vision. Looking at participants' interactions with the system (Fig. 8, actions timeline), we observe that they were able to create new layers, and navigate the layer stack, to alternate work on different layers (e.g. P2's work on the bushes stack, spanning three appearance layers). They also experimented heavily with blending mode changes, and opacity editing through the opacity slider and gradients (e.g. Fig. 8a (7) & (9), Fig. 8b (7)). This was reflected in the comparable or better rating of our approach relative to Quill, on NASA-TLX dimensions (see Fig. 10, lower scores are better): average physical demand (3.6 comparable to Quill 3.4), better performance (3.8 vs. 5.2) and lower effort (4.8 vs. 5.6). Conversely, participants reported being slightly more frustrated with our approach (3.4 vs. 2.8), partly attributable to using a novel research prototype (as opposed to a familiar professional tool).

Participants appreciated the approach of decoupling shape and color/appearance **(G1)**, relative to Quill: *"With Quill, lighting and modeling are the same thing. It's not a step. There's no step between them. [...] I have to be very precise with the shape of the geometry and the poly count so I can specifically curate the colors I'm using." (P3)* The non-destructive color mixing **(G2)** of 3D-Layers was found to be highly flexible and appealing. Participants quickly applied changes in style when instructed on the "plot twist" in Task 1. For instance, P2-P3-P4-P5 applied a global color change by painting over the whole scene in one color (e.g. orange, red, purple) and blended it with their previous result by using a gradient and changing the blending mode (see Fig. 8a, (6), (7)) ; and P3-P4 selected previously drawn strokes in an appearance layer (e.g. highlights or shade) and re-colored them with a new envisioned color (see Fig. 8b (6)). The contrast between our approach, where *"nothing is rasterized, everything is still very dynamic" (P3)* , with vertex coloring where one has to *"destroy the geometry" (P3)* to make color changes, because it *"hard bakes [the color] into it [the geometry]"* was evident, enabling rapid exploration of color and appearance: *"I feel like I could make multiple versions of the same scene [...] and I could have them all separately and they can all exist at the same time, which would be*

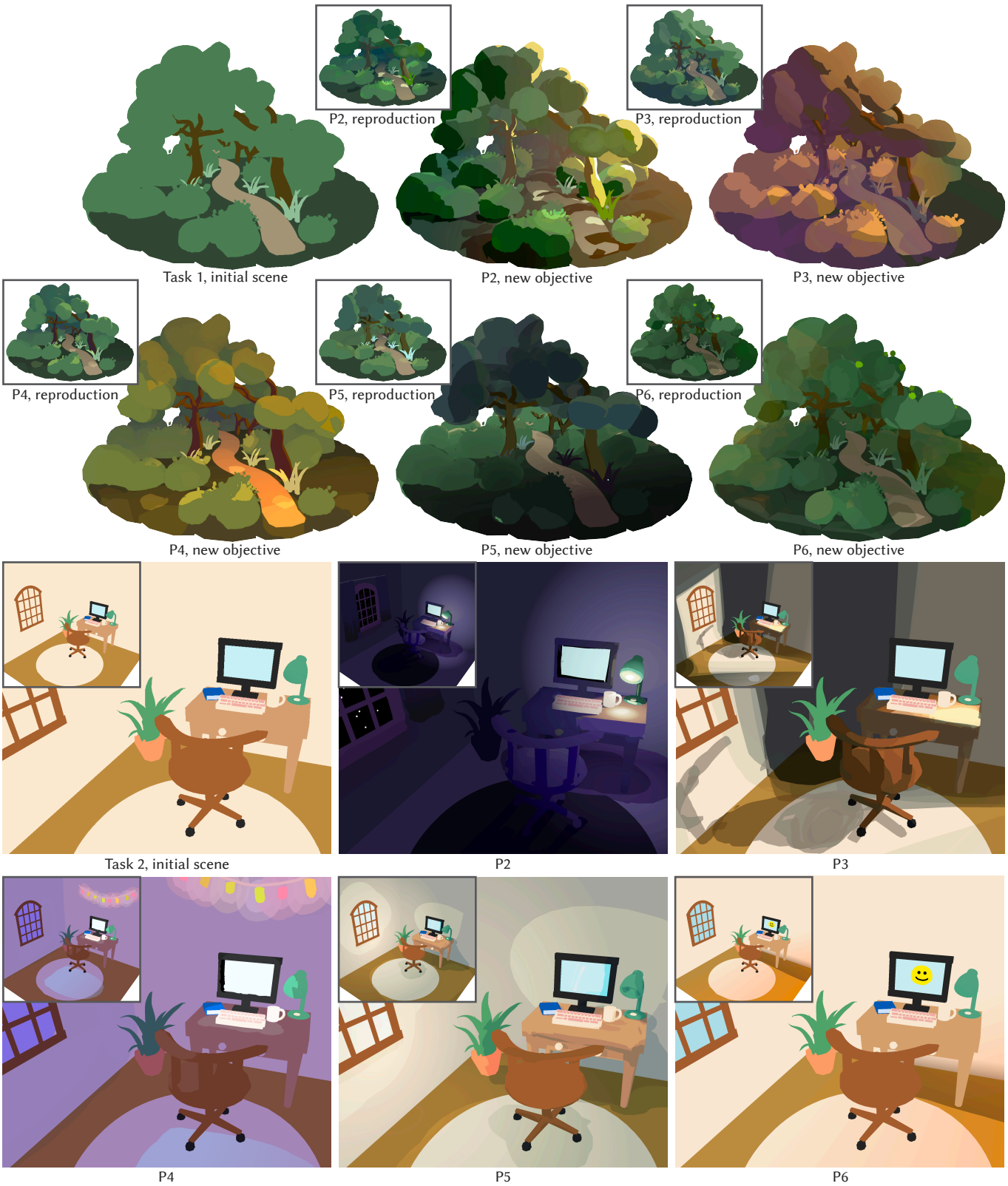


Fig. 9. Painting produced by the participants of our usability study, for the guided task (top) and the open-ended task (bottom). Participants successfully leveraged 3D layers to give unique looks and atmospheres to their respective artworks.

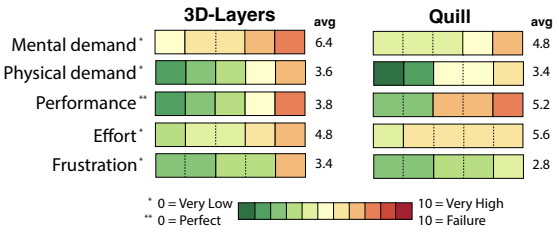


Fig. 10. Participants' answer to the NASA Task Load Index usability questionnaire [Hart 2006], after using 3D-Layers (left) and after using Quill (right) to complete Task 1.

nice because it would also be non-destructive." (P2) . Going further, P3 felt that this would empower the creation of VR-painted animations: *"I don't think I've ever seen a single artist in Quill do like a day-night transition in real time [...] because it's too much work" (P3)* .

Finally, while participants did not explicitly comment on the precision of layering effects (**G3**), precision came into play effectively, for example, when P6 created a smiley face on the screen, or P2 painted stars on the window (Fig. 9). With Quill, P3 went through precise editing of the geometry as an attempt to re-align strokes with the bush for a more convincing shading effect, designed to work best within a reasonable range of viewpoints (Fig. 12a): *"it is basically just like modeling, I match it very closely so you won't really notice unless you're an inch away from it and most people won't look that close."* In contrast, they quickly laid broad strokes onto the bush at the exact intended location with 3D-Layers (Fig. 12b).

iii) Support for creativity. The final outcomes of Task 1 (Fig. 9, two first rows) and the open-ended Task 2 (Fig. 9, two last rows) illustrate the range of diverse effects our participants were able to achieve using 3D-Layers, in just 2 sessions of 30 minutes. For the open-ended task, P2 achieved a night-time atmosphere by applying a dark purple tint, combining recolored substrate strokes and dark purple appearance layers with gradients. Light is expressed with spherical gradients depicting the lamp lighting the wall and desk. In contrast, P3 achieved a highly contrasted day-time atmosphere by overlapping multiple semi-transparent dark strokes that create stepped gradients suggesting light and shadows produced by an opened door, including the shadow of the chair. P5 produced yet a different atmosphere by painting colored spheres in an appearance layer in overlay mode over the wall to create fairy lights, along with a light blue region on the floor faded with a linear gradient to depict moonlight shining through the window.

Participants commented on the merits of our approach to support creativity through quick and fast experimentation: *"It was nice to be able to very quickly establish that [all] I needed to was to make that orange brighter or dimmer just by adjusting a few sliders [...]. I could have changed it to blue or something. It would have made no difference, it would have been fast." (P2)* ; *"In terms of values and colors and playing around with them it's a lot easier for me to do I find with the app. [With Quill] I'm not experimenting as much because there's so much more on the table that it's so hard to experiment with ideas quickly and see if they work." (P3)*

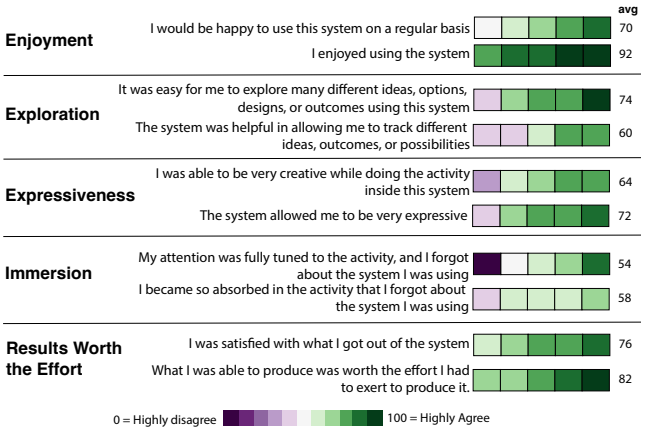


Fig. 11. Participants' evaluation of 3D-Layers, along four of the CSI questionnaire dimensions [Cherry and Latulipe 2014].

These impressions were also reflected in the CSI [Cherry and Latulipe 2014] scores for 3D-Layers (Fig. 11), in terms of enjoyment (avg: 81 out of 100), exploration (avg: 67), and expressiveness (avg: 68), indicating an overall positive creative experience.

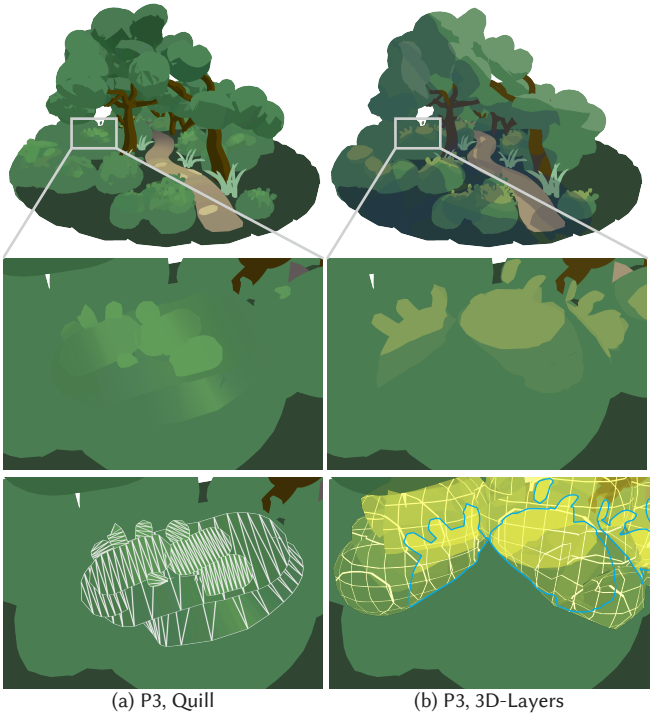
6 CONCLUSION, LIMITATIONS AND FUTURE WORK

We have introduced the foundational concept of *3D-Layers* to bring non-destructive appearance editing to VR painting. Our key conceptual contribution is to explicitly separate strokes into a geometric *substrate* and a stack of *appearance layers* that are rendered and composited in sequence to produce complex, editable visual effects. From a user standpoint, 3D-Layers combines the directness of the re-coloring brush (a popular tool in VR painting), with the flexibility of the layer stack (a ubiquitous feature of 2D painting).

Our experiments with 3D layers and our usability study also revealed limitations that warrant further investigation, as well as functionalities that would extend the possibilities of 3D layers:

Layer navigation. Navigating large layer stacks is complex in 2D software [Shimizu et al. 2019], and it does not get easier in 3D. A common strategy is to toggle the visibility of individual layers on/off to see which parts of the artwork it affects. While we included similar functionality to highlight a layer's content (Fig. 5b), VR poses the additional challenge that the layer panel and its content in the 3D painting might not be simultaneously visible, either due to occlusion or the 360° field of view. Optimization-based methods for in-situ 3D UI layouts [Cheng et al. 2021; Evangelista Belo et al. 2022] or layer selection based on spatial interactions [Ramos et al. 2006; Shimizu et al. 2019] might help alleviate this issue.

Clutter. The fact that appearance strokes only affect substrate strokes at their intersections relaxes the need to draw these strokes precisely. As noticed by P5, *"I could just make a huge brush cause I knew it won't be seen outside of the volume."* However, the same participant then realized that *"When I started doing too much of that, suddenly the whole thing was too complicated [...], your workspace becomes very messy."* In particular, spatial interactions such as stroke



(a) P3, Quill (b) P3, 3D-Layers

Fig. 12. In Quill (a), P3 created highlights on the bushes by painting additional strokes and placing them roughly in front of the bush. With 3D-Layers, the same participant created highlights by painting rough strokes in an appearance layer that color the bush exactly on its surface, where the strokes intersect the bush (blue outline).

selection and color picking become more difficult to perform in the presence of clutter. Redefining those interactions to ignore parts of strokes that do not contribute to the final appearance could be a potential solution.

Compatibility with downstream applications. VR painting is an emerging medium for which there is not yet an established standard. Since our prototype system relies on a dedicated rendering algorithm to display 3D layers in real-time, the artworks produced in that system cannot be directly exported and displayed in other systems. One option could be to *convert* our multi-layer paintings into single-layer colored meshes, similar in spirit to the "flatten image" functionality of Photoshop. Mesh booleans [Cherchi et al. 2022] could be used to introduce new edges in the substrate strokes such that the mesh captures the color boundaries produced by the intersecting appearance strokes well, while adaptive mesh subdivision could be used to best capture smooth color variations.

Additional functionalities of layers. Our prototype includes many functionalities found in 2D layer-based painting software, and could include more. For example, while we offer users the possibility to control the opacity of a layer via linear and spherical gradients, it would be useful to also allow users to *paint* opacity, akin to a *layer mask* [Adobe 2023a]. P4 expressed the need for such a feature when painting the layer that depicts light cast by the window on the ground (Fig. 9), as she would have liked to mask part of that layer to depict the grid of the window. Similarly, layers could be filled-in with volumetric textures to depict repetitive patterns.

Realistic 3D scenes stylized as layered paintings. Another exciting direction of research would be to develop algorithms for creating layered 3D paintings from existing 3D models or from captured 3D scenes, similar in spirit to prior work on vector graphics generation [Eisemann et al. 2009, 2008; Lopez-Moreno et al. 2013] and layered vectorization [Du et al. 2023; Favreau et al. 2017]. Such automatically-generated content could bootstrap the creation process by providing an initial set of strokes that users can import in our system for further refinement.

Animation. Finally, while we have used our prototype system to create simple 3D animations via rigid transformation of the layers, further research is needed to combine 3D layers with more advanced animation tools, such as deformations, skinning and physical simulations. A similar approach has been proposed to animate offset strokes in 3D paintings [Bassett et al. 2013], and might be extended to support our volumetric coloring edits. Such animation features would be especially useful to create dynamic stylized lighting effects in immersive VR paintings [Petikam et al. 2021], which is very tedious to achieve with existing software.

ACKNOWLEDGMENTS

We thank our study participants for their time and invaluable insights into their craft. We thank Berend Baas and Gilda Manfredi for help with testing the interface. We thank the anonymous reviewers for their helpful comments. This work was supported by ERC Starting Grant D3 (ERC-2016-STG 714221), Mitacs Globalink Research Award, NSERC (RGPIN-2018-05072) and by software donations from Adobe.

REFERENCES

Adobe. 1990. Photoshop. https://www.adobe.com/products/photoshop.html.
Adobe. 2023a. Layer Masks. https://helpx.adobe.com/photoshop/using/masking-layers.html.
Adobe. 2023b. Techniques for nondestructive editing. https://helpx.adobe.com/photoshop/using/nondestructive-editing.html.
Maneesh Agrawala, Andrew C Beers, and Marc Levoy. 1995. 3D painting on scanned surfaces. In *Proceedings of the 1995 symposium on Interactive 3D graphics*. 145–ff.
Yağız Aksoy, Tunç Ozan Aydın, Aljoša Smolić, and Marc Pollefeys. 2017. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM Transactions on Graphics* 36, 2 (2017), 19:1–19:19.
Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. SymbiosisSketch: Combining 2D & 3D sketching for designing detailed 3D objects in situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–15.
Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George W Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR.. In *CHI*, Vol. 17. ACM, 5643–5654.
Rahul Arora and Karan Singh. 2021. Mid-air drawing of curves on 3D surfaces in virtual reality. *ACM Transactions on Graphics (TOG)* 40, 3 (2021), 1–17.
Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 151–160.
Ilya Baran, Johannes Schmid, Thomas Siegrist, Markus Gross, and Robert W Sumner. 2011. Mixed-order compositing for 3D paintings. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–6.
Katie Bassett, Ilya Baran, Johannes Schmid, Markus Gross, and Robert W. Sumner. 2013. Authoring and animating painterly characters. *ACM Transactions on Graphics* 32, 5 (Sept. 2013), 1–12.
David Benson and Joel Davis. 2002. Octree Textures. *ACM Trans. Graph.* 21, 3 (jul 2002), 785–790.

Yifei Cheng, Yukang Yan, Xin Yi, Yuanchun Shi, and David Lindlbauer. 2021. Semanticadapt: Optimization-based adaptation of mixed reality layouts leveraging virtual-physical semantic connections. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 282–297.

Gianmarco Cherchi, Fabio Pellacini, Marco Attene, and Marco Livesu. 2022. Interactive and Robust Mesh Booleans. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

Erin Cherry and Celine Latulipe. 2014. Quantifying the creativity support of digital tools through the creativity support index. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21, 4 (2014), 1–25.

Franklin C Crow. 1977. Shadow algorithms for computer graphics. *ACM Siggraph* 11, 2 (1977), 242–248.

David DeBry, Jonathan Gibbs, Devorah DeLeon Petty, and Nate Robins. 2002. Painting and rendering textures on unparameterized models. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 763–768.

Michael F Deering. 1995. HoloSketch: a virtual reality sketching/animation tool. *ACM Transactions on Computer-Human Interaction (TOCHI)* 2, 3 (1995), 220–238.

Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. 2007. The mental canvas: A tool for conceptual architectural design and analysis. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. IEEE, 201–210.

Tobias Drey, Jan Gugenheimer, Julian Karlbauer, Maximilian Milo, and Enrico Rukzio. 2020. VRSketchIn: Exploring the Design Space of Pen and Tablet Interaction for 3D Sketching in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 1–14.

Zheng-Jun Du, Liang-Fu Kang, Jianchao Tan, Yotam Gingold, and Kun Xu. 2023. Image vectorization and editing via linear gradient layer decomposition. *ACM Transactions on Graphics* 42, 4 (2023), 1–13.

Elmar Eisemann, Sylvain Paris, and Fredo Durand. 2009. A visibility algorithm for converting 3D meshes into editable 2D vector graphics. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28 (2009).

Elmar Eisemann, Holger Winnemoeller, John C. Hart, and David Salesin. 2008. Stylized vector art from 3d models with region support. *Computer Graphics Forum (Proc. of EGSR)* 27, 4 (June 2008).

Hesham Elsayed, Mayra Donaji Barrera Machuca, Christian Schaarschmidt, Karola Marky, Florian Müller, Jan Riemann, Andrii Matviienko, Martin Schmitz, Martin Weigel, and Max Mühlhäuser. 2020. VRsketchpen: unconstrained haptic assistance for sketching in virtual 3D environments. In *Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology*. 1–11.

João Marcelo Evangelista Belo, Mathias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, and Kaj Grønbæk. 2022. AUIT – the Adaptive User Interfaces Toolkit for Designing XR Applications *(UIST '22)*. Association for Computing Machinery.

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2017. Photo2ClipArt: Image Abstraction and Vectorization Using Layered Linear Gradients. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia Conference)* 36, 6 (November 2017).

Jack Goldfeather, Jeff P M Hultquist, and Henry Fuchs. 1986. Fast Constructive-Solid Geometry Display in the Pixel-Powers Graphics System. In *Proc. Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*.

Gravity Sketch. 2017. Gravity Sketch. https://www.gravitysketch.com/.

Pat Hanrahan and Paul Haeberli. 1990. Direct WYSIWYG painting and texturing on 3D shapes. *ACM SIGGRAPH computer graphics* 24, 4 (1990), 215–223.

Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage publications Sage CA: Los Angeles, CA, 904–908.

Rorik Henrikson, Bruno Araujo, Fanny Chevalier, Karan Singh, and Ravin Balakrishnan. 2016a. Multi-Device Storyboards for Cinematic Narratives in VR. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 787–796.

Rorik Henrikson, Bruno De Araujo, Fanny Chevalier, Karan Singh, and Ravin Balakrishnan. 2016b. Storeoboard: Sketching Stereoscopic Storyboards. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4587–4598.

Laura M. Herman and Stefanie Hutka. 2019. Virtual Artistry: Virtual Reality Translations of Two-Dimensional Creativity. In *Proceedings of the 2019 on Creativity and Cognition (C&C '19)*. Association for Computing Machinery, 612–618.

Perry Hurt. 2013. Never Underestimate the Power of a Paint Tube. https://www.smithsonianmag.com/arts-culture/never-underestimate-the-power-of-a-paint-tube-36637764/. *Smithsonian magazine* (2013).

Icosa. 2020. Open Brush. https://openbrush.app/.

Savage Interactive. 2011. Procreate. https://procreate.com/.

Johann Habakuk Israel, Eva Wiese, Magdalena Mateescu, Christian Zöllner, and Rainer Stark. 2009. Investigating three-dimensional sketching for early conceptual design—Results from expert discussions and user studies. *Computers & Graphics* 33, 4 (2009), 462–473.

Bret Jackson and Daniel F Keefe. 2016. Lift-off: Using reference imagery and freehand sketching to create 3D models in VR. *IEEE transactions on visualization and computer graphics* 22, 4 (2016), 1442–1451.

Ying Jiang, Congyi Zhang, Hongbo Fu, Alberto Cannavò, Fabrizio Lamberti, Y K Henry Lau, and Wenping Wang. 2021. HandPainter - 3D Sketching in VR with Hand-based Physical Proxy. In *ACM Conference on Human Factors in Computing Systems (CHI)*. ACM.

Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. 2002. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 21, 3 (July 2002), 755–762.

George Katanics and Tasso Lappas. 2003. Deep Canvas: Integrating 3D Painting and Painterly Rendering. In *Theory and Practice of Non-Photorealistic Graphics: Algorithms, Methods, and Production Systems*.

Yeojin Kim, Byungmoon Kim, and Young J Kim. 2018. Dynamic deep octree for high-resolution volumetric painting in virtual reality. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 179–190.

Joon Hyub Lee, Hanbit Kim, and Seok-Hyung Bae. 2022. Rapid design of articulated objects. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–8. Publisher: ACM New York, NY, USA.

Jorge Lopez-Moreno, Popov Stefan, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. 2013. Depicting stylized materials with vector shade trees. *ACM Transactions on Graphics* 32, 4 (2013).

Mayra Donaji Barrera Machuca, Wolfgang Stuerzlinger, and Paul Asente. 2019. The Effect of Spatial Ability on Immersive 3D Drawing. In *Proceedings of the ACM Conference on Creativity & Cognition (C&C'19)*. ACM, 173–186.

Wendy E Mackay. 2023. DOIT: The Design of Interactive Things. Selected methods for quickly and effectively designing interactive systems from the user's perspective. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–3.

Priya Mistry. 2022. Journey. https://www.priyamistry.co.uk/portfolio/product/departures-journey/.

Brad A Myers, Ashley Lai, Tam Minh Le, YoungSeok Yoon, Andrew Faulring, and Joel Brandt. 2015. Selective undo support for painting applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 4227–4236.

Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. XRDirector: A Role-Based Collaborative Immersive Authoring System. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 1–12.

NVRMIND. 2018. AnimVR. https://www.meta.com/en-gb/experiences/pcvr/1741124389277542/.

Lohit Petikam, Ken Anjyo, and Taehyun Rhee. 2021. Shading rig: Dynamic art-directable stylised shading for 3D characters. *ACM Transactions on Graphics (TOG)* 40, 5 (2021), 1–14.

Owen D. Pomery. 2021. Owen Pomery - AIRBNB. https://owenpomery.com/airbnb.

Thomas Porter and Tom Duff. 1984. Compositing digital images. *SIGGRAPH* 18, 3 (1984), 253–259.

Gonzalo Ramos, George Robertson, Mary Czerwinski, Desney Tan, Patrick Baudisch, Ken Hinckley, and Maneesh Agrawala. 2006. Tumble! Splat! Helping Users Access and Manipulate Occluded Content in 2D Drawings. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '06)*. Association for Computing Machinery, 428–435.

Christian Remy, Lindsay MacDonald Vermeulen, Jonas Frich, Michael Mose Biskjaer, and Peter Dalsgaard. 2020. Evaluating creativity support tools in HCI research. In *Proceedings of the 2020 ACM designing interactive systems conference*. 457–476.

Miguel A Renom, Baptiste Caramiaux, and Michel Beaudouin-Lafon. 2023. Interaction Knowledge: Understanding the 'Mechanics' of Digital Tools. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–14.

Christian Richardt, Jorge Lopez-Moreno, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. 2014. Vectorising Bitmaps into Semi-Transparent Gradient Layers. *Computer Graphics Forum (Proc. EGSR)* 33, 4 (July 2014), 11–19.

Scott Robertson and Thomas Bertling. 2014. *How to Render: the fundamentals of light, shadow and reflectivity*.

Enrique Rosales, Chrystiano Araújo, Jafet Rodriguez, Nicholas Vining, Dongwook Yoon, and Alla Sheffer. 2021. AdaptiBrush: Adaptive General and Predictable VR Ribbon Brush. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia)* 40, 1 (2021).

Enrique Rosales, Jafet Rodriguez, and Alla Sheffer. 2019. SurfaceBrush: From Virtual Reality Drawings to Manifold Surfaces. *ACM Transaction on Graphics* 38, 4, Article 96 (2019), 15 pages.

Steven Schkolne, Michael Pruett, and Peter Schröder. 2001. Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 261–268.

Johannes Schmid, Martin Sebastian Senn, Markus Gross, and Robert W Sumner. 2011. OverCoat: an implicit canvas for 3D painting. In *ACM SIGGRAPH 2011 papers*. 1–10.

Evan Shimizu, Matt Fisher, Sylvain Paris, and Kayvon Fatahalian. 2019. Finding Layers Using Hover Visualizations. In *Proceedings of the 45th Graphics Interface Conference on Proceedings of Graphics Interface 2019*. 1–9.

Sketchsoft. 2022. Feather: 3D Sketchbook. https://feather.art

Smoothstep. 2021. Quill. https://quill.art/.

Jianchao Tan, Marek Dvorožňák, Daniel Sýkora, and Yotam Gingold. 2015. Decomposing Time-Lapse Paintings into Layers. *ACM Transactions on Graphics (TOG)* 34, 4, Article 61 (July 2015), 10 pages.

Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-space Geometry. *ACM Transactions on Graphics (TOG)* 36, 1 (Nov. 2016).

Unity. 2023. Unity Engine. https://unity.com/products/unity-engine.

W3C. 2023. Compositing and Blending. https://www.w3.org/TR/compositing/.

Xue Yu, Stephen DiVerdi, Akshay Sharma, and Yotam Gingold. 2021. ScaffoldSketch: Accurate Industrial Design Drawing in VR. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST).*