UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Alicia Sudlerd

UE5 Flair's Integration – UI, Shaders, and Image Algorithms for NPR

Master's Thesis (30 ECTS)

Supervisor(s):
Ulrich Norbisrath, PhD
Santiago Montesdeoca, PhD

Tartu 2024

UE5 Flair's Integration – UI, Shaders, and Image Algorithms for NPR

Abstract:

This thesis presents an initial implementation and setup of the Flair plugin for Unreal Engine 5, aiming to bring the same easy-to-use features from Autodesk Maya. This work is made for 3D artists and Unreal Engine Developers, helping them use the plugin smoothly and effectively across both major digital graphics tools. The scope covers porting key features including Style Presets and Material Presets, along with their sub-modules, meticulously adapted to align with the architecture and functional paradigm of Unreal Engine 5. The initial version of the plugin is dedicated to a usability studying with the target groups—Unreal Engine developers and 3D artists. The findings from this research not only highlight the intuitive of the implemented features but also reveal areas for improvement. This work establishes the first step towards building a tool to support creative workflows across 3D digital art and game development fields.

Keywords:

Non-Photorealistic Rendering (NPR), Slate widgets, User Experiences, Shaders, Computer Graphics, Synthesis, GPU Pipeline, Post-processing

CERCS: P170 Computer science, numerical analysis, systems, control

UE5 Flairi integratsioon – kasutajaliides, varjutajad ja pildi algoritmid NPR jaoks Lühikokkuvõte:

See väitekiri esitab Flair'i plugina esialgse rakendamise ja seadistamise Unreal Engine 5 jaoks, eesmärgiga tuua samad kasutajasõbralikud funktsioonid Autodesk Maya'st. Töö on suunatud 3D-kunstnikele ja Unreal Engine arendajatele, aidates neil plugina sujuvalt ja tõhusalt kasutada mõlemas peamises digitaalse graafika tööriistas. Ulatus hõlmab peamiste funktsioonide, sealhulgas stiilipresettide ja materjalipresettide, koos nende alammoodulitega, portimist, mis on hoolikalt kohandatud Unreal Engine 5 arhitektuuri ja funktsionaalse paradigma järgi. Plugina esialgne versioon on pühendatud kasutatavusuuringutele sihtrühmadega—Unreal Engine arendajad ja 3D-kunstnikud. Uuringu tulemused mitte ainult ei rõhuta rakendatud funktsioonide intuitiivsust, vaid paljastavad ka parendamist vajavad alad. See töö loob esimese sammu tööriista loomiseks, mis toetab loovaid töövooge nii 3D digitaalkunsti kui ka mängude arendamise valdkondades.

Võtmesõnad:

Mittefotorealistlik renderdamine (NPR), Slate vidinad, Kasutajakogemused, Varjutajad, Arvutigraafika, Süntees, GPU torustik, Järeltöötlus

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Table of Contents

1 Introduction

The evolution of digital art and computer graphics has advanced the landscape of visual storytelling and design, bringing tools and creativity to artists' fingertips. Artists can now express their visions more dynamically by transitioning from paper or 2D art sketches to 3D models which often face a challenge from the complexities of mimicking hand-drawn art in a 3D space. This is where Non-Photorealistic Rendering (NPR) [1] comes into play.

1.1 Non-Photorealistic Rendering (NPR)

Non-Photorealistic Rendering (***ie*** *NPR*, stylized rendering, artistic rendering, or expressive graphics) is a set of techniques within computer graphics that contain a range of algorithms to render arts and visual effects that diverge from the traditional *photorealistic rendering (PR)* [1]. NPR simulates art styles such as painting, drawing, and cartoony looks in 3D environments. The techniques are also suitable for artists to portray artistic expressions, such as ideas or emotions [10], directly within a 3D environment while still maintaining the looks of 2D art styles.

NPR is actually used in many digital creations, starting from movies to video games, 3D modelling tools, or even in Japanese anime. Nonetheless, it is less common than photorealistic rendering (*PR*) in 3D graphics.

Figure 1. Zelda: Breath of the Wild (2017)

Figure 2. Run Totti Run (2022)

With the abundance of digital media content, standing out becomes challenging. Digital media consumers are actively looking for unique aesthetic visual experiences. NPR offers 3D artists a way to characterize their work with distinctive styles, letting artists explore fresh and creative paths through 3D computer graphic tools.

[1] *What is Photorealistic Rendering?* https://it-s.com/what-is-photorealistic-rendering/

There are numerous tools related to NPR: *PPixel* [2]from Polygon Pictures, as well as Blender NPR (BNPR) which involved many contributors in the Blender community. These tools enable 3D artists to configure the rendering style. Flair by Artineering provides a tool and a stand-alone engine that provides extensive stylized presets for digital media artists.

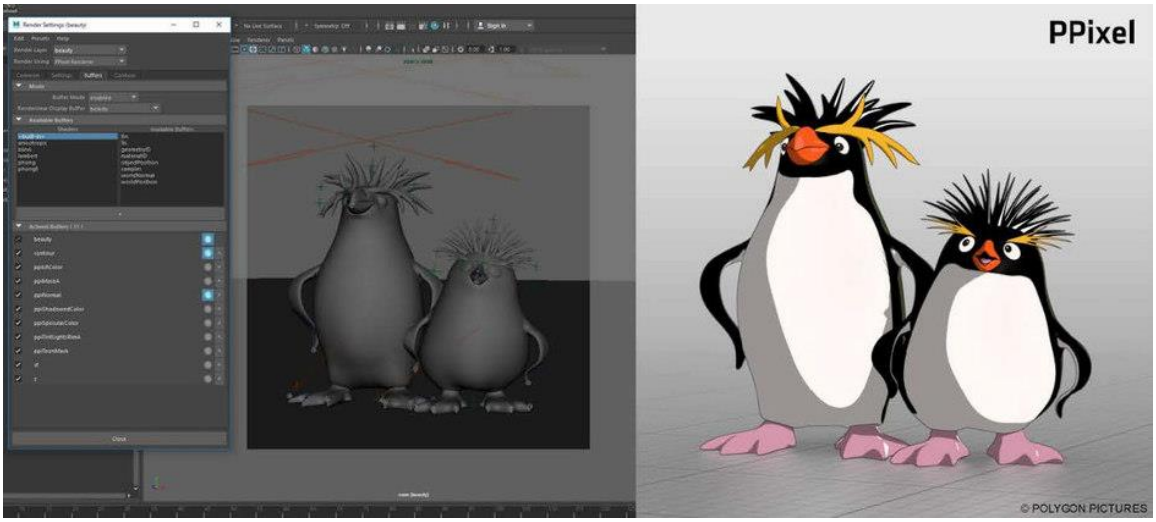

Figure 3. PPixel by Polygon Pictures

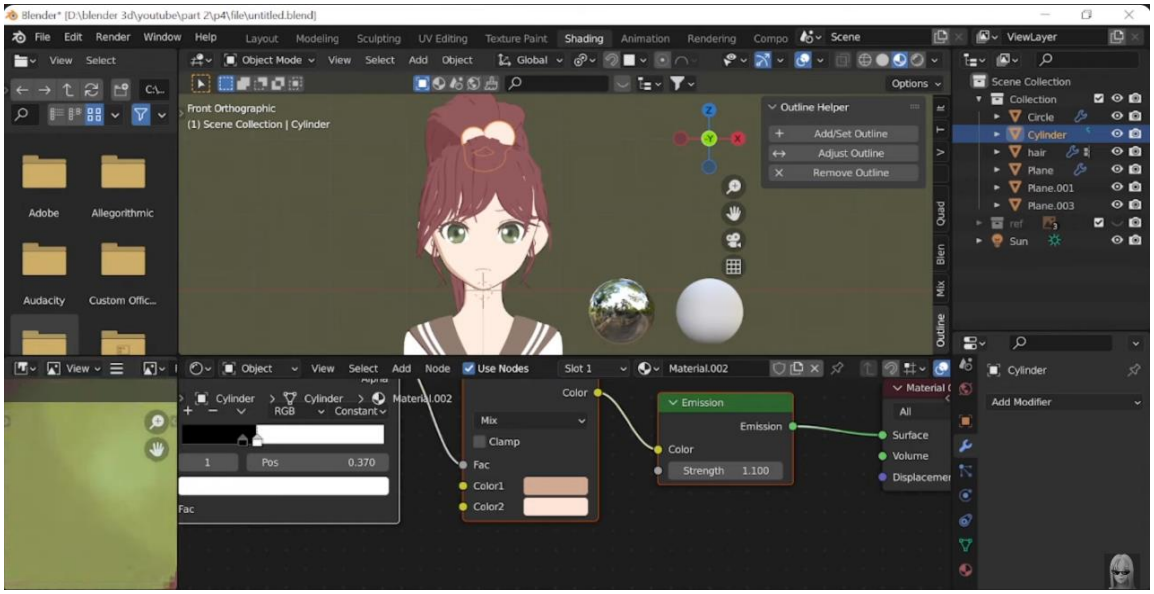

Figure 4. Modeling + Blender NPR shading anime schoolgirl character - Blender 3.0[3]

[2] https://www.awn.com/news/polygon-pictures-announces-ppixel-non-photorealistic-rendering-software
[3] https://www.youtube.com/watch?v=3h8iC5e2A8k

1.2 Flair

Flair[4] is a real-time engine developed by Artineering exclusively for NPR. Its node graph interface enables 3D artists to visualize and modify image processing pipelines. While Flair is not a native node-based engine, Flair Graph provides a user-friendly node-based interface for configuring image processing pipelines to achieve various styles. Additionally, Flair is designed to integrate with third-party engines.

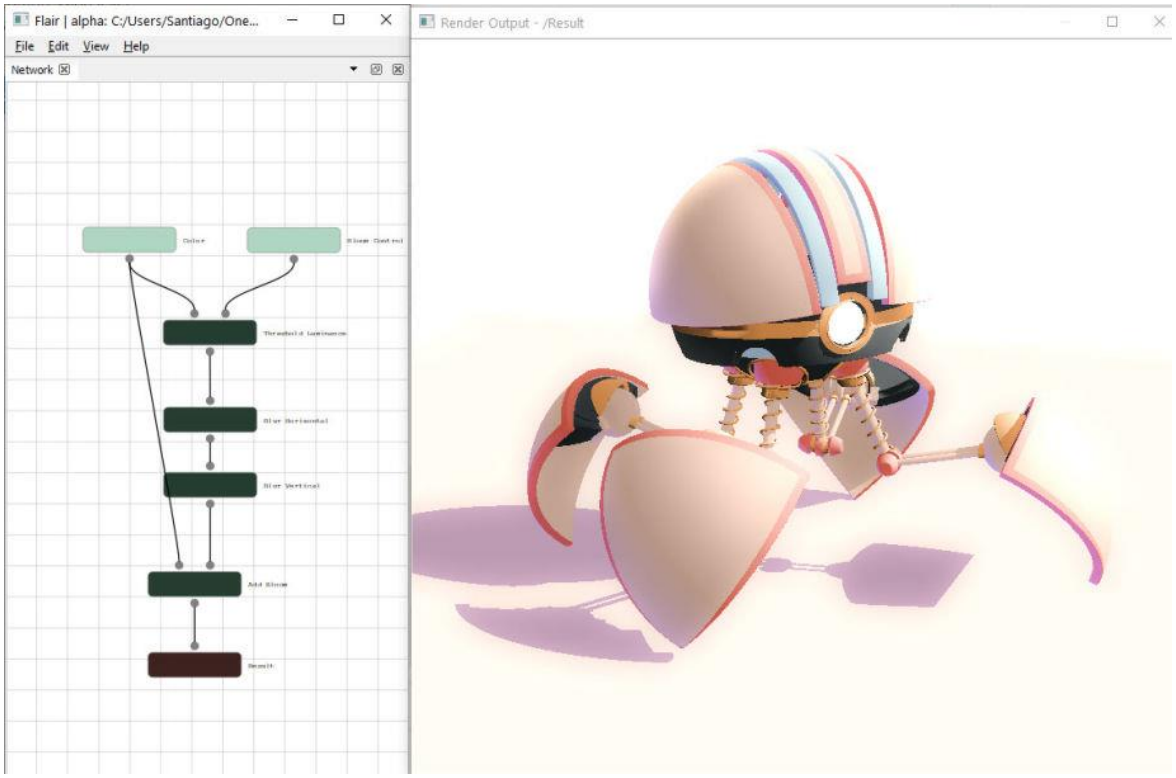

Figure 5. Flair Graph

1.3 Flair in Autodesk Maya

The stand-alone Flair shader plugin is integrated exclusively with Autodesk Maya[5]. The plugin has real-time rendering capabilities and seamless Maya integration. Flair also comes with a set of stylized presets. Additionally, artists can create custom style presets using the Flair engine, empowering them to explore and customize styles and shaders.

[4] https://80.lv/articles/flair-an-overview-of-a-node-based-engine-for-stylized-3d-art/
[5] https://en.wikipedia.org/wiki/Autodesk_Maya

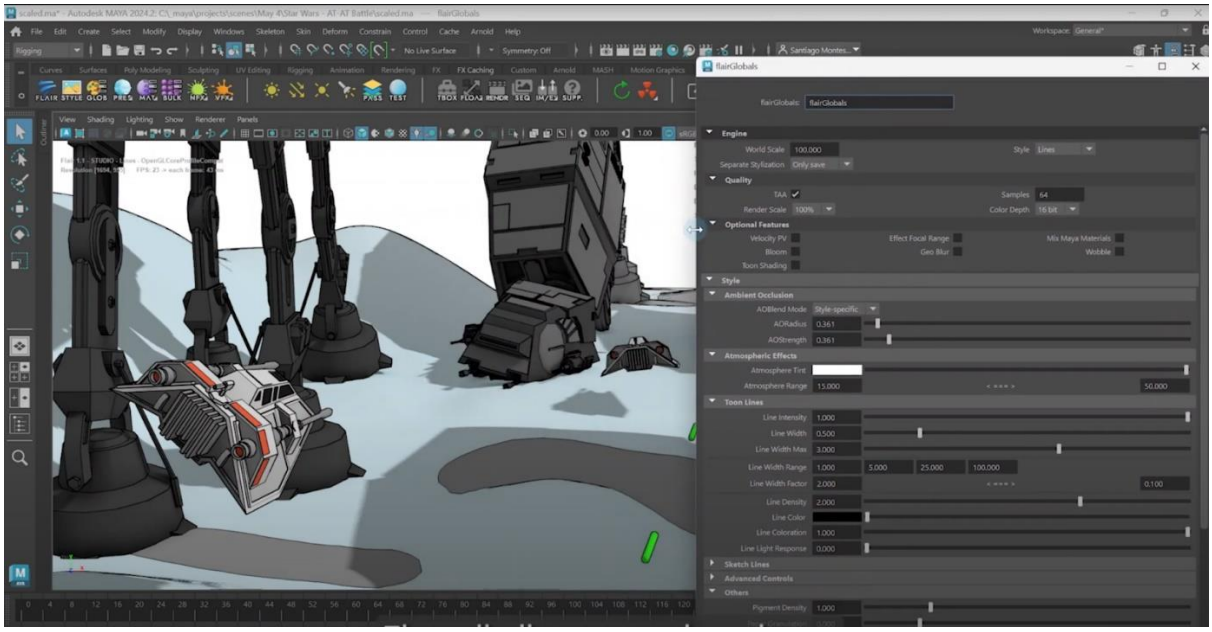

Figure 6. Flair for Maya

1.4 Integration to Unreal Engine 5 (UE5)

While Unreal Engine[6] is well-known for its photorealistic (PR) style; it does not offer much extensive NPR tools. There are limited NPR plugins available on the marketplace, making it challenging to access such functionality.

Often, artists [6] and developers have to create and use their own NPR solutions and work-arounds [8], which is inconvenient. Artineering sees this as an opportunity to expand Flair's market into UE5 users, especially 3D artists. Thus, the goal of this thesis is to fill in this gap and make NPR more accessible.

This thesis explores the integration of Flair's existing features, currently exclusive to Maya, into UE5. Integrating Flair into UE5 will give artists and developers direct access to Flair's NPR techniques, so that they can create and use stylized content without the necessity of knowing Maya and other complexities in the production pipeline. This will simplify the production process and expand creative possibilities within the 3D graphics realm.

1.5 Thesis Structure

[6] https://www.bairesdev.com/blog/what-is-unreal-engine/

2 Existing NPR, Maya, UE5 Plugins, and Documentation

The research and analysis presented in this chapter lay the groundwork for Flair in Unreal Engine 5. It includes a [2] comprehensive review of individual studies and existing works such as the design thinking behind the user interface and the overall stylization pipeline provided by Artineering [3]. Understanding these aspects helps us envision areas that need improvements and maintain existing effective design elements to keep in development.

2.1 Level of Control in Maya for NPR

Non-Photorealistic Rendering (NPR) tools in software like Maya were primarily built around effects and post-processing algorithms. These tools were designed to automatically generate stylized outputs without expecting customization from the user. [2] This often resulted in a rigid workflow, limiting artists' ability to interact and personalize the rendering outcomes. To bridge this gap, Flair introduces art-direct tools for artists to adjust their styles at various levels of control in real-time.

Flair in Maya has a shelf that organizes as a set of art-directed tools to prioritize different levels of controls from making NPR rendering for globally, some specific area, or in a particular 3d mesh. The design of the control has solid proof of usability from current users.

2.1.1 Global Control: Style Presets

Style presets (Figure 7) give users the highest level of control by applying an NPR style over the rendering scene – such as watercolor, charcoal, or toon – with predefined control parameters which can be controlled from the Attribute Editor.

The users can also create their own style through the Flair Graph, typically indicated by the prefix "_g". These customized shader graphs can then be saved and loaded from the Preset panel under a specified name for use in other projects.

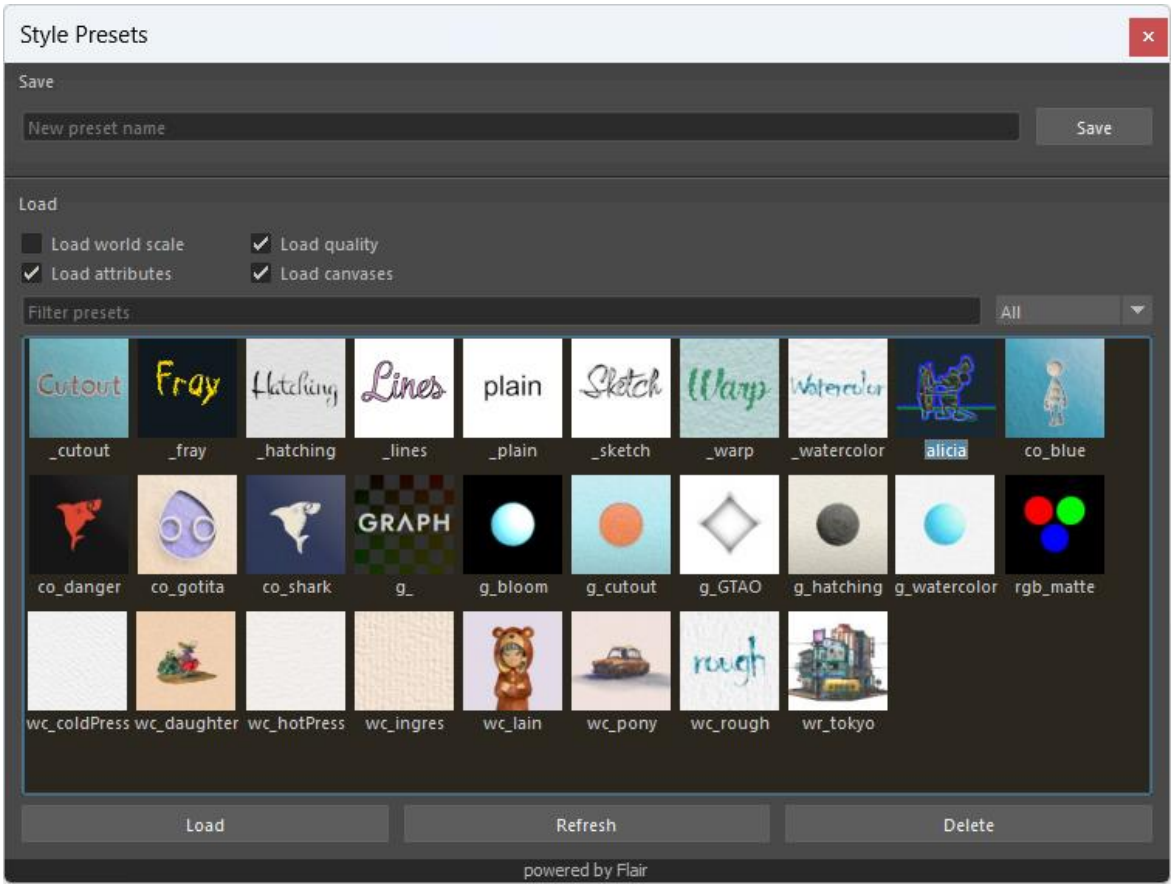

Figure 7. Flair Style Presets

Flair Graph

Flair Graph, a core feature inside the engine of Flair, is a node-based graphics engine used to manage shader code, and image processing. Users can loading any style preset prepended with "g_". Then, click “Show Flair Graph” (Figure 8) in Maya Attribute Editor to open the Flair Graph user interface (Figure 9). The users can write their own custom style using GLSL within Flair and it will be displayed in real time in Maya.

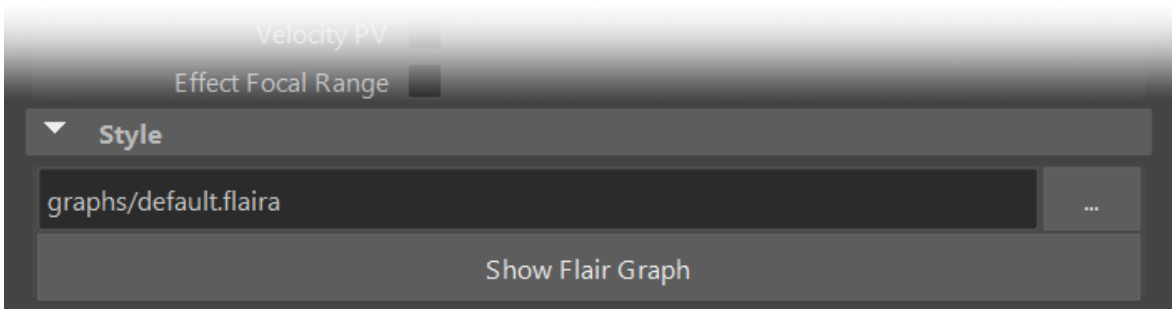

Figure 8. Maya Attribute Editor’s “Show Flair Graph”

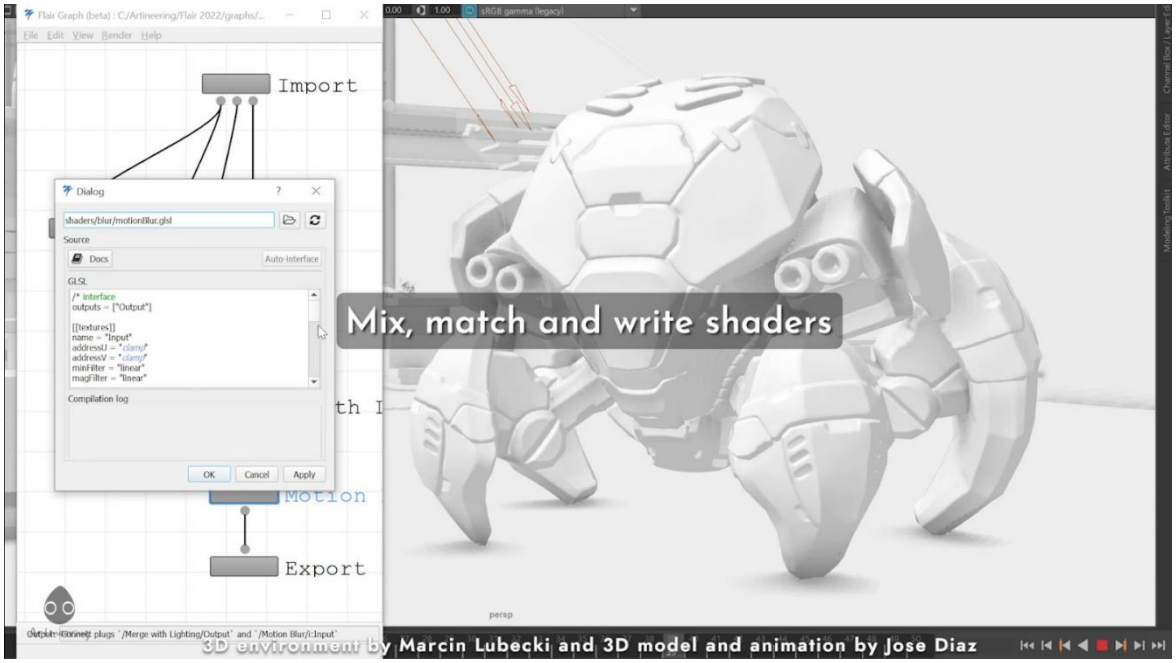

Figure 9. Flair Graph

2.1.2 Mid-Level Control: Material Presets

Material Presets serve as the mid-level of control, where artists can save or load predefined material parameters. Artists also have the flexibility to control these parameters individually outside of the preset. This level has a more detailed approach than global presets by allowing modifications to the shader code and parameters for each material, thereby creating and customizing a versatile material preset.

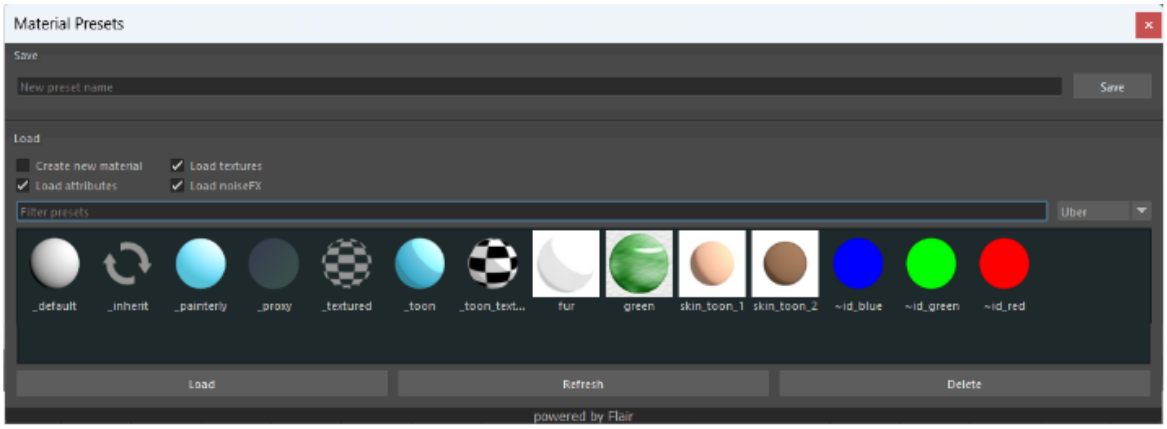

Figure 10. Flair Material Presets

2.1.3 Mapped Control

Mapped Control represents a more particular level of control, where specific effects can be applied locally to objects or parts of objects. Artists can apply detailed adjustments and stylization effects directly within the 3D space, providing the finest detailed control.

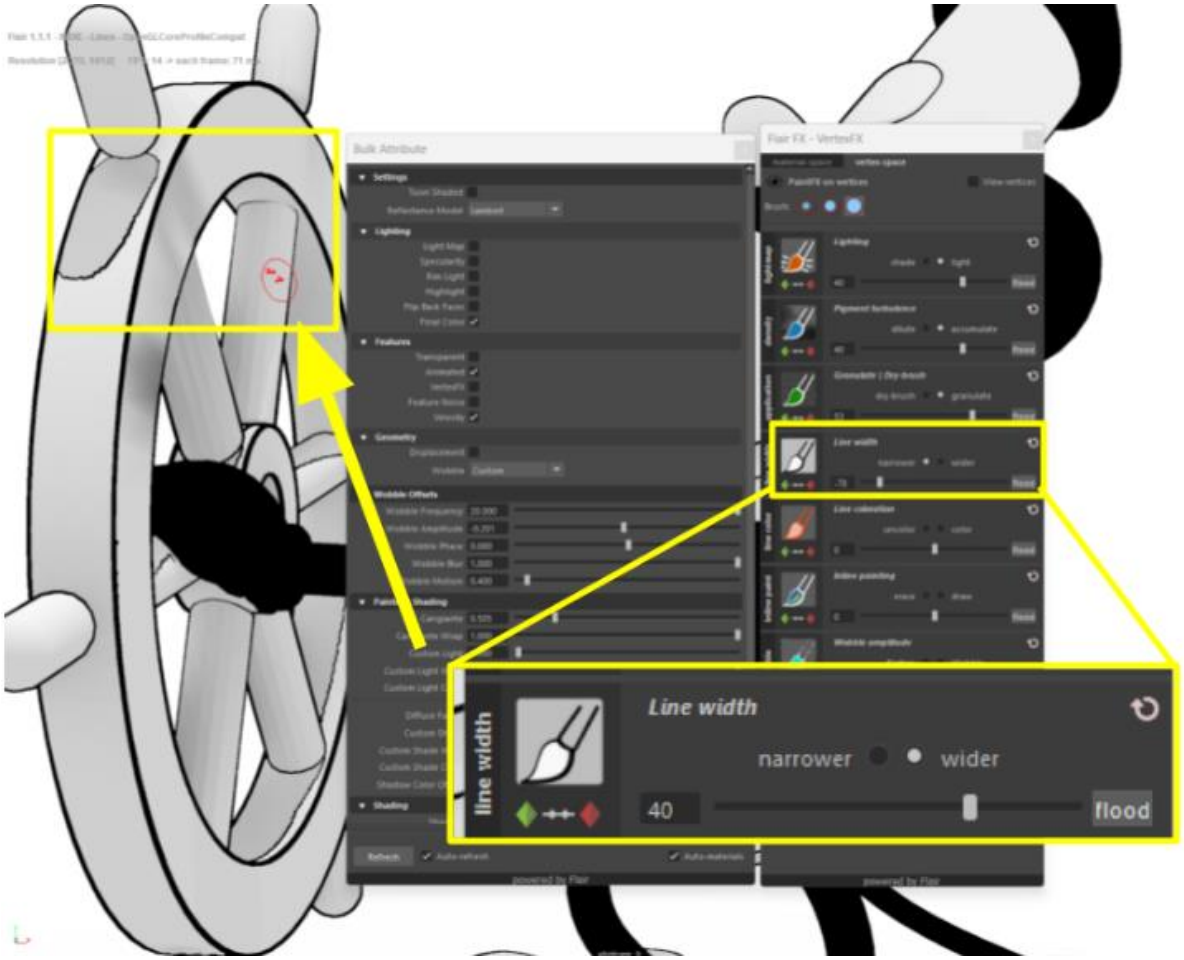

Figure 11. Mapped Control for local outline adjustment

2.1.4 Proxy Control

Proxy Control, usually cooperating with Mapped Control, allows for manipulating invisible 3D elements that only influence the render with localized stylization effects alongside the other levels. This approach offers broad control over stylization parameters, enabling artists to affect the entire scene or specific areas within the 3D space.

2.2 Shelf: UI's Plugin Analysis

The Flair shelf in Maya is designed to facilitate artists to engage with NPR tools without prior technical experience. Levels of control were made using the user-centric approach[7], which is sequenced from global to local controls. This hierarchy mirrors the workflow of artists, allowing for both broad and precise adjustments to stylization effects.

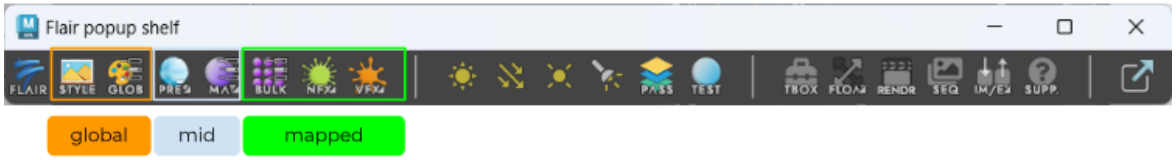

Figure 12. Flair Shelf

2.2.1 Global

At the highest level of control, Artists can define styles across the entire scene, setting a foundational tone or mood with just a few clicks. This level of control is essential for setting up the art-direction and keeping consistency across all elements within the scene.

Tools in Global: Style Presets and Globals

2.2.2 Mid

For mid-level control, artists can tune the appearance of specific objects' materials. This level gives more precise manipulation, allowing artists to distinguish and highlight elements according to their design preferences without overwhelming them with too much details.

Tools in Mid: Bulk Attribute, Material Presets and Material Attributes

2.2.3 Mapped

At the lowest level, mapped offer local control. Artists can apply stylization effects to specific areas or objects within the scene. This control level provides the most detailing and refinement to ensure that each element satisfies the artist's vision.

Tools in Mapped: Bulk, NFX, and VFX

[7] https://www.oreilly.com/library/view/user-centered-design/9781449359812/

2.3 User-Centric Control with Flair's Control Hierarchy

With the levels of control on the shelf, users can simply navigate through the toolset with tooltips provided [12], focusing more on their creative expression and less on navigating complex software features. This design not only supports the usability of Flair but also encourages exploration and creativity within the world of NPR.

The user-centric approach in Flair's design simplifies the learning curve for new users while giving advanced users the flexibility to craft sophisticated visual styles. Flair enables a wider range of artists to participate in the creative process without requiring deep technical knowledge of shader programming or rendering algorithms. This inclusiveness supplies a collaborative and smooth working environment where technical artists and traditional artists can easily share and refine ideas.

For smooth integration, a thorough understanding of Flair's architecture is needed. This understanding lays the foundation for the subsequent exploration into the varieties of tools, shaders, and pipeline integrations.

2.4 Flair's Plugin Architecture in Maya

Flair is implemented using the Model-View-Controller (MVC)[8] pattern with Python Qt for the GUI and Maya's API for system interactions. Each Flair tool—such as Style Presets, Material Presets, Bulk, and NoiseFX—operates under its own MVC framework, which manages user interactions and its services within Maya. This design pattern distributes responsibilities across each tool and increases the system's modularity and scalability. The diagram below demonstrates a user interaction for each tool in Flair to visualize how each tool utilizes its own MVC framework for operation with Flair in Maya:

[8] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

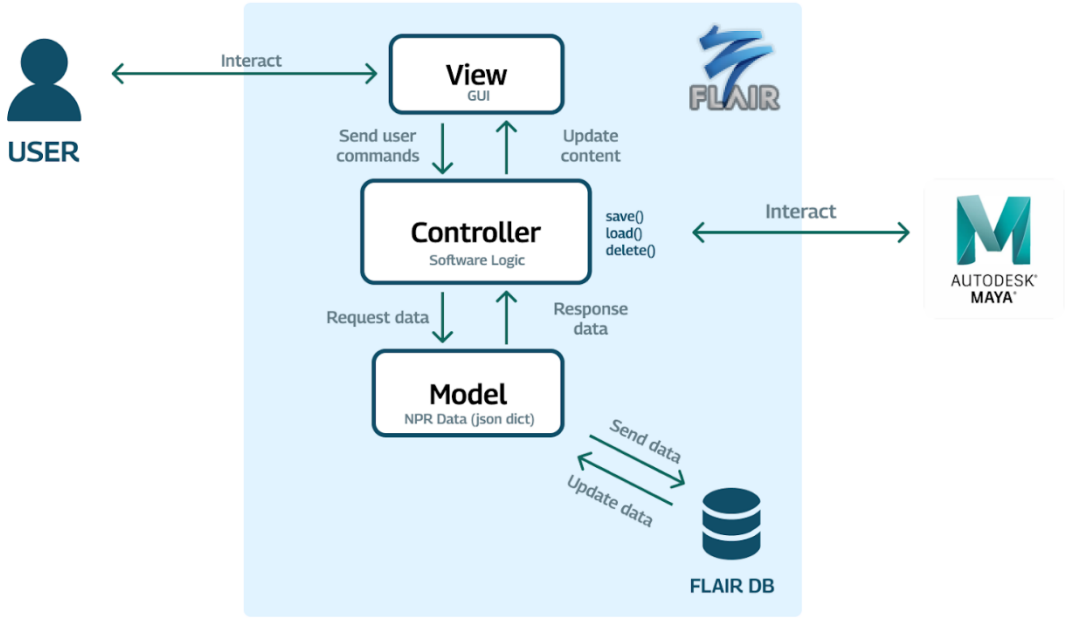

Figure 13. Flair MVC Architecture and User Interaction

2.4.1 Model (Data Layer)

The Model is the data management layer, interacting with the Flair local database to store and retrieve NPR styles and material presets. It encapsulates Flair's core data logic, facilitating the translation of artistic value adjustments into customizable parameters that Maya can process and apply to scenes or objects.

2.4.2 View (GUI Layer)

Flair's View layer, implemented in most tools using PySide—a Qt wrapper for Python—provides a graphical user interface (GUI) that displays Flair's dialogs, such as the Style Presets and Global Settings. This GUI allows users to visually navigate through presets and set predefined parameters easily. The View serves as the intermediary between the user and the controller, ensuring a smooth and responsive user experience.

2.4.3 Controller (Logic Layer)

The Controller handles Flair's business logic, acting as a middleman that translates user inputs from GUI into actions. It processes commands, communicating with the Model (data layer) to modify and apply data as needed. The Controller's responsibilities include invoking *save()*, *load()* and *delete()* operations on Style Presets and Material Presets' views, which facilitate the application of NPR in Maya.

2.4.4 Interaction with Autodesk Maya

The Controller interacts with Maya, while the Model reads and applies settings received from the database. The Controller also triggers actions that modify Maya's rendering settings, such as setting all parameters defined in the preset.

2.4.5 Database (JSON Files)

The Flair database system stores NPR styles and material presets in JSON dictionaries. Each JSON file is a standalone record. This setup enables quick access and update of presets. This format offers a human-readable and easily editable structure for storing complex configurations. Here's an example of how Flair's data is structured, displaying various attributes and settings that control the rendering:

```
{
    "host": "Maya",
    "version": 20231017,
    ... (additional configuration data) ...
    "_renderScale": 1,
    "_colorDepth": 1,
    ... (other attributes) ...
    "gradient": 2.5,
    "smoothness": 3.0
}
```

Figure 14. Flair Style Presets Dictionary

2.5 Shader and Pipeline Integrations Across Platforms.

Flair’s NPR shaders are written in GLSL (OpenGL Shading Language) for stylized effects. The user-adjustable variables, which allow for customization of these effects, can be stored as JSON dictionaries, so that users can fine-tune their visuals with their preferences.

To integrate these effects into Maya, Flair uses hard-coded C++ pipelines, or the Flair graph, which interprets a custom shader graph that interprets GLSL and Maya's native rendering environment. This graph reads the parameters from the host application and maps the parameters to the GLSL shader. The integration includes:

2.5.1 Shader Adaptation

Flair reads (AOVs) generated by its custom materials and does image processing on them using GLSL shaders. After multiple operations, the final stylized image is produced and shown back on viewport.

2.5.2 Parameter Mapping

The JSON preset files, which contain adjustable shader parameters, are parsed by the preset tools and set onto the respective node's attributes within Maya. These attributes, then, dynamically set the shader uniform values within Flair to change the respective effects.

2.5.3 Cross-Platform Consistency

Flair is designed for cross-platform compatibility across 3D applications and rendering engines if there is support for integrating them. This flexibility enables Flair’s NPR effects to maintain functionality regardless of the platform.

After the breakdown of architecture and the UI design within Maya's environment, Flair's NPR shaders using GLSL, and data within JSON files, we now have information ready to move on to see the opportunities in UE5.

2.6 Conceptualizing the UI for Flair in UE5

Designing UI in UE5 involves a meticulous process. The goal is to maintain a smooth and engaging experience that supports the unique workflows of NPR artists. Firstly, an analysis of the existing UI elements and tools within UE5 is conducted. I examined the layout, docking options, and user interaction within the UE5 workspace.

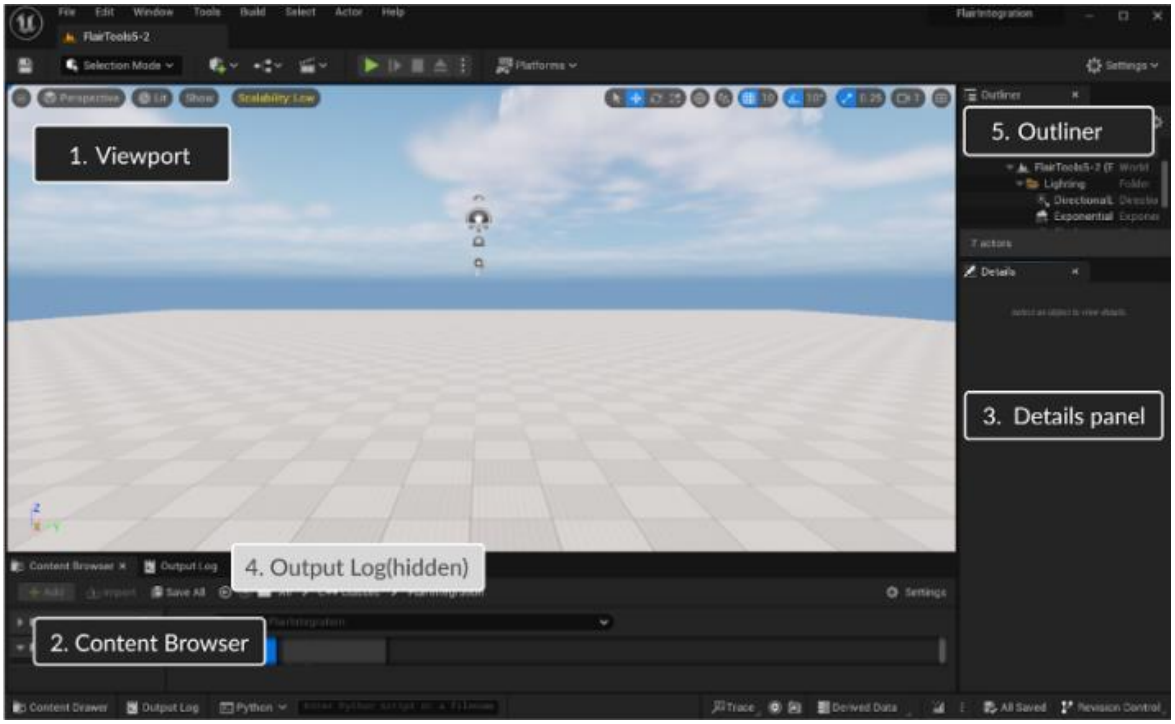

Figure 15. UE5 Workspace

2.6.1 Layout

UE5 has highly customizable docking options. Almost every element can be moved, grouped, docked, or floated based on the user's preference. Extra viewports[9] can be added and configured for different views and tasks.

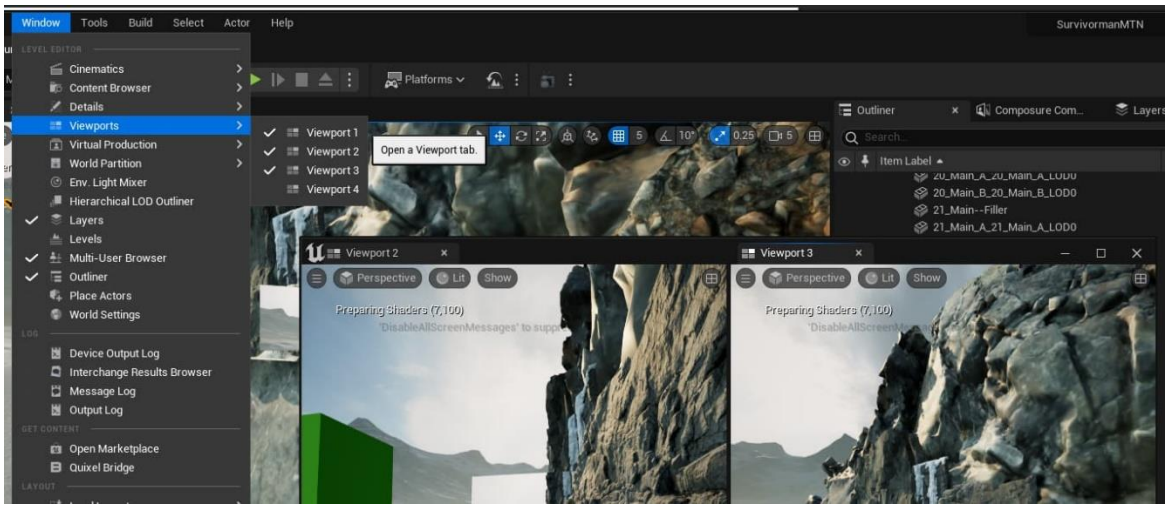

Figure 16. UE5 Extra Viewports

[9] https://forums.unrealengine.com/t/can-we-display-more-than-one-camera-viewport-within-a-viewport/648497

2.6.2 Docking Options

Although Maya, built on the Qt framework, supports flexible docking of panels and windows, Flair's current implementation is static. UE5 also highly supports a dynamic docking system for customization directly by the user. For instance, the Blueprint Editor can be docked on a secondary monitor or as a tab adjacent to the Material Editor, offering flexible visual adjustments and scripting without losing workflow context.

2.6.3 User Interaction

UE5 is designed for users to drag and drop objects from the content browser directly into the viewport for general usage. In addition, the engine supports switching between modes within the same viewport, facilitated by toolbars and menus.

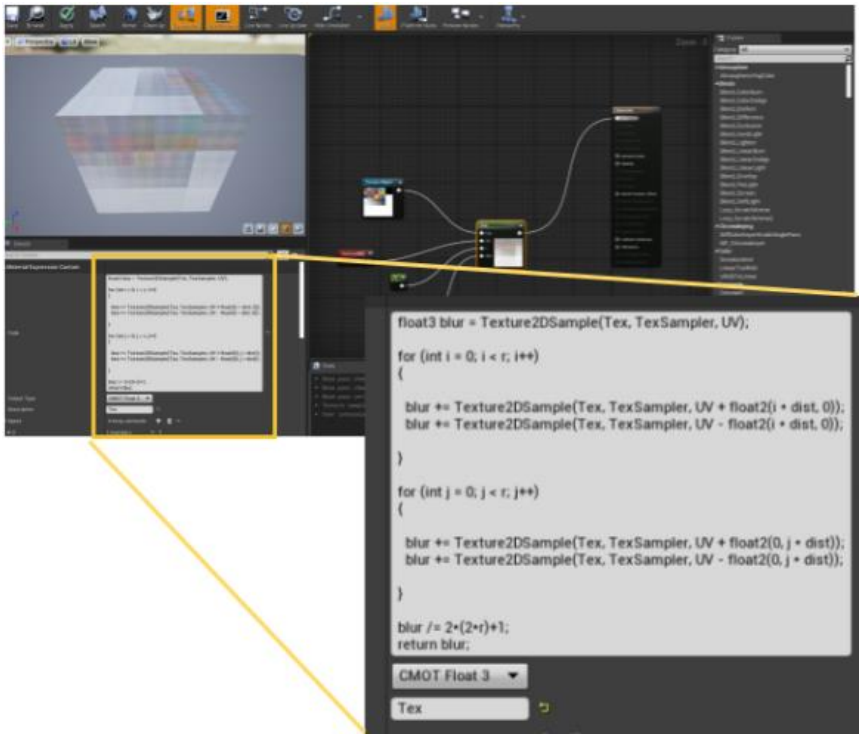

Figure 17. Shader Code in UE5

A designer, in terms of environmental or level creation, usually spends most of the time on their Material Blueprints, Outliners, and Detail panels. The user typically inspects the viewport for the rendering results. Users might opt for an external IDE to write HLSL code for custom shader development, as the native UE5 does not support an HLSL editor[10].

[10] https://discourse.techart.online/t/i-cannot-use-texturesample-with-custom-hlsl-gaussian-blur-function/13935

Having examined the overview of the UE5 workspace, now it's time to compare the previously mentioned points with Autodesk Maya. A side-by-side comparison will visualize the difference and guide the adaptation of the Flair GUI for UE5 users.

Feature	UE5	Autodesk Maya
Layout	• Highly customizable interface • Central main viewport with detachable views for each task • Supports extensive real-time interaction	• Customizable, but more rigid layout • Central main viewport with fixed surrounding panels • Less focus on real-time feedback
Docking Options	• Flexible: multiple panels can be docked inside the main window or as separate windows	• Limited: certain panels docked, some floating windows
User Interaction	• Mode switching by Tab within the viewport using toolbars and menus • Drag-and-drop assets from the content browser to the viewport • Flexibility and adaptability for game development workflows	• Marking menus and hotbox commands for quick tool access • Apply assets using keyboard modifiers such as Shift or Ctrl combined with mouse actions • Focused on efficiency in linear content workflows

Up to this point, we have laid initial research for the upcoming Flair in UE5, which includes an initial implementation of some NPR shaders and preset levels of control found in Flair for Maya, as well as the GUI and architecture. After the evaluation of the workspaces in both Maya and UE5, the next step is user research and persona creation to understand what UE5 users expect from Flair.

3 User Research and Design

This chapter covers the comprehensive design and research process for integrating Flair into UE5. We start with an identification of the target user groups, then explore how Flair was adapted from Maya to meet the requirements of UE5 users, including technical artists, 3D artists and developers. Our user experience research and key findings lead us through the development of user journey maps [11], which help pinpoint specific challenges. The persona and user story are then created to visualize features to be kept or improved in the plugin within UE5 later on with a persona-driven design strategy.

In digital product design, it's crucial to start with thorough research and clear goals to understand the product's main purposes, its users, their needs, and how it will be used. In our case, the target groups are UE5 technical artists and 3D artists who are a tech-savvy user and work daily with digital 3D tools. Our goal is to integrate and design a plugin that's not only functional but also visually appealing and easy to use, helping users to achieve their goals efficiently.

Flair was initially designed for 3D artists working with Autodesk Maya for animations and artistic renderings who are accustomed to Maya's interface. Integrating Flair directly into Maya as an additional shelf makes it straightforward and user-friendly for 3D artists.

However, the transition to UE5 will have a broader user base. Not only should Flair be made familiar to 3D artists and animators, but also to product designers and game developers who may not have used Maya before. This requires the design that resonates with this diverse audience, ensuring Flair in UE5's experience is as smooth as it is in Maya.

3.1 User Experience (UX) Research

After identifying the target group, it is time to enter the Unreal Engine's world by exploring tutorials and engaging with the dev communities. Understanding the workflows, preferences, and challenges people face with current NPR tools narrows the way down to meet user needs. The research approach involves a few methods to gather qualitative insight:

3.1.1 Observations

I observe how UE5 developers and artists use current NPR workflows and share their insights into existing tools' usability, pipelines, and limitations. Their techniques point out the current state of tool usability and its complexity. For example, I analyzed a tutorial on Unreal Engine's developer community [13] where artists construct NPR effects from scratch.

These processes often involve complex blueprint designs or advanced HLSL programming that includes mathematical operations like convolution.

Additionally, I reviewed an article on 80.lv [6] that showcased the creation of a fantasy NPR environment in UE5, revealing a blueprint-intensive process. It also illustrated the lighting setups and the integration of low-poly meshes to achieve the desired artistic effect. These resources are crucial for identifying the areas where our plugin can simplify the NPR process for artists.

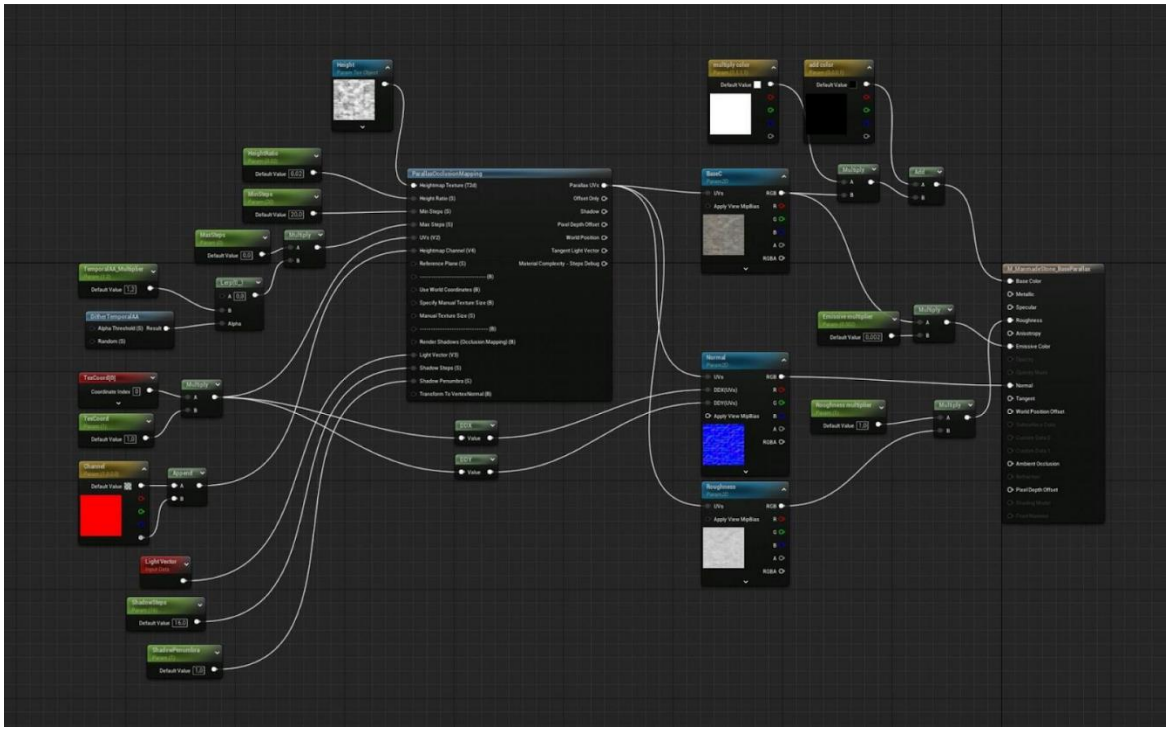

Figure 18. Material Blueprints

3.1.2 Interviews with the Supervisor

I conducted detailed discussions with my supervisor, Santiago Montesdeoca from Artineering, who developed the original Flair and its plugin for Autodesk Maya. Santiago provides valuable views from 3D artists, helping me to understand their specific needs and expectations for an NPR plugin. His assistance includes the reason behind the Flair shelf's design from his direct experience with the challenges artists faced when there were no presets or material adjustments available. Previously, artists had to manually write shader code or create complex shader node graphs for each individual material, a process that lacked the simplification provided by the plugin.

3.1.3 Community Engagement

I actively participate in group chats and share knowledge on Discord to interact with a broad range of UE5 users. This engagement provides diverse viewpoints that enrich our understanding and response to user needs.

3.1.4 Review of Existing Research

As mentioned in Chapter 2, I analyzed several research papers to understand the context of NPR and pipeline architecture better. These papers and online articles also help me become familiar with relevant NPR terminology.

3.1.5 Use of Flair in Maya

Personally, I started using Flair on a simple project in Maya, where I applied Flair presets and materials into the scene. After familiarizing myself with the workflow, I continued testing on a more complex scene, "Steamboat Willie". These experiences gave me a firsthand experience for designing the plugin for UE5.

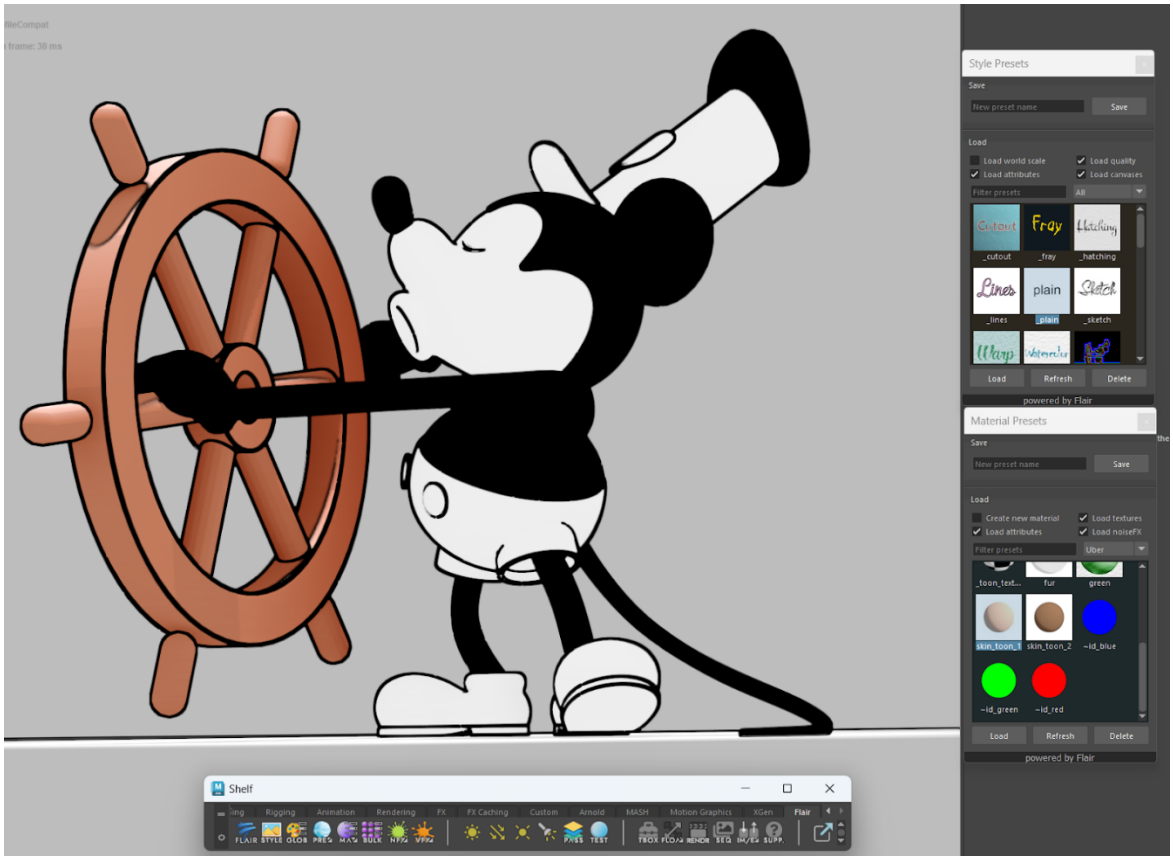

Figure 19. Steamboat Willie Provided by Artineering

3.2 Key Findings

This section outlines additional thoughts which were not covered in Chapter 2, by using personal intuition and hypotheses developed during the research process:

3.2.1 Ease of Use

Flair is designed to be user-friendly right from the start; its shelf organizes different levels of control. Users can quickly understand and handle the tools according to their needs.

3.2.2 Expertise and Enjoyment

Typically, experienced developers and individuals with a strong background in mathematics tend to enjoy NPR creation. They engage deeply in technical aspects of shader creation and algorithms.

Flair offers GLSL scripting [11]within its engine, allowing users to customize their own style presets for further use. This provides a handy tool for those who are proficient in coding.

3.2.3 Artist-Friendly

In contrast, artists with basic knowledge of shader writing may find their way to exploit advanced NPR features limited. The Flair plugin addresses this challenge by bridging the gap between complex coding requirements and artistic creativity.

For users who prefer not to write their own shaders, Flair offers a variety of presets to choose from. These presets can be easily adjusted by tweaking values or even painting parameters onto objects, allowing non-coders to create unique, stylized NPR shaders. The customized presets can be saved just as those created from GLSL.

Following the collection of these findings, we will proceed to create user journey maps. These maps will illustrate how users interact with the platform to achieve their objectives.

3.3 User Journey

A user journey maps the interaction between a user and the application, giving an overview of the experiences and actions. The journey is organized into two major components: actions and tasks in chronological order. Each task presents a logical sequence designed to guide the user through the application's feature. Emojis are placed at the end of each task as the user's potential emotional response.

[11] https://thebookofshaders.com/01/

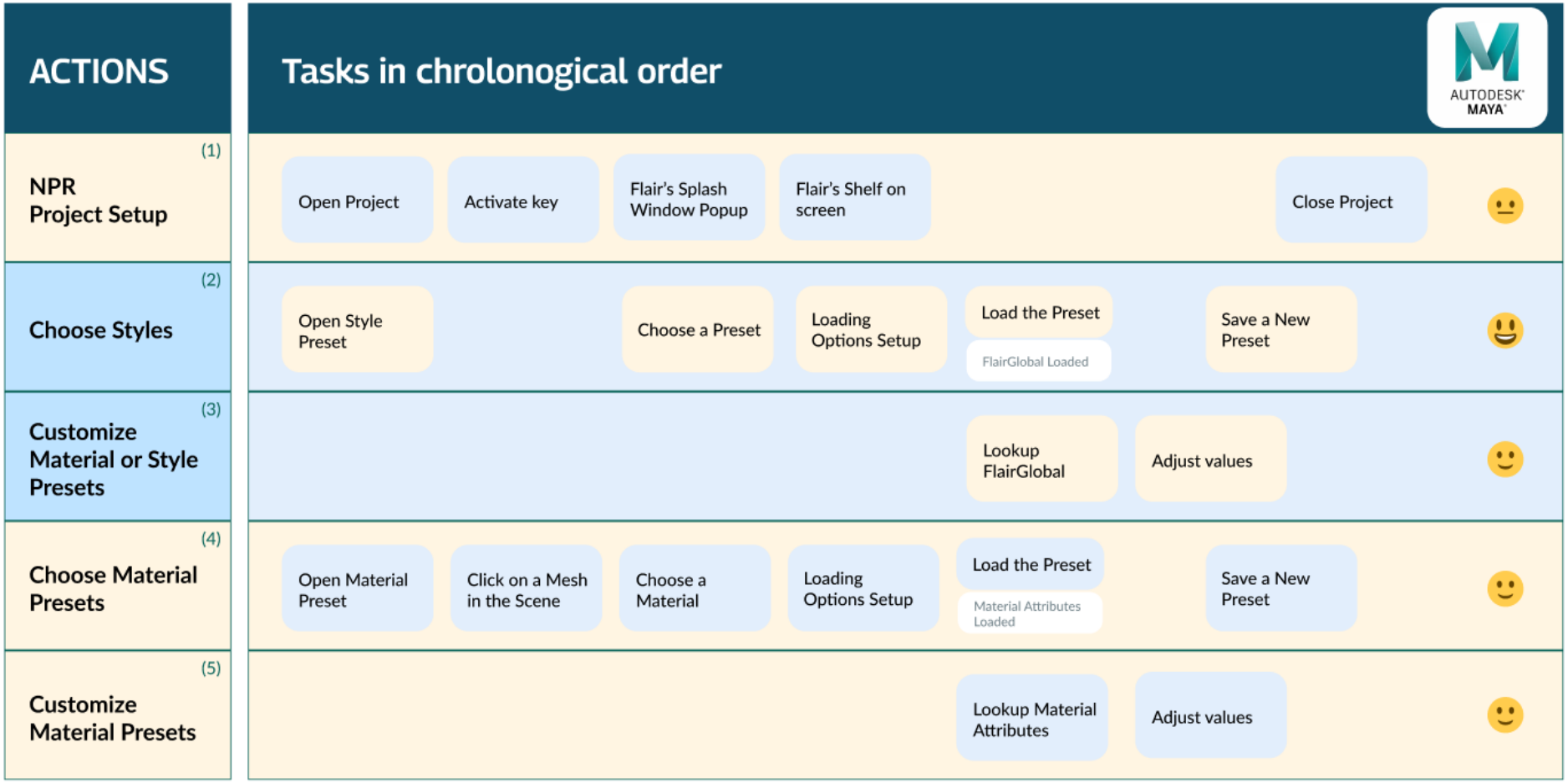

Figure 20. The User Journey of Flair in Maya

3.3.1 The User Journey of Flair in Maya

The user journey of Flair in Maya, shown in Figure 20, contains five actions in total:

1. NPR Project Setup

The journey begins when the user opens the project and activates the plugin. These tasks are straightforward and might be only slightly confusing, with no significant emotional spikes, thus it's marked without an excitement emoji.

2. Choose Styles

The user selects a style preset and may adjust the setup options with *flairGlobals* or Flair Globals. Many users are likely to be excited as they see the default rendering transform into NPR. Thus, there is a big smile emoji at the end of the task.

3. Customize Styles

When the user wants to save new style presets and adjust specific values within the preset. The positive feeling comes from manipulating parameters in *FlairGlobal* and observing real-time changes in the viewport.

4. Choose Material Presets

For applying styles to specific meshes, the user opens a material preset and clicks on a mesh in the viewport to apply the material. Since this action focuses on local objects, the excitement level might not surpass that of action 2, as the NPR is already visible globally.

5. Customize Material Presets

This action is similar to action (3) but involves adjusting parameters in Flair Shader material attributes instead of *FlairGlobal*. The user can customize material presets, including the loading and saving of new presets and detailed adjustments to the material attributes.

This user journey is already implemented in Flair for Maya. While this process may seem straightforward and perhaps common for any application, we would like to draw the readers' attention to the user journey of manually creating an NPR pipeline in UE5.

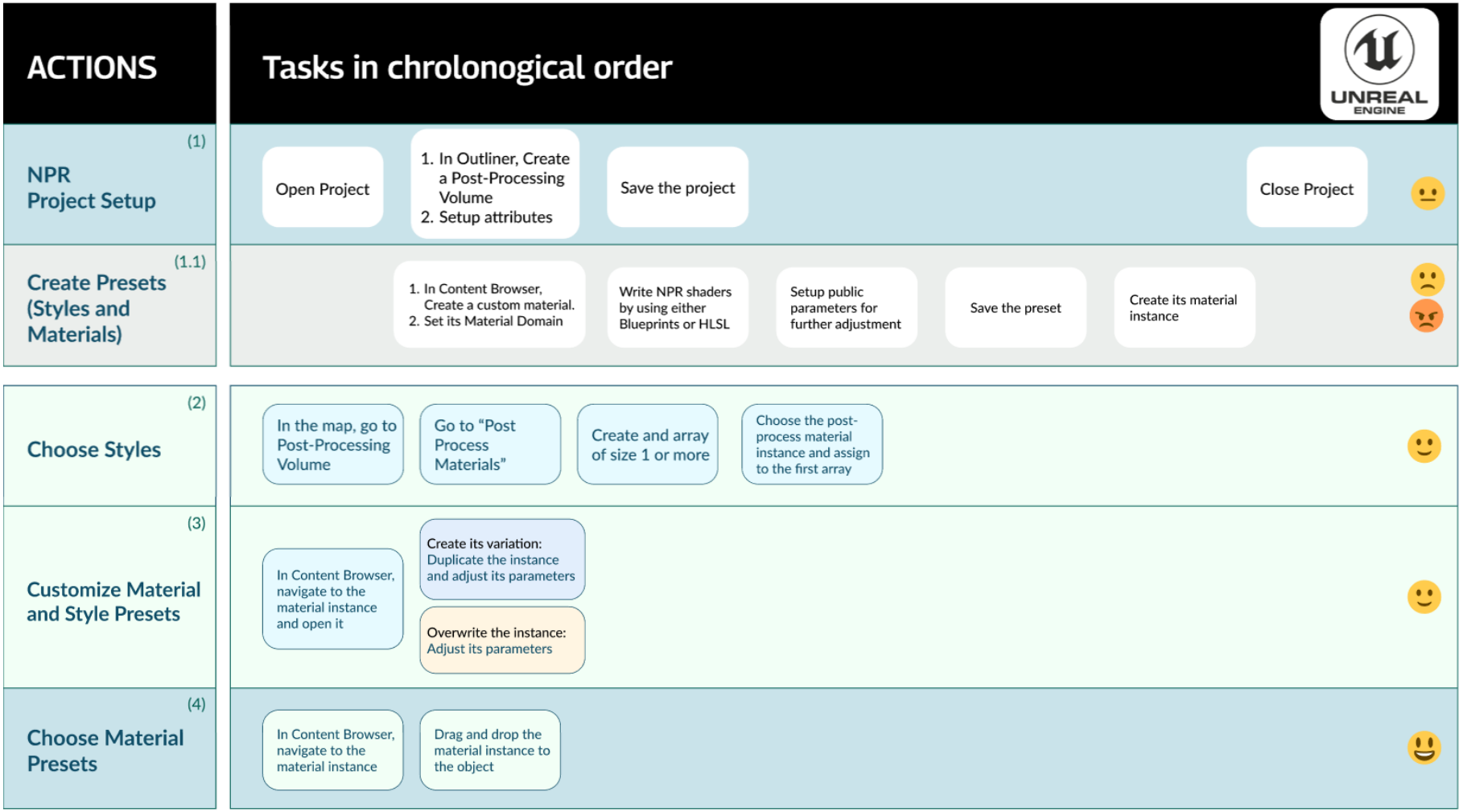

Figure 21. The User Journey of NPR creation in UE5

3.3.2 The User Journey of NPR creation in UE5

The user Journey of NPR creation in UE5, illustrated in Figure 21, contains four main actions. This structure shares similarities with the journey of Flair in Maya (Figure 20), particularly in actions (3) and (4), which focus on assigning materials and post-processing. These steps are as user-friendly and straightforward in UE5 as they are in Maya.

In contrast, the journey includes a distinctive and challenging action (1.1) that often presents difficulties for users. In this step, setting up the project for NPR can be daunting, frequently leading users to look up UE5's extensive, and unclear code documentation. This action is represented by neutral and anxious emojis, reflecting potential user frustration.

According to these summaries of both journeys, a persona and user story are created to point out characteristics, needs, and goals of a larger group of users. It also helps visualize the end-user and product decisions according to the user needs.

3.4 Persona and User Story

In the development process, personas and user stories are primary tools to guide the design and functionality of a product.

Personas are characters created based on research to represent different user types who might use a product in similar ways.

User Stories, on the other hand, describe features from the perspective of the user, focusing on their needs and the reasons behind these needs. This narrative form helps us prioritize features from user expectations and satisfaction.

To illustrate, we have designed a persona that includes an embedded user story, reflecting the combined insights and expectations from our target user group:

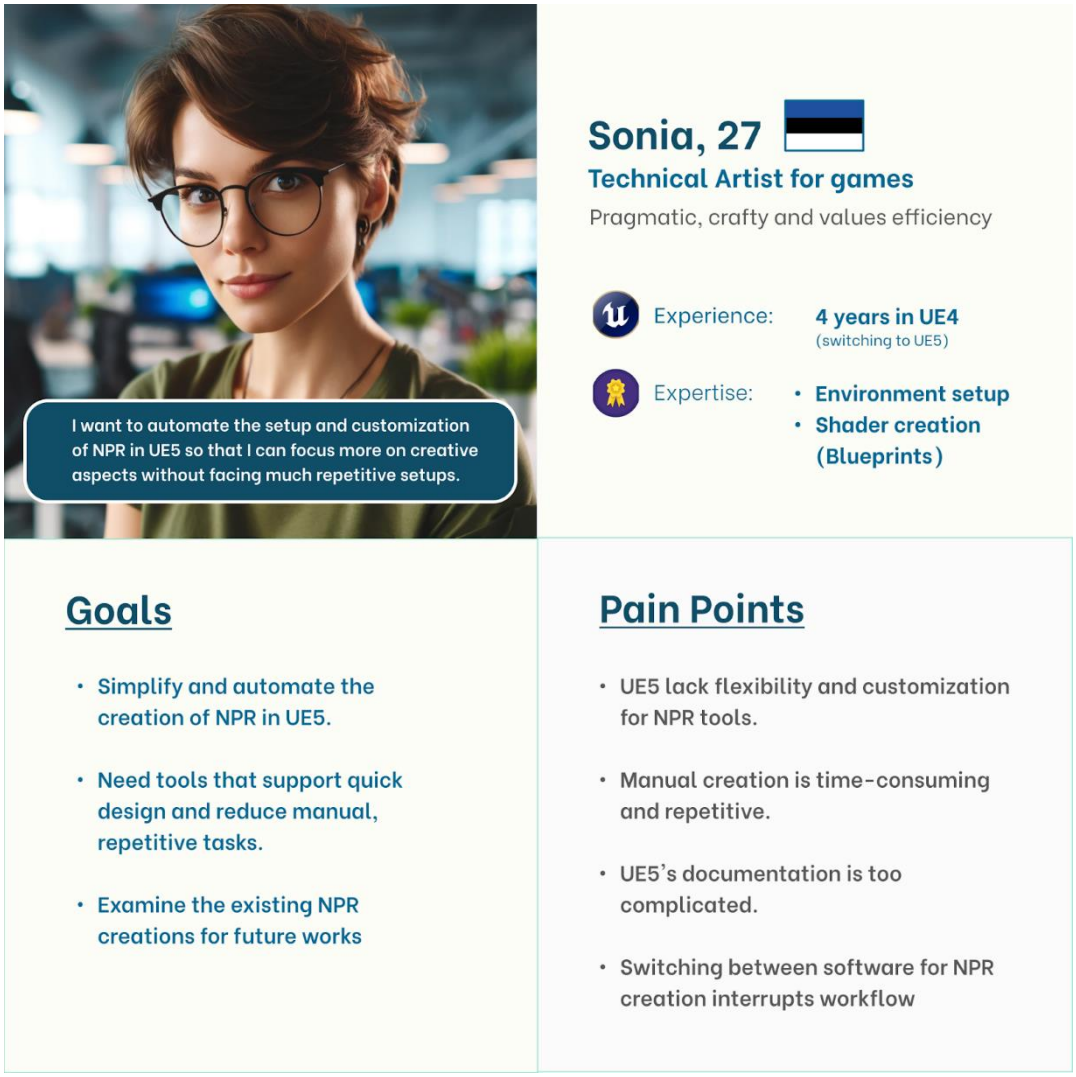

Figure 22. Persona

3.4.1 Persona — Sonia

Sonia, a 27-year-old technical artist for games and 3D animations, specializing in environment setup and shader creation using UE4 and transitioning to UE5. Representing our user base, she is looking for tools that help with her creative workflow. Her current working process is facing these pain points:

(1) Lack of Flexibility: UE5 has limited customization for NPR.

(2) Time Consumption: Setting up and customizing NPR is repetitive and time-consuming.

(3) Complex Documentation: UE5 documentation is hard to navigate as an artist.

(4) Workflow Interruptions: Switching between software for NPR tasks is distracting.

3.4.2 User Story

"As a technical artist, I want to automate the setup and customization of NPR in UE5 so that I can focus more on creative aspects without facing much repetitive setups."

The following sections will discuss proposed improvements of these challenges for Sonia and others like her in their daily tasks.

3.5 Improving UE5's NPR Creation Process

These improvements are designed to reduce NPR pipeline overhead, giving users more flexibility for advanced customization. By addressing these areas, we will make the NPR setup more intuitive and accessible, thus broadening its appeal and usability across a range of users from general 3D artists to UE5 developers.

3.5.1 Simplify and Automate Post-Processing Volume Setup

Users could apply NPR effects without manual setup overhead once Flair automates the creation and configuration of post-processing volumes for NPR. If users do not find the default settings satisfying, they can adjust these settings to fit their needs.

3.5.2 Offer Preset Accessibility

To address pain points in Figure 21 - step (1.1), having a library of ready-to-use presets would significantly reduce the technical anxiety during early stages. It will be beneficial for non-technical users like 3D artists by providing existing presets, users would be able to select and apply any material or style preset without the need to create them from scratch. Thus, they can mainly focus on creative aspects over the technical setups. Furthermore, offering a selection of HLSL shaders and Blueprints as examples would help users understand and learn to customize their materials more.

3.5.3 Maintain User-Friendly Adjustment Features

Keep simplicity and functionality of steps (3) and (4), where users adjust values and customize their projects as they are already straightforward and have a satisfying experience.

These proposed improvements also address pain points and user goals. We can better support the persona developed to guide the design process. The user story, which encapsulates these personas, serves as a foundational element in our strategy, ensuring that the enhancements align with user needs and expectations.

Now we have a persona and user story presenting what the user needs, we move on to create a mock-up in the next section.

3.6 Design the First Mock-up(s)

The **mock-up** designs present four potential workspaces within UE5. This approach doesn't cost much time compared to the interactive prototype one. In addition, we can visualize the end product early on. At this stage, the project is small and flexible, making it fast to produce varied mock-ups to ensure everyone at Artineering is aligned.

Considering that Unreal Engine supports responsive and adjustable dockable panels, the UE5's Flair shelf is designed to be both responsive and customizable. Users can organize their workspace according to personal preference—whether it resembles the existing Flair shelf, is free-floating, vertically oriented like a sidebar, or organized as a rectangular panel.

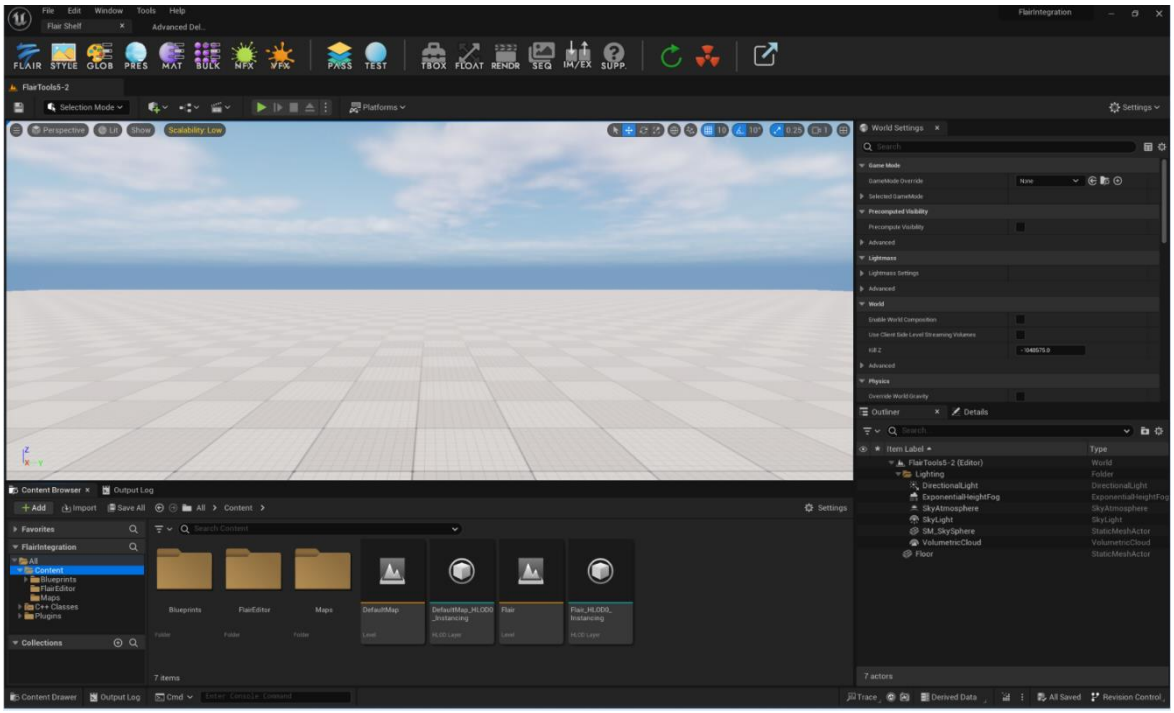

Figure 23. UE5 Flair - Horizontal Layout

This mock-up (Figure 23) displays the Flair shelf in a horizontal layout. The shelf is positioned at the top of the workspace for easy access to Flair's features. This setup mimics the shelf in Maya, suits the users who are accustomed to the original layout.

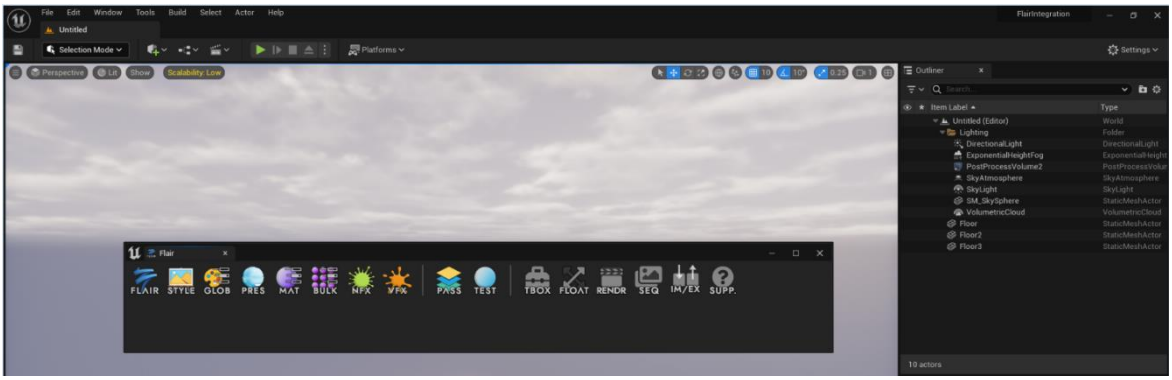

Figure 24. UE5 Flair - Floating Freeform

The shelf can be detached and floats freely over the workspace. This mockup (Figure 24) illustrates the freeform version of the Flair shelf, where it is undocked and floats over the workspace. This design is ideal for users who require a dynamic setup, allowing them to position the shelf wherever in the workspace without being bound to a fixed location.

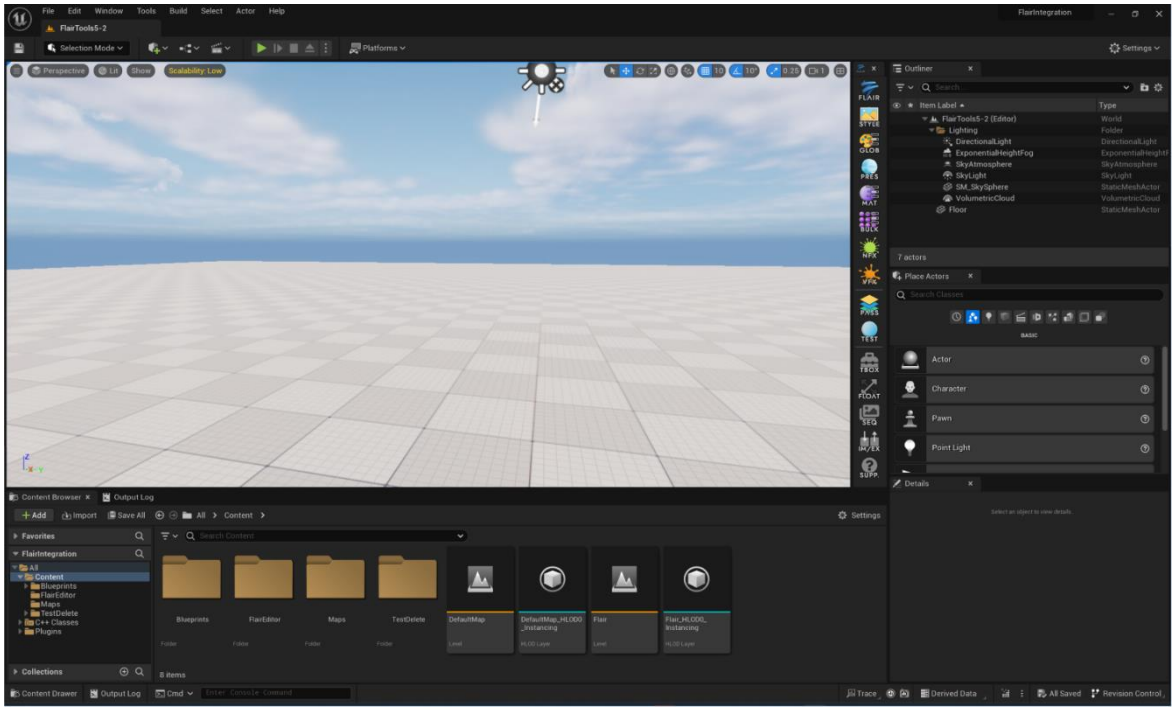

Figure 25. UE5 Flair - Vertical Sidebar

In Figure 25, the shelf is adapted to a vertical sidebar arrangement, similar to toolbars in applications like Photoshop or Illustrator. The icons are adjusted responsively to fit the vertical orientation. This layout benefits users who prefer to maximize the size of the viewport without overlapping the main working area.

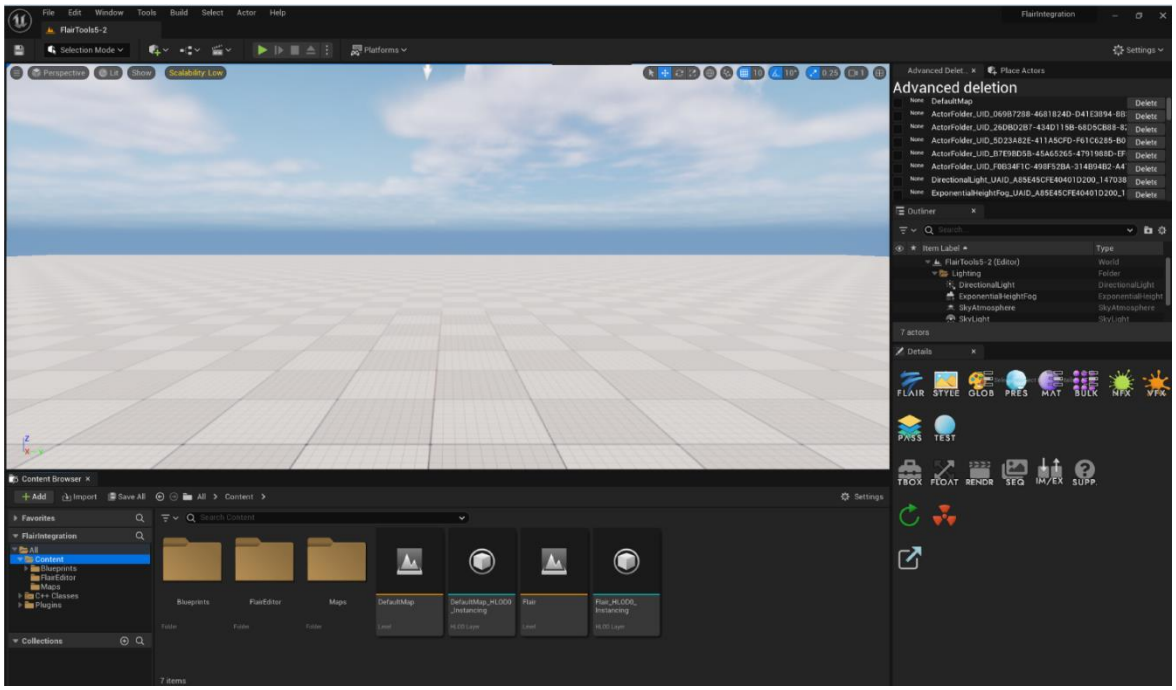

Figure 26. UE5 Flair - Box Form

Lastly, the box form in Figure 26 depicts a compact panel, docked alongside the Details panel on the right side of the workspace. The box form is organized by arranging icons in a grid-like layout. This suits users who view the shelf to be a core feature of UE5 along with the Details panel and Outliners. The design extends the responsive adaptability based on user preference and immersion of the plugin within the game engine, tailored to user preferences.

Finally, after achieving clear visual of these designs, we are nearly ready to begin the actual application development. We plan to utilize the core C++ library from UE5's Slate Widget to make the shelf functional according to the design. More detailed information on our integration with UE5 will be discussed in the dedicated chapter, which will be in the integration phase at Chapter 5.

4 Image Processing Integrations and Algorithms

Image processing plays a key role in Non-Photorealistic Rendering (NPR) to achieve aesthetic stylized effects. The process involves the writing and fine-tuning of shaders during the post-processing phase to achieve the desired look. This chapter will discuss the shader pipeline of the original Flair and the research on the post-processing and material pipeline for the upcoming Flair plugin for Unreal Engine 5.

4.1 What is Post-Processing in NPR?

Post-processing refers to techniques applied after the initial rendering process to further refine the visual output. This phase is where the NPR truly comes to life, utilizing algorithms, such as blurs, edge detections, and image enhancements, to manipulate the rendered image toward the desired aesthetic. It is simply adding a layer on the top as a filter.

Most 3D graphic programs usually offer post-processing effects for users to play with the values from the given presets. When the users want to further customize the rendering, they have to do so through the shader graph and write their own shader code, such as GLSL (for OpenGL) or HLSL (for DirectX). Flair also offers a unique engine for users to customize the image processing according to the user's preferences.

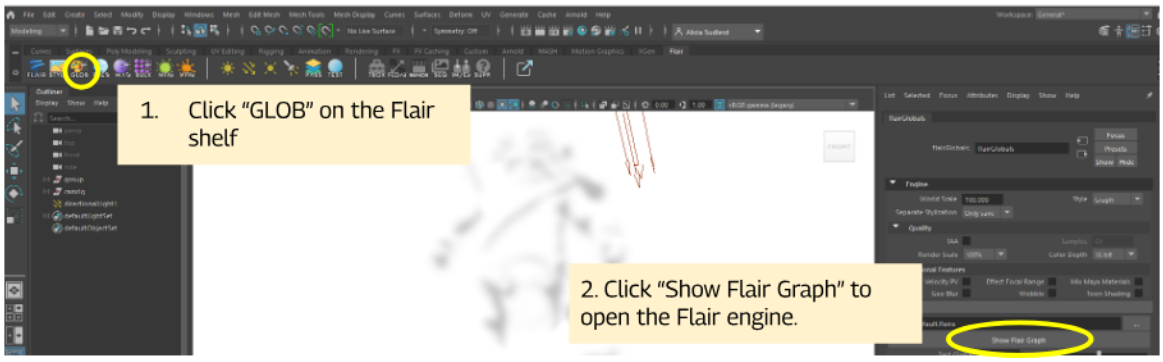

Figure 27. Maya Flair - Global Settings

4.2 GLSL Shader for Flair in Maya

As previously discussed in Section 2.5, most Flair's shaders are written in GLSL. These shaders are designed to allow parameter adjustments within a third-party application, in this case, Maya. The construction of a Flair Graph shader[12] is divided into two main parts: the Interface Definition and the Shader Code.

[12] https://docs.artineering.io/flair/graph/shader-node/shader-anatomy/

The *Interface Definition* is defined using TOML syntax[13] within a block *(/* interface */)*. This definition specifies how the shader be manipulated within the *Parameters* panel within the Flair Graph, as seen in Figure 28. Flair GLSL and the Parameters Panel. This setup allows for the definition of the shader's inputs, outputs, and uniforms that users can interact with in the parameter panel. For instance, as shown in Figure 28. Flair GLSL and the Parameters Panel, a parameter named *Radius* is defined in the interface section. To make Radius visible in the parameters panel, we have to declare the variable inside the *Shader Code*.

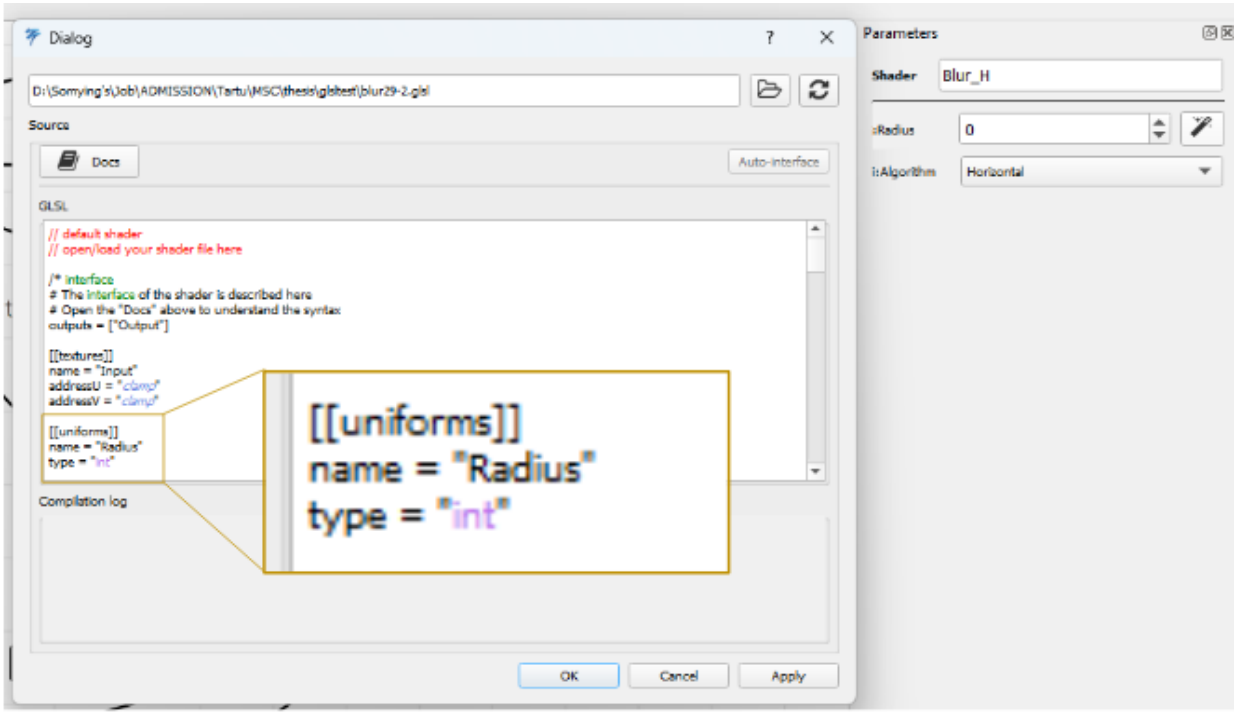

Figure 28. Flair GLSL and the Parameters Panel

The Shader Code is where the logic of the shader is written, defining how it processes inputs to produce outputs. After the pre-defined parameter in the interface part, for visibility in the panel, the declaration inside the logic part must match the definition in the interface. In this example, it must be *uniform int Radius.*

Once *Radius* appears in the parameters panel of the shader graph in Flair. Users can adjust this Radius by assigning the value in the input box or using a slider. This value is then used in the rendering in Maya.

[13] https://toml.io/en/

To show the parameter directly in Maya's interface (Figure 30), users can define a global variable within the Flair Graph so that it automatically appears in Maya's Attribute Editor. The newly defined global within the Flair Graph can then be used as an Expression (Figure 29) for the desired shader parameter. That way, the Global within the Maya interface can directly control the uniform passed to the Flair shader.

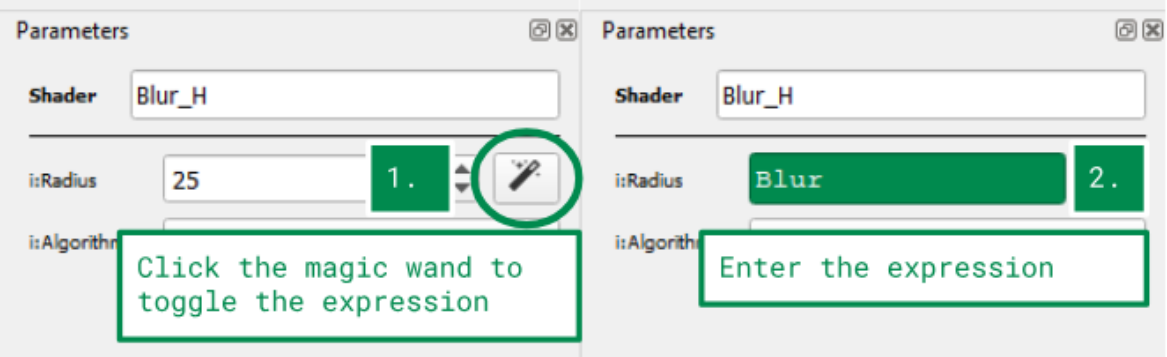

Figure 29. Flair Parameters – Expressions

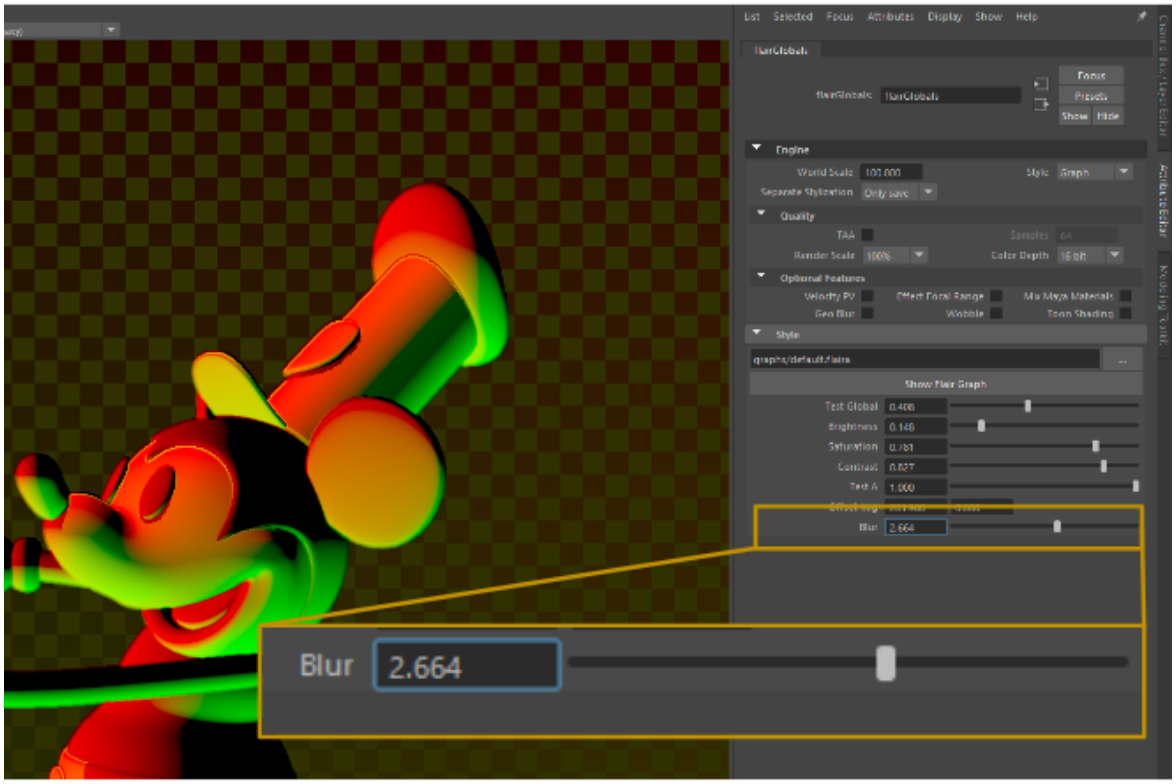

Figure 30. Maya Flair - Global Settings

4.3 HLSL Shader for Flair in UE5

In Unreal Engine, users can create materials by utilizing the components available in the material editor, often referred to as visual scripting. Additionally, these materials can include the custom shaders written in HLSL (High-Level Shading Language). Using shaders enables users to write iterations and complex algorithms within a single component. These HLSL shaders must be written within a material component known as Custom Material Expressions.

4.3.1 Custom Material Expressions

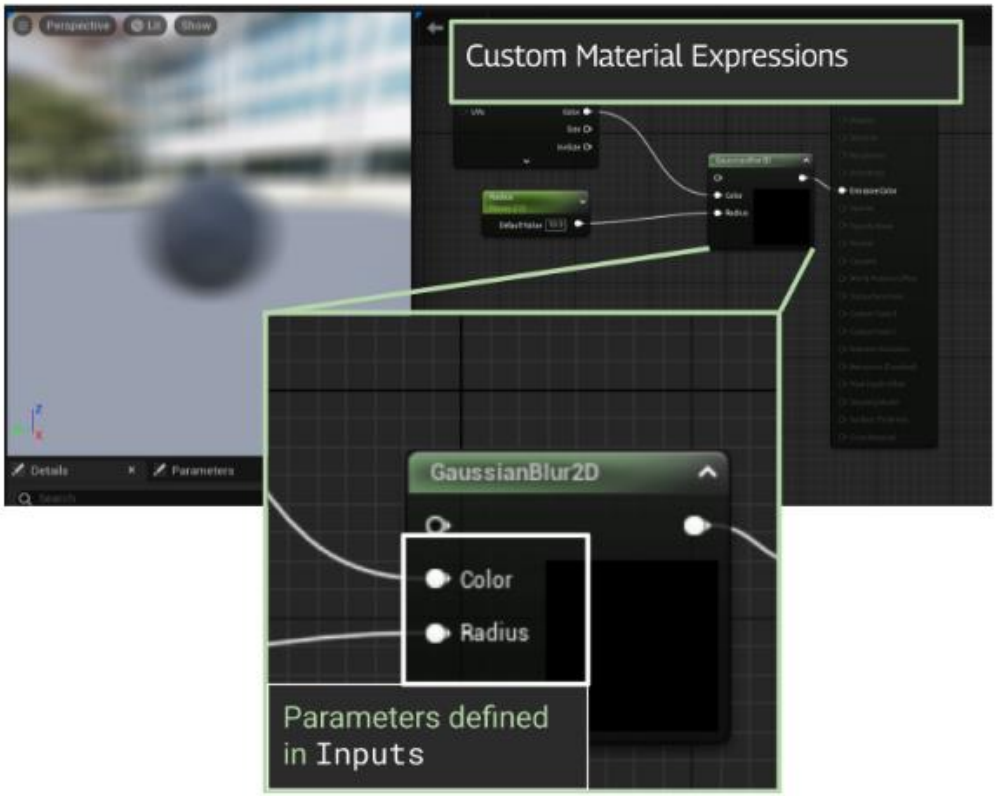

Figure 31. Material Editor - Custom Material Expressions

Custom Material Expressions, shown in Figure 31, is an expression or a component that allow the use of custom shaders within the Material Editor[14]. This component enables the creation of HLSL in the material editor, by writing and applying specific HLSL (High-Level Shading Language) code snippets to materials. This component is useful for creating writing shaders that are highly customized, which cannot be achieved with components.

[14] https://dev.epicgames.com/documentation/en-us/unreal-engine/custom-material-expressions-in-unreal-engine?application_version=5.0

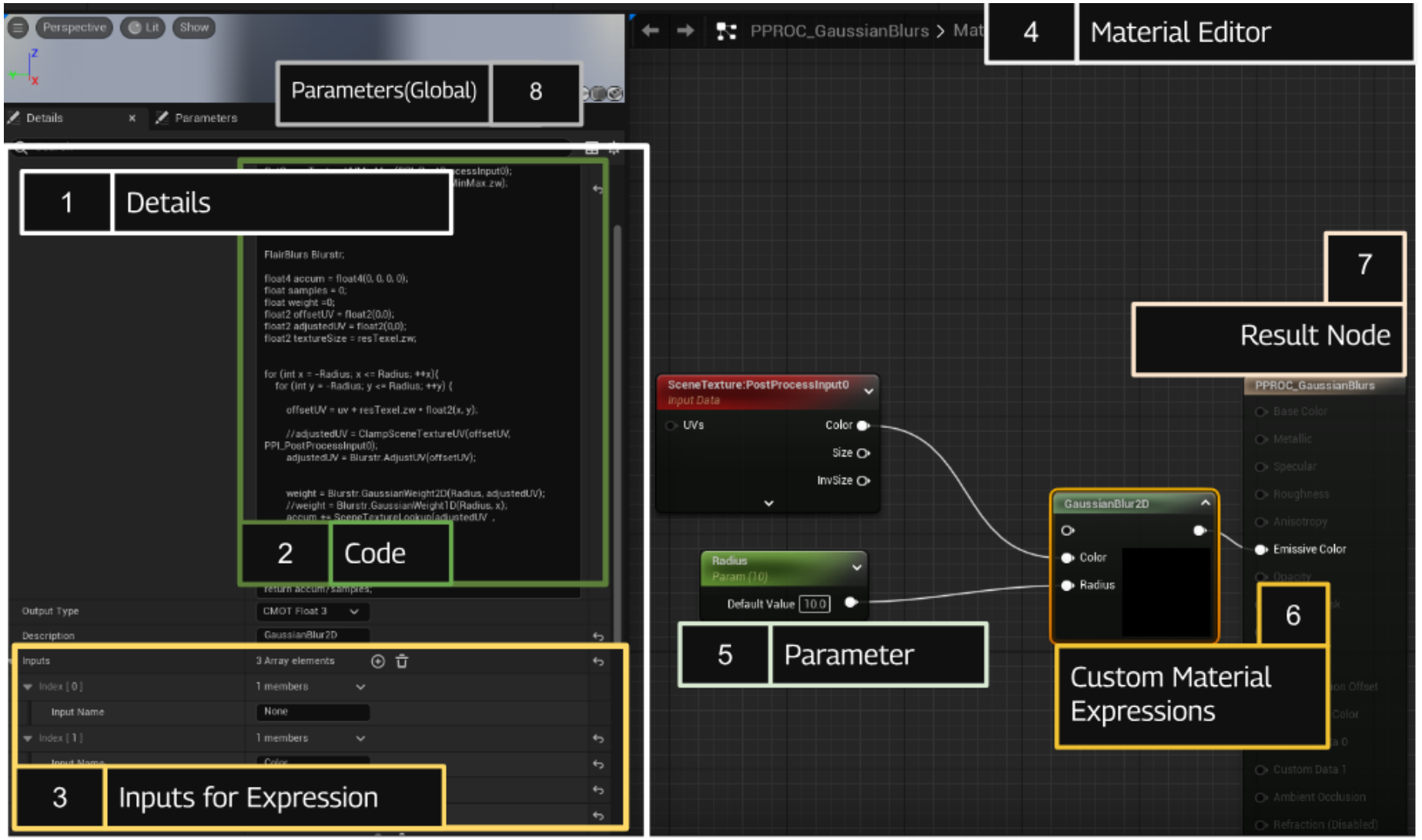

Figure 32. Custom Material Expressions - Details

4.3.2 Utilizing Integration

As shown in Figure 32, users can write stylized effects using HLSL. Within the *Details* panel (1) of the Material Editor (4), users can insert shader codes (2) into the property named Code, functioning similarly to the *Shader Code* of Flair in Maya (see Figure 28).

Once the HLSL code is written, the next step is to create the input variables in the Input property (3). After defining these inputs, users then customize other corresponding components in the Material Editor as needed. Lastly, connect the custom material (6) to the *Result Node* (7) to see the rendering result.

Additionally, it is necessary to create components corresponding to the collection of inputs (3) and convert them to parameters (5) and link them to the Custom Material Expressions component (6). This conversion enables these parameters to be dynamically adjusted outside the editor (4), whether in the Material Instance or via any third-party APIs.

Furthermore, both Material Editor and Material Instance provide global settings for all parameters which are declared in the editor. This allows users to fine-tune the value without going to each parameter and adjust the value one-by-one.

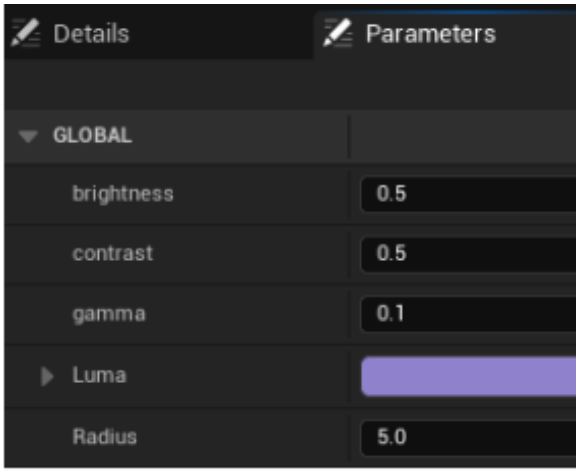

Figure 33. Material Editor - Global

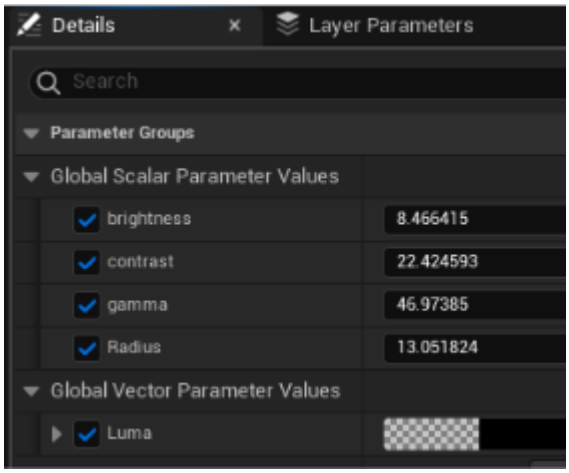

Figure 34. Material Instance - Global

4.4 Image Processing Algorithms: Gaussian Blurs

In this section, we introduce image processing algorithms that will be implemented and tested within UE5. We want to demonstrate and discuss how Gaussian Blurs are adapted using UE5 HLSL and Flair GLSL. The research serves as primal NPR algorithms implementation of the Flair plugin in UE5.

Gaussian blur is considered to be a core technique in NPR as it helps smooth out noises and details in images. It facilitates users to achieve NPR effects. This approach is ideal for emphasizing the visual impact of the art, emphasizing mood and expression.

Additionally, the algorithm can be used to implement effects similar to watercolor paintings. It transforms realistic images into more artistic representations by focusing on general shapes and colors, rather than fine details.

In the following section, we present code snippets for implementing Gaussian blur in both GLSL (used primarily in Maya) and HLSL (used in UE5) writing in the same style to demonstrate how the algorithm is applied within each shader language.

4.4.1 Flair GLSL Snippet for Gaussian Blurs

In Flair GLSL, shaders are structured with an interface definition, specifies the shader's inputs, outputs, and uniforms, making these parameters adjustable within both Flair Graph and Maya's *flairGlobals*. For example, the Radius parameter is declared in the interface section as uniform to be interact as an expression.

```
/* interface
[[uniforms]]
name = "Radius"
type = "int"
--other variables are declared here
*/
// GLSL shader
uniform sampler2D Input;
uniform int Radius;
out vec4 Output;
/*
Gaussian Weight and helper functions.
*/

// Main shader function applying Gaussian blur
void main() {
    // Accumulator for the color values
    vec4 texAccum = vec4(0.0);
    // Total weight for normalization
    float totalWeight = 0.0;
    for (int x = -Radius; x <= Radius; x++) {
        for (int y = -Radius; y <= Radius; y++) {
            // Offset UV by radius
            vec2 uv = gl_FragCoord.xy + vec2(x, y);
            // Calculate Gaussian weight
            float weight = gaussianWeight2D(Radius, vec2(x, y));
            // Weighted sum of texture colors
            texAccum += texture(Input, uv) * weight;
```

```
            // Sum of weights for normalization
            totalWeight += weight;
        }
    }
    // Normalized blurred output
    Output = texAccum / totalWeight;
}
```

The main shader function uses *gl_FragCoord.xy* to access the current pixel coordinates and applies the Gaussian blur by iterating over a kernel defined by the Radius. The UV coordinates are offset by the radius, and Gaussian weights are calculated for each offset.

4.4.2 UE5 HLSL Snippet for Gaussian Blurs

```
float2 uv = GetDefaultSceneTextureUV(Parameters, PPI_PostProcessInput0);
float4 resTexel = GetSceneTextureViewSize(PPI_PostProcessInput0);
FlairBlurs Blurstr;
#define PI 3.14159265358979323846

float4 accum = float4(0, 0, 0, 0);
float samples = 0;
float weight =0;
float2 offsetUV = float2(0,0);
float2 adjustedUV = float2(0,0);
float2 textureSize = resTexel.zw;

struct FlairBlurs
{
/* a struct that contains Gaussian Weight and helper functions.*/
};

for (int x = -Radius; x <= Radius; ++x){
    for (int y = -Radius; y <= Radius; ++y) {

        offsetUV = uv + resTexel.zw * float2(x, y);

        //adjustedUV = ClampSceneTextureUV(offsetUV,
PPI_PostProcessInput0);
        adjustedUV = Blurstr.AdjustUV(offsetUV, resTexel.xy);
        weight = Blurstr.GaussianWeight2D(Radius, adjustedUV);
        //weight = Blurstr.GaussianWeight1D(Radius, x);
        accum += SceneTextureLookup(adjustedUV   , PPI_PostProcessInput0,
true) * weight;
        samples += weight;
    }
}
return accum/samples;
```

In UE5 HLSL, shaders can be written through *custom expressions*. Users write HLSL code within the Material Editor and define input variables directly in the Details panel. The HLSL snippet does not require an interface definition like GLSL; instead, it uses UE5's built-in functions to get *default scene texture UVs* and *texture sizes*. The Gaussian weights and UV adjustments are encapsulated in a struct (`FlairBlurs`), and the main logic iterates over a kernel defined by the Radius, similar to GLSL. The computed weights are used to accumulate the texture samples, and the final blurred output is normalized by the total weight.

4.5 Shader and Image Processing Comparison

In this section, we compare the implementation and integration of Gaussian blur in GLSL for Maya and HLSL for Unreal Engine 5 (UE5). By examining the differences in how these shaders are constructed in each environment, it can benefit the implementation of the Flair plugin, collecting insights from these comparisons.

4.5.1 Parameter Handling

Both languages use *uniforms* to handle parameters that can be adjusted outside the editor. In GLSL, parameters are declared in the *Interface Definition* to be used within Flair Graph and Maya. In HLSL, parameters are defined in the *Inputs* property of the custom material expressions. These parameters can then be connected by other UE5 components, making them accessible for dynamic adjustments.

4.5.2 Shader Logic

- GLSL accesses pixel coordinates using *gl_FragCoord.xy*.
- HLSL gets UV coordinates and texture sizes through built-in UE5 functions such as *GetDefaultSceneTextureUV* and *GetSceneTextureViewSize*.

We have explored how image processing works, particularly through Gaussian blurs, which serve as a foundation for many stylized effects. The algorithm has been implemented and examined in both GLSL for Maya and HLSL for UE5. In the next chapter, we will begin to implement the plugin, applying the research and methods discussed up to this point to initialize the integration of Flair in UE5.

5 Implementation

This chapter gathers all the insights discussed in all previous chapters to show how everything is put together in a test version of the plugin. We designed this first version to validate the design goals mentioned in Chapter 3, focusing on how easy it is for users to interact with the system and how the system performs. Therefore, the test implementation focuses only on some of Flair's core features, namely Style Presets and Material Presets.

Figure 35. Technology Stack of Flair UE5

5.1 Technology Stack

The selection of tools is primarily influenced by the original Flair, as to avoid code duplication, re-use production proven toolsets and simplify future maintenance of code across applications. The aim is to make shareable code between Flair for Maya and Flair for UE5. For the *Qt Dialogs*, the original code was mostly integrated and adapted to be compatible with UE5 [13]. Although this adaptation process seemed straightforward, it was necessary to redesign the controller, the API and the model to facilitate communication between the Qt view and the UE5 Python API.

Additional tools for this integration were chosen based on my prior experience with UE5. The shelf was recreated using *C++ Slate Widget*. Rendering, materials, and post-processing styles are written in HLSL, instead of the GLSL used in the original Flair. To store shader data, We keep using dictionaries in Python as the model in Model-View-Controller (MVC) pattern; however, we fetch data directly from the *Material Parameter Collection* in UE5.

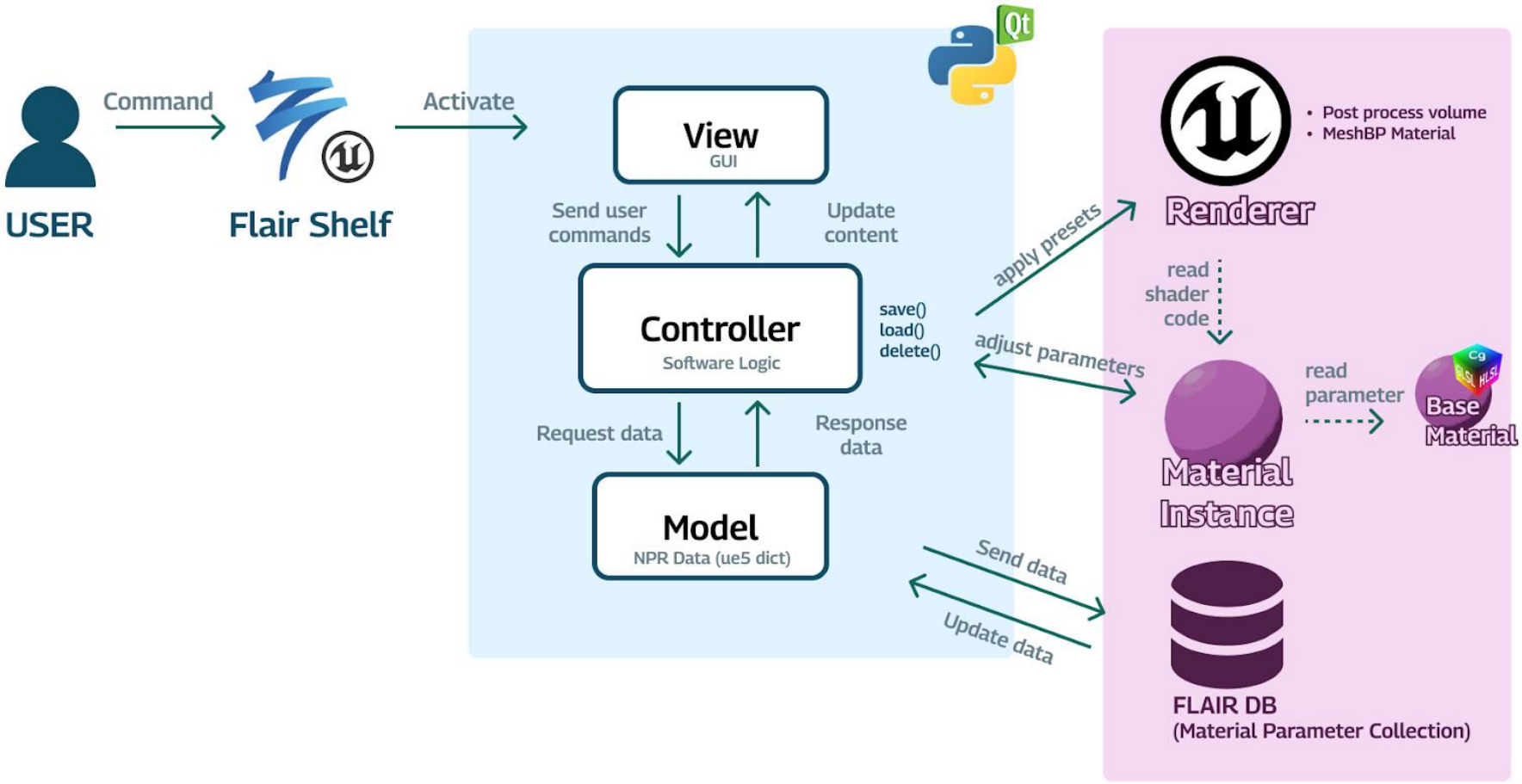

Figure 36. Architecture Blueprint of Flair UE5

5.2 Plugin Architecture

The plugin architecture in UE5 connects various parts of the plugin to the game engine as shown in Figure 36. The shelf contains available tools within the UE5 workspace. This shelf is implemented using a C++ Slate widget, which includes a view and a setup class that registers itself within the project and the UE5 editor. When a user selects a tool on the shelf, it triggers the Qt View to open. The Qt View is constructed using an MVC framework, similar to the original version of Flair.

For the style and material preset, the Qt controller receives commands from the user into UE5's rendering system. When using the *Style Presets*, the controller locates the existing *Post Process Volume* and applies the chosen style to the rendering viewport. For *Material Presets*, the user must select a mesh in the viewport and then apply the material preset to it.

For adjustments such as *Global Settings* and *Material Attributes*, the tool applies adjustments to the material instance. This applies to both style and material preset adjustments, allowing UE5 to re-render changes in real-time. Flair materials, including post-processings and regular materials, are programmed in HLSL supported by UE5.

5.3 The Shelf with Slate Widget

To integrate a responsive and dockable shelf within the UE5 workspace, we chose the Slate Widget framework written in C++ to construct the shelf. The framework allows users to easily access the shelf through the tools menu by selecting the "Flair" option.

Shown in Figure 38, this shelf plugin is located in `/Plugins/FlairManager`, consists of 3 classes shown in Figure 37, which facilitate the functionality and integration of the shelf within UE5. These classes are crucial for the modular implementation of the shelf, each serving a specific role.

The main class (3) `FlairManager.cpp/.h,` manages the configuration of the shelf within the UE5 editor. It ensures the shelf is available when needed and is properly removed when the editor is closed or the plugin is disabled.

The class (1) `FlairStyle.cpp/.h` is responsible for creating and managing a custom Slate style set, which includes defining icons (*FSlateImageBrush*) and properties.

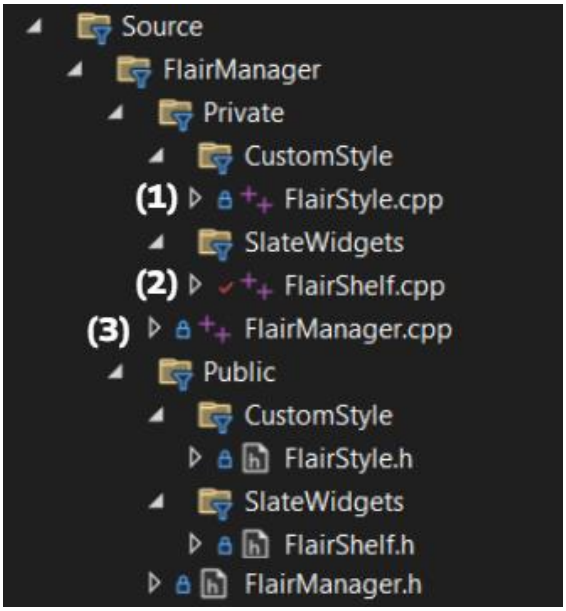

Figure 37. C++ Architecture the Shelf

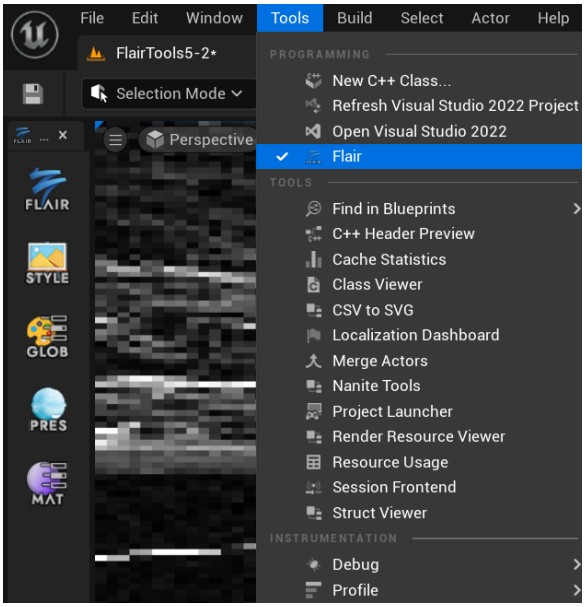

Figure 38. Flair Shelf in UE5

The class (2) `FlairShelf.cpp/.h`, is derived from *SCompoundWidget* for constructing the shelf, allowing to have child widgets within itself to form a cohesive GUI. Its child contains the *SWrapBox* widget for layout flexibility and responsiveness. The shelf can be customized into different layout styles based on user preferences, whether vertical, horizontal, or as a floating window.

5.4 QT GUI Integration with UE5

When the user selects a tool from the shelf, the corresponding *Qt Dialog* will open. The Qt **Dialog builder** is located in the `/Plugins/FlairUI` directory inside the UE5 project. It is written using the MVC framework, similar to the original version of Flair.

However, all controllers in this version are rewritten to exclusively communicate with UE5 instead of Maya. Presented in **Figure 39**, **there** contains the central controller in which any tool can access called `UEController`. Additionally, each tool can also have its local controller, for example, `GlobalAPICall`, which is accessible only within *Global Settings*.

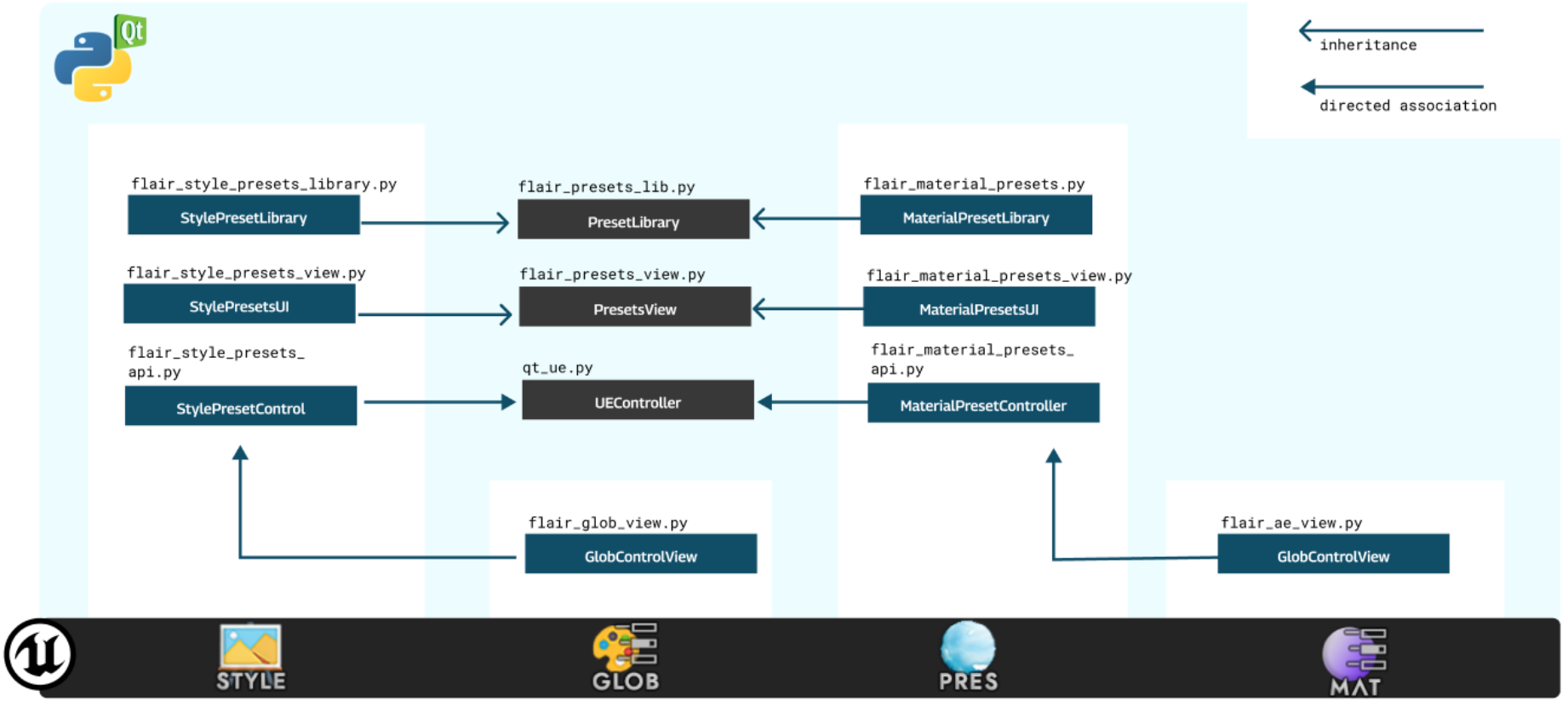

Figure 39. QT GUI Integration Architecture

5.4.1 Style Presets

The Styles Presets tool is created to automatically apply a post-processing material and add the material into *Post Process Volume* in the *Outliner*. The tool also detects whether to replace the existing materials or insert a new one. The implemented styles include *Blur*, *Cel*, *Sobel*, *Gaussian* and *Pixel*. This tool can also be considered as a filter selection. The post-processing presets are provided internally in the UE5 project.

This tool includes two main parts: *StylePresetLibrary* and *StylePresetsUI*. Within the module, the Style Preset class uses *StylePresetsUI* as the view to interact with the user, providing a GUI for applying the stylized post-processing styles based on the user's command. The command is then sent to *UEController* (the controller) to apply changes to UE5 Editor. The *StylePresetLibrary* manages the post-processing data received from the controller.

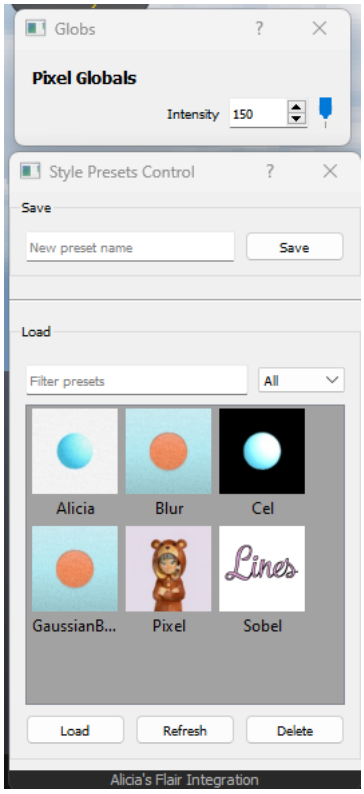

Figure 40. UE5 Style Presets and Globals Dialogs

5.4.2 Global Settings

The Global Settings tool provides a GUI containing the collection of global parameters from all active post-processing materials. Users can tune the parameter to influence overall environment appearance all at once without manually going through each of its instances in the UE5 Content Browser.

The Global class includes *GlobalView*, providing the GUI and displaying all global parameters and `GlobalAPICall`(the controller) contains the command to communicate between the view and UE5's Material instance.

5.4.3 Material Presets

The Material presets is a tool to easily create and load stylized materials within UE5. The tool is opened by clicking on the *PRES* shelf icon. The GUI construction is identical to Style Preset as they are derived from the same *View*. The available materials, similar to the Style Presets, include *Blur*, *Cel*, *Sobel*, *Gaussian* and *Pixel*.

The Material Preset class, `MaterialPresetLibrary` and `MaterialPresetsUI`, handles the automation to assign a material or add the material into the selected Mesh Blueprint in the workspace. The tool can replace the existing material or assign a new one to the mesh.

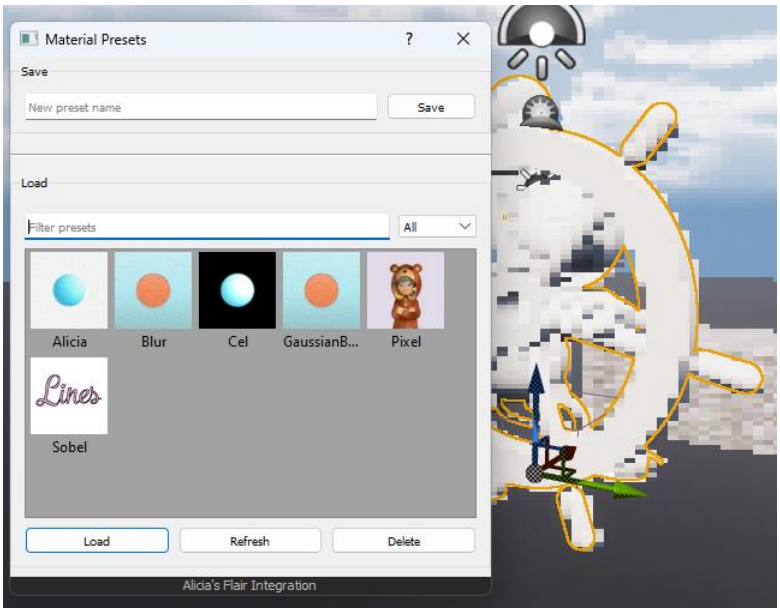

Figure 41. Material Presets GUI

5.4.4 Material Attributes

The Material Attributes tool, provides a GUI that looks similar to the *Style Presets*', for adjusting the attributes of materials used within the selected mesh blueprint. Users can adjust a set of parameters of each material assigned in mesh without manually going through each material and adjust it one by one.

The Material Attributes class includes `AttrView`, providing the GUI and displaying all material parameters and `AttrAPICall`(the controller) contains the command to communicate between the view and UE5's Material instances and Mesh Blueprints.

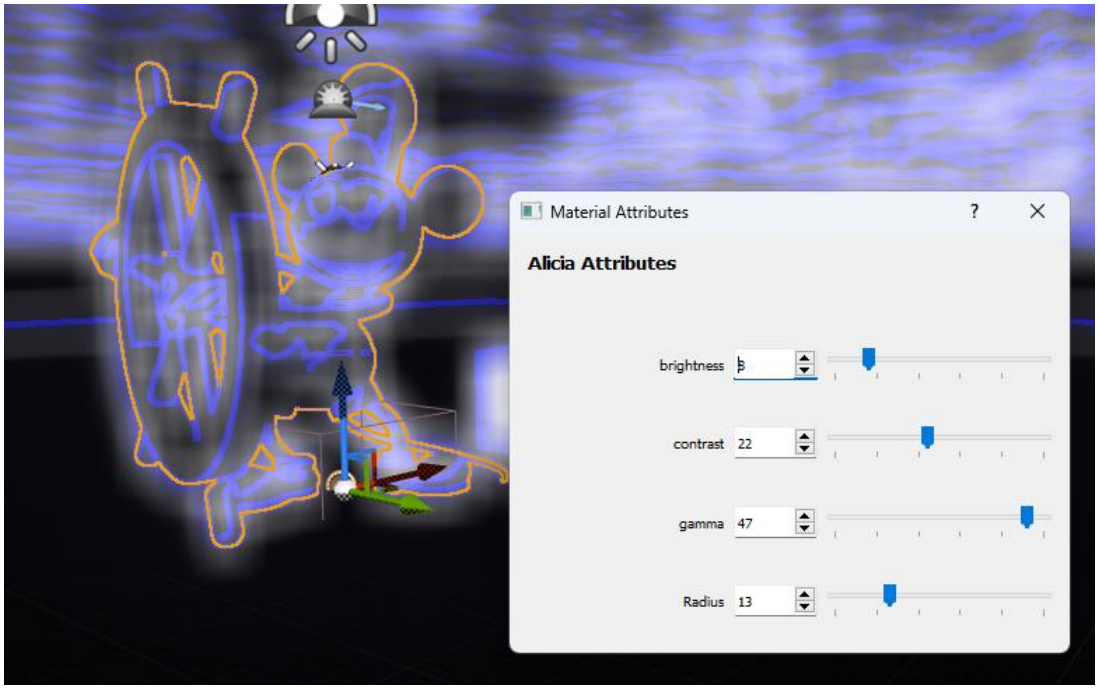

Figure 42. QT GUI Integration Architecture

5.5 NPR Materials and Construction

Material plays a key role with the Flair UE5 plugin, which is designed for creativity and ease of use with NPR (Non-Photorealistic Rendering) shaders and materials. The structure of each Flair's material (Figure 43) is modular, with each element having a specific role.

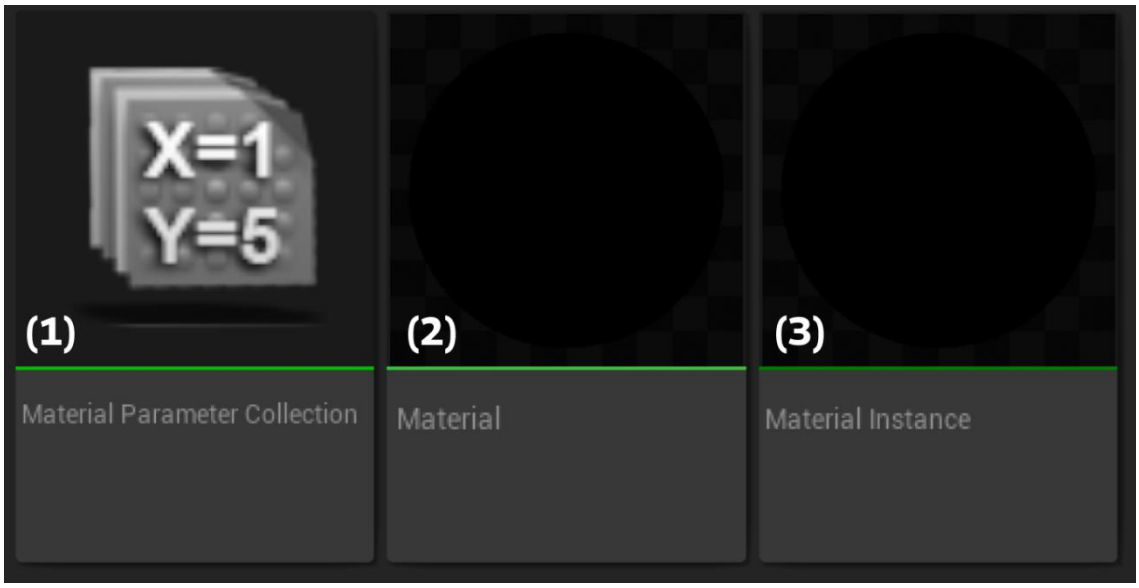

Figure 43. Materials' Architecture

5.5.1 Material Parameter Collection as Parameter Configuration

This component acts as the default configuration and provides predefined values for each Flair material. The configuration is manually adjusted in the editor mode and cannot be modified through the *Qt Framework*. This standard applies to all presets under the plugin.

5.5.2 Material Asset

Material Assets are central to the material implementation in UE5. These are created and modified within the Material Editor, where we utilize HLSL (High-Level Shader Language) along with UE5's material components to achieve specific rendering effects.

5.5.3 Material Instance

A Material Instance is derived from a Material Asset and can be used within both the Qt Pyside2 and UE5 environments. Modifications to this instance can be performed directly or through external mechanisms such as Flair globals or attributed materials.

This chapter has wrapped up a detailed overview of the Flair UE5 plugin's development, from the initial blueprint to its full integration with UE5's tools like C++, Slate Widget, HLSL, and the UE5 Python API, combined with the Qt framework. This design prepares the plugin for future scalability and extensive testing.

The upcoming testing phase is crucial to validate if the plugin meets our design goals, focusing on user-friendliness and usability. With the customized architecture from the original Flair, based on the MVC framework, and its ability to automate settings and material attributes, the plugin has potential to advance rendering capabilities in UE5.

6 Results and Evaluation

This chapter details the procedures and outcomes of the testing sessions for UE5's Flair Plugin, particularly focusing on the Styles and Material Presets. Testing involved UE5 artists and developers, who represent the target groups identified early on in the User Research and Design process (referenced in Chapter 3).

These sessions not only uncovered unanticipated issues but also validated elements that correspond to the design goal. The feedback will be used for further refinements after all testing sessions. Thus, the version of the plugin presented during all testing sessions was in a slightly different state compared to the finished product, but the core feature of the plugin remained the same.

6.1 Testing Objective and Methodology

The objective was to validate the usability and user-friendliness of "`Style Presets`" and "`Material Presets`", as well as the integrated Qt Dialog and Flair's shelf. The qualitative approach was used to capture individual thoughts and user experiences that quantitative methods might overlook.

Following the methodology proposed [15] by Jakob Nielsen, the test consisted of 5 sessions with each tester participating in a one-on-one setup. During each session, the tester was asked to perform specific actions without explicit guidance, such as opening the preset tab or applying a style to a scene. This approach helped observe the practical use of the plugin and captured the testers' experiences.

6.2 Participant Recruitment

The recruitment took place from April 16th 2024 to April 26th, 2024. Participants booked their available date and time through a Doodle form [15]titled "UE5 Flair NPR Usability Test." I recruited participants by personally writing and posting the advertisement on the University of Tartu Game Jam Discord channel, which led to the recruitment of two testers.

Additionally, Raimond Tunnel, the head of the Computer Graphics and Virtual Reality Study Lab(CGVR) at the University of Tartu, significantly assisted me by posting the ads in Estonian within the Tartu Art School community forums, successfully attracting three more testers whose profiles closely matched the target user group. Following registration

[15] https://doodle.com/meeting/organize/id/b8B1YJ2e

through Doodle, the date and time of each session were booked in Google Calendar, with details subsequently sent via email.

6.3 Testing Location

The testing environment took place in the CGVR lab (Delta room 2007) with each testing session using the same hardware and software configurations to ensure consistency. Unreal Engine 5.2.1 was installed on a personal computer with the Flair plugin pre-installed.

6.4 Testing Procedure

Each participant was informed that the testing session would be recorded, including audio. If a participant preferred not to be video recorded, the camera could be disabled. However, audio recording and screen capture were mandatory.

The participant sat in front of the personal computer while recording using Zoom, enabling the use of webcam and screen sharing features. The recordings were automatically saved at the end of each session.

6.4.1 Phase 1 - Initial Q&A

The test began with a brief interview to collect background information about each tester, including their name, experience with Unreal Engine 5, familiarity with stylized rendering, and prior usage of any UE5 plugins to facilitate their workflow.

Table 1. Information of each participant in the session.

Session	Date	Professional Backgrounds	UE5 Experience (Years)	Experience with NPR	Duration (minutes)
1	2024-04-22	Generalist UE5 Developer	1	None	14.13
2	2024-04-22	Senior UE5 Technical Artist	7	Advanced	30.34
3	2024-04-22	Generalist UE5 Developer	4	Intermediate	20.38
4	2024-04-23	UE5 and Unity Environment Artist	1	Intermediate	12.16
5	2024-04-26	Generalist UE5 Developer	10	Intermediate	19.56

6.4.2 Phase 2 - Task Performance

This phase served as a blind test to assess if the UI design is intuitive. Participants were given specific tasks but not instructed on how to proceed, testing the UI's clarity and their ability to navigate it independently.

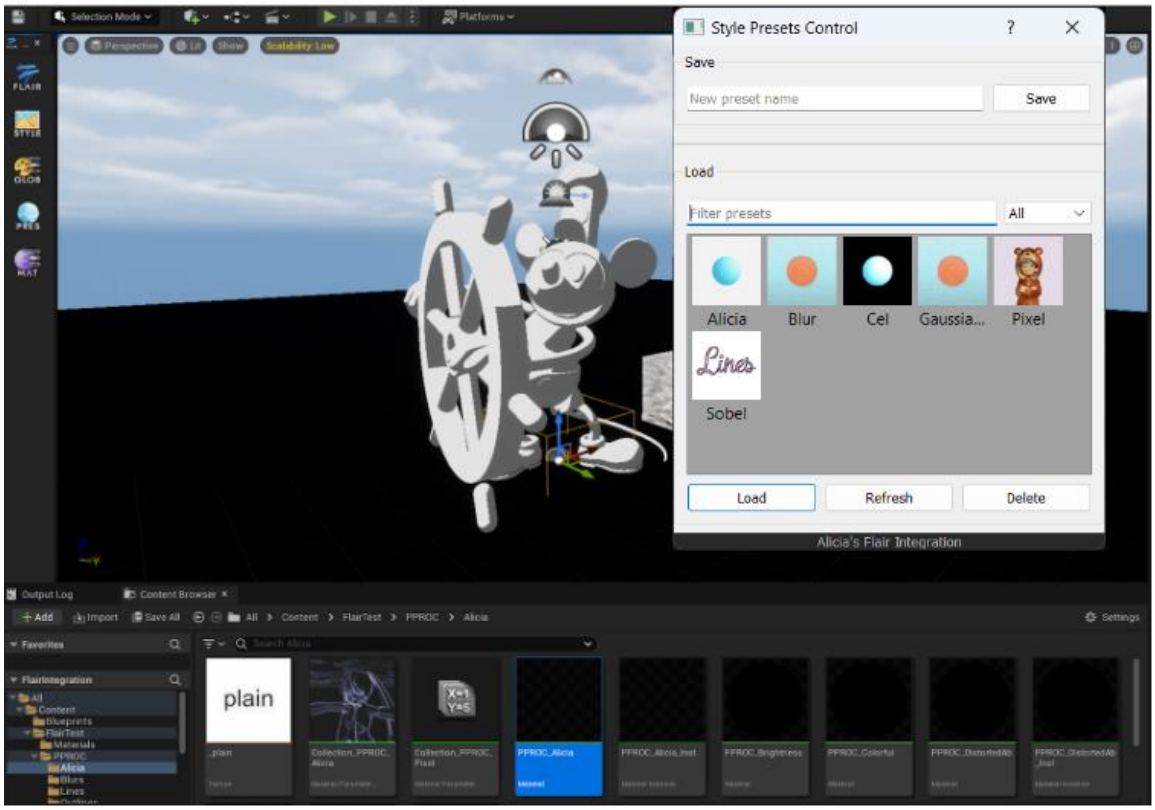

Figure 44. UE5 Editor - Flair Test Environment

Table 2. Task List

ID	Tasks	Description
1	Open Flair Shelf	Participants accessed the NPR tools within UE5.
2	Explore Plugins	Participants discussed their previous experience with post-processing effects. Those unfamiliar with these effects were given a brief introduction.
3	Open Documentation	Participants were asked to locate the Flair documentation on the shelf to determine how to proceed if they got lost with specific actions.
4	Use Style Presets	Participants were asked to open "*Style Preset*" from the shelf, load one or two NPR styles (e.g., *Blur*, *Outline*, *Cel*) then inspect the viewport
5	Adjust Globals	Participants accessed the Flair Globals node by clicking the GLOB icon and modified global parameters, assessing effectiveness in impacting the entire scene.
6	Create and Save New Preset	After adjusting global parameters, participants were asked on how to save these presets for future use.
7	Delete Preset	Participants deleted custom style presets as needed.
8	Conceptualize Material Presets and Material Attributes	-Participants were asked to find where on the shelf to access Material Presets and adjust material attributes. -This task was designed to compare with the Style Preset task to gather feedback on similar functions.
9	Manual NPR Creation	-Participants manually implemented post-processing volumes, wrote HLSL code for post-processing, creating a simple material, and integrating these into the scene. -This task was for users to compare the ease of operations with and without the plugin.

Table 3. User Task Performance in Flair Plugin Testing

Session ID	Professional Background	Experience (year)	1. Open Shelf	2. Explore Plugins	3. Open Documentation	4. Style Presets	5. Adjust Globals	6. Save New Preset	7. Delete Preset	8. Access Material Presets	9. Manual NPR
1	Generalist UE5 Developer	1	✔	✔	❓	✔	❓	✔	😐	😐	❌
2	UE5 Technical Artist	7	✔	✔	✔	✔	😐	✔	😐	😐	✔
3	Generalist UE5 Developer	4	✔	✔	❓	✔	❓	✔	✔	😐	😐
4	Environment Artist	1	✔	✔	😐	✔	✔	✔	✔	✔	❌
5	Generalist UE5 Developer	10	✔	✔	😐	✔	😐	✔	✔	😐	✔
		Sum	✔ (all)	✔ (all)	😐 (2) ❓ (2) ✔ (1)	✔ (all)	😐 (2) ❓ (2) ✔ (1)	✔ (all)	✔ (3) 😐 (2)	😐 (4) ✔ (1)	❌ (2) ✔ (2) 😐 (1)

✔ : Figured out with no problem. 😐 : Figured out after some effort. ❓ : Needed guidance. ❌ : Completely unable to figure out

Table 3. presents participants performing the tasks from 1 to 9 validating the UI's intuitiveness. Each session revealed common issues, notably in Task 3 and 5.

Additionally, participants' feedback often reflected their professional backgrounds; generalists and juniors primarily suggested design improvements, whereas developers and technical artists focused on technical and performance aspects of the plugin.

Task Completion Success:

All participants successfully completed tasks Open Shelf (1), Explore Plugins (2), Style Presets (4), and Save New Preset (6), proving that initial access to the plugin is intuitive across all user levels.

Complex Tasks and User Levels:

Developers and generalists experienced difficulty using Global Settings (5), mentioned that the word "*Global*" is common among technical artists and specialized 3D artists, not with general developers.

Delete Preset (7) showed varied results based on user prior experience. Senior users generally handled them better, possibly due to their familiarity with other tools and intuitions of long experience.

Session 1 and Session 4 were unable to perform the task Manual NPR Creation (9) as they had no prior experience creating shaders using HLSL and implementing post-processing volume from scratch. This results in their testing sessions ending early within 12-14 minutes. The participant in Session 3, in spite of years of experience and coding skills, needed guidance to understand the process. These user types matched the design goal as the *Style Preset* and *Material Preset* are made to aid artists who have limited knowledge in coding and technical overhead.

In contrast, participants who could smoothly implement NPR pipelines were the senior technical artist with 7 years of experience and the senior UE5 developer/generalist with 10 years of experience (Sessions 2 and 5). These users will only use the plugin if the plugin offers stunning NPR presets and they can access a bit of shader code to help them be creative.

6.4.3 Phase 3 - Q&A - User Feedback

Post-Testing Questions:

1. What is the overall experience so far?
2. Should the Glob tab be merged with the same Style Preset tab?
3. Is the plugin worth having in your project?
4. Anything else you want to add?

6.5 Discussion

The overall outcomes of the user testing sessions contain the successes, challenges, and critical insights derived from the UE5 users from wide ranges of experiences with the plugin.

6.5.1 Interactive User Observations

The hands-on sessions proved the core strengths in the plugin design and functionality, aligning well with the design goals. During these interactive observations, participants were able to navigate and utilize the *Style Preset* tool effectively, showcasing how integration could enhance their workflow efficiency.

Most participants appreciated that they can customize and explore several NPR styles. Traditionally, creating NPR materials from scratch and setting *Post Process Volume* is time consuming and unconventional for users such as 3D artists and environment artists. Once the all the NPR styles and presets available in Flair for Maya are ported over and implemented in UE5, the plugin will reach its full potential allowing to quickly iterate within stylized workflows.

The plugin appears to suit target groups like environment artists and technical artists, who found the *Global settings (Glob)* intuitive based on my interactions with the five participants. However, other generalists and developers occasionally struggled with the terminology. This suggests that while the plugin meets the needs of some users quite effectively, the clarity of its language and user interface need to be refined more to serve a broader range of professional users.

6.5.2 Areas for Improvement

Several areas still require improvement despite the positive feedback. Users faced difficulties figuring out the icon from the shelf to open the documentation and the global settings. The lack of intuitive design interrupts the workflow and affects the user experience.

In addition, when users successfully saved a customized preset, the content browser did not navigate to its file's location, and its thumbnails were missing. This left users unsure whether the preset was successfully saved or not. It has been suggested to improve visual feedback when presets are saved. Notably, this functionality is available in Flair for Maya but has not yet been implemented in Flair for UE5.

Moreover, the "*Delete*" button in "*Style Preset*" and "*Material Preset*" tabs were quite concerning; it is recommended that this button should be hidden unless the preset is actively selected. This change would simplify the GUI and smoothen the user experience by removing non-applicable options and adding a confirmation step to prevent accidental deletions.

6.6 Issues and Improvements Priority

After all testing sessions were completed, I evaluated the comments and findings, leading to the identification of main issues necessary for improving the design and usability of the system. These issues have been prioritized based on their impact from Section 6.4.3 Phase 3 - Q&A - User Feedback.

The table below categorizes all identified issues according to their relevance to different features of the Flair Plugin. Each issue is coded with a letter and numbers: 'u' stands for usability, 'd' for design. The second digit indicates the specific feature affected: 1 for documentation, 2 for style presets, 3 for global settings. The third digit specifies the sub-feature involved. For example, '`u22`' refers to a usability issue concerning the clarity of the "`save`" function within style presets.

Most of these issues will be addressed during the upcoming thesis defense, with critical ones such as `u1` and `u3` being the main priority before the presentation. This enables staff and other stakeholders to see the improvements made. The commitment to refining the project and enriching the quality is crucial as we prepare for the plugin's future growth.

Table 4. Prioritized Issues and Recommended Actions

Issue#	Description	Recommended Solutions	Proportion	Priority
u1	**Documentation Clarity:** struggles with accessing documentation.	Implement Tooltips in UE5	100%	High
u3	**Global Settings Clarity:** improve intuitive use and access.	Display Global Settings after a preset selection.	100%	High
u22	**Save Function Clarity and Responsiveness:** lacks clear feedback on action completion.	Implement navigation to the saved preset's location in the UE5 content browser.	40% (1,5)	High
d24	**Delete Function Risk:** design causing accidental deletions	Modify the Delete function: **Either** 1. Enable it only when a preset is actively selected. 2. Include a confirmation step **Or** Right click on the preset node to delete	40% (1,5)	Medium
d1	**The Plugin's Immersiveness Issues:** QT dialogs don't seem to blend well with UE5	Reskin GUI design to be more engaging and intuitive; using the similar color scheme according to UE5	40% (1,3)	Low

Lastly, this chapter reflected the generally positive feedback towards the Flair Plugin, despite a few issues that were identified. The findings from all testing sessions helped in identifying and prioritizing issues and tasks for future improvements based on users' experiences. After resolving the tasks listed in this chapter, the next step is to improve the plugin's quality and usability to ensure the smooth integration in professional workflows.

7 Conclusions

This thesis documented partially integration of the Flair plugin from Autodesk Maya into Unreal Engine 5. The journey began with overviews on non-photorealistic rendering (NPR) and Flair in Maya. Then we continued with a detailed survey of levels of control. I also get familiarized with technical terms essential for mutual communication with our target audience —technical and 3D artists.

The design phase included crafting user journeys to identify essential features, creating a persona and a user story to define design goals, and the hands-on experience of developing the plugin's shelf in UE5. This phase was crucial for integrating the plugin into the new application, which caters to game development and possibly wider target audiences.

During the development phase, the integration of the Slate widget with the QT framework, and the planning of the software architecture resulted in an immersive interface by combining these frameworks. This phase provided a valuable learning opportunity, allowing me to explore new areas, such as writing Python APIs, writing HLSL and GLSL, which enriched my technical skills as a developer.

The testing phase confirmed that the plugin functions well with our main target groups, including junior developers and generalists. The core design of the plugin, such as the Style Preset and Material Preset, was well-received by participants, aligning with our target group's needs. With the integration improvements, including the well-crafted NPR presets provided by Artineering, are expected to contribute to the success of this project.

This project showcased how integrating Flair's into UE5 opens up to new possibilities for creative expression in digital art and in game development. As computer graphics continue to grow, tools such as real-time rendering engines, easy-to-use shading languages, and intuitive plugins are essential. They make digital art more accessible and expressive for artists, which suits Unreal Engine's core mission of democratizing access to powerful creative tools.

8 Acknowledgments

First and foremost, I would like to express my deep gratitude to Ulrich Norbisrath, the professor at The University of Tartu, and Santiago Montesdeoca from Artineering, for their continuous support, patience, and guidance throughout the journey of this thesis. Their expertise and insights were invaluable to this work.

I am also thankful to the CGVR team at the University of Tartu for providing essential resources and a supportive research and testing environment. The staff were consistently responsive and helpful, assisting with a range of tasks from recruiting testers and thesis writing to submission, planning, and coding.

Special thanks to all participants, whose feedback and suggestions significantly enhanced the quality of this work. To my friends and family, for their continuous encouragement, and for being my sanctuary of strength during this journey, I cannot thank you enough.

I appreciate the UE-Developer Community and the Discord server from the Udemy course led by Vince Petrelli for their willingness to engage in technical discussions, assist with troubleshooting my code, and support me through moments of doubt and clarity.

Last but not least, to all those who indirectly contributed to this thesis and those I may have inadvertently omitted, I am grateful for your role in my academic journey.

9 References

[1] N. Moon, M. Reddy and L. Tychonievich, “Non-photorealistic ray tracing with paint and toon shading,” 2021. [Online]. Available: https://doi.org/10.1145/3450618.3469173.

[2] S. E. Montesdeoca, "Real-time watercolor rendering of 3D objects and animation with enhanced control," 2018. [Online]. Available: https://dr.ntu.edu.sg/handle/10220/47356. [Accessed 2024].

[3] S. E. M. e. al., “MNPR: a framework for real-time expressive non-photorealistic rendering of 3D computer graphics,” 2018. [Online]. Available: https://doi.org/10.1145/3229147.3229162.

[4] J. H. R. Z. Paulius Liekis, “Art pipeline: transition from offline to realtime CG,” 2012. [Online]. Available: https://doi.org/10.1145/2343045.2343096.

[5] L. Z. Eric Chu, “Exploring alternatives with Unreal Engine’s Blueprints Visual Scripting System,” 2021. [Online]. Available: https://doi.org/10.1016/j.entcom.2020.100388.

[6] V. Colpaert, “NPR Environment in UE5,” 2024. [Online]. Available: https://80.lv/articles/learn-how-to-make-a-fantasy-npr-environment-in-ue5/.

[7] A. F. Donaldson, H. Evrard, A. Lascu and P. Thomson, “Automated Testing of Graphics Shader Compilers,” 2017. [Online]. Available: https://doi.org/10.1145/3133917.

[8] Aeros, “NPR rendering study on Unreal Engine 5.1,” 2024. [Online]. Available: https://dev.epicgames.com/community/learning/tutorials/l7kR/npr-rendering-study-on-unreal-engine-5-1.

[9] A. Hertzmann, “Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines,” 1999.

[10] D. Mould, R. L. Mandryk and H. Li, "Emotional response and visual attention to non-photorealistic images," 2012. [Online]. Available: https://doi.org/10.1016/j.cag.2012.03.039.

[11] A. Giordano, Study and development of a mobile-oriented application for the efficient management of a radiation test, 2020.

[12] MINIMALEFFORTTECH, "Simple UX Tips For Developers," 2021. [Online]. Available: https://minimaleffort.tech/simple-ux-tips-for-developers/.

[13] MINIMALEFFORTTECH, "QT TO SLATE TRANSITION GUIDE," 2021. [Online]. Available: https://minimaleffort.tech/qt-to-slate-transition-guide/.

[14] M. Wadstein, "Editor Utility Widget in Unreal Engine 4," 2019. [Online]. Available: https://youtu.be/C9UAuH72z6M?si=Z1x-vpLbMdui9LYL.

[15] J. Nielsen, "Why You Only Need to Test with 5 Users," 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[16] Fingent, "Top 10 Tech Stacks That Reign Software Development in 2024," Fingent, [Online]. Available: https://www.fingent.com/blog/top-7-tech-stacks-that-reign-software-development/.

Appendix

I. Glossary

Non-Photorealistic Rendering The bar (or other symbol) marking the active editing point.
MNPR [2] A framework for real-time expressive NPR of 3D computer graphics
Live Coding A tool that can be used within the Editor, that allows the hot-reload of C++ code for faster iteration when developing
Shaders A set of instructions that are executed all at once for every single pixel on the screen. [11]
Pipeline a. (General) -- A series of processes or steps through which digital content is created, developed, and finalized. [3] b. (Game Development) -- The structured process through which game assets, code, and functionalities are developed, integrated, tested, and refined to create a complete, playable game. c. (Flair Integration) -- The pipeline focuses on adapting and extending Flair's functionalities to be compatible with UE5's architecture, ensuring artists and developers can leverage its NPR capabilities across platforms.
Third-Party API An API that internal infrastructure didn’t develop and provided by some other cooperation

Technology Stack [16] A combination of programming languages, frameworks, libraries, tools, and technologies that are used to develop and deploy a software application or system
Modularity A programming concept where developers separate program functions into independent pieces. These pieces then act like building blocks, with each block containing all the necessary parts to execute one aspect of functionality.[16]
Mesh A part of a 3D model which defines a set of vertices of an object.
Material A set of properties that defines how an object reflects or emits light, such as color, transparency, shininess, roughness, and so on.[17]
Blueprints a. (General) -- A plan or a set of guidelines for achieving something. b. (Unreal Engine) -- A gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor.[18]
Learning Curve A process where people develop a skill by learning from their mistakes.[19]

[16] https://www.tiny.cloud/blog/modular-programming-principle/#:~:text=Modular%20programming%20is%20a%20general,execute%20one%20aspect%20of%20functionality.
[17] https://www.linkedin.com/advice/0/what-best-materials-textures-use-3d-rendering-3cwcc
[18] https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/GettingStarted#:~:text=The%20Blueprint%20Visual%20Scripting%20system,or%20objects%20in%20the%20engine.
[19] https://www.collinsdictionary.com/dictionary/english/learning-curve

II. Plugin Guide

To access and use Flair for UE5, one must follow these:

1. Extract `FlairIntegration.zip`
2. Open the `Alicia-FlairIntegration/FlairIntegration` folder
3. Open *FlairIntegration.uproject*
4. On the top of the screen go to `Tools > Flair`

Requirements:

- Unreal Engine 5.0 up to 5.2.1
- Windows 10 or 11
- Python *Qt*, *Numpy* and *Reload* installed.
- C++ for game development enabled.

III. Accompanying Files

- *Alicia-FlairIntegration/UsabilityTesting* – A folder of recordings from all testing sessions. Each video includes an interview with the usability testing.
- *Alicia-FlairIntegration/FlairIntegration* – A project directory containing the plugin builds (Flair Shelf and Qt Dialogs)
- *FlairIntegration/Plugins/FlairManager* – A Flair Shef directory written in C++
- *FlairIntegration/Plugins/*– A directory containing Python API and Qt Dialogs activated after a user pick a tool from the shelf.

IV. Source Code

Available via request on

- https://github.com/Alitcher/UE5ShaderIntegration

Please send an email to alicia.sudlerd@gmail.com to request access.

V. Usage of AI Tools

This thesis includes ChatGPT-4 for research and writing up until May 15, 2024. The tool is categorized in two primary areas: Thesis Writing Assistance and Programming Assistance.

ChatGPT-4 for Thesis Writing Assistance:

1. Help in drafting and structuring a title and chapters of the thesis.
2. Tone adjustments and word choices for non-developers.
3. Summarize research papers.
4. Help with explaining and simplifying complex academic language.
5. Format references according to academic standards.
6. Explain the table or diagram and help drafting them.
7. Analyze testing sessions.
8. Improve clarity and readability of sections.
9. Write ads to recruit participants for testing.
10. Revising the tone and content of a conclusion chapter.
11. Translate the abstract to Estonian

ChatGPT-4 For Programming and Debugging:

1. Debug syntax errors in Python.
2. Improve efficiency of image processing algorithms.
3. Compare different code snippets for performance and quality.
4. Handle communications between Slate Widget in C++ and PySide2.
5. Evaluate the pros and cons of using different UI tools.
6. Custom UI tools using Slate and other tools such as UMG and Editor Utility Widgets.
7. Handle shader programming errors.
8. Handle compile error during live coding in UE5 is on.
9. Help with creating responsive panels using Slate Widget.
10. Manage material parameters through Python APIs.
11. Manipulate materials inside Post Process Volume through Python APIs.

Impact on Thesis

ChatGPT-4 significantly improved the quality of both the written and technical components of the thesis. Screenshots demonstrate the specific uses and benefits. The AI's support was essential for meeting deadlines and maintaining high academic and technical standards.

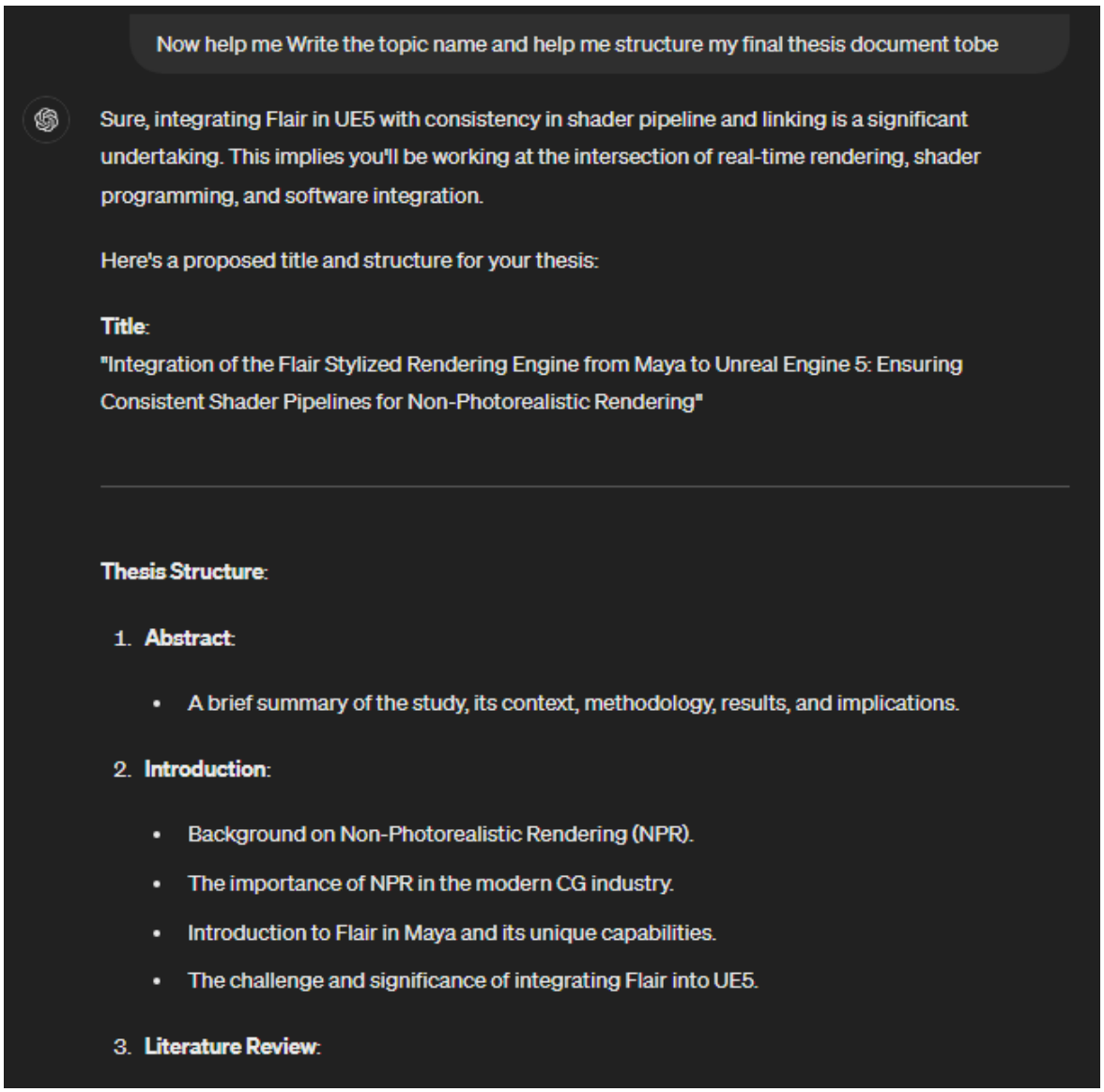

Figure 45. Thesis Writing Assistance

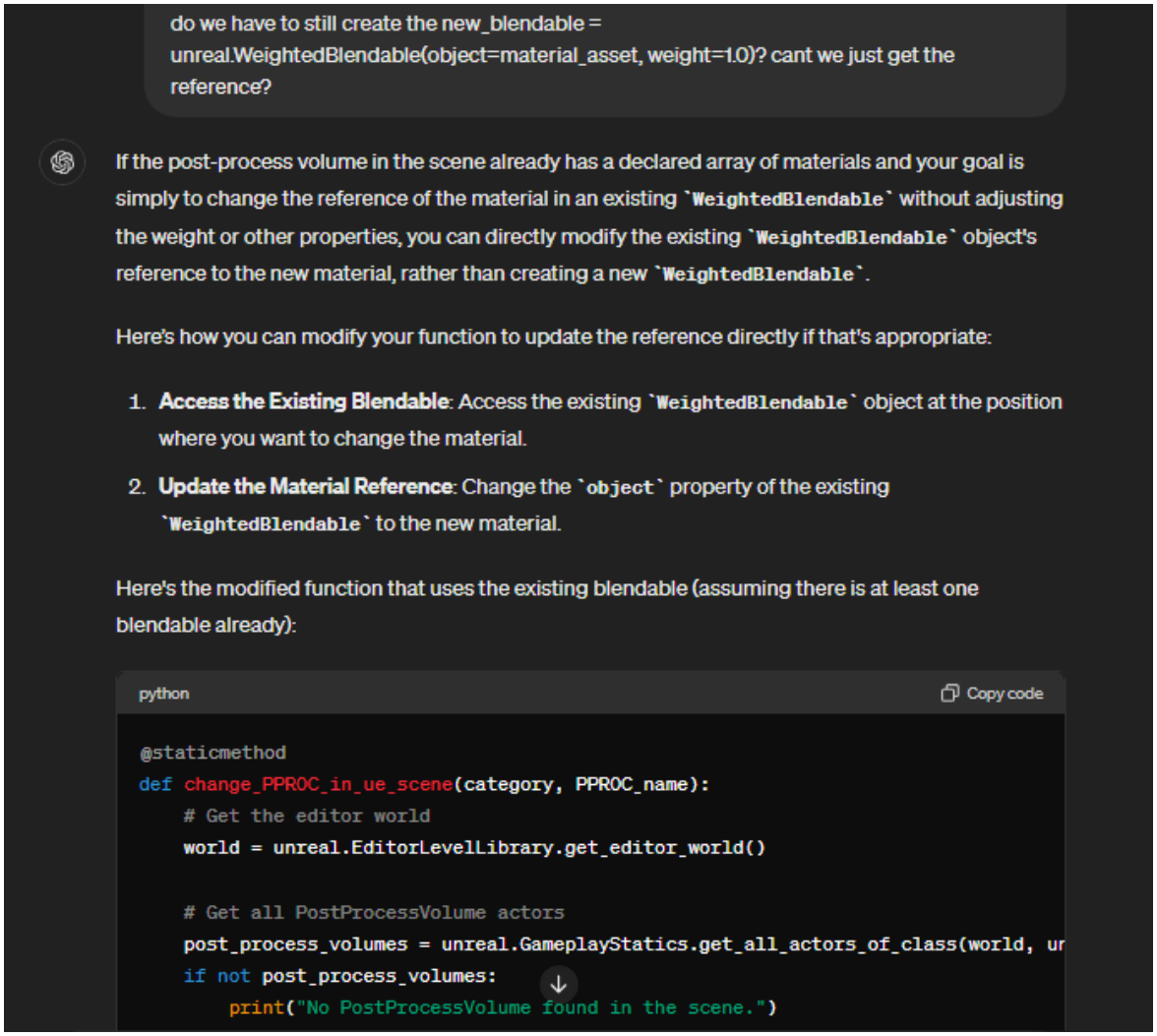

Figure 46. For Programming and Debugging

VI. License

Non-exclusive licence to reproduce thesis and make thesis public.

I, **Alicia Sudlerd**,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

UE5 Flair's Integration – UI, Shaders, and Image Algorithms for NPR,

(title of thesis)

supervised by Ulrich Norbisrath, PhD and Santiago Montesdeoca, PhD

(supervisors' name)

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **17.05.2024**