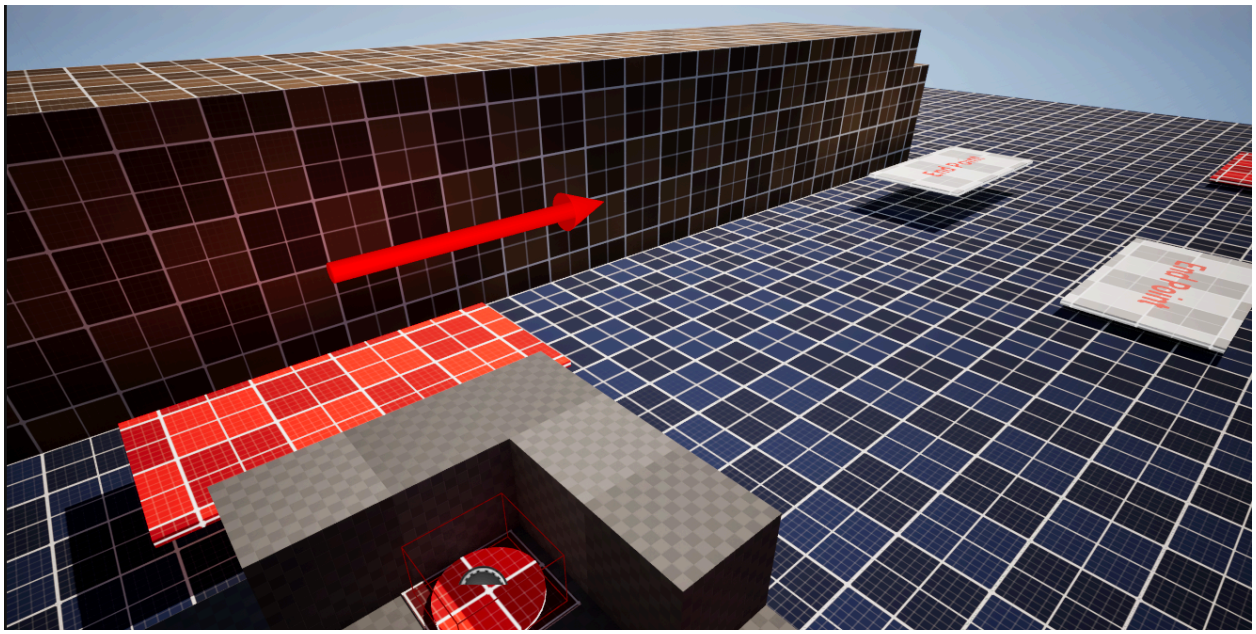
BP_MovingPlatform & BP_Pressurepad_Platform



ViKing Production
Prepared by: Joey Vanlanduyt
Unreal Engine Version 5.1.1

Overview

This guide walks you through the creation of an interactive Pressure Pad and Moving Platforms System in Unreal Engine. Designed for both beginner and intermediate developers with a foundational understanding of Blueprints, this system offers a modular, reusable solution for dynamic environmental interactions. Whether you're building puzzles, platforming sequences, or time-sensitive challenges, this system integrates seamlessly into a wide variety of gameplay scenarios.

The pressure pad responds to specific actor interactions, such as players or designated objects, activating or deactivating linked moving platforms. By combining dynamic visual feedback, sound effects, and smooth platform movement, the system ensures a polished and intuitive player experience. In addition, the setup is fully customizable within the Unreal Engine editor, allowing level designers to easily configure and expand the system to suit their project's needs.

This guide emphasizes functionality, modularity, and player feedback, with detailed explanations of the Blueprints, setup processes, and design decisions behind the system. By the end, you'll have a versatile tool that enhances both gameplay and level design while serving as a foundation for future expansions or mechanics.

In this guide, you'll learn:

- How to set up a pressure plate Blueprint that reacts to player or object interaction.
- How to link moving platforms to the pressure plate using an array for dynamic control.
- How to create smooth platform movement using timelines and lerp functions.
- How to incorporate sound effects and visual feedback to enhance player interactions.

This tutorial assumes a basic understanding of Unreal Engine's Blueprints, including:

- Actor placement in a level.
- Using variables, arrays, and casting.
- Working with timelines for animation.
- Configuring details in the editor.

Key Features of This System:

1. Interactive Pressure Plate:
    - Responds to overlapping actors such as the player or a designated object (e.g., a crate).
    - Plays a timeline animation to simulate the plate depressing when activated.
    - Updates its state dynamically (activated or deactivated) and broadcasts this change to linked platforms.
2. Modular Moving Platforms:
    - Moves between two points based on activation signals from the pressure plate.
    - Uses timelines to ensure smooth, realistic movement between locations.
    - Can be linked to multiple pressure plates for complex gameplay scenarios.

3. Editor Configurability:
    - Platforms to be controlled are assigned through the pressure plate's details panel, ensuring flexibility and ease of setup.
    - Parameters such as movement speed, platform paths, and feedback effects are adjustable directly in the editor.
4. Feedback Systems:
    - Visual indicators, such as platform movement and material changes, ensure players understand the interaction's impact.
    - Sound effects provide additional feedback for activation and deactivation events, enhancing immersion.

By the end of this guide, you'll have a modular, reusable system that can be integrated into any Unreal Engine project, offering a polished and interactive gameplay experience. Whether you're creating puzzles, platforming challenges, or dynamic environments, this system provides the foundation for endless possibilities.

Design Choices and Philosophy:

Player Feedback To ensure players intuitively understand their interactions, this system combines visual and auditory feedback to create a clear and engaging experience. When the pressure plate is activated, it dynamically changes its material to a distinct color, providing immediate visual confirmation that an interaction has occurred. This feedback not only helps players associate their actions with in-game responses but also strengthens their sense of engagement and immersion. Linked moving platforms respond instantly upon activation, with smooth and fluid motion achieved through timelines and interpolation (lerp) functions. This seamless transition ensures the interaction feels polished and intentional, adding to the overall immersion of the gameplay. Optional sound effects further enhance the experience by providing auditory cues, such as a "click" for activation or a mechanical noise as platforms begin to move. These sound effects reinforce the mechanical nature of the interaction, guiding players effectively even when they are not directly observing the platform movement. Together, these feedback mechanisms create a cohesive and intuitive system that ensures players are always aware of the impact of their actions.

Modularity: The pressure pad and moving platform system is designed with adaptability in mind, allowing for effortless integration into a wide range of gameplay scenarios. The standalone pressure plate blueprint (BP_PressurePad) is designed to function independently, making it reusable throughout a level without requiring redundant setup. Each instance can be customized directly in the editor to control specific platforms or linked objects, offering unparalleled flexibility for level designers. The system supports linking multiple platforms to a single pressure pad through an array in the editor, enabling intricate puzzles and sequences where a single action can affect multiple environmental elements. For example, a designer could create a scenario where activating a pressure pad causes several platforms to move simultaneously, forming a path to a new area or triggering a timed challenge. The moving platform blueprint (BP_MovingPlatform) is equally modular, allowing designers to set unique start and end positions directly in the editor. With adjustable movement speeds and customizable paths, the platforms can be adapted for diverse applications, from straightforward platforming challenges to complex, interconnected puzzles. This modularity ensures the system remains versatile and scalable, meeting the needs of both simple and advanced gameplay designs.

Iterative Prototyping: The design process for the pressure pad and moving platforms system emphasizes rapid prototyping techniques to maximize efficiency and adaptability. During the initial development stages, block meshes are used to represent the pressure pad and platforms. This simple approach enables rapid iteration, allowing designers to test functionality, positioning, and interactions without being encumbered by the complexities of final art assets. By focusing on mechanics first, designers can quickly make adjustments to platform placement, pressure pad behavior, or interaction logic without the need to rework intricate visual details. This streamlined workflow not only saves significant time and resources but also ensures that the core gameplay remains the priority.

Once the system’s functionality is thoroughly tested and finalized, detailed assets can be introduced to replace the placeholder block meshes. This smooth transition allows the final product to achieve a high level of visual polish while ensuring that the underlying mechanics remain solid and reliable. The iterative

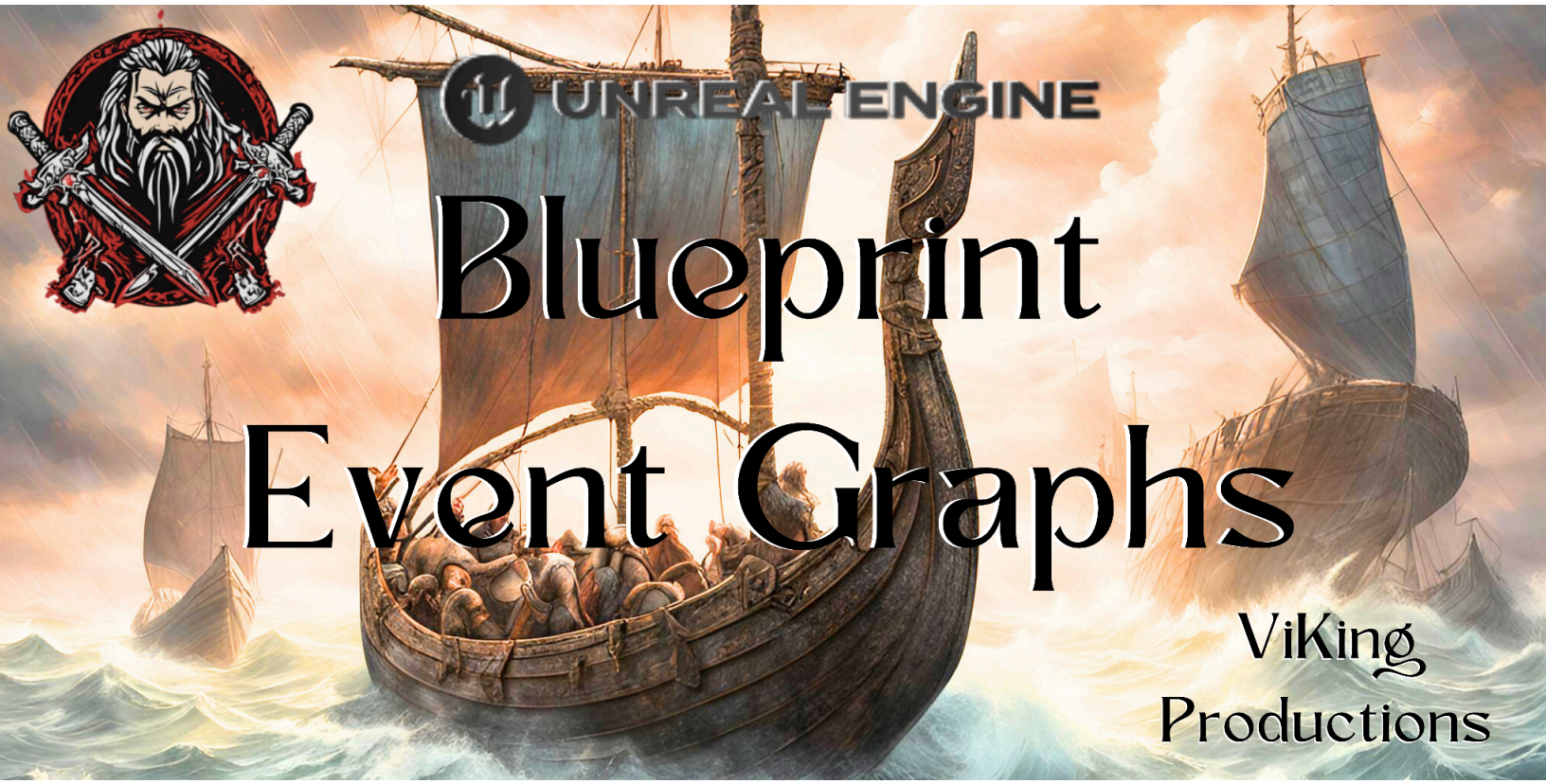
nature of this process ensures a balance between functionality and aesthetics, producing a system that is both mechanically sound and visually appealing.

Philosophy Behind These Design Choice: The design philosophy guiding this system is rooted in clarity, usability, and adaptability. Every element of the design, from the pressure pad's color change to the moving platform's smooth motion, is crafted to provide immediate and clear feedback to the player. This focus on clarity ensures that players can easily understand the cause-and-effect relationships between their actions and the system's responses, reducing frustration and enhancing the overall gameplay experience.

For designers, the system's modular design philosophy ensures that it is not only functional but also highly flexible. These Blueprints can be easily integrated into various levels, enabling designers to reuse and customize them without needing to recreate functionality. This flexibility saves time, maintains consistency across levels, and reduces the potential for errors during implementation.

The system's adaptability also makes it a valuable asset for the broader project. The modular design and iterative prototyping approach mean that the system can evolve to meet the needs of the game. It can be used for simple tutorials in the early stages of the game and then scaled up for complex, multi-layered challenges in later levels. This versatility ensures that the pressure pad and moving platforms system remains a robust and essential tool for a wide range of gameplay scenarios, supporting the game's growth and development at every stage.

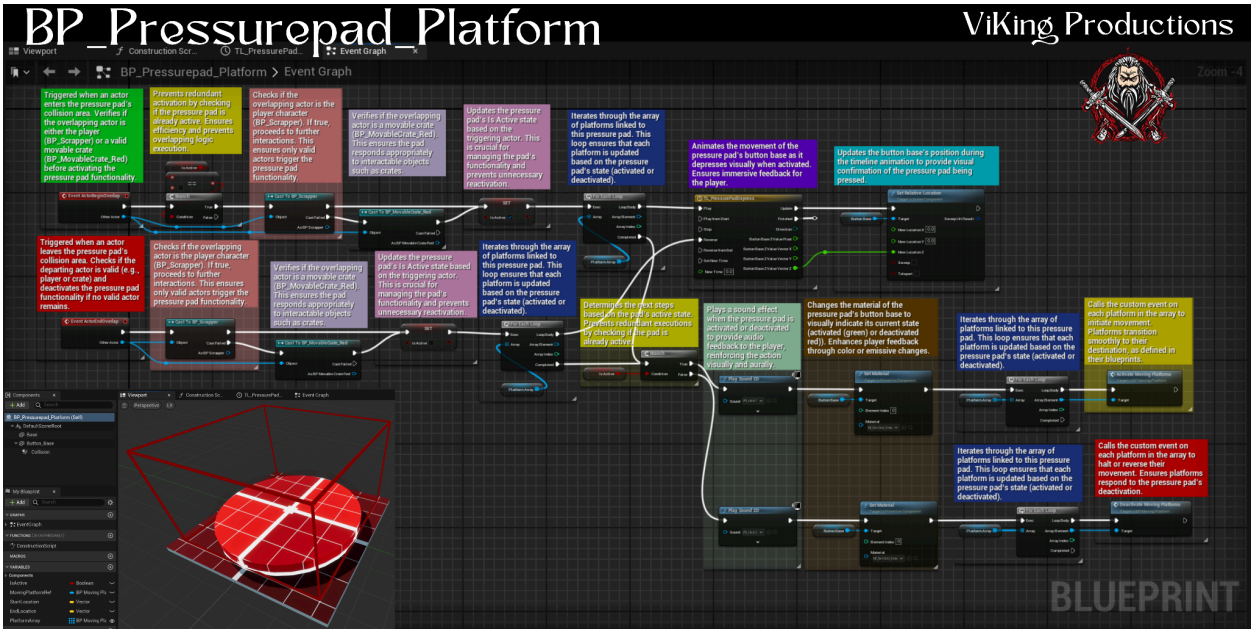
These design choices aim to create an intuitive, player-friendly system that is equally rewarding for designers, offering a balance of functionality, adaptability, and polish.



Unreal Blueprint: **Pressure Pad and Moving Platform** Difficulty Level: **3 out of 5:** ★★★☆☆

Blueprint Logic

1. Pressure Pad Blueprint (BP_PressurePad_Platform)



The BP_PressurePad_Platform blueprint is a multi-functional system designed to enhance player interaction and environmental storytelling by seamlessly integrating detection, activation, and feedback mechanisms. It serves several critical purposes in gameplay and level design:

The BP_PressurePad_Platform blueprint is a versatile and modular tool designed to enhance gameplay by detecting specific actor interactions, controlling linked platforms, and providing immediate feedback. Using dynamic actor recognition through overlap events, the pressure pad identifies when particular actors, such as the player character, NPCs, or designated interactable objects like crates, step onto or leave the pad. The system is highly customizable, allowing designers to define which actors can interact with the pad. This ensures precise control over gameplay mechanics by distinguishing between valid and invalid triggers, providing clarity and intentionality in interactions. To maintain a smooth gameplay experience, the blueprint includes state management, tracking whether the pad is "active" or "inactive" to prevent redundant interactions. This feature is particularly beneficial for puzzles or sequences that require timed or sequential activations.

The pressure pad also controls linked moving platforms, offering a wide range of gameplay possibilities. Through an array system, multiple platforms can be linked to a single pressure pad in the editor. When the pad is activated, it triggers the platforms to move along predefined paths. The blueprint supports bidirectional platform control, enabling platforms to return to their original positions upon deactivation. This functionality is ideal for creating intricate puzzles or multi-step gameplay challenges. Designers can populate the platform array with any number of platforms, ensuring adaptability for scenarios ranging from simple interactions to complex platforming sequences. The modular design allows seamless

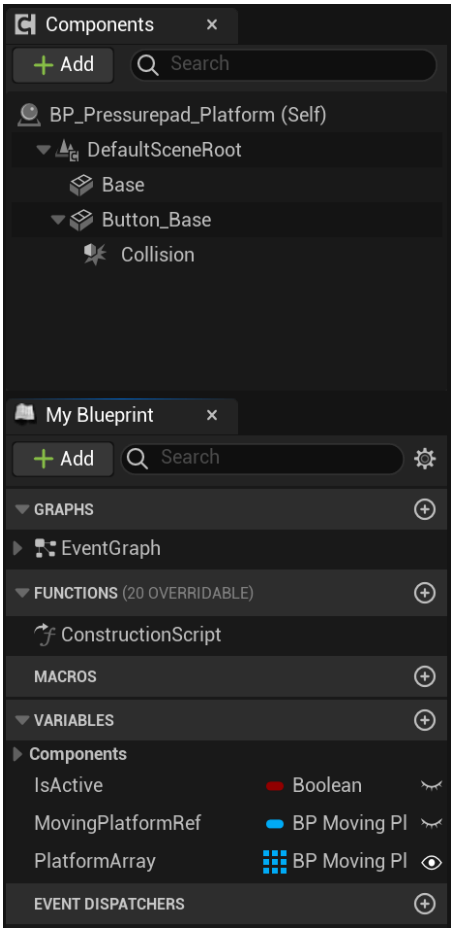
integration into various levels, with easy setup and maintenance that ensures changes to one pressure pad or platform do not affect others.

To enhance player engagement, the blueprint incorporates visual and audio feedback, making interactions intuitive and immersive. Upon activation, the pressure pad changes its material color, providing clear visual confirmation that the interaction was successful. This material change is customizable, allowing for different visual styles or themes. Optional animation depresses the pad when activated, adding a tactile, mechanical feel to the interaction. Complementing the visual elements, audio feedback includes a "click" sound effect upon activation or deactivation, offering auditory confirmation and reinforcing the mechanical nature of the system. Designers can replace these sounds with custom audio to align with the game's aesthetic, ensuring consistency across levels.

The blueprint's modular design further supports its utility in diverse gameplay scenarios. It can be dropped into any level and linked to existing moving platforms or other interactable elements without requiring extensive modifications. This flexibility ensures that the pressure pad can facilitate dynamic interactions while maintaining clean and efficient level design. Changes to one instance of the blueprint are localized, preserving the integrity of other instances and simplifying level creation.

In gameplay, the BP_PressurePad_Platform blueprint encourages critical thinking and problem-solving, particularly in puzzle-based levels. By linking player interactions to environmental changes, such as opening pathways or enabling new mechanics, the blueprint adds depth to player engagement. Immediate visual and audio feedback enhances immersion, aligning player actions with tangible in-game outcomes. The system's adaptability allows it to support everything from introductory tutorials to complex, multi-stage puzzles, making it a foundational component for dynamic interactions. In summary, the BP_PressurePad_Platform blueprint combines robust functionality, intuitive feedback, and modular design, ensuring its seamless integration into any project.

Key Variables:



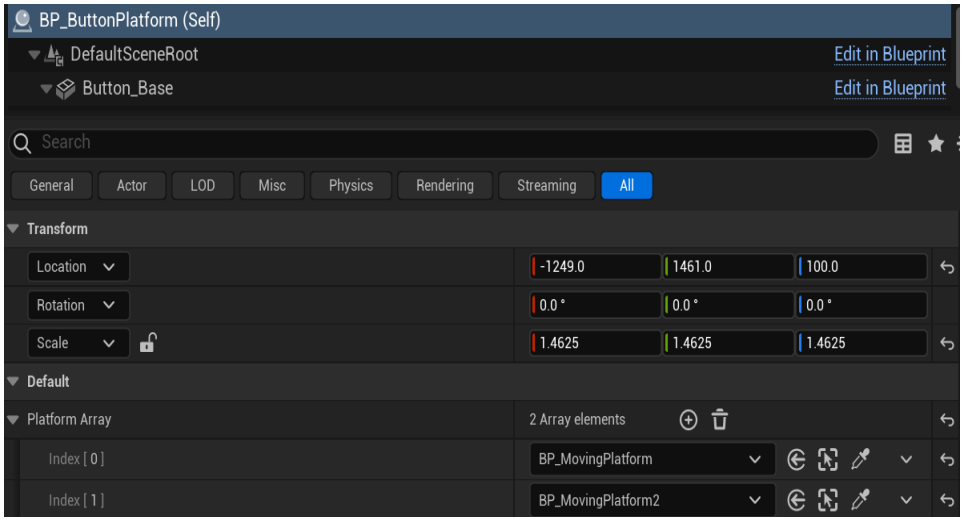
PlatformArray: An array storing references to linked moving platforms.

IsActive: Boolean that tracks the current state of the pressure pad (active or inactive).

ButtonBase: Represents the visual element of the pad that depresses when activated.

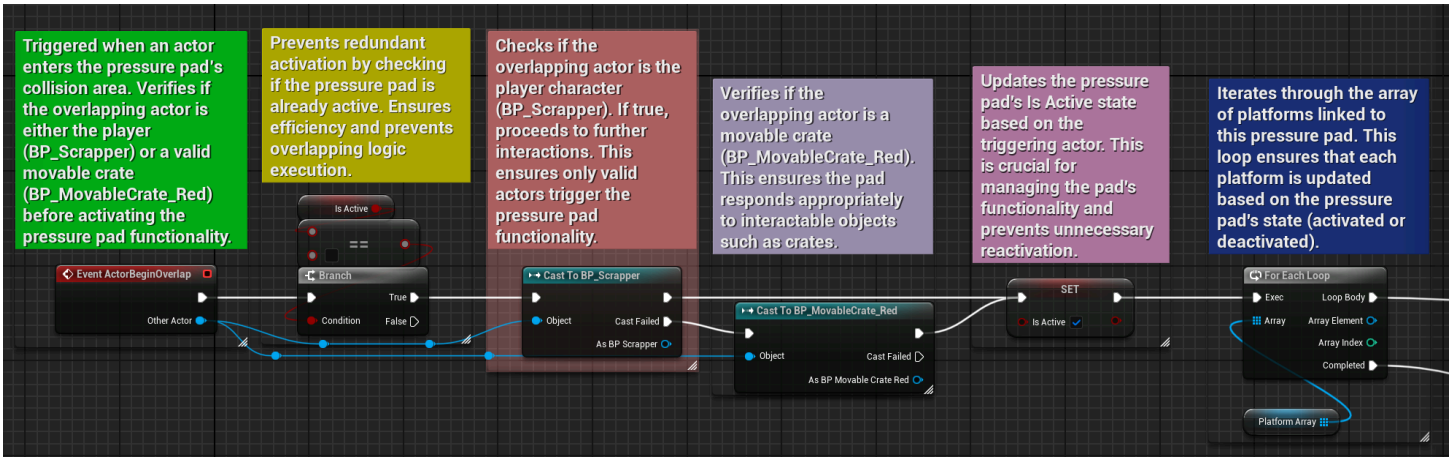
NOTE!

Make sure to assign the desired moving platforms to the button by selecting them in the button's Details panel.

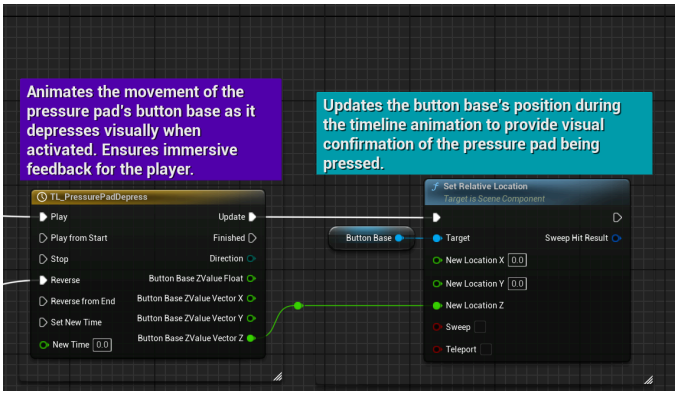


Logic:

Event: Actor Begin Overlap



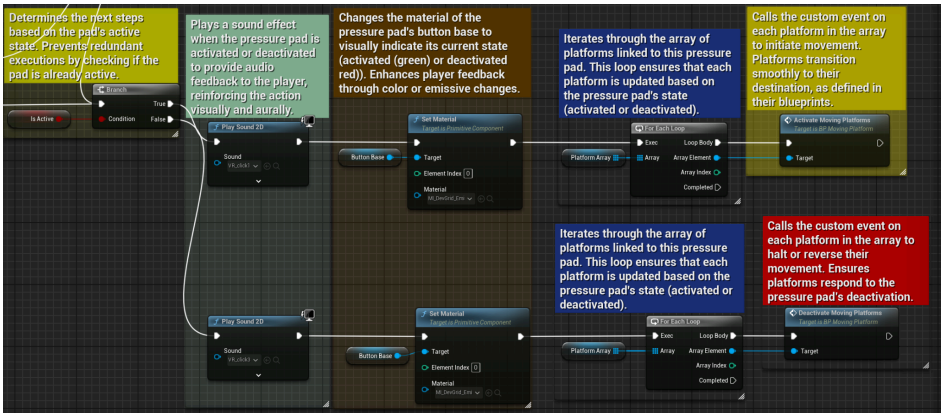
Detects if the overlapping actor is either the player (BP_Scrapper in the case of this tutorial) or a crate (BP_MovableCrate_Red). If valid, sets IsActive to True.



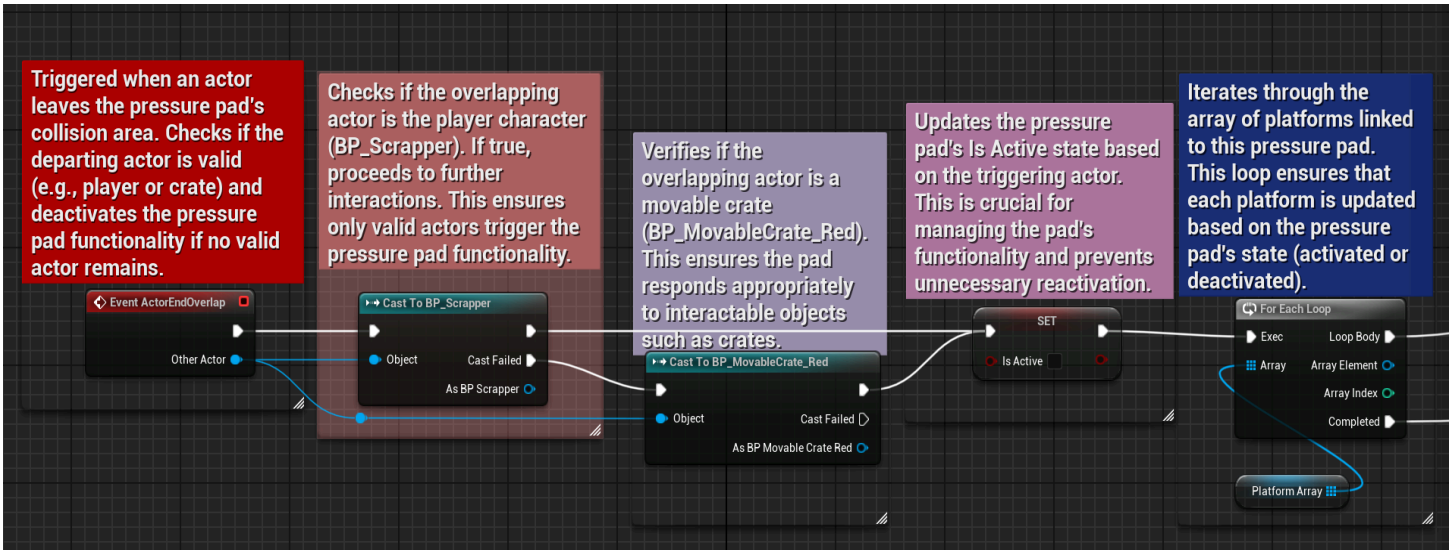
Activates the Timeline TL_PressurePadDepress to visually depress the button.

Plays a 2D sound for feedback.

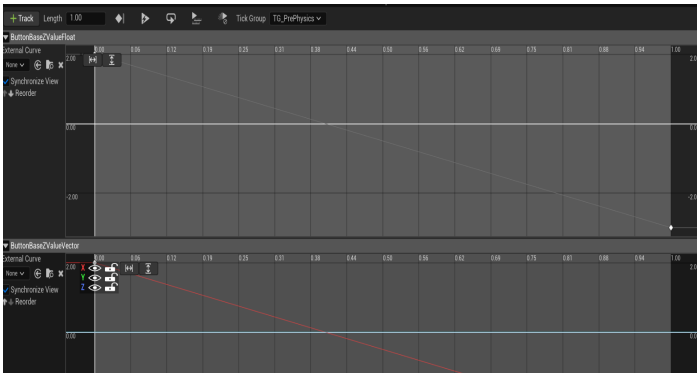
Iterates through the PlatformArray and triggers the ActivateMovingPlatforms event on all linked platforms.



Event: Actor End Overlap

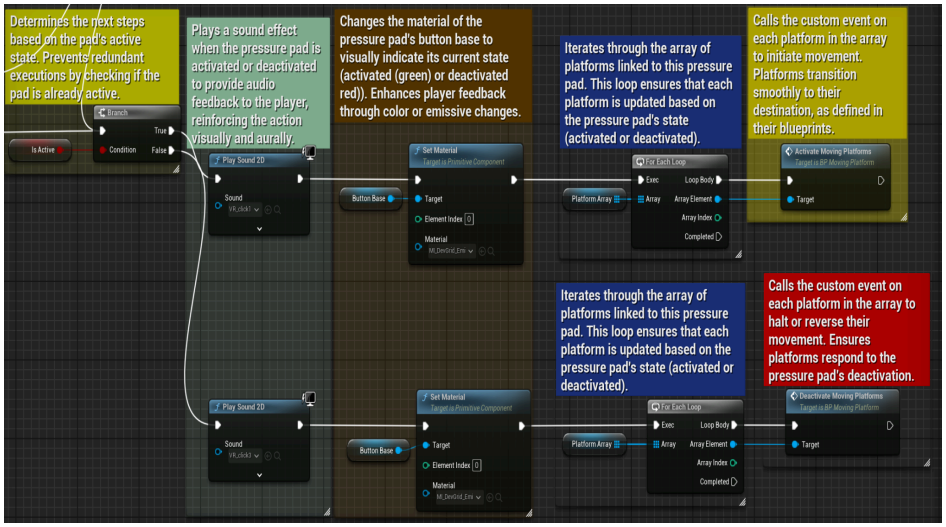


Detects when the actor leaves the pressure pad and sets IsActive to False.

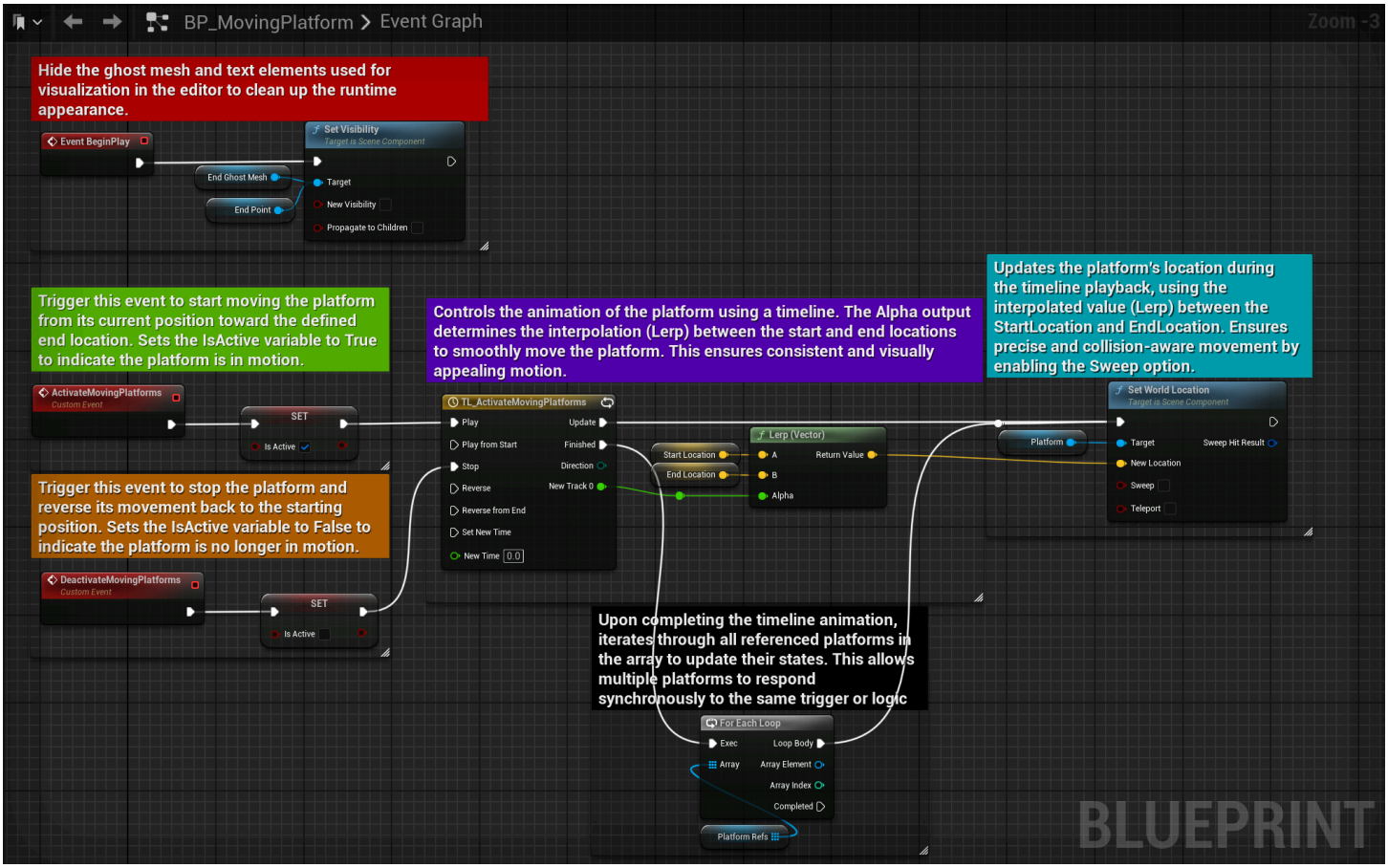


Activates the Timeline in reverse to visually reset the button.

Iterates through the PlatformArray and triggers the DeactivateMovingPlatforms event on all linked platforms.

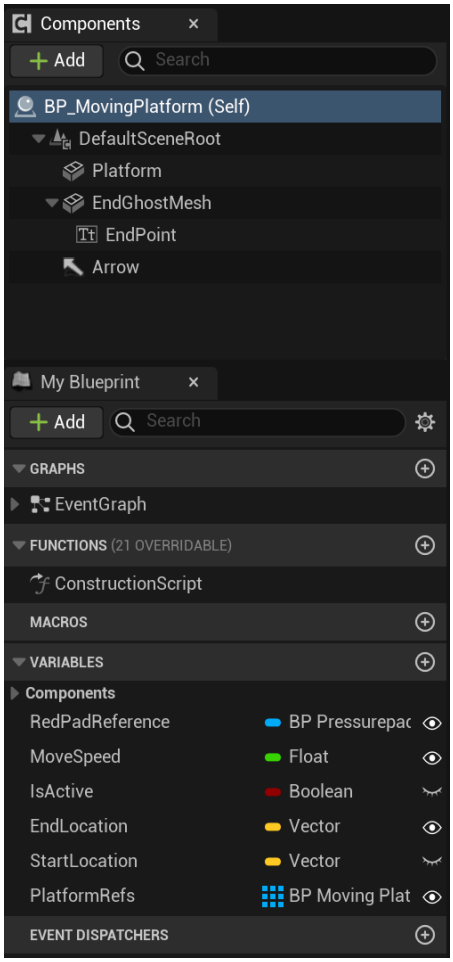


2. Moving Platform Blueprint (BP_MovingPlatform)



The BP_MovingPlatform blueprint manages the seamless movement of platforms between their designated start and end points, activating only when triggered. This system ensures fluid, precise motion, enhancing gameplay mechanics by integrating dynamic environmental changes. Its customizable design allows level designers to easily define start and end locations directly in the editor, making it adaptable for various gameplay scenarios, from simple transitions to intricate puzzles and challenges.

Key Variables:



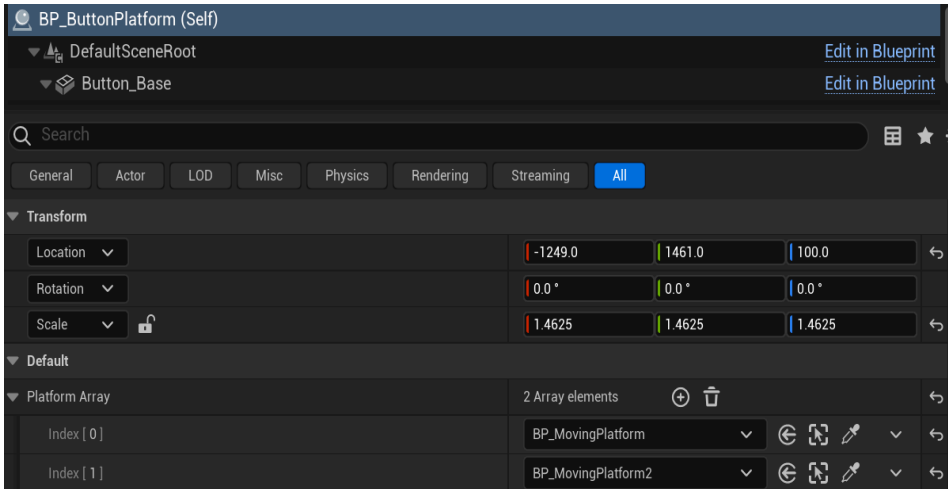
PlatformRefs: Stores references to the platform instances.

StartLocation and EndLocation: Define the movement range, set in the editor with ghost meshes.

IsActive: Boolean indicating whether the platform is currently moving.

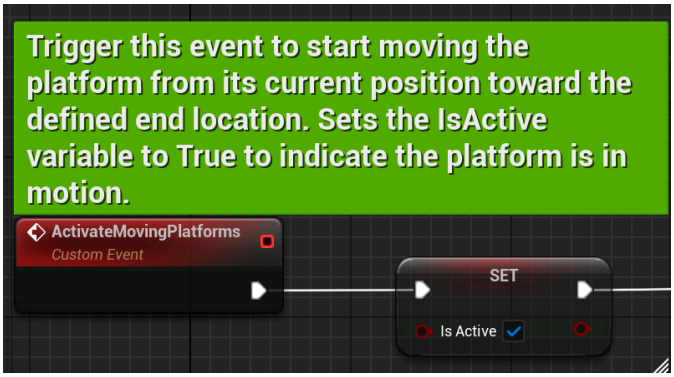
REMINDER!

Make sure to assign the desired moving platforms to the button by selecting them in the button's Details panel.



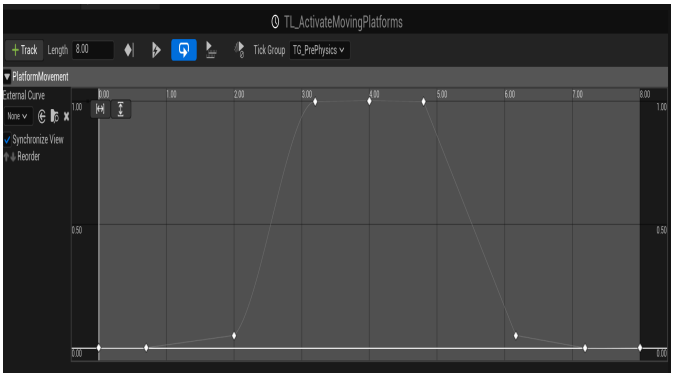
Logic:

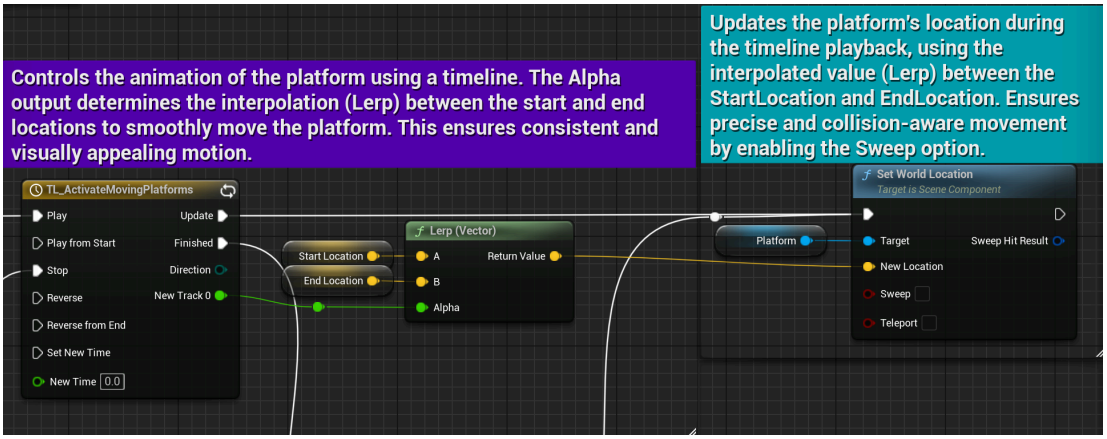
Event: Activate Moving Platforms

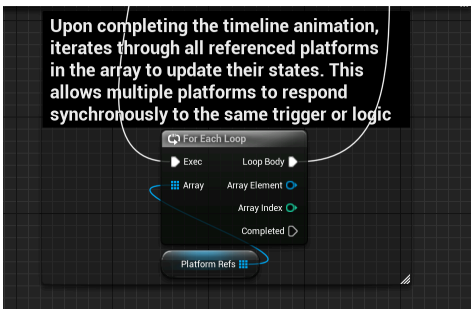


Sets IsActive to True.

Starts the Timeline TL_ActivateMovingPlatforms, which interpolates the platform's location from StartLocation to EndLocation using a Lerp (linear interpolation).





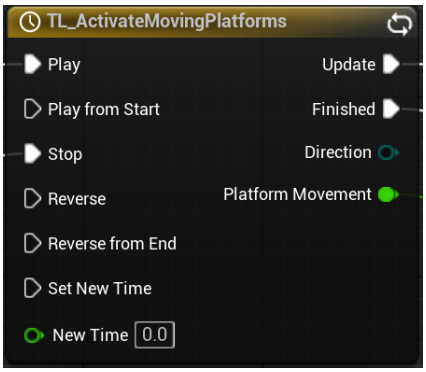


Ensures all linked platforms update correctly using a ForEachLoop.

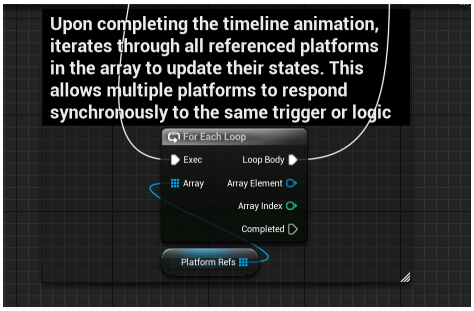
Event: Deactivate Moving Platforms

Sets IsActive to False.

Plays the Timeline to stop to stop the platforms where they are



.



Iterates through all platform references to synchronize their states.

Challenges and Solutions

Implementing the BP_PressurePad_Platform and BP_MovingPlatform systems in Unreal Engine presented several challenges during development. Each challenge required careful problem-solving to ensure the system was robust, modular, and intuitive for both players and designers.

1. **Platforms Moving to Unexpected Locations (e.g., 0,0,0)**

   Issue: Platforms would often reset to the world origin (0,0,0) when their start or end locations were not properly set in the editor.

   Solution: To ensure proper initialization of the platform's movement, logic was added to the Construction Script to verify and set the StartLocation and EndLocation during blueprint setup. A visual ghost mesh for the EndLocation was included, allowing designers to easily manipulate and define the platform's path directly within the editor. Additionally, error-checking mechanisms were implemented to display warnings if the EndLocation was not initialized, providing clear guidance to designers and preventing unintended behavior during gameplay.

2. **Activating Multiple Platforms Simultaneously**

   Issue: The pressure pad initially struggled to trigger more than one platform reliably, especially in complex puzzle setups.

   Solution: To manage the activation of multiple platforms, an array system (PlatformArray) was integrated into the pressure pad blueprint to store references to all linked platforms. A ForEachLoop was implemented to iterate through this array, triggering the ActivateMovingPlatforms or DeactivateMovingPlatforms events for each referenced platform, ensuring synchronized functionality. The editor setup process was further enhanced by allowing designers to easily populate the array directly in the Details panel, streamlining the configuration of platforms and improving usability.

3. **Platforms Continuing Movement After Deactivation**

   Issue: Platforms sometimes kept moving after the pressure pad was deactivated, leading to unexpected behavior.

   Solution: A boolean variable, IsActive, was introduced in the platform blueprint to track whether the platform should currently be moving. The timeline nodes were updated to check the IsActive state, ensuring the platform stopped immediately when deactivated. Additionally, precise timeline reverse play was implemented to smoothly return the platform to its original position, providing seamless and polished movement behavior.

4. **Lack of Clear Feedback for Player Interactions**

   Issue: Players were initially unsure if their interaction with the pressure pad had triggered a result.

   Solution: Dynamic material changes were added to the pressure pad's ButtonBase to visually indicate its activation and deactivation states, ensuring players could easily identify when the pad was engaged. Audio cues were incorporated for both states, with a "click" sound signaling activation and a "reset" sound marking deactivation, enhancing the player's sensory feedback. Additionally, a Timeline was used to animate the pressure pad as it depresses and resets, providing a tactile and visually engaging confirmation of the interaction.

5. **Debugging Overlap Issues with Actors**

   Issue: The pressure pad would sometimes activate or deactivate unintentionally due to collision overlap inconsistencies.

   Solution: Collision checks were refined to cast specifically to valid actor types, such as the player (BP_Scrapper) or designated crates (BP_MovableCrate_Red), ensuring only intended interactions triggered the pressure pad. The collision setup was further improved to prevent activation by unintended objects, such as stray physics actors or environmental debris, maintaining the integrity of gameplay interactions. Debugging tools like Print String were utilized extensively during testing to identify and resolve overlap issues, ensuring reliable and consistent functionality.

6. **Iterative Design Delays Due to Final Assets**

   Issue: Early development stages were slowed by the lack of final assets for platforms and pressure pads

   Solution: Block meshes were used during the prototyping phase to prioritize functionality over aesthetics, enabling rapid iteration and testing of interactions without the constraints of final designs. This approach allowed designers to focus on refining the mechanics and ensuring reliable functionality. Once the system was finalized, block meshes could be seamlessly replaced with detailed assets, streamlining the development process while maintaining high-quality results.

7. **Crate Integration and Compatibility with Pressure Pads**

   Issue: Ensuring crates (BP_MovableCrate_Red) worked seamlessly with the pressure pad, especially during respawn scenarios.

   Solution: Logic was added to the crate blueprint to detect when it fell below a defined kill floor, triggering a teleportation back to its designated respawn location. Compatibility with pressure pads was ensured by casting and verifying actor types during overlap events, guaranteeing seamless interaction. Additionally, the crate's physics and velocity were reset upon respawn to prevent unintended behaviors such as sliding or floating, maintaining consistent and reliable functionality throughout gameplay.

These challenges and their solutions highlight the iterative process of refining game mechanics, demonstrating the importance of problem-solving and testing to create polished, engaging gameplay.

General Lessons Learned

Effective debugging tools, such as Print String and visual aids like ghost meshes, proved invaluable for identifying and resolving setup errors during development, ensuring smooth progress. Adopting a modular design approach for Blueprints allowed for reusable and scalable systems, saving time and effort, particularly in levels with multiple pressure pads and platforms. Incorporating visual and auditory feedback significantly enhanced player understanding and engagement, creating more intuitive and enjoyable interactions. Additionally, prioritizing editor usability by making setup parameters easily configurable minimized errors and streamlined the level design process, contributing to a more efficient and user-friendly workflow.

Final Result

The completed Pressure Pad and Moving Platforms System is a fully functional and modular solution designed to create dynamic environmental interactions in Unreal Engine. Combining robust logic, clear player feedback, and flexible design, the system enhances both gameplay and level creation by providing a polished and adaptable framework.

This system features interactive pressure pads that react dynamically to valid actors, such as the player or crates, stepping onto or leaving the pad. The pads offer immediate visual and auditory feedback through dynamic material changes, smooth animations, and sound effects. These elements not only create a more engaging experience for players but also ensure clarity and responsiveness during gameplay. The pressure pads are fully modular and reusable, making them a versatile tool for various levels and scenarios.

The moving platforms are designed with dynamic movement, allowing for smooth transitions between customizable start and end locations. These positions are easily defined in the editor using ghost meshes, which provide a clear visual aid for level designers. Configurable speeds and states add further flexibility, enabling the platforms to adapt to diverse gameplay needs. The synchronization between pressure pads and moving platforms, managed through an array in the pressure pad blueprint, ensures seamless functionality and efficient setup.

The system also integrates crates effectively. These crates interact seamlessly with the pressure pad, triggering platform activation as needed. When a crate falls below a defined kill floor, it teleports back to a designated respawn location, with its physics and velocity reset to ensure consistent behavior. This integration adds depth to gameplay mechanics, making the system ideal for puzzles and interactive challenges.

Testing demonstrated the system's reliability and versatility. Pressure pads successfully activated and deactivated linked platforms with no delays or unintended behavior. Crates worked flawlessly with the pressure pad, resetting properly when necessary and maintaining expected functionality. The modular setup allows designers to customize and expand the system for various gameplay mechanics, from intricate puzzles to platforming challenges.

In the editor, designers can easily configure pressure pads and platforms through the Details panel, with ghost meshes simplifying level design by visualizing platform paths. Additional functionality, such as crate respawns or linked puzzles, can be implemented effortlessly, making the system highly adaptable for different project needs.

The final result offers a polished, engaging, and intuitive gameplay experience while maintaining flexibility for designers. With its modular design and robust functionality, the pressure pad and moving platforms system is a valuable addition to any Unreal Engine project, serving as a foundation for diverse gameplay scenarios. Additional guides and assets will further expand this system's potential, providing even greater versatility for game developers.