Horse Show Jumper Simulator Report

By:

Acacia Williams

December 8, 2025

CART 415, Game Studio #1 Fall 2025

Prof. J. Lessard

Situational Game Design Breakdown

This project is a small 3D show-jumping game built in Unreal Engine 5.6, where the player rides a horse around an arena and tries to complete a course of jumps in the correct order and as cleanly as possible. This project grew directly out of my own history with horses and show jumping. I've been riding since I was six and competed in hunters and jumpers. Instead of trying to simulate every part of riding (grooming, feeding, stable management, etc.), I deliberately concentrated on a very specific moment jumpers equestrians know well: the specific feeling I was chasing: the rhythm and adrenaline of a jumper round: with the primary goal of accomplishing a fast round time with as little faults, which in reality involves factors like counting strides, committing to a line, taking calculated risks, and knowing that one bad decision can mean a pole, a refusal, or even a fall. The game tries to capture that same edge between mastery and mistake, but in a safe, replayable way.

The player controls the horse character from a third-person view. Movement and steering are handled with WASD-style input, shift to sprint, and jumping is mapped to the space bar, with the option to trigger jumps using the left mouse button as well for players who find that more comfortable. The course itself is built

out of modular jump blueprints containing meshes I modeled in Blender. Each jump blueprint contains the two standards, the pole, and two invisible colliders: a trigger on the ground before the fence that detects approach, and a trigger in the air behind the fence that detects a successful clearance, with the condition the player hits the two triggers in a second, and only then counts the jump as cleared. The physical pole is its own blueprint that can simulate physics and fall when hit, and it notifies the jump when it has been knocked. This supports a simple but recognizable scoring model: clears versus faults, which is displayed on the HUD during the round with time.

Above the jump, a Niagara "waypoint" effect appears to show which fence is currently active. The game mode tracks the overall run: which jump index is current, how many poles have been knocked, whether the run is active, and the elapsed time from the first correct jump to the last. This lets the system enforce correct order (and notify "Wrong jump order!" when needed), handle start and end of the round, and eventually display a summary of the player's performance, where the player can add their name to the score board.

Around this core loop sits a simple UI flow: a main menu with a Start button and access to a scoreboard, and a game-over screen where the player sees their score, can enter their name, and save it. Pressing "Save" returns them to the main menu so they can either start a new round or review scores. The intention is to keep the whole experience lightweight and replayable, more like a small training tool or a prototype for a larger equestrian game rather than a full simulation.

Playtest

Observation 1: Jumps felt too close together

Observation: Several playtesters commented that the jumps were crammed too close, making it hard to steer and line up properly, especially at speed.

Design analysis: The original layout was designed more like a tech test: "can the logic handle multiple jumps in order?" rather than a comfortable learning curve. In practice, riders needed a bit more space between fences to adjust their line and correct mistakes. When jumps are too close, the pacing feels frantic in a bad way, and it's easy to blame the controls rather than your own timing.

Modifications: I spaced the jumps further apart and gave the player longer approaches. The course now reads more like a simple training pattern with clear

straight lines and gentle turns instead of a tight obstacle maze. This also works better with the fixed chase camera, since players have more time to see what's coming and adjust their trajectory.

Observation 2: Camera angle made steering frustrating

Observation: Players reported that they were constantly fighting the camera: they had to keep adjusting it just to see where they were going. Some actually lost track of jumps because they were busy wrestling the view instead of riding.

Design analysis: A fully free third-person camera gives flexibility, but it increases cognitive load. For a game that's really about rhythm and timing, constantly managing camera yaw and pitch gets in the way. What players wanted was a camera that simply followed the horse and let them focus on the course.

Modifications: I reworked the horse's camera setup so the camera is essentially locked behind the horse. The character now rotates to face its movement direction, and the spring arm uses a fixed relative rotation instead of letting controller yaw spin it 360 degrees. I also removed the mouse "look" inputs that rotated the controller. The result is a straightforward chase camera that stays behind and slightly above the horse, with optional lag for smoothness. Steering immediately felt more controllable and less disorienting.

Observation 3: Players could sometimes "score" a jump without really jumping it

Observation: Some testers managed to trigger a successful jump without properly clearing the fence, for example by clipping around the side but still hitting the before/after triggers within a second.

Design analysis: The logical scoring system was too generous. It only cared that the player touched the two trigger zones in time, not how they travelled between them. Because the pole itself wasn't tied tightly into the scoring, you could exploit the collision layout and get a "clean" result that didn't match what visually happened.

Modifications: I adjusted the positions of the two trigger volumes so they tightly frame the intended jump line, and refined the one-second timing to better match a realistic takeoff/landing window. On top of that, the pole knock system feeds into the scoring: if the pole reports that it has been hit and knocked, the jump can record a fault even if the triggers were technically satisfied. This greatly reduces the "fake clear" feeling and makes the results match what players see.

Observation 4: Poles didn't fall when hit

Observation: As I was still working out the collision for the playtest, early the pole jumps were like decorations since they couldn't be knocked over and a lot of players

got stuck at jumps, affecting game flow. The horse could clip through without any physical reaction, which broke immersion for players.

Design analysis: In show jumping, poles falling is the core feedback for a mistake. If the pole doesn't move, there's no emotional punch to knocking it, and the arena feels lifeless. The problem was that the pole mesh wasn't completely set up as a proper physics body with collision and hit events.

Modifications: I gave the pole a proper collision shape and turned on physics simulation and gravity. When the horse collides with it, the pole now responds physically and sends a notification to the jump, which in turn increments a "poles knocked" counter. This not only looks better but also directly supports the scoring system and reinforces the risk-reward of aiming tight to the fence, as well as enabling players to fully immerse.

Observation 5: Start/stop glitch after a round

Observation: Players could play one round, get to the game-over screen, save their name, and return to the main menu, but after that the menu stopped responding. The only way to play again was to stop and replay the level in the editor.

Design analysis. This wasn't an obvious "visual" bug, but a flow and state problem. The UI looked fine, but input wasn't reaching the buttons or the horse. Under the hood the game was still stuck in a menu-style input mode and/or paused state after returning to the main menu, and the new widgets weren't created with a proper owning player. In other words, the game thought it was still in "UI only / paused" mode even though the player expected a fresh start.

Modifications: I made level reloads more deliberate and reset input whenever the menu is shown or hidden. After saving on the game-over screen, the game now reloads the main level and, on BeginPlay, explicitly unpauses, recreates the main menu widget with the correct owning player, sets the input mode to UI only with the menu focused, and shows the mouse cursor. When the player presses Start, the menu switches input back to "game only," hides the cursor, and removes itself. This cleaned up the start/stop glitch and made the restart flow reliable.

Personal Narrative

I've ridden horses since I was six years old and competed in hunters and jumpers. I chose to make a jumper simulator because I love the thrill of jumpers: riding against the clock, balancing speed with precision, and trying to leave all the poles up. In real shows, one of my favorite parts is watching other riders do their rounds. You learn so much from that: how many strides they take between fences, where they risk a tight turn and clip a pole, or where they choose to play it safe. I wanted the game to capture that feeling of planning and executing a course.

Some of my proudest riding memories aren't just ribbons; they're the moments where a horse and I finally "clicked" after months of work. My favorite horse was a mare named Kisses, show name Bravissima. She came from Thunderbird Show Park in Langley, B.C., and she needed a lot of retraining. With patience, persistence, and some specialized tack, my coach and I worked through her challenges. Eventually, I was showing her in the jumper ring, cantering to fences with total trust, feeling completely in sync. Kisses helped me build a lot of confidence. My coach had an incredible eye for stride count, she could see exactly how many strides we needed on approach. She pushed me to ride forward, fast but

controlled, instead of staying in overly collected, “safe” strides which hunter jumping is known for.

Knowing how many strides to take to a jump, and when to collect or extend, becomes crucial as the fences go up. Some horses will simply refuse if the distance is wrong. Others, whether because of their nature, training, or trust in the rider, may “help you out” and launch from a deep or long spot. Those moments can save a round.... or cause a problem, which is why stride management matters so much.

Of course, things don’t always go to plan. Sometimes you commit to extending the stride and it just doesn’t work out. I’ve misjudged distances in the jumper ring more times than I’d like to admit, and I’ve hit the ground for it. Once at a show, on grass, a bad distance led to a fall and a concussion. This was before MIPS helmets were common, so the impact wasn’t as well absorbed. True to the “always get back on” mentality I’d been trained with, I insisted on riding my next round, even against the paramedics’ advice. When it was my turn to go in, I realized I couldn’t remember the course.

These highs and lows are part of what makes riding so compelling. From blue ribbons and soaring over oxers to miscalculations that leave you in the dirt, there is never a dull moment.

I also have fond memories of playing the jumper mini-game in The Legend of Zelda: Tears of the Kingdom. That experience reinforced how satisfying a well-designed jumping challenge can be, even in a virtual world. With my jumper simulator, my goal was to convey at least a slice of what goes through an equestrian's mind during a simple four-jump exercise: stride choices, lines, risks, and corrections. In real lessons, you might school the same pattern for an hour, refining it over and over. A game is a perfect space to explore those interactions safely and playfully. Overall, I enjoyed building this project, and I hope that anyone who plays it can feel a bit of the tension, focus, and joy that make show jumping so special to me.