Systems: Grapple Hook

Zach Phillips

2/7/19

Table of Contents

Table of Contents **2**

Intent Statement **3**

Research History **3**

Thesis **4**

Mechanic **4**

- Spinning the Grapple Hook 5
- Throwing the Grapple Hook 5
- Rope Physics 5
- Bringing the Grapple Hook back 5
- Attaching to a Hook 6
- Swinging on a Hook 6
- Releasing from a Hook 6

Game Video **7**

Post Mortem **7**

Annotated Bibliography **10**

Intent Statement

Through a physics based grapple hook system, using a controller joystick as the core game interface, I intend to create an experience which is intuitive, fun, and easy-to-learn, hard-to-master.

Research History

When I started this project, I wanted to think about how I could create something that was entirely skill based. I initially thought about making a rhythm based game, but as I thought more about it, I realized that it would, one, be incredibly hard to make it feel right, and two, would take more time than I had available to get it to a place I wanted it to be in. After realizing that, I focused on grappling hooks. Swinging in games had always fascinated me, and I wanted to create something that met my imagination. I looked at Spiderman games, specifically Spider-Man (PS4) and Spider-Man 2 (PS2) for their web-swinging mechanics, since both were praised for it. Through that research, I found my first source, a blog post from the creator of the Spider-Man 2 web-swinging system. That was my bread and butter for this project, and I looked over that page a lot.

For my other research, I was mainly looking for the physics behind how grappling hooks worked, and how to implement them into my game. I found a short webpage that contained physics equations for swinging off of ropes, and I decided to use that as a base for a lot of my calculations. After I found that, I decided to turn my focus over to the experience. I found two articles. The first was about how to add depth to mechanics, and the second talked about why players liked hard games. The first was useful since I was worried my mechanic would be too simple. While the article focused on puzzle design, I was able to use it to look at my mechanic from a new lens and think about the choices I was going to be making. The second article, designing games to be tough, helped me understand the audience I was making this system for. I've always enjoyed the *fuego* of a hard game, but I had never designed for one. And I certainly didn't want to have just myself in mind as my target audience.

I also knew that I wanted to make my grapple rope fun to play with. I didn't want it to be a single shot, jump to target type of system. I wanted my grapple rope to actually feel like a rope. Wrapped into that, I also knew I wanted to focus on making the

swinging feel fun as well. This mechanic I felt more lenient on in terms of realism, since I had some ideas already on how to make the swinging and jumping off of hooks fun.

Keeping all of this in mind, I felt ready to start working on my system.

Thesis

By creating a physics based grappling hook system which uses repeated joystick motion as an input, the player feels as if they have a tighter control over the way they move through the game, and thus feel able to master the system.

Mechanic

I decided to make this game in Unity and use a controller as my input. There are 7 main mechanics at work in the grappling hook system. I'm removing movement, jumping, and ladder climbing from this list, since I consider them to be mainly separate.

As a quick overview to them, however, my movement is controlled through through the left joystick. Jumping is triggered by the A button, and the player can only jump once before hitting the ground again. Basic movement is halved while in the air to account for air control. Lastly, ladders can be climbed after entering their collider and holding up on the left joystick. The player can also press the jump button to automatically climb the ladder while inside it's trigger.

Now, onto the main system at play. As I said before, I'm dividing it into 7 mechanics.

- Spinning the Grapple Hook
- Throwing the Grapple Hook
- Rope Physics
- Bringing the Grapple Hook back
- Attaching to a Hook
- Swinging on a Hook
- Releasing from a Hook

1) Spinning the Grapple Hook

Swinging is controlled through the right joystick. By tilting the joystick in a direction, the grapple hook adds force in that direction. The grapple hook is constricted in a circle around the player character, preventing it from flying away. By spinning the joystick in a circle, and at a constant speed, the player can build up momentum for the grapple hook. If the player stops spinning at any time, gravity takes over the grapple hook, and it will slowly come to rest at the bottom of the constricting circle.

2) Throwing the Grapple Hook

At any time, the player can press the Right Bumper button to release the Grapple Hook. When it is released, the constricting circle surrounding the player is removed, and the Grapple Hook gets a small boost in its current velocity vector. The Grapple Hook is immediately affected by gravity and any force that it comes in contact with, such as a wall. The player is unable to affect the Grapple Hook after it has left their grasp.

3) Rope Physics

After the Grapple Hook is thrown, a rope is created between the player and the Grapple Hook. After a set amount of distance, a GameObject, called a Rope Point, containing a RigidBody2D and BoxCollider2D is created at the Grapple Hook's position. It is then given half the velocity of the Grapple Hook and a becomes affected by gravity. This repeats each time the preset distance is exceeded from the Grapple Hook to the new Rope Point.
At the end of every frame, a Line Render is used to draw a line between the Player Character, Grapple Hook, and all existing Rope Points. This creates the rope that the player can see.

4) Bringing the Grapple Hook back

If the player presses the Right Bumper button while the Grapple Hook has been thrown, regardless of where it is or the state it contains, it returns to the player. This is done by removing all of the existing Rope Points and reenabling the Grapple Hook constricting circle around the Player. The Grapple Hook's physics then take over, and it moves back towards the player until it reenters the constraining circle. The System then returns to the beginning, and the player can spin and release the Grapple Hook normally again.

5) Attaching to a Hook

While the Grapple Hook has been thrown, in tandem with the Rope Physics mechanic, raycasts are sent between the Player Character, Grapple Hook, and all existing Rope Points. The raycasts follow the same path and angles as the Line Renderer, effectively making it so the Line Render can sense what it collides with. These raycasts react when they hit a Hook object, and nothing else. When a raycast hits a Hook object, regardless of where it hit on the rope, the Grapple Hook immediately repositions itself to be on top of the Hook object. All existing Rope Point objects are destroyed as well, creating a straight line between the Player and the Hook object.
Once the connection has been established, the player teleports to the halfway point between itself and the Hook object. This is to allow room for the player to swing without having to move forward. This will be explained in the next section. The Grapple Hook then activates a HingeJoint2D component that had been laying dormant. This allows the player to pivot on the Grapple Hook's rotation.

6) Swinging on a Hook

After the Player has been attached to a Hook, their movement mechanic changes slightly. Gravity takes over, and the player starts to swing back and forth, pivoting on the Hook's position. By tilting left and right on the left joystick, the player can now add force in that direction. This allows the player to control the speed that they swing on the Hook.

7) Releasing from a Hook

While swinging from a Hook, the Player can press the Right Bumper button to release from the Hook. This acts similar to releasing the Grapple Hook, but instead, the Player is the Grapple Hook in this scenario.
The Player receives a small boost in the direction of their current velocity vector, and the Grapple Hook enters its Return to Player state, which I called Bringing the Grapple Hook back. The Grapple Hook's HingeJoint2D component is deactivated, and Player movement returns to normal.

Below, I have my VDD for the Grapple Hook system.

Game Video

I created a video showing off this system. You can find it here: https://youtu.be/1yxolBIKXEM

Post Mortem

First, I want to talk about the project process. I used Rewired, a Unity Asset Store plugin, as my input manager. This allowed me to get more functionality out of my controller quicker and easier than I would normally be able to in a standard Unity project. This was ultimately a great decision, but I'm worried it had an effect on my inputs. I noticed that my controller wouldn't register sometimes, and other testers notes this as well. Unfortunately, I found out after testing, and after buying a new controller, that I was using an outdated version of the software. If I were to restart and reimport Rewired, I think that would solve a lot of the issues.

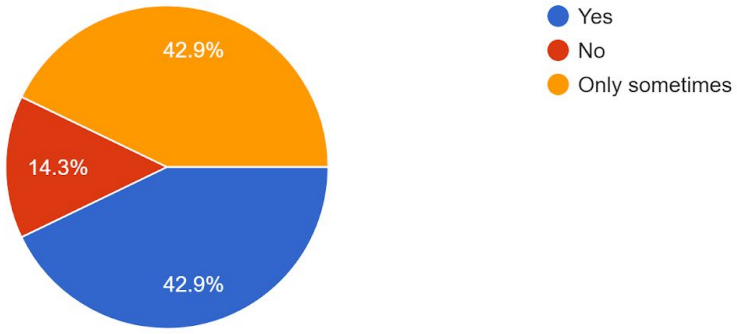
Now that that's out of the way, let's discuss the player experience. Unfortunately, I got everything working later than I liked, and as such only got 7 people to playtest my game. For playtesting, I had them play a short level. They were allowed to quit whenever they wanted. 3 completed the course entirely, with the rest having all but one person getting to the very end.

I wanted to focus on the Grapple Hook experience and how the players felt while using it. As such, the questions I asked were:

- Did you feel like you had control over where the grapple hook went?
- Do you feel, given more time and levels, you could master this system?
- What was your favorite part(s) about the demo?
- What was your least favorite part(s) about the demo?
- (Optional) Why did you choose those answers?

Did you feel like you had control over where the grapple hook went?

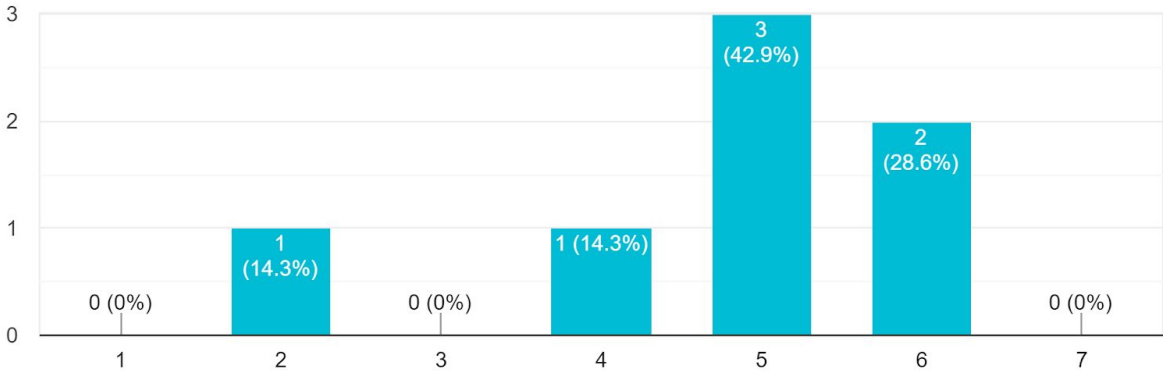
7 responses



This question was the biggest indicator of my success, in my opinion. I'm not happy with these results. To me, "Only sometimes" leans more to "No", since the player wasn't able to accurately use the Grapple Hook. However, the "Yes" answers were encouraging. This meant that at least ~43% of the testers felt like they were able to control their Grapple Hook.

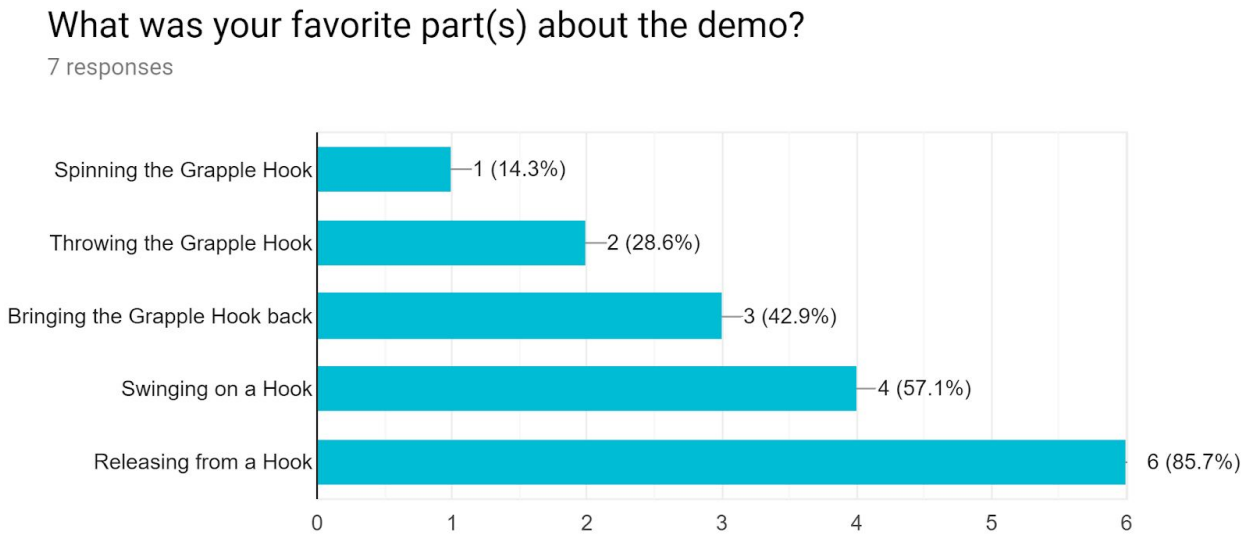
Do you feel like, given more time and levels, you could master this system?

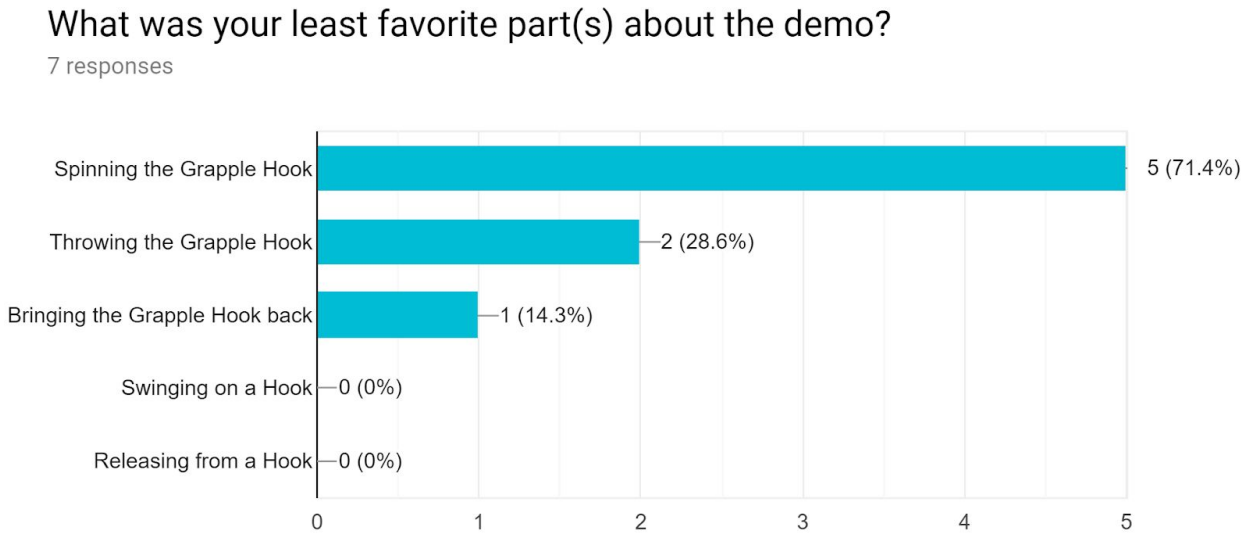
7 responses



The scale for this question is 1: "Absolutely Not" to 7: "I've already mastered it!". This question goes hand-in-hand with the previous question, and it's where I feel like my system has promise. I wanted to create a system that was easy-to-learn, but hard-to-master, and I feel like these results show that I have achieved something close

to that. Over 70% of the testers felt like they could eventually improve in this system and move towards mastery.





Next, I wanted to see what the testers thought of the independent mechanics present. This also exposed the strong and weak points of my system. From this data, I can tell that my Swinging and Releasing mechanic feels good to testers. This is a relief, since I saw that set of mechanics as the second most important set in the system. Unfortunately, my most important system, Swinging the Grapple Hook, felt horrible to testers. The next question I asked related to the testers answers. I won't be showing it,

only referencing their answers, since only 4 responded. Looking back on it, I should have made that question mandatory.

The reasoning behind 3 of their answers was the controller responsiveness. They said it felt laggy and buggy since it didn't register the button presses. The other response talked about how they liked swinging and jumping off of a Hook, but how they disliked when the Grapple Hook would sometimes get stuck when returning to the player.

Based off of all this data, I think that I hit a midpoint my intended experience. While a lot of the mechanics felt good when used together, my core mechanic of Swinging the Grapple Hook messed with the players and took them out of the experience. I think, given one or two more iteration cycles, I would be able to definitively meet my intended experience.

If I had to rate this as a Success/Failure, I would call it a Success, but just barely. If I could do this again, and what I will plan on doing for the next projects, I would start my prototyping in tandem with my research. That way I can get more time done on the prototype and get more iteration cycles in. I also want to test as I go. I only tested at the end, when I felt like I had enough of the system made in order to complete my play experience. If I had tested earlier, I might have been able to catch things like my buggy input system and my unsatisfactory Spinning mechanic.

Annotated Bibliography

Fristrom, Jamie. "Swinging Physics For Player Movement (As Seen In Spider-Man 2 And Energy Hook)". Game Development Envato Tuts+, 2013, https://gamedevelopment.tutsplus.com/tutorials/swinging-physics-for-player-movement-as-seen-in-spider-man-2-and-energy-hook--gamedev-8782.

This article by Jamie Fristrom talks about swinging physics. He has worked on games such as Spider-Man 2 and Energy Hook, which both feature fun and engaging swinging mechanics. The article goes into depth about implementing swinging movement and the various methods of doing so.

The important points in this article are:

- Two types of swinging, Physical Simulations or Canned Movement.
- I would be working towards Physical Simulations.
- Canned movement tends to slow down momentum.

- I can constrain movement through tethers. This makes the circular motion of the rope swing without using the physics based rope. Instead, all calculations are done on the character themselves.
- Code snippet inside of the article to help explain.
- To make it feel more realistic, add slack and springiness to the rope.
- Also, let the player have a lot of air control.

This article is one of the best informational pieces I've found. It's really going to help me think about how swinging should feel in the game. The article also contains diagrams and code snippets to help me understand the topics presented. Along with this, there are also ideas about systems to add to make the experience better for the player, even if it breaks physics as we know it.

Stout, Mike. "Evaluating Game Mechanics For Depth". Gamasutra.Com, 2010, http://gamasutra.com/view/feature/134273/evaluating_game_mechanics_for_depth.php.

This post by Mike Stout, former Insomniac designer, talks about how to see if your mechanics will hold up through playthroughs, or as Mike puts it, make sure you don't make "a design that looks great on paper doesn't turn out as well in practice as you'd hoped".

The important points in this post are:

- When testing look out for testers saying certain buzzwords such as "boring" or "repetitive".
- This means that the underlying mechanic isn't engaging, and no amount of visual polish or feedback will make the players overlook it.
- "Challenges [should] never stay the same long enough to be boring and yet they also don't change so fast that the player can't enjoy his mastery over the game."
- Meaningful skills need to be established first before anything else is added.
- "For example, a simple platforming challenge could be described in an Activity Statement as: "I want the player to jump up to that platform." In this case "jump up" is the meaningful skill and "that platform" is the objective."
- "By altering the Activity Statement during the design phase to more explicitly encompass the meaningful skill (and thereby altering the underlying mechanic), your whole design will get deeper and more satisfying."

When I first thought of the idea for my mechanic, I immediately started thinking of other things to add, such as different grapple heads or platforms. I fell right into the trap that

this article described. I didn't take enough time to really think about my main mechanic and how to make it stand on its own. I'm going to be using action statements from now on, and I'm going to try and resist bloating my mechanic with unnecessary fluff.

Sinha, Ravi. "Fun In Mastery: Why We Seek Tough Games". Gamingbolt.Com, 2018, https://gamingbolt.com/fun-in-mastery-why-we-seek-tough-games.

Ravi Sinha writes this article about why hard games are fun. Ravi brings in multiple examples and stories of famously hard games, such as Dark Souls, Celeste, and monster Hunter World. The article then continues to analyze why these games were so hard and rewarding, and why players keep coming back.

The important points in this article are:

- "These games engage the player, presenting unique challenges within the confines of their framework, and provide a stimulus that keeps one coming back. Therein lies the core concept of fun in mastery and why many of us seek tough games."
- The basic idea is to choose one mechanic that is small and easily used. From there, you can expand on the mechanic and think about how different systems could interact with it.
- "Sebastian Deterding, a user experience designer and researcher for gameful design, believes too much feedback without enough challenge can make things dull overtime.
- Players are looking for challenging games, and many want the experience of mastery.
-
- This article will help me get a better view of the people I'm making this game for. I want this System to be challenging but rewarding. I need this to help me understand how other games did it so well, and what I can take from them to make my own game better.

"Buzz Blog: Let Go, Tarzan!". Physicscentral.Com, 2012, http://www.physicscentral.com/buzz/blog/index.cfm?postid=12341073632472744 30.

This quick blog post analyzes the physics behind a rope swing. Specifically, it's looking at it through the movie Tarzan. While the topic isn't important to the mechanic I'm making, the blog post includes formulas, equations, and diagrams explaining the rope swing from a physics standpoint.

The important points in this blog are:

- The optimal swing should start below 45 degrees.
- Various physics equations.
- A link to a paper which goes more in depth in the math.
- Equations for how the release works and what the optimal angle for it is depending on the speed.

This blog post, while short, will be the basis for a lot of the coding in my system. I'm not planning on using Unity's default swing system, so references like this will help me create something that's realistic. Once I get it realistic, I can make it feel good.