ROBERT TITIRI

APPLIED GAME DESIGN

SOMBER FIGHTER

Combat System GDD

An Exploration of Player Agency &
Tactical Emergent Gameplay

Table of Contents

Introduction & Research Foundation

Problem Overview

Many third-person action games (i.e. beat 'em ups, hack n' slash games, etc) struggle to enable deep player agency within their core combat systems. Instead, they often rely on external elements such as build customization or level design to create variation, rather than embedding tactical choice directly into moment-to-moment gameplay. This results in combat that can feel repetitive or superficial, reducing strategic depth and player engagement.

Addressing this is important as player agency fosters a stronger sense of control, investment, and retention, which are key to the success and longevity of single-player action titles. This project aims to create a tactical combat system that empowers players to develop their own strategies and playstyles through deliberate decision-making and meaningful choices during combat, all within a minimalistic 3D environment that keeps focus on the core gameplay loop.

Design Goals

- Create a combat system that prioritizes player agency, allowing for diverse tactical approaches and personal playstyles.
- Facilitate strategic thinking through a balance of risk and reward on every action.
- Maintain a clean, minimalistic environment to avoid distracting from combat focus.
- Design enemy AI and combat mechanics that reward reading opponents and mastering how to abuse AI behaviours over memorizing combos.
- Enable emergent gameplay to create unique experiences for each player in how they tackle the same challenge.

Inspiration

This approach is inspired from online competitive games, primarily fighting games (e.g., *Tekken*, *Street Fighter*), known for their emergent strategic depth, as well as tactical action titles (e.g., *Sekiro*) that reward situational awareness but also enable direct offense. My emphasis on player driven strategies also takes a lot from modern trends in game design within the industry, valuing emergent gameplay over linear or defined solutions to gameplay challenges.

Research Summary

Industry Conventions to Player Agency

Player agency varies between Japanese and Western design philosophies. Japanese games often focus on clear, precise systems like stances or timing-based counters, supporting skilful counterplay but sometimes feeling predictable. Western games offer broader toolsets and metagame mechanics, encouraging experimentation and player freedom but sometimes

sacrificing depth. Examples include FromSoftware for Japanese design and Assassin's Creed for Western. Common design approaches include:

- Using counterplay systems like rock-paper-scissors for clarity, though risking predictability
- Emphasizing strong game feel and clear visual communication
- Encouraging player experimentation with multifaceted toolsets
- Supporting distinct playstyles to categorize player behaviour

These are general trends, not strict rules, and many games blend or break these conventions.

Genre Research

Analysis of 3D action games reveals common design patterns such as progression systems, scripted enemy behaviour, and complex environments. While these elements add depth and variety, they often overshadow pure combat skill, resulting in a diluted tactical experience. In contrast, fighting games focus on balanced move sets, equal screen presence, and simplified environments, placing player skill and tactical decision-making at the core of gameplay at the cost of confusing and convoluted inputs and control schemes. This difference highlights how fighting games emphasize immediate mastery, whereas action games blend combat with exploration and progression. This researched helped me understand what makes different combat systems appealing across the genres and guided me in how I should design my own.

Demographic Research & Categorisation

Throughout my research and experience with players of the genre, I was able to categorise the way players engage in action games into 4 categories. This distinction in players is important, as different people want different things within a combat system, and while this is not necessarily in service of creating a game that can appeal to everyone (since a game designed for everyone is a game designed for no one), it does help to understand what it is that people want when organically discovering their playstyle in a given game, something that can be designed around.

Core Combat Design

Initial Gameplay Loop Layouts

1st Iteration

The initial design incorporated a rock-paper-scissors (RPS) system layered with 3D movement, actions like jumping, grabs, and other context-sensitive manoeuvres were intended to offer ways of initiating your offense in creative ways, similar to a 2D fighting game like Street Fighter. However, the system proved overly complex, both for players to intuitively understand and for development, often introducing bugs and implementation challenges. As a result, the focus shifted toward expanding attack variety within a simpler framework, rather than relying on a wide array of unique combat actions.

2nd Iteration

The system later shifted to a more grounded style of combat, emphasizing a wide array of attacks with niche functions and finely tuned properties. This version relied heavily on situational awareness, with positioning playing a much more punishing and decisive role. While it offered deep tactical possibilities, it proved difficult for new players to grasp and was less appealing to those not interested in diving deep into its mechanics. Additionally, balancing such a nuanced system became increasingly time-consuming and complex. At this point I realised I was leaning too much onto fighting games as inspiration for the system, which led to the overly obtuse implementation of mechanics that I aimed to avoid from the beginning, so my next iteration focused on tweaking such issue.

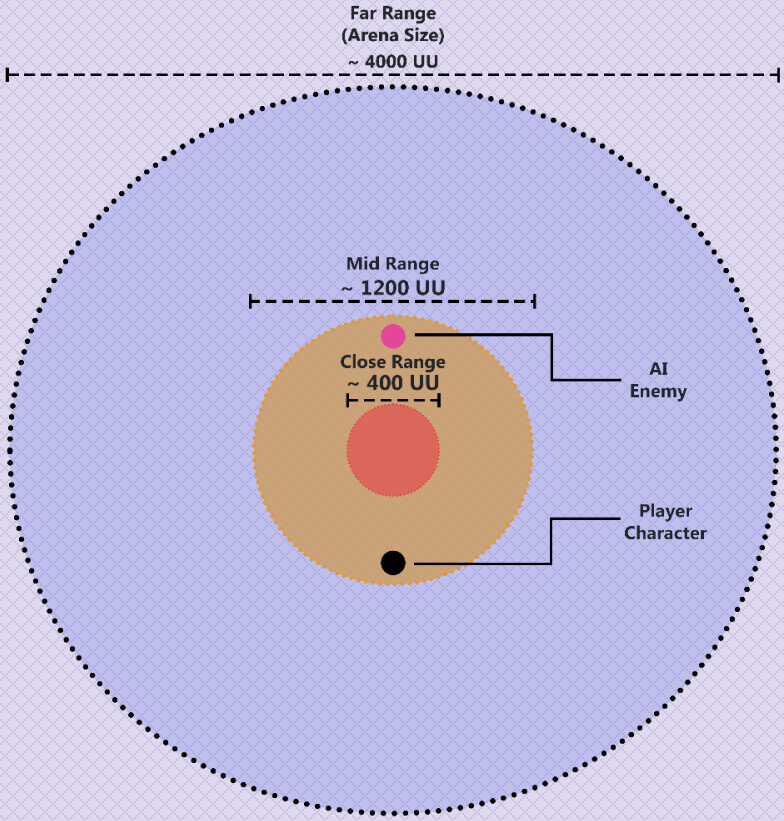
3rd Iteration

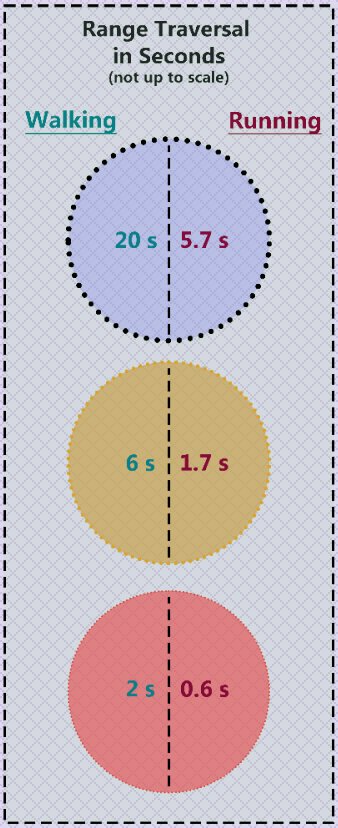
The combat system was greatly simplified. Attacks were made more general but given distinct, impactful properties, each with meaningful strengths and weaknesses. This added weight to player choices and reduced redundancy in the move set. Movement was also refined, with the gameplay loop shifting toward strike baiting and identifying gaps in the opponent's defense.

As offensive and defensive options deepened, it became clear during early testing that players didn't fully grasp the consequences of their actions. To address this, I exaggerated the strengths and weaknesses of combat actions to improve clarity and feedback. These changes preserved the original design constraints while enhancing the system's tactical depth and readability.

Combat Ranges & States

The combat system was designed around an array of different ranges and states that direct how the player engages the enemy. The goals and approach to combat, as well as the advantages, disadvantages, and sometimes moves available to either character depend on these states, which are dictated by the range dynamics and what situation the characters are in.



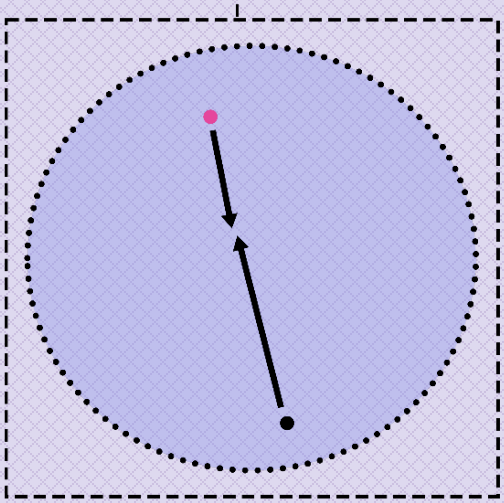


* Note: 1 UU = 1 Unreal engine Unit = 1 cm

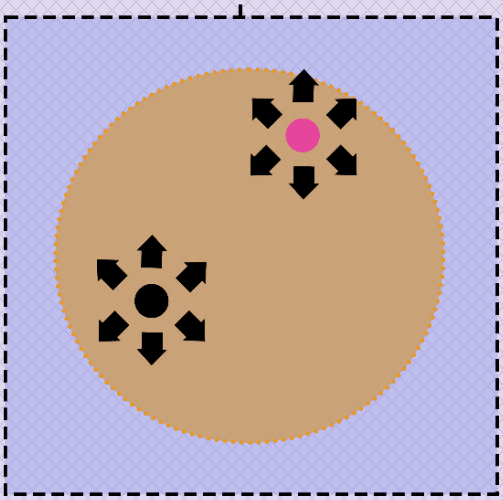
The ranges and how fast traversal is through these help balance the gameplay loop on a fundamental level, helping pace offense and defense, and moderate mobility, which often tends to be an extremely fun yet difficult to balance aspect of combat.

The various state in combat are defined here, with what options each character has at their disposal at each one, and what win conditions they have.

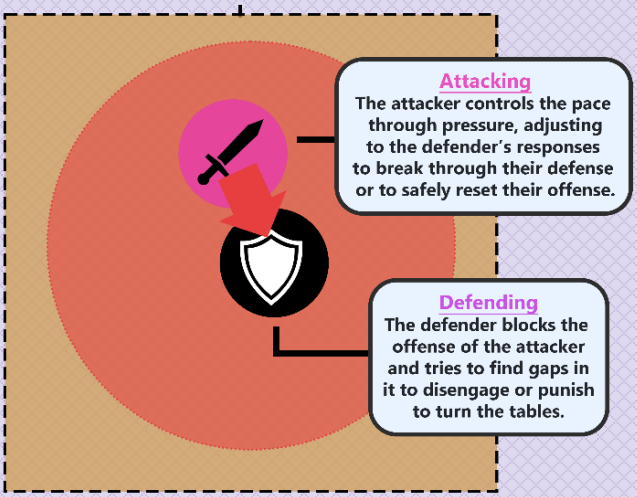
Out-Of-Contact
Opponents are too far away for any interaction. A transitionary state.



Neutral
Both opponents are nearby with no clear advantage, maneuvering for an opening or to stop the other from engaging.



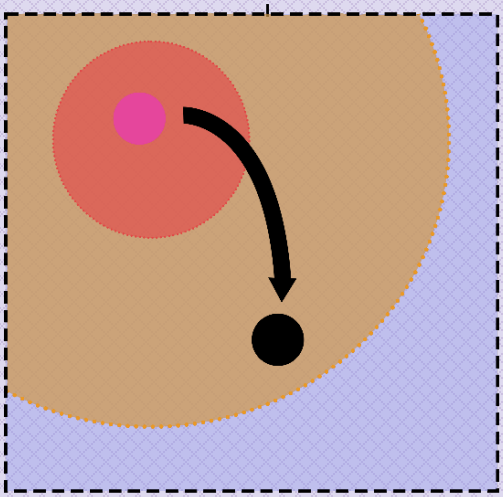
Engaged
One of the characters has engaged, closing the gap and starting their offense. There is an attacker and a defender.



Attacking
The attacker controls the pace through pressure, adjusting to the defender's responses to break through their defense or to safely reset their offense.

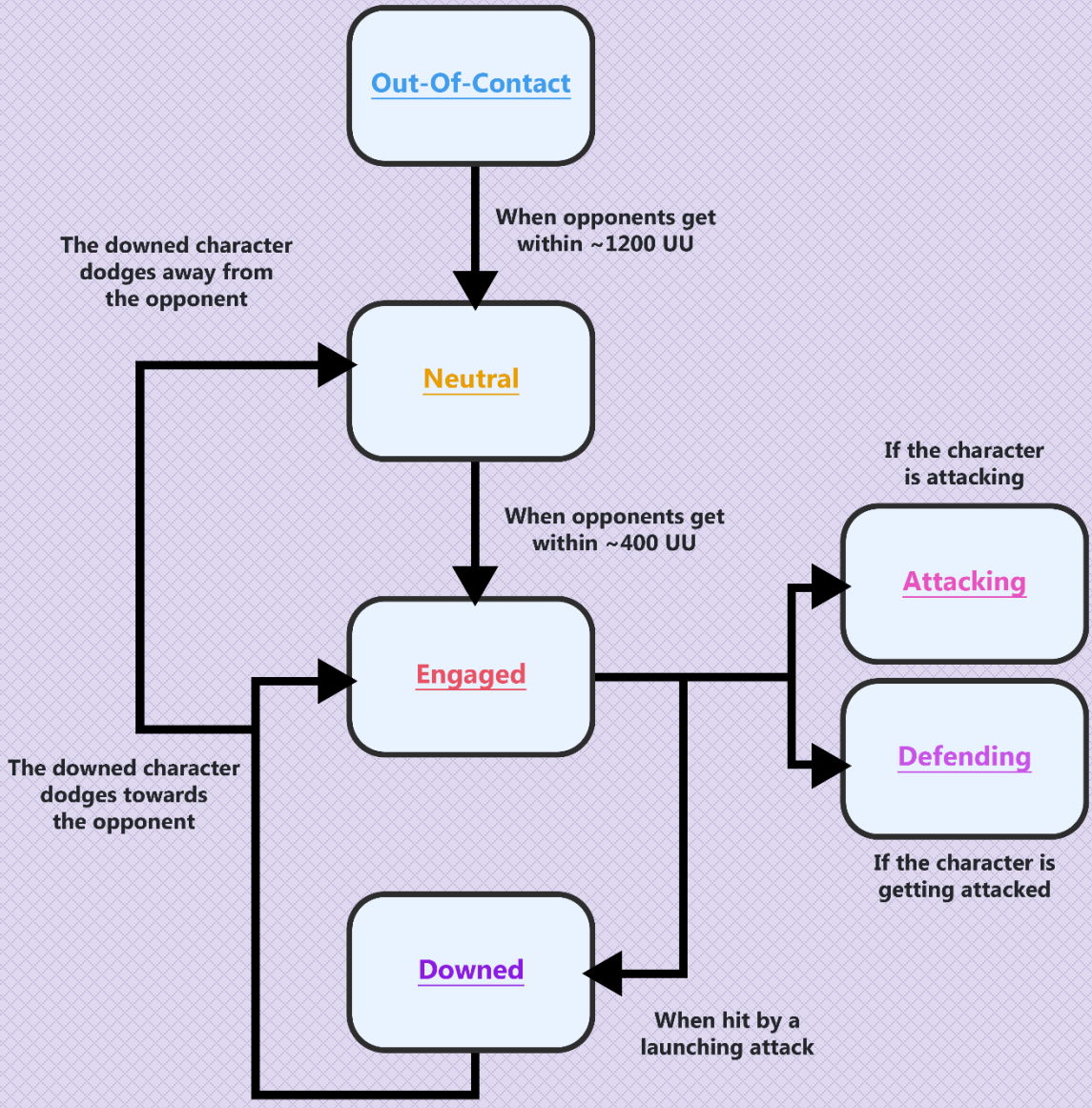
Defending
The defender blocks the offense of the attacker and tries to find gaps in it to disengage or punish to turn the tables.

Downed
After being launched by a launching attack, the character is knocked to mid-range and falls to the ground. They can recover by rolling in any cardinal direction.



The objective with this gameplay framework was to make a simple, yet interactive loop with players having plenty of ways on how to approach their corresponding gameplans. Combat systems can often end up encouraging non-engagement with characters not being forced to really interact directly much with each other to win, and that was one of the pitfalls I wanted to avoid.

For a more simplified breakdown, here is how the combat flow changes throughout each state.



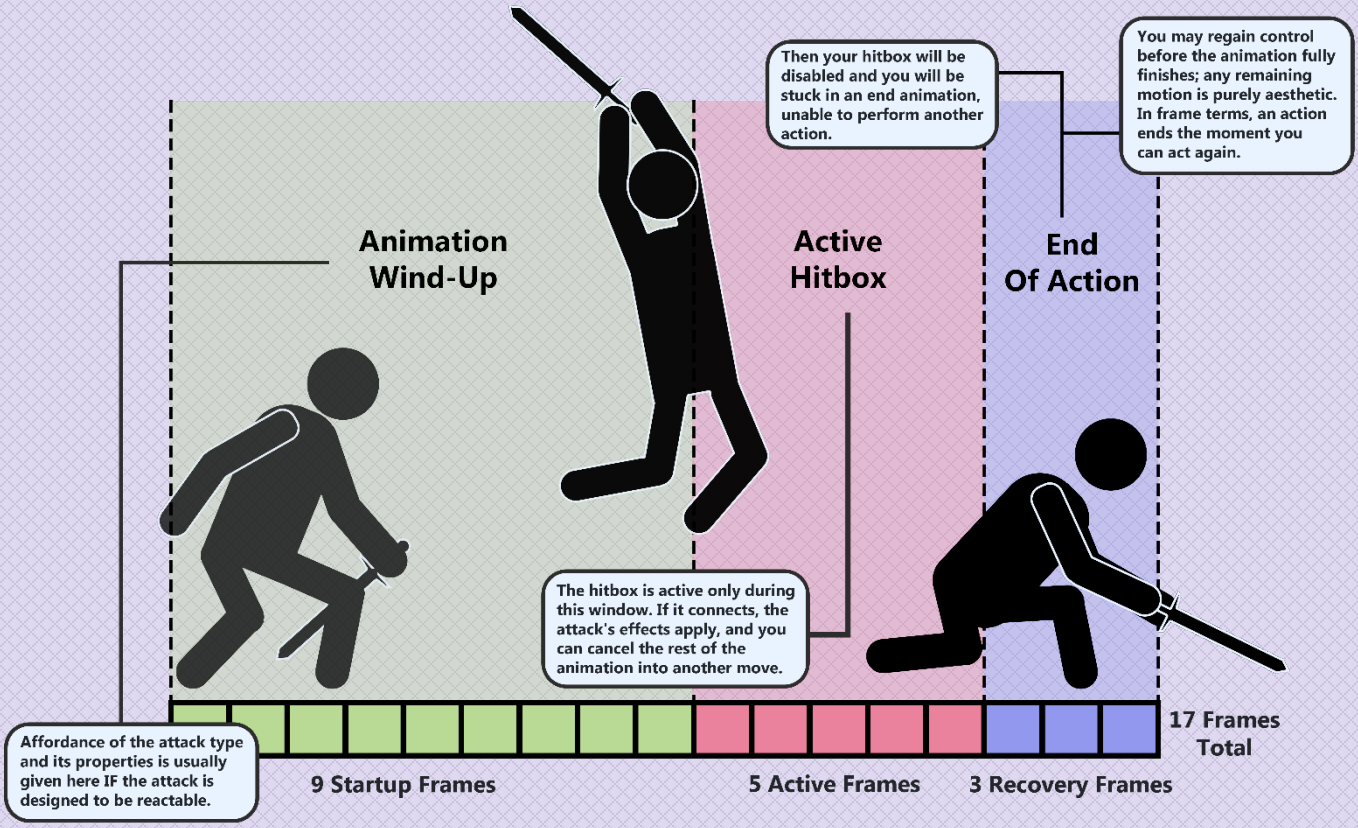
Attack Properties

Frame Data

For this project I thought the best approach to designing and implementing new actions and moves was through a more numerical lens. Often action game attacks tend to be balanced around more qualitative feedback, with more emphasis on what feels right and in iteration. While this can be a good approach for simple mechanical games with auto-combos that do not need to be timed, it brings issues for games that have a plethora of attacks with different properties that are not necessarily tied to a specific combo, like fighting games.

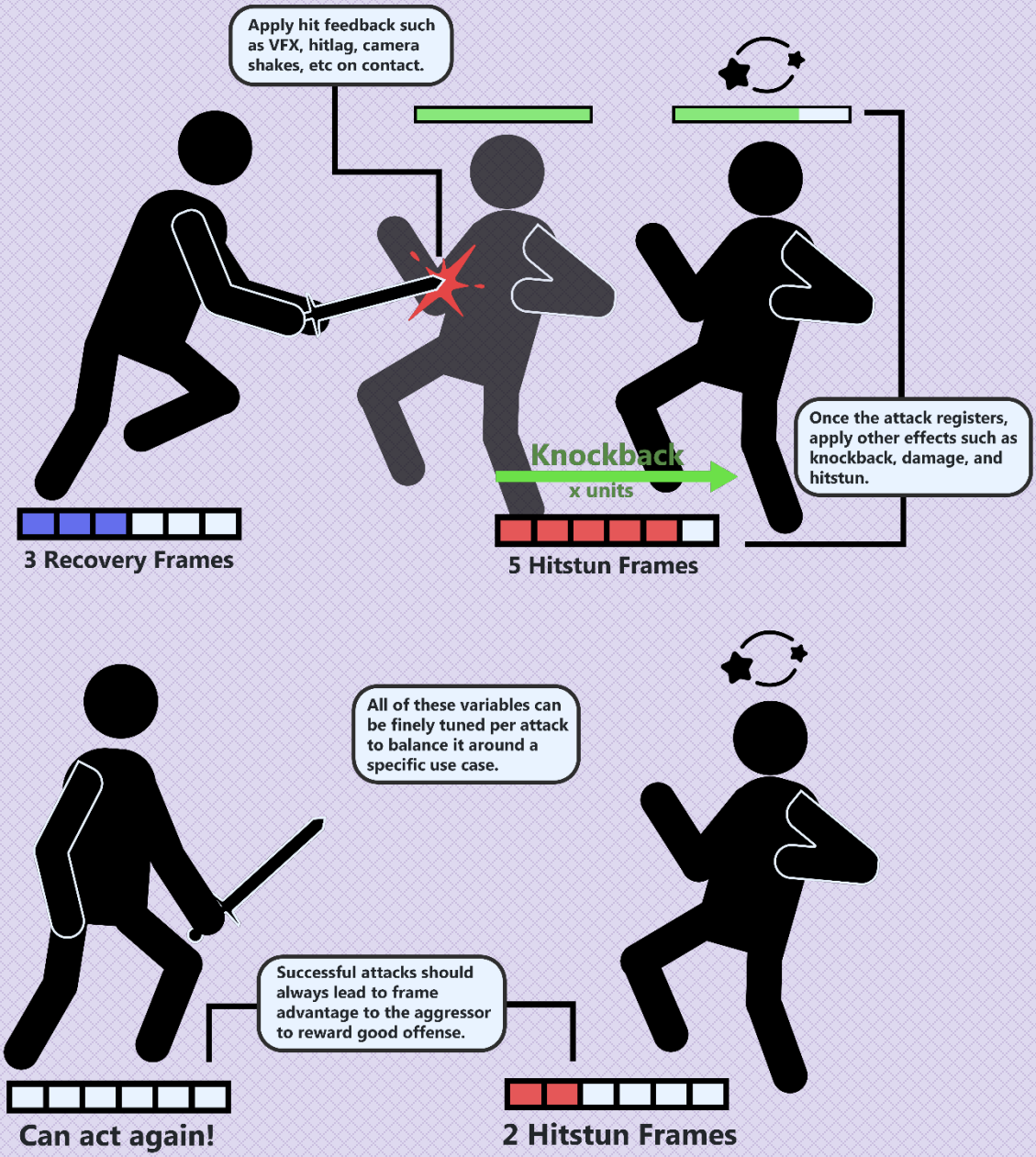
Fighting games use frame data as a way of breaking down actions into numbers that can be easily adjusted to create different effects. Fine tuning of these numbers can be less intuitive for a designer but can end up giving more control over action timings and balance.

Here is a generic action timelapse broken down into frame data.



Frame data does not only apply to the duration of the animation of a move, but can define hitbox durations, punish periods, windups, as well as when a character can act. The properties of an action can change contextually, such as with windows of invulnerability or animation cancellations, so defining properties numerically can help organise attacks and compare strengths and weaknesses amongst each other. This

balance heavy approach is necessary for a tactical game that takes such inspiration from competitive games.



Property Terminology

Here is a list of common terms that are going to be used when describing actions and attacks.

Startup (frames): The number of frames that it takes for an attack to start. If an attack has too few, or if the animation does not show very well if the action has started, the attack can be unreactable and must instead be predicted.

Active (frames): The number of frames that it takes for a hitbox to disappear. Even if initially not touched by the hitbox when created, if a character walks into it, hit properties will be applied, so attacks with high amounts of active frames have lingering hitboxes.

Recovery (frames): The amount of time in frames that it takes a character to recover from an action. During this period, the character will be vulnerable to attacks, so using attacks with lots of recovery is risky.

Hitstun: The amount of time, in frames, a character is stuck unable to act after being hit by an attack. During this period, the enemy will be able to reposition, or attack again to start a combo.

Blockstun: The same as hitstun, but for a character is blocking. Usually when blocking an attack, although unable to move, you will still be safe from future attacks.

Launcher/Launching attack: This is a property of certain moves. When hit by such an attack, the enemy will be downed, a special state you can only recover from by dodging in a direction.

Armour: An action with armour can ignore an x amount of hits without being put in hitstun or blockstun.

Pressure: Attacks thrown to open up the defense of an opponent. They might not directly hit the opponent, but they are advantageous still by helping you gain terrain or condition the enemy to expect a certain pattern.

Combat Actions

The following actions will be described in general terms of frames and other properties and might not reflect the end state of the attack exactly. The data shown are just default property values used for balance.

General Actions

Walking – the player will normally walk at 200 cm/s. Walking will be disabled during set actions such as attacking or blocking an attack.

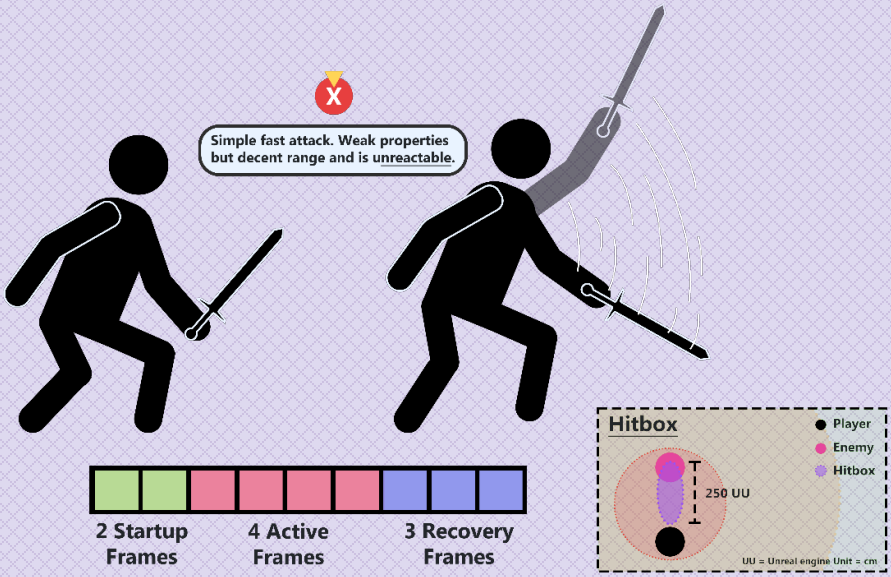
Looking – You can freely move the camera around the player if not locked. More information on the camera section.

Running – While holding a button, the player will now move at 700 cm/s. You cannot run while blocking. There is no obvious incentive to not run at all times since it does not take resources, but walking helps throw off enemy attack timings and bait certain actions.

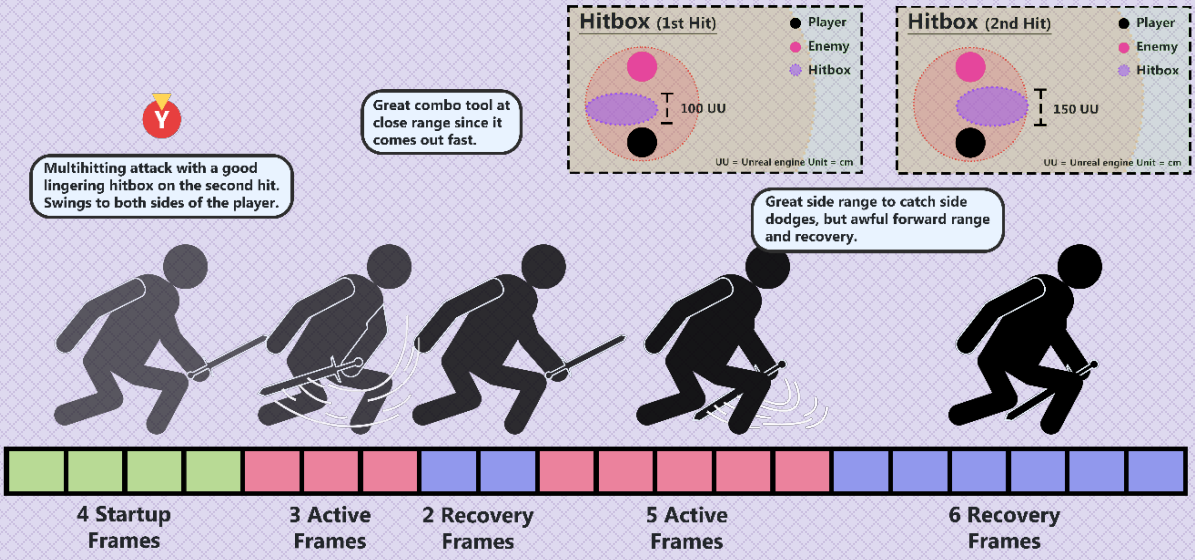
Lock-On – Locking-on will lock the camera at the enemy ‘s location, stopping the player from freely moving it. There is a cooldown between enabling and disabling it. Move information on the camera section.

Attacks

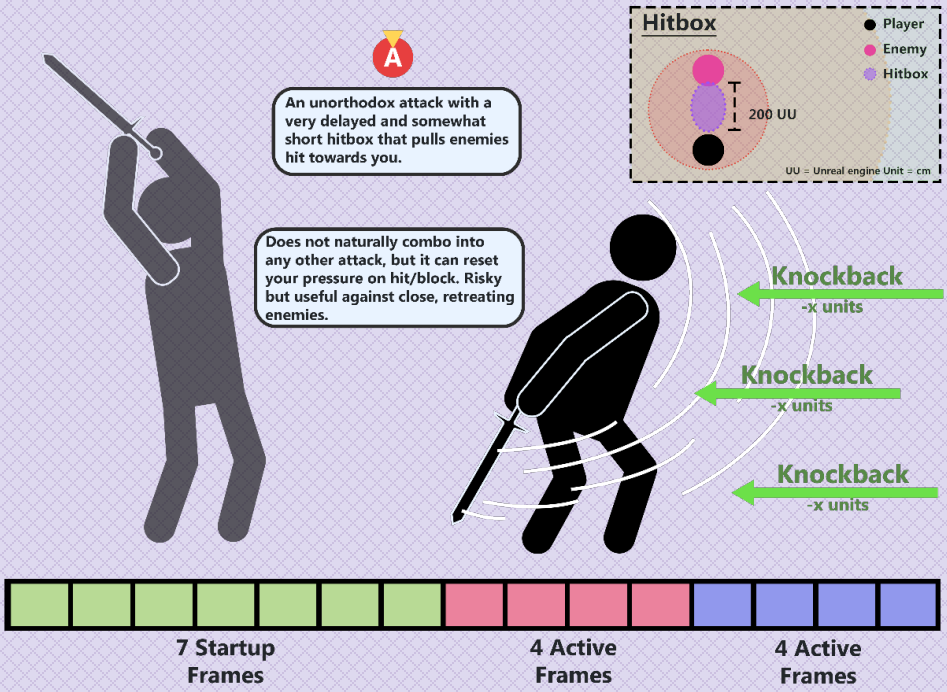
Light Attack 1 – A quick attack with decent range, but negligible hitstun and damage. It is a safe combo starter but can be inconsistent and unrewarding at longer range.



Medium Attack 1 – A multihitting medium attack with a great horizontal hitbox to catch side-rolls and a lot of active frames, making it useful for offense and defense. Its good damage and quick startup make it useful to extended combo at close range.

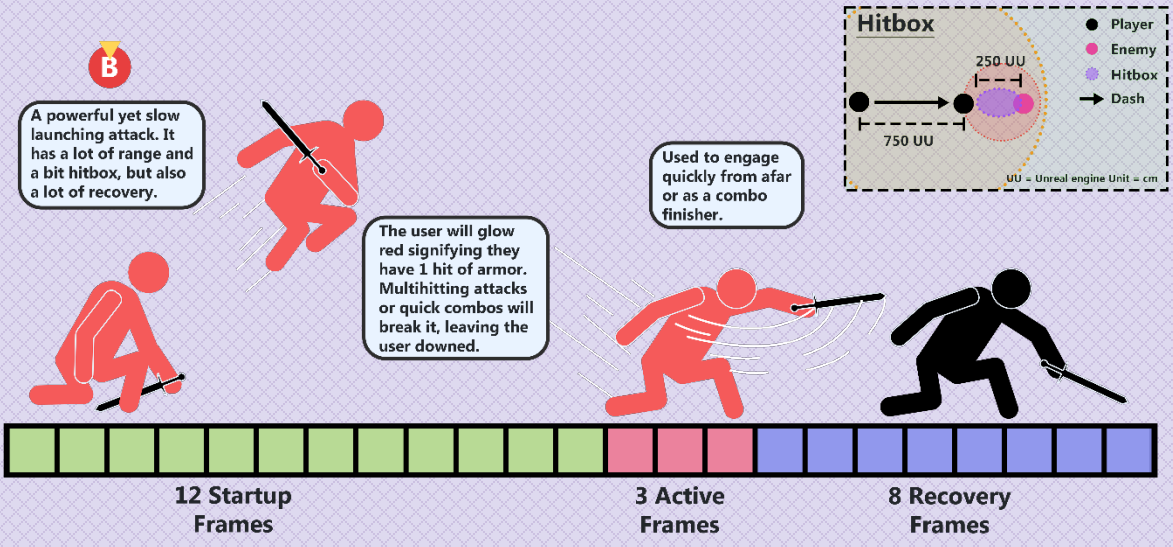


Special Attack* – Has a very slow startup and doesn't naturally combo into any other attack. It has negative knockback, meaning it pulls the enemy towards the player, so one can reset combos or pressure on hit with it.



*Note: This attack used to be called Medium Attack 2, but it was changed during the player moveset size rebalancing due to its unique properties, making it stand out on its own.

Heavy Attack 1 – A slow startup attack with great reach that shifts the player forward at quick speeds. On hit, it will deal great damage and launch the player, knocking them onto the ground, so it is good as a combo ender. Can tank 1 hit, but if hit twice, the user will be knocked down at close range, being left open for a punish.



Removed Attacks

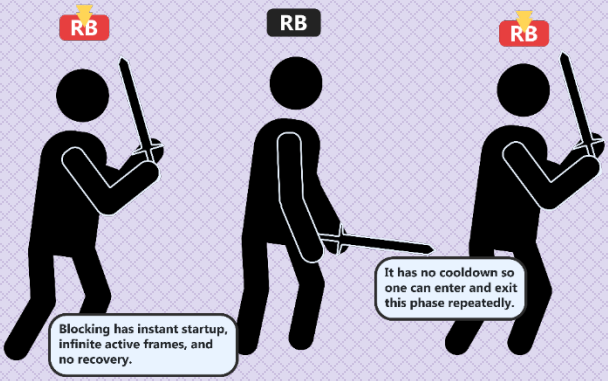
Originally there were 6 different attacks available to the player, but during balancing some got cut out as testers felt like the moveset was confusingly big, and some of these attacks had uses that were better fulfilled by other attacks.

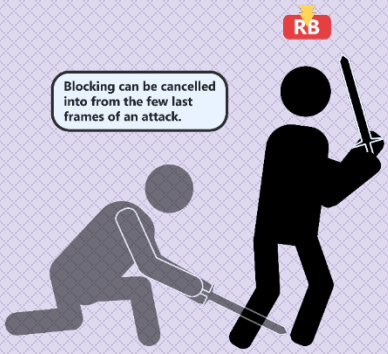
Light Attack 2 - Originally added as a faster alternative to Medium 1 for a multihitting attack, it ended up seeing little use. It was a more damaging Light 1 but much more situational, as its high knockback and limited range made it clunky to use. A niche application for this move was to be used as a Frame Trap (i.e. a move with a gap between hits far enough for the opponent to recover and act, baiting them into getting hit mid action), but this property was hard to communicate and was not very rewarding as enemies would rarely respond to this. Overall, the move lacked an identity and was not interesting to use.

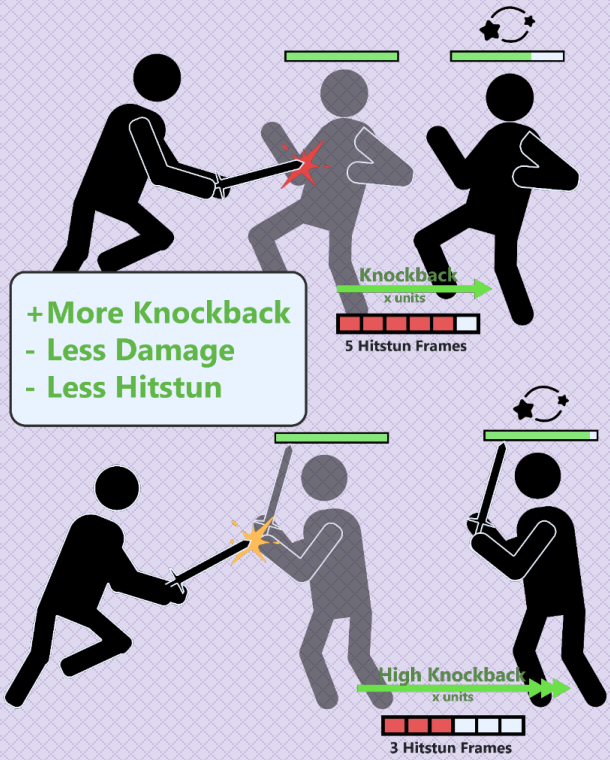
Heavy Attack 2 – An extremely slow attack with great damage, hitstun, and knockback. It was too high risk and too high reward to be fun, hitting it felt undeserved and missing it felt frustrating. Later I played with the idea of making it break guards so that you had to dodge it, but I thought the mechanic was hard to communicate and non-interactive, and by this point the prototype was already suffering a little from feature creep. It was abandoned as Heavy 1 did everything this attack did in a more balanced and engaging way.

Defensive Tools

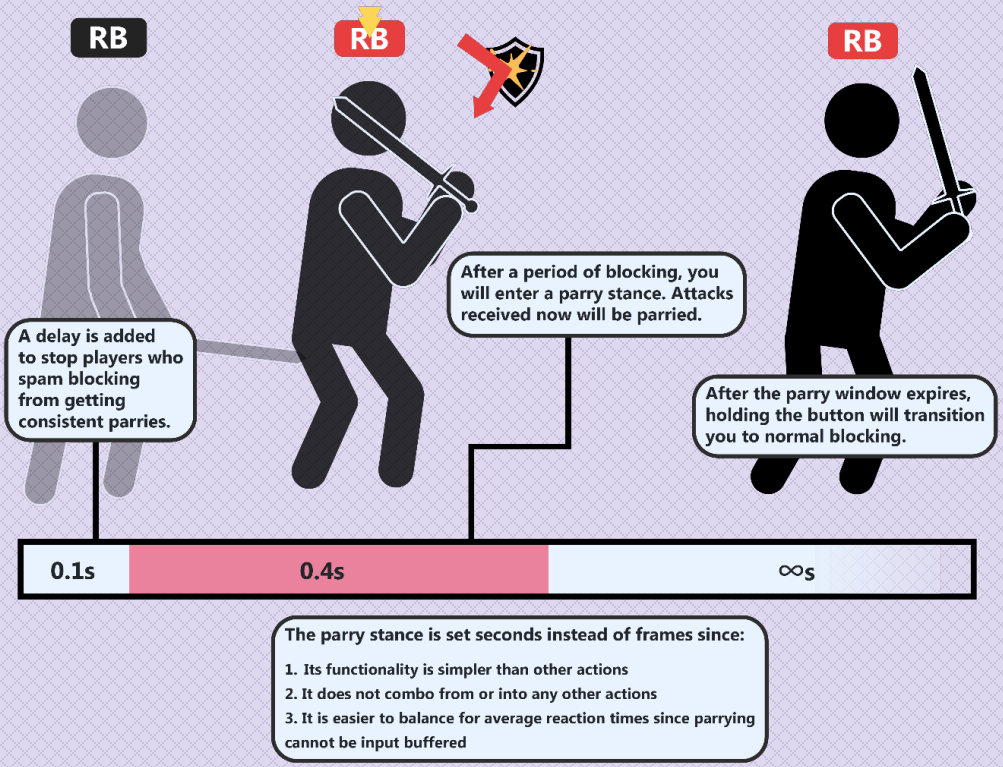
Blocking – Holding the button will put the user in a stance where they will block all attacks, receiving less damage and hitstun, and pushing the aggressor further away from them. You cannot block while you are in the middle of another action, but if pressed at the end of certain actions you will cancel the remainder of their animation.





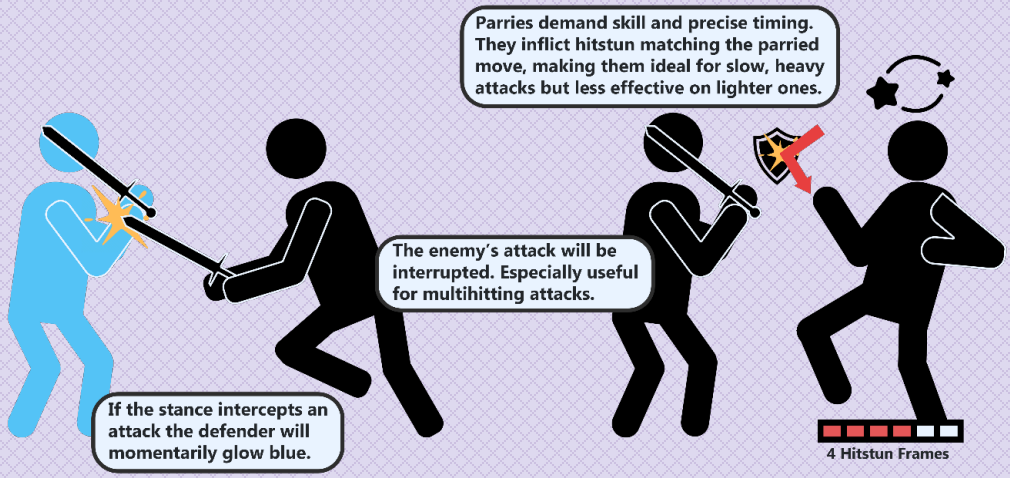


Parry – Pressing block at the very moment the user is going to be hit by an attack (the window is around 500 ms), will grant very little hitstun, no damage, and will interrupt the enemy's action, allowing the defender to punish them.

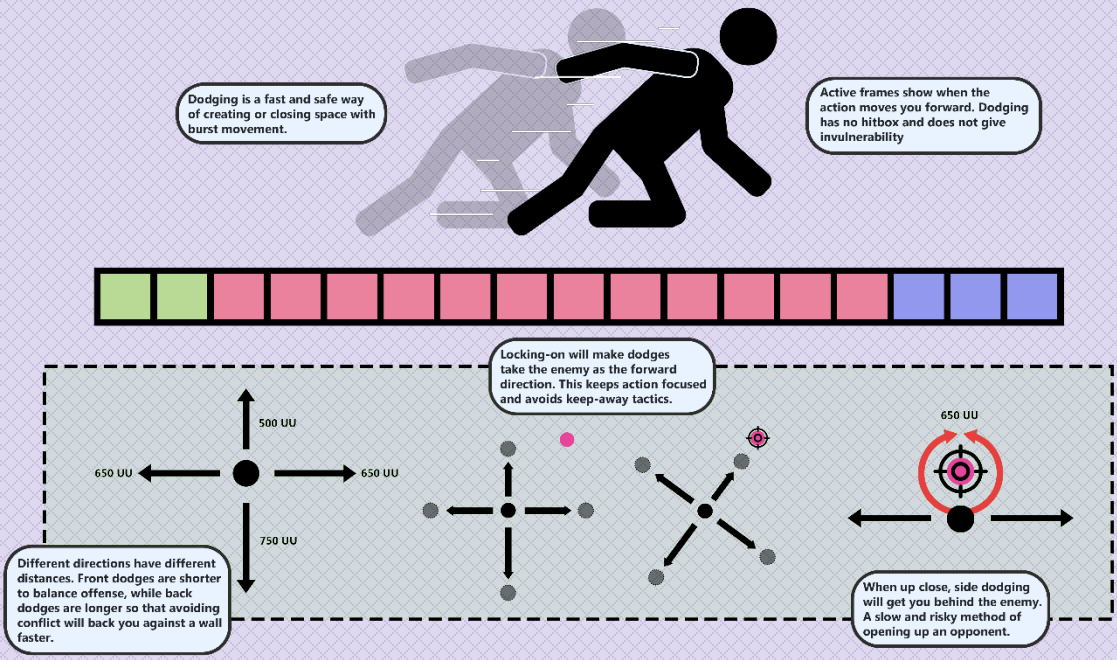


The parry stance is set seconds instead of frames since:

1. Its functionality is simpler than other actions
2. It does not combo from or into any other actions
3. It is easier to balance for average reaction times since parrying cannot be input buffered



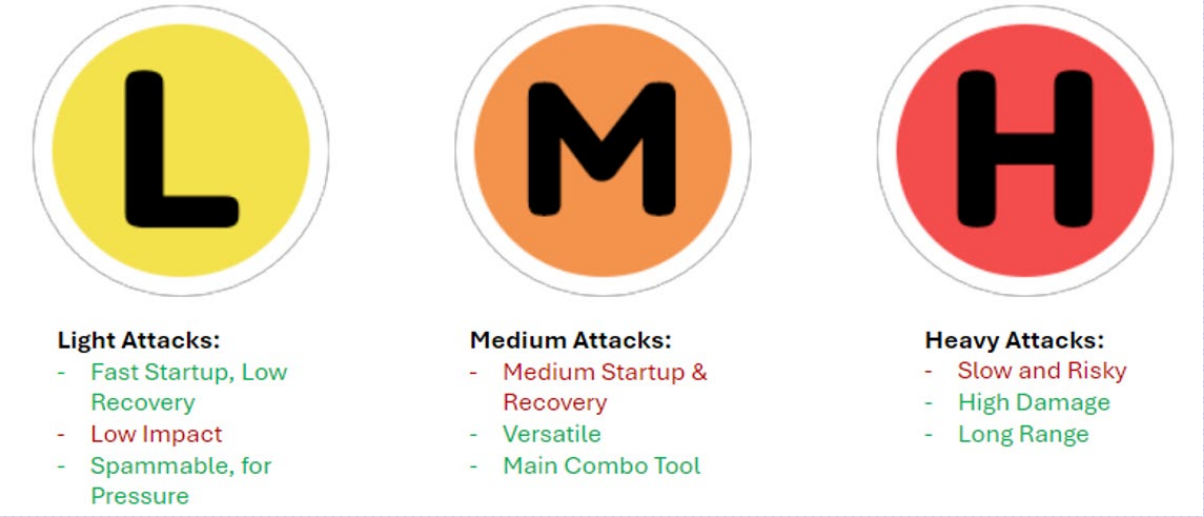
Dodging – A movement ability in 4 cardinal directions, used to avoid attacks, create space with the enemy, and continue offense. The end of the attacks' animation can be cancelled with a dodge for defensive and offensive purposes. This action **does not have invulnerability** throughout its duration.



Combo Structure

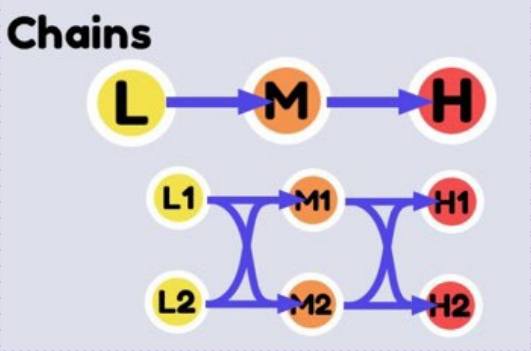
Offense is built around combos using 3 distinct attack types, each serving a different purpose. Unlike typical games in the genre with fixed combo structures, players must time their inputs carefully and rely on situational awareness to freestyle link attacks in unique ways.

The prototype includes 4 attacks instead of the original 6 (2 per type) for simplicity and balance reasons. Notably, the current "Special Attack" is treated as a medium attack internally but stands apart due to its unique behaviour outside the combo structure, since it doesn't have typical combo properties. Despite the reduced number of attacks, this structure still works as of the latest prototype and could easily be expanded with new moves if development were to continue.



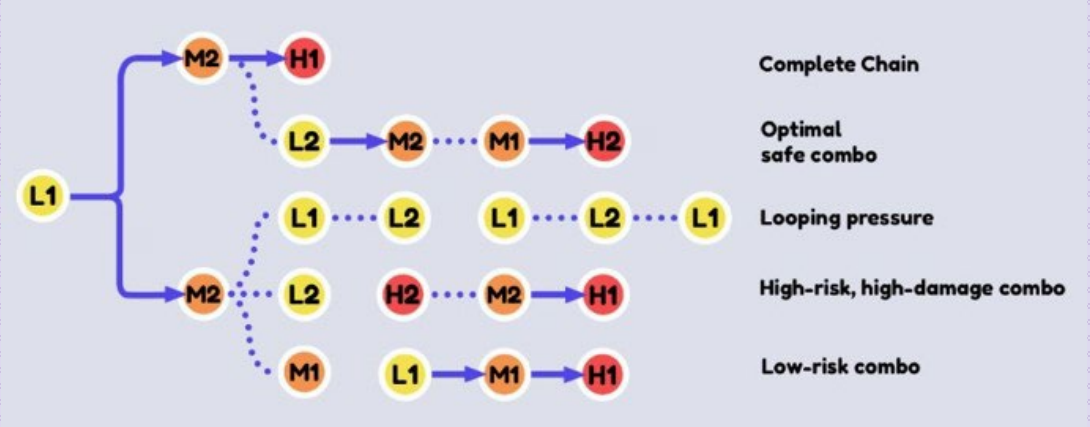
There are 2 ways of connecting attacks, with chains or links:

- **Chains:** If an attack lands, an attack of a "higher" category can cancel the remaining animation time, guaranteeing a new hit. This does not happen if the attack misses, but it does if the attack is blocked.



- **Links:** When an attack lands, the character that was hit will be unable to act for a period. If the attacker inputs another attack that hits the enemy before they can block again, the attack will be confirmed. Since no animation is being cancelled, this type of combo is harder and requires more experimentation to discover.

These types of connecting attacks can be mixed and weaved in together, offering a variety of combos for every situation and effect, and a playstyle can be naturally created through player experimentation and iteration with this mechanic. This type of emergent gameplay is conscious of player skill, so there are easier to perform but less rewarding alternative combos for those who are not interested in improving their skills mechanically.



Note: The combos shown in the diagrams are just examples, and might not reflect the final prototype, especially due to the reduction of inputs that happened in later iterations.

Camera Implementation

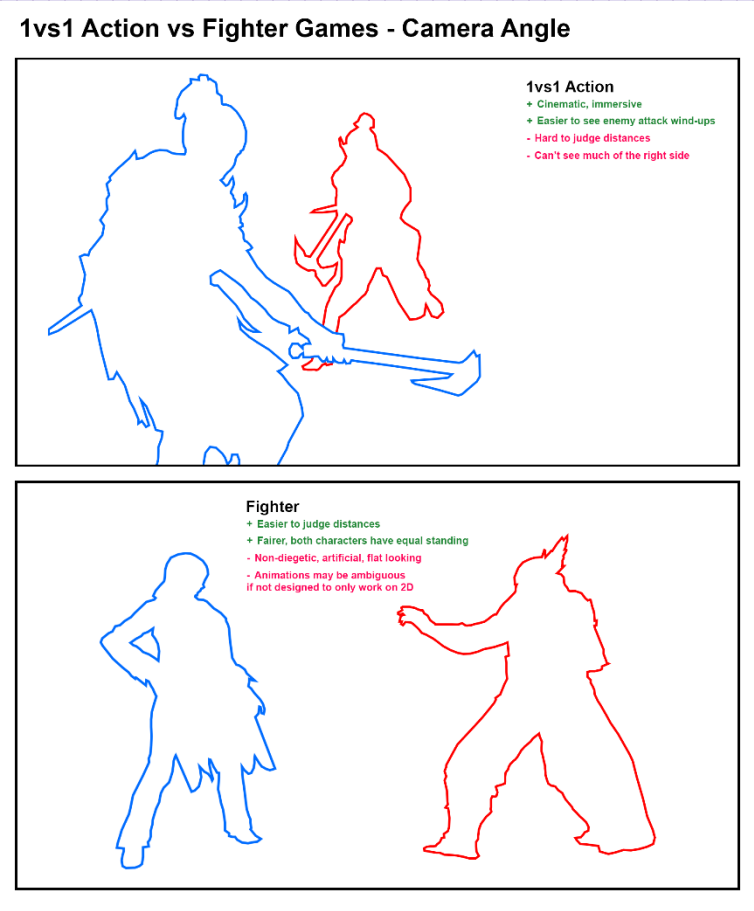
The prototype lets players switch between locked-on and free-roaming cameras to fit their playstyle. I thought about cutting free roam at first, but testing showed few select players liked aiming attacks themselves and coming up with unique tactics, such as unlocking the camera to back dodge instead of front dodging directly for extra distance. Keeping both options adds more variety to how people play the prototype, although in a full game development this would need to be considered further as it could add unintended strategies that could break balance.

Here are some of the camera design considerations I made.

1v1 Lock-On Camera Types

The lock-on camera angle is vital for visibility and player awareness in 1v1 combat, typically positioning characters near the centre of the lower third of the screen to maximize forward view. Inspired by For Honor, I implemented a similar lock-on system that supports deliberate, tactical duelling. I enhanced it with more dynamic movement to suit the faster pace of my prototype, helping players track both their own and their opponent's moves while maintaining spatial context. This mirrors fighting games, where equal screen presence and movement along a single axis improve readability and balance, even in 3D fighters like Tekken.

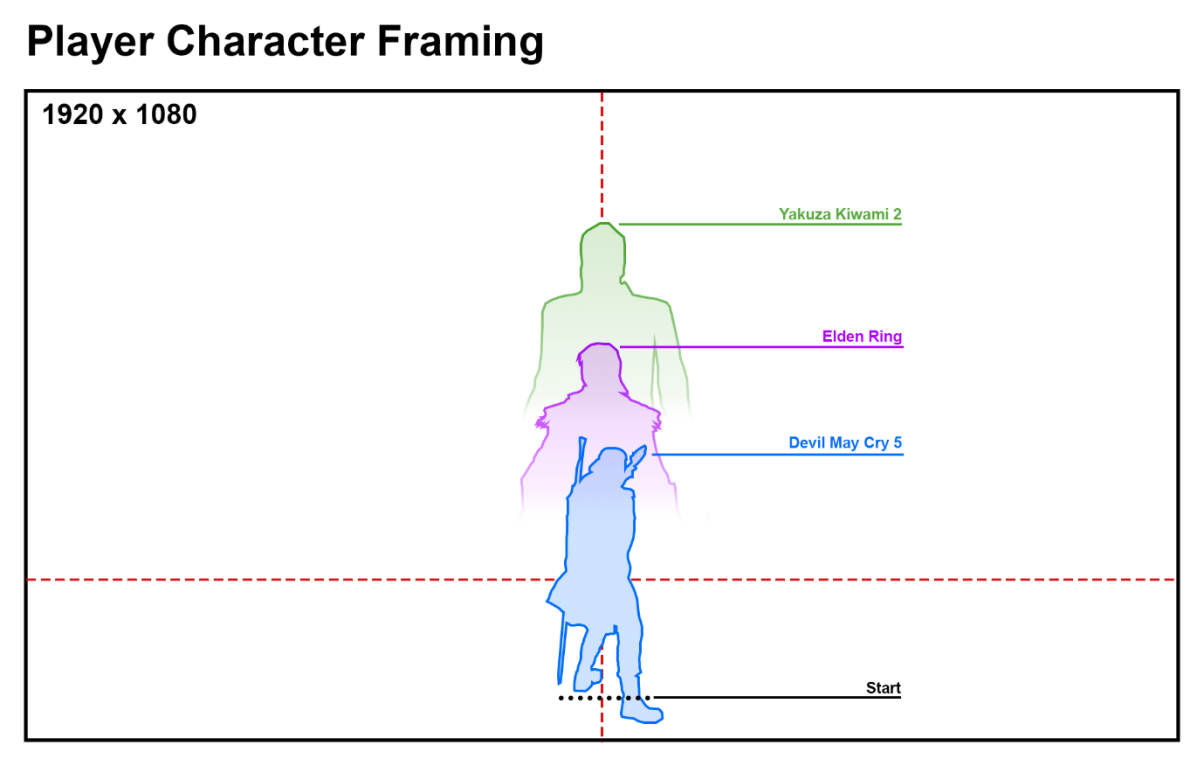
For non-locked-on gameplay, I adopted a traditional free-camera style, akin to Dark Souls, which offers broader environmental awareness and player control at the cost of precision, since attacks are no longer automatically aimed at enemies.



Character Framing

Games that emphasize exploration or complex environments often position characters smaller and lower on screen to increase visibility. The size and placement of on-screen elements guide player attention, with larger elements signalling greater importance.

In my game, the environment plays little to no role, and the enemy is the primary threat. To keep the focus on combat, I kept the character large and centred when locked-on, ensuring clear visibility during intense encounters. While unlocked, the camera smoothly transitions to a more zoomed out version better aimed at playing around the edges of the arena, yet it still ensures that the player is not too distant from the screen as this could impact game feel and gameplay visibility.



Balance Framework

Before moving into testing, I wanted to get a solid grasp of how the mechanics I designed were meant to interact. One of my main goals was to ensure variety, so players could approach the game in different ways and naturally discover new strategies. With each iteration, I created evaluation schemes like the one below to help me organize feedback, track patterns, and spot where the prototype started drifting from the original design goals.

Quick Point Table

I developed quick point tables for the player's moveset, outlining my intended strength and properties for each attack. Using a qualitative scale from 'Very Low' to 'Very High' and a colour-code to outline advantages, I assessed each move based on perceived impact rather than raw numerical data. This approach allowed me to evaluate how the mechanics feel from a player's perspective. It also helped establish acceptable ranges for each property, ensuring that no attack strayed too far from the overall balance and design intent of the full moveset.

The original 6 attacks where analysed this way, and while there was an obvious progression in strength, it also showed that some attacks were too like each other to grant any fun use cases for casual players.

	Startup	Active	Recovery	Overall Duration	Damage	Hitstun	Knockback	Block Damage	Blockstun	Block Knockback	Special Properties?
Light1	Low	Low	Low	Low	Low	Low	Low	Low	Low	Low	NONE
Light2	Mid	Mid	Low	Mid	Low	Mid	Mid	Low	Low	Low	MULTIHIT
Medium1	Low	High	Mid	Mid	Mid	High	Mid	Mid	Low	Mid	MULTIHIT
Medium2	High	Low	Low	Mid	Mid	Mid	High	Low	Mid	Mid	PULLING
Heavy1	Very High	Low	High	High	High	High	Very High	Mid	Mid	High	LAUNCHER
Heavy2	Very High	Mid	Mid	High	Very High	High	Very High	High	Mid	High	LAUNCHER

After stripping down some actions, the situations in which each attack should be used became clearer, and players showed a deeper engagement with the moveset. It also simplified the balance process a lot more.

	Startup	Active	Recovery	Overall Duration	Damage	Hitstun	Knockback	Block Damage	Blockstun	Block Knockback	Special Properties?
Light	Low	Low	Low	Low	Low	Low	Low	Low	Low	Low	NONE
Medium	Low	High	Mid	Mid	Mid	High	Mid	Mid	Low	Mid	MULTIHIT
Heavy	Very High	Low	High	High	High	High	Very High	Mid	Mid	High	LAUNCHER
Special	High	Low	Low	Mid	Mid	Mid	High	Low	Mid	Mid	PULLING

Note: Gray cells refer to not-applicable data or non-classifiable data. Purple cells refer to data that cannot be compared with other cells of the same colour.

Resonance Table

Resonance tables were used to determine if the interaction between 2 mechanics one-on-one would be favourable to the user (the user's action is found in the columns). An action is seen as favourable (in green) if:

- The action successfully defends the user from an attack.
- The action successfully damages the enemy.
- The action nullifies, cancels, or ignores the enemy's action without any major drawback.
- The action gives the user frame advantage to act before the enemy.

And vice versa for if an action is seen as unfavourable (in red). An action is seen as net neutral (in yellow) if:

- Neither character gets the upper hand from the interaction
- Both characters are hit by the same move at the same time.

If 2 mechanics cannot interact with each other in any way, they are not applicable (in grey), and if the interaction has a situational result, it will be in purple and explained below the chart.

	Light1	Light2	Medium1	Medium2	Heavy1	Heavy2	Dodge	Block	Parry	Walking/ Running
Light1	=	✓	✓	×	×	×	×	×	×	×
Light2	×	=	✓	×	×	✓	×	✓	✓	×
Medium1	×	×	=	×	×	×	✓	=	✓	✓
Medium2	✓	✓	✓	=	✓	✓	✓	×	✓	×
Heavy1	✓	✓	✓	×	=	×	✓	=	✓	✓
Heavy2	✓	×	✓	×	✓	=	✓	=	✓	✓
Dodge	✓	✓	×	×	×	×	=	~	~	×
Block	✓	×	=	✓	=	=	~	=	n/a	n/a
Parry	✓	×	×	×	×	×	~	n/a	=	n/a
Walking/ Running	✓	✓	×	✓	×	×	✓	n/a	n/a	=

~ Dodge & Block/Parry technically do not interact but given that dodging has recovery time while blocking/parrying does not, one could punish the enemy from block if they dodged towards you.

The table helped point out bugs and exploits in the system that would trivialise certain aspects of the gameplay, but it was difficult to process and analyse so much data. It also pointed out how many moves fulfilled the same roles and beat similar options, so it helped me in deciding what to get rid of once I swapped to the 4 attack moveset model.

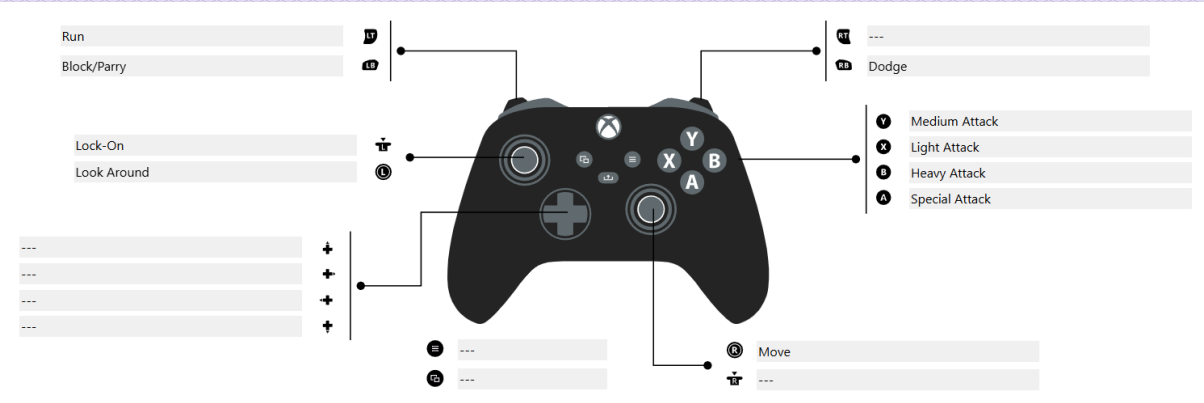
	Light	Medium	Heavy	Special	Dodge	Block	Parry	Walking/ Running
Light	=	✓	×	×	×	×	×	×
Medium	×	=	×	×	✓	=	✓	✓
Heavy	✓	✓	=	×	✓	=	✓	✓
Special	✓	✓	✓	=	✓	×	✓	×
Dodge	✓	×	×	×	=	~	~	×
Block	✓	=	=	✓	~	=	n/a	n/a
Parry	✓	×	×	×	~	n/a	=	n/a
Walking/ Running	✓	×	×	✓	✓	n/a	n/a	=

Inputs and Controls

Control Scheme

I chose to design the game with home console controls (specifically Xbox but this scheme can be translated to other consoles) because this genre performs best on home consoles, where players can enjoy more precise and comfortable inputs. This controller layout has each attack assigned to the main face buttons, which is derivative of how fighting games usually assign their actions. I found this setup during testing to be

intuitive, and comfortable for long input sequences and play sessions.



Input Buffer

An input buffer feature was added later in development after players reported sluggish controls and missed inputs. Since the average human reaction time is around 400 ms, the buffer window was set to 450 ms, longer than the typical 100-300 ms that is seen in other games within the genre, to accommodate and reduce the skill requirements for the prototype, allowing for more natural testing. For a full game, the window would likely be reduced to tighten skill progression and control precision.

The way it works is that the game saves the latest input detected and plays the appropriate action whenever the character is available within 450 ms, after which the input is ignored. If the player is allowed to cancel the input pressed from the action currently playing, the input buffer will do so when available.