**Game Design Document**

**Jake Berger**

**Part 1: Philosophical Framework**

Before sharing my design, it’s important to first provide my overarching philosophical framework for feature development. Then I’ll review a gameplay mechanic in Madden/NCAA Football; in this case, I’ll be looking at how quarterback throw accuracy is determined and how the results of that determination are manifested on the field. Finally, I’ll provide my design for a functionally comparable mechanic i.e., pass accuracy. In both cases, I’ll be analyzing each design – current and proposed – through the lens of my framework, analyzing their implications on gameplay.

When first brainstorming a new feature, it should be viewed within the context of existing gameplay. How compatible will it be with existing systems and mechanics? Will it promote gameplay in a way that aligns with the vision of the game? How will it affect other gameplay mechanics? It can be difficult to answer these questions without a clear and defined set of guidelines that regulate feature development; thus, a framework for feature design is essential. Ultimately, an authoritative framework would help futureproof new features (given their adherence to established game design principles), and it would ensure that all ideas interact harmoniously and build upon existing mechanics.

Framework for Dynamic Gameplay

So what goes into this framework? What are its core tenets? It depends on one’s philosophy and goals, but I believe the key objective is to develop features that engender dynamic gameplay and accurately replicate the game of football. While the latter goal is relatively easy to assess (though still difficult to design), the former goal is far less understood. “Dynamic gameplay” is an amorphous, yet pervasive term ingrained within the community’s lexicon, but it needs to be properly defined to be properly understood. And it needs to be properly understood before features can be created that engender this so-called dynamic gameplay. So what is dynamic gameplay?

Well, dynamic gameplay originates from mechanics that, in my estimation, adhere to the following set of core principles:

1. Mechanics should possess “mechanical depth”
2. User-controlled mechanics should maximize the degree of real time player control
3. Mechanics should synergize with other mechanics in its gameplay system (a system is defined as a group of related mechanics)
4. Mechanics should be built to allow future iterations to expand on its depth

In the following section, I’ll examine throw accuracy in Madden/NCAA, and I’ll be using the core principles as a lens to assess its design. However, given the lack of user control over the landing spot of thrown balls, the second principle does not apply. Likewise, the third principle is beyond the scope of this analysis. Consequently, I’ll only review throw accuracy within the context of the first and fourth principles.

But before analyzing the existing and proposed landing spot mechanic, we first need to review two concepts: “possibility space” and “continuous vs. discrete variables.”

Possibility Space

Game designers must establish the possibility space for both (A) player choice and (B) outcomes of interactions. Possibility space regarding player choice represents the total number of options at the user’s disposal, whereas possibility space pertaining to outcomes represents the range of possible outcomes as determined by player choice and other variables. The complexity of a game mechanic (considering possibility spaces for both player choice and outcomes) is referred to as mechanical depth. The breadth of possibility space should be medium to high for player choice and extremely high for outcomes.

The possibility space established for player choice is set precisely by its designers. More options available to the user increases the game’s complexity, thereby providing more mechanical depth. Perhaps most importantly, a wide possibility space with respect to user choice will increase the game’s level of strategy. A well-crafted possibility space constrains a player’s actions to a range they understand clearly; too few choices and the player will become bored, but too many choices and the player will feel overwhelmed.

On the other hand, the possibility space derived from possible outcomes of interactions is set by the variables in a game’s formulas. Here, we’ll look at two different types of variables, assessing each based on the degree of mechanical depth they provide.

Continuous Vs. Discrete Variables

If a variable can take on any value between its minimum value and its maximum value, it is called a continuous variable; otherwise, it is called a discrete variable (a variable with a defined number of outcomes). A discrete variable is binary if there are exactly two possible outcomes. Continuous variables can exponentially increase a mechanic's possibility space, while binary variables limit possibility space to two outcomes. A wide possibility space promotes “organic gameplay,” but a narrow possibility space promotes “scripted gameplay.” In a college football game especially, where excitement is largely dependent on the feeling that anything can happen, continuous outcomes are usually preferable to discrete outcomes.

**Part 2: Madden/NCAA Football Design**

Core Design

With the philosophical framework in place, we can now use it as a lens to evaluate existing and proposed mechanics. So let’s first analyze how throw accuracy is determined in Madden/NCAA Football. When a ball is thrown, the quarterback’s throw accuracy rating is put into a formula. The formula then churns out a binary outcome; the ball is either perfectly thrown, or the ball is wildly inaccurate in the form of an extreme overthrow or underthrow. There is no variation in a quarterback’s accuracy when the game decides the ball will be a completion. The accuracy rating only affects *if* the pass will be accurate or not; it does not alter the degree of inaccuracy on any given pass.

Moreover, the exact landing spot of a thrown ball is probably determined by factors that calculate the receiver’s projected position at the time of the catch interaction, including receiver speed and throw power.

Next, player states appear to affect throw accuracy. When the quarterback is standing tall in the pocket (default player state), his accuracy rating is unaffected when placed into the formula. But when the quarterback is throwing on the run, or if the quarterback is engaged in a collision with a defender, his accuracy rating is lowered when entered in the formula.

Lastly, throw accuracy is affected by the distance of the throw. Throw distance is categorized into three discrete states: short, medium, and deep passes. Likewise, quarterbacks have a short, medium, and deep pass accuracy rating that corresponds with these three states. The throw distance determines which accuracy rating is used in the passing formula. To simulate the increased difficulty of a longer throw, quarterbacks typically have a lower deep pass rating than medium pass rating, which is lower than their short pass rating. Ultimately, this rating classification system does functionally alter pass accuracy even if it doesn’t alter the throw accuracy formula directly. One important note is that distance seems to be determined by yards downfield rather than yards traveled through the air. If, for example, a quarterback throws a ball horizontally from sideline to opposite sideline (53.5 yards), the short throw accuracy rating would be used in the formula. We’ll explore the implications of this design decision in the next section.

Mechanic Evaluation

With an understanding of how the mechanic is designed, it’s finally time to evaluate it within the context of the framework. Question #1: Do the factors that dictate the landing spot of a thrown football offer sufficient mechanical depth, and if so, to what degree?

With respect to player choice, there are very few ways to affect throw accuracy. However, throw accuracy is not a user-controlled mechanic; instead, it’s governed by formulas. Shouldn’t a non-user-controlled mechanic preclude player choice by its very nature? Well, not exactly. Choices still exist, though they are indirectly tied to the mechanic. In this case, players can skillfully and intentionally control the different states of accuracy a user-controlled quarterback is in. For example, holding onto the ball too long will put the quarterback at risk of being hit while throwing, which alters the accuracy rating used in the formula. Likewise, attempting to escape pressure and throwing on the run is subject to an accuracy penalty. In both cases, users must be cognizant of the player states that exist and how they affect throw accuracy, and as a result, players must make choices about how long they stay in the pocket or whether to risk throwing outside the pocket or on the run. Therefore, while user choice does exist, the possibility space is extremely narrow, resulting in a mechanic devoid of almost any strategy.

Although little strategy is offered here, any mechanic responsible for landing spots is primarily concerned with possibility space regarding outcomes. In Madden/NCAA, a formula determines if a ball is perfectly thrown or uncatchable. As such, there are only two possible outcomes. There is no variation in a quarterback’s accuracy when the game decides the ball will be a completion. The ratings only affect the number of uncatchable passes, not the degree of inaccuracy on any given pass. Predictably, the breadth of possibility space for outcomes is very low.

Now let’s turn our attention to the fourth principle and ask the following question: Does this mechanic lend itself to being iterated upon to increase mechanical depth? In this case, since the passing formula yields a binary outcome, no number of additional variables to the formula would alter the number of possible outcomes. More variables would only affect how often a ball is perfectly thrown. The binary nature of the mechanic therefore precludes increases in mechanical depth; to truly alter outcomes on the field would require a total rebuild of the mechanic.

Though this is but one mechanic within its gameplay system that helps define the passing game, it is fundamentally unrealistic and lacks mechanical depth. The binary nature of the mechanic results in a very narrow possibility space, and the passing game by extension consequently feels inorganic and predetermined.

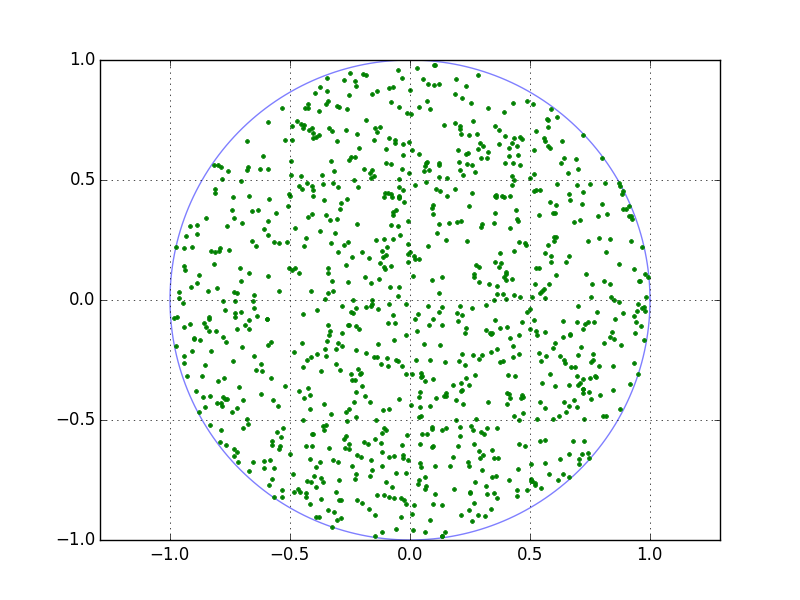
**Part 3: My Design**

In this section, I’ll share my feature design. I’ll once again analyze it within the context of my framework. However, the structure of this analysis will differ from the previous section. In the previous section, I first explained – in aggregate - how the mechanic was programmed. Then I evaluated it against each core principle of the framework in sequential fashion. In this section however, I’ll analyze components of my proposed mechanic one at a time, drawing on the framework when applicable. So let’s begin.

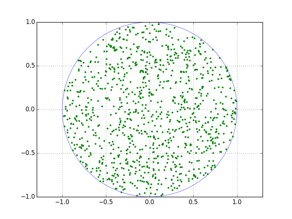
Core Design of the Mechanic

We’ll first review the core design of the mechanic: how the landing spot of a thrown ball is determined. First, the game will determine the end position of the perfect pass. This point in space is based on the projected location of the catch interaction, accounting for the receiver’s current speed and direction, route, and ball velocity. This point will henceforth be referred to as the “perfect pass.” The perfect pass is calculated for every pass attempt, and its location is the same for every quarterback.

After the perfect pass has been determined, a “landing circle” forms around this point. The size of the landing circle corresponds to the quarterback’s throw accuracy rating, and it encompasses all possible landing spots of a thrown football. A formula will generate a random point within the circle, and this point will be the exact landing spot of the football. A larger landing circle increases the range of distribution around the perfect pass (increases the amount of possible landing spots), thereby decreasing the predictability of a throw. Likewise, a smaller circle decreases the range of the distribution around the perfect pass (decreases the amount of possible landing spots), thereby increasing the predictability of a throw. As you might imagine, a higher throw accuracy rating will correspond to a smaller landing circle and – therefore - increased precision, and vice versa. Below is a visual example comparing the distribution range of landing spots between a quarterback with a high vs. low accuracy rating:



QB with lower pass accuracy = larger landing circle = increased possible outcomes = decreased precision



QB with higher pass accuracy = smaller landing circle = decreased possible outcomes = increased precision

Mechanic Comparison

In a vacuum, this mechanic permits an extremely wide possibility space regarding outcomes. There are hundreds of possible landing spots for a given pass (though the degree of variance changes with the QB accuracy rating). In this system, throw accuracy essentially determines how “catchable” a ball is; in Madden/NCAA, throw accuracy determines *if* a ball is catchable. In this system, the ball’s landing spot is a continuous variable and increases the game’s mechanical depth; in Madden/NCAA, the ball’s landing spot is a binary variable that limits the game’s mechanical depth. With this system, the passing game will feel organic and unpredictable; in Madden/NCAA, the passing game is predictable and feels scripted.

With a binary mechanic, it is impossible to widen the possibility space of outcomes. Case in point: the throw accuracy rating in Madden/NCAA yields one of two options - a pass is either catchable or uncatchable. In my proposed design however, given the wide possibility space inherent in the mechanic, it is possible to layer additional complexity to the system (core principle #4) in ways that appreciably alter the possibility space. Accordingly, in the following paragraphs, I’ll identify multiple factors that affect throw accuracy. I’ll also explore directly related mechanics that are necessary to accompany my design, including a proposal for a complementary feature that meshes well with the core mechanic (core principle #3).

Factors Affecting Throw Accuracy

As we discussed in the previous section, although a thrown ball’s landing spot is determined by a formula, possibility space with respect to user choice can still be widened if the user can influence the equation. The following variables and proposed mechanics will all increase user choice – let’s find out how.

First, let’s examine the factors that affect the throw accuracy rating. Like Madden/NCAA, throw distance and a variety of player states influence the throw accuracy rating. We’ll first review throw distance. In this design, throw distance wouldn’t be categorized into three discrete states (short, medium, long) – it will be a continuous variable calculated more realistically i.e., yards traveled in the air. Additionally, quarterback accuracy will be inversely proportional to throw distance, preventing scenarios where a quarterback can throw from sideline to sideline without being subject to an accuracy hit. In Madden/NCAA, distance is a function of yards downfield – not distance traveled in the air. Therefore, if a quarterback throws a ball horizontally from sideline to opposite sideline (53.5 yards), the short throw accuracy rating would be used. As a result, user quarterbacks can roll to either side of the field safe in the knowledge that there is no consequence in throwing to the opposite side. Contrarily, with my method of calculating throw distance, user-controlled quarterbacks would benefit more from staying in the pocket to have both sides of the field within practical throwing range. As is the case in real life, if a quarterback rolls to one side of the field, they are usually reading to that side of the field; a throw toward the opposite sideline would be less accurate and less safe.

Moreover, multiple player states will be considered when determining throw accuracy. Similar to Madden/NCAA, player states include the default player state, throwing on the run, and throwing while engaged with a defender, but additional player states will include throwing while shuffling in the pocket and throwing under duress (quick throw). Each player state will affect throw accuracy differently, but to build on this system in future iterations, each animation within a particular state can alter throw accuracy with its own rating multiplier. By tying accuracy to a specific animation, designers can ensure that the landing spot of the football makes sense given the animation.

In addition to throw distance and player states, other ingredients impact throw accuracy, or in this case specifically, the landing spot of passes. Other such ingredients include the wide receiver’s route, a mechanic that allows the quarterback to lead the receiver, and a “throw away” mechanic. I’ll briefly explain how each ingredient affects pass accuracy – in particular, an in-depth examination of a user controlled “lead the receiver” and “throw away” mechanic is beyond the scope of this feature analysis.

When the “perfect pass” is calculated, the receiver’s route must be considered. In Madden/NCAA, the landing spot of a thrown ball is determined by the direction of the receiver at the time of the pass without accounting for forthcoming movements in the receiver’s route. For example, if a receiver is running a deep out route, and if the quarterback throws while the receiver is still in his vertical stem (before breaking outside), the pass will lead the receiver as though he is running a fly/go route. The pass will not account for the receiver’s upcoming change in direction, and when combined with receivers’ telepathic awareness of the ball’s location at the time of the throw, the system unrealistically allows receivers to change their intended direction as quarterbacks alter routes on the fly. A more authentic passing game needs to account for where the receiver will be based on his route, not where he is currently going.

Quarterbacks can already lead receivers in Madden/NCAA, and the feature would be implemented similarly to its current design. In my design, when leading receivers with the left stick, the landing circle moves correspondingly. The relationship between the movement of the left stick and the movement of the landing circle does not require examination here.

A throw away mechanic already exists in Madden/NCAA, but it only allows the quarterback to throw the ball far out of bounds. Throwing the ball away in the pocket will result in an intentional grounding penalty. There should be an option to throw an intentionally inaccurate and – nearly - uncatchable ball to a specific receiver, which would enable quarterbacks to throw the ball away more safely while in the pocket. As with the ability to lead receivers, the precise determination of the landing circle for this mechanic is for a future design document.

Lock-on Mechanic

The recipe that governs pass accuracy is almost complete. There is but one more ingredient directly tied to the passing “gameplay system” (group of related mechanics) that needs to be baked in – this ingredient is the length of time a quarterback locks onto a receiver. Unfortunately, this variable does not exist in Madden/NCAA – it isn’t simulated under the hood, it isn’t visually represented in the game, nor is it tied to a user mechanic. That all changes with my proposed “lock-on” feature, a mechanic that aims to replicate the effects of a quarterback going through his progressions.

With this feature, the user will control the eyes of the quarterback, which subsequently alters animation and throw accuracy. We’ll start with how the feature affects quarterback animation. At the start of the play, the quarterback will be locked on the primary receiver as determined by the play call, and his head and body will continually adjust to face that receiver. To throw to that receiver, the user will simply press the corresponding wide receiver button. At this moment, apart from the quarterback visually tracking the receiver, this mechanic is functionally the same as the current system. However, my design begins to differ from the current system when attempting to lock-on to a different receiver. In Madden/NCAA, there is no locking on to a receiver; you simply press the corresponding wide receiver icon to throw to him. With my proposed mechanic, the user will first need to press the corresponding wide receiver icon to lock-on to him and then press the same button a second time to throw to that receiver. Basically, after the initial button press, the quarterback’s head and body will turn to the intended receiver; the re-tap of the button will prompt the throwing animation.

This lock-on feature isn’t solely for aesthetics however; it also has implications on pass accuracy. The length of time locking on to a receiver determines the bonus or penalty applied to the throw accuracy rating. Throw accuracy will increase the longer a quarterback is locked on to a receiver (an upper bound will be applied to the accuracy bonus). Likewise, throw accuracy will decrease with a shorter lock-on period. For example, if the user double taps a wide receiver icon (one press to begin locking on, and a second press to throw the football), a penalty will be applied to throw accuracy. This bonus or penalty provided from the lock-on duration ultimately simulates the importance of surveying the field, finding a target, and then making an accurate throw.

Aside from being a more accurate replication of quarterback play, a lock-on mechanic increases the passing game’s mechanical depth, especially regarding player choice. While in the pocket, users must make strategic decisions about when and how long to lock on to a receiver, as well as how many reads to make over the course of a play. There exists a constant risk/reward with cycling through receivers. Does the quarterback survey the entire field, spot an open receiver, select his icon, and then throw to that receiver? Well, with the short lock on duration, throw accuracy will take a hit. Ok, so instead of surveying the entire field all at once, the user opts to go through progressions. A similar risk/reward scenario arises; does the user risk going through too many progressions at the expense of a possible short lock-on phase and proportional accuracy penalty? Keep in mind the pocket is collapsing. A quarterback going through too many progressions risks succumbing to harsh penalties from pocket pressure. So maybe a quarterback plays it safe and limits a play to two reads. But remember, the quarterback’s eyes and body are in tune with the receiver, so a long lock-on duration enables defenders to read the quarterback’s eyes to better position themselves or jump routes. No problem: combat this by looking off safeties to take them out of the play. At its core, this lock-on feature is mechanically simple, yet many complex micro-decisions must be made every play, substantially increasing the level of real-time strategy in the passing game.

Pocket Pressure

The aforementioned variables fall within the same gameplay system i.e., mechanics directly related to throwing. They act in harmony to be greater than the sum of their individual parts, and just as importantly, there is still room for more parts to be added to further synergize the whole system. But relationships also exist between mechanics across different gameplay systems. Take the offensive and defensive line – on the surface, lineman interactions and mechanics may be thought to have no bearing on throw accuracy. But they can. Enter a variable called pocket pressure.

On a rudimentary level, pocket pressure can be defined by the distance between a defender and the quarterback. Predictably, as pocket pressure increases (the distance between the defender and quarterback decreases), quarterback accuracy will decrease via a rating modifier. The rating modifier penalty will stack as more defenders/blitzers collapse the pocket. Moreover, as a variable, pocket pressure needs to be subjected to an exponential curve rather than a linear one. Though pockets collapse over time, there is a point – in this case a distance between the QB and defender - where a quarterback begins to feel this pressure. Before this point, the pressure generated would still allow a quarterback to stand tall in the pocket and survey options without fear of being hit. The quarterback only begins to feel this pressure once defenders breach this point. Though the precise tuning of how pocket pressure alters throw accuracy is beyond the scope of this document, it does deserve proper consideration. To take things further, a new quarterback rating, “pocket presence,” would affect the exponential curve by altering the rate at which increased pressure penalizes accuracy. This rating can also change the point when a quarterback begins to feel pressure. But regardless of how pocket pressure is tuned, by quantifying it as a variable that influences throw accuracy, mechanics from different gameplay systems are working together here. Mechanics shouldn’t work in isolation; they should affect and be affected by other gameplay systems to promote more organic and realistic gameplay.

Other Considerations

Finally, I’d like to take an inventory of what my proposed design either does not account for or would require to be fully effective. First, a player feedback system would need to be developed. There exists a problem with player perception of skill-based mechanics that take player ratings into account: is the inability to translate intent into desired reality because of a lack of skill or some problem with the game? What’s important is the player’s perception; visible and immediate feedback – based on player input – is required to inform the player why a particular input did not lead to the desired outcome. The user must be aware if the unintended outcome of a pass was the result of insufficient skill or a factor beyond his or her control. Finally, the passing gameplay system needs to be paired with a realistic player movement system. Although much improved over the years, players can still change direction on a dime, covering vast seas of grass with little respect for physics. To render a truly authentic passing game and catapult this system to “Heisman” level, player movement should be physics-based. The technology is there.

**Part 4: Conclusion**

Since its inception, the Madden and NCAA Football franchise has been built on a network of binary mechanics operating independently of one another, not acting in harmony to promote more organic gameplay. But by adhering to the tenets of my philosophical framework, individual mechanics will offer high levels of depth and strategy, and designs from different gameplay systems can coalesce into a coordinated and complementary whole.

I’d like to leave a parting insight about game feel. Most gamers can’t identify what makes a game feel good or bad, dynamic or scripted, tight or loose, but they can identify if a game possesses these characteristics and where these feelings fall on a continuum. These feelings may be difficult to articulate. They may be difficult to understand. But these feelings do exist, and they shouldn’t be dismissed as inaccurate, useless, or trivial - they’re not.

Game feel is determined by the math behind the scenes, and though the math isn’t understood, it can undoubtedly be felt. The math is a tangible, yet invisible and inexplicable force, and as designers and programmers, it’s our job to break down, define, and implement the game’s feelings. We create the player’s reality and we’re responsible for providing and shaping their experience in our world.

Why does this matter? Because it’s not the shiny new back of the box features and the easily digestible, mass-marketable gimmicks that change a player’s experience. What shapes a player’s experience has always been the core of a game – the math. The player doesn’t have to know why the game feels good, deep, authentic, strategic, fun, etc., but they do have to perceive these feelings, and if features are developed that follow my framework, the game will simply feel right. Just remember - it’s all in the math.