Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Interactive Display of Football Action

Bachelor Thesis

Thiago Knill

June 16, 2025

Advisors: Prof. Dr. Ulrik Brandes, Hugo Fabrègues

Department of Computer Science, ETH Zürich

Abstract

In football (soccer), tactics boards are widely used by coaches to visualize and communicate their strategic instructions. While physical whiteboards with magnets remain common, digital alternatives are increasingly being adopted for their flexibility and potential for richer interaction. This thesis investigates how interaction techniques on multi-touch surfaces can be leveraged to manipulate digital tactics boards in a manner that is both intuitive and efficient. We will develop and propose an interaction design and tool design aimed at maximizing efficiency, functionality, and user experience, particularly in the context of interactions involving player selection, player movement, player scaling, and view manipulation. To provide an overview of existing tools, we begin with a review, and to support further development of the proposed design, we present a guideline for implementation. Our design demonstrates that there is significant potential in leveraging multi-touch interfaces and football-specific structures to enhance the user experience with such a tool.

Acknowledgement

Firstly, I would like to thank my supervisor, Prof. Dr. Ulrik Brandes, and co-supervisor, Hugo Fabrègues, for their advice, support, and valuable feedback throughout this thesis. I also appreciate the helpful input and tips from the Social Networks Lab of ETH Zürich. A special thanks goes to my family and friends for their constant support and encouragement throughout the writing of this thesis. Lastly, I would like to mention that AI assistance tools were used to improve wording and phrasing at times when I was stuck, but not for the content of this thesis.

Contents

Chapter 1

Introduction

1.1 Motivation

Tactics boards, which are regularly used by professional coaches, are essential tools in football. However, since football is a sport played and followed by people at all levels, these boards are also widely used by amateur coaches, fans, and even players themselves to communicate and analyze football strategies. Traditional physical boards with magnetic player pieces are simple and accessible, but they come with several limitations. Typically, only a few magnets can be moved at once and only individually, and the size of the player pieces is often disproportionately large relative to the pitch. Moreover, these boards have a fixed size, cannot be zoomed in, and do not allow for saving tactics for reuse. Furthermore, they do not support the representation of more complex tactical structures, such as predefined formations, player roles, or dependencies between players [1,2].

Digital tactics boards, especially those designed for multi-touch devices, offer new opportunities to address these limitations. They allow for precise and scalable manipulation of elements and enable richer representations of tactical relationships. For example, players can be grouped into units like a defensive line or can be given positions. By incorporating these structural relationships, a digital system can support more powerful and intuitive interaction techniques. This reduces the time and effort required to effectively communicate tactical ideas [3–5].

1.2 Research Goals

The goal of this thesis is to develop an interaction design and tool design for multi-touch surfaces that supports intuitive and efficient manipulation of tactical boards in football. In addition to basic configurational settings, such as player numbers or pitch dimensions, our main focus is on interactions involving player selection, player movement, player scaling, and view manipulation. This focus is chosen because these interactions are most critical during the manipulation of tactics boards, and there is significant potential for developing new and useful features in this area. Domain-specific structures, such as player positions, should be effectively leveraged, and the time required to perform actions needs to be minimized. A key challenge is balancing the trade-off between creating a tool that is easy to use for beginners, but also offers advanced features for expert users.

Our approach begins with a review of existing tools for visualizing and manipulating tactical diagrams, identifying common features and interaction patterns. Based on this research, we derive a set of requirements tailored specifically to multi-touch devices. The result is a conceptual design that combines the strengths of current tools, established practices of professional coaches, and innovative ideas, while optimizing efficiency, functionality, and user experience. To further support our approach, we present a theoretical implementation using pseudocode.

Chapter 2

Review of Existing Tools and Interaction Design Theory

With the rise of digital applications, there has also been a growth in digital tactics boards for football. In this chapter, we will review existing tactics board tools, including digital options such as apps and websites, as well as traditional physical tools. Since our main focus is on interactions involving player selection, player movement, player scaling, and view manipulation, we will mainly examine how these interactions are used in existing tools. In addition, relevant principles from interaction design theory are included to provide a theoretical foundation.

2.1 Digital Tools

Digital tools, ranging from mobile apps and desktop applications to websites and interactive touchscreen systems, are now available for almost all purposes. Football tactics boards are no exception. A variety of tools with different functions and for different needs are available.

Our review covers fourteen tools for creating and manipulating tactics boards [6–19].

All these tools provide a variety of configuration options, such as selecting the pitch type, customizing styles and colors for various objects, saving boards, and adjusting settings [6–19]. Most also allow users to add shapes, draw on the pitch, or place training equipment and balls [6, 8–19].

Although some tools support multi-touch interactions [11, 17], most do not [6–10, 12–16, 18, 19].

2.1.1 Player Selection Interactions

Player selection interactions refer to the process of selecting players on the pitch, which is required for subsequent actions. Among the reviewed tools, only two types of selection interactions were observed:

- select a player directly
  - touch a player directly
  - touch a touch-assistant-zone
- select players of an area or shape
  - tap and drag to create a selection rectangle
  - tap on a shape

The first type of interaction selects a single player. This is typically achieved by touching a player on the screen [6–19]. One tool introduced a variant in which the user does not need to touch the player directly, but instead interacts with a circle associated with the player [10]. We refer to these as touch-assistant-zones, as they assist in selecting a player. The selection ends when the touch is released. This interaction is commonly used as part of a drag-and-drop gesture. Drag-and-drop and touch-assistant-zones will be discussed later in more detail.

Some tools also support the selection of multiple players. This is often implemented by allowing the user to tap and drag to create a rectangular area, selecting all objects within that area [8,14,16,19]. In addition, one tool includes a feature that allows players to be connected using a shape. Once connected, the entire group can be selected by tapping on the shape [16].

2.1.2 Player Movement Interactions

Player movement interactions are typically used after a selection has been made, allowing the user to reposition selected players on the pitch. The movement interactions observed in the reviewed tools can be grouped as follows:

- drag-and-drop
  - move a player
  - use the touch-assistant-zone
  - move an area
  - move a shape
- tap buttons

    - change the formation
    - move an entire team
    - narrow/expand a team
- swap players
- undo/redo actions

As mentioned in the previous section, drag-and-drop interactions involve selecting an object, moving it by dragging a finger without lifting it from the screen, and then releasing the touch [6–19]. This interaction can be performed by touching the player directly or by using the touch-assistant-zones introduced earlier [10]. Some tools allow for the selection of an area that can then be moved using the same drag-and-drop gesture [8,14,19]. One tool includes the selection of players connected by a shape, which can also be moved with drag-and-drop [16].

Among the reviewed tools, three features were identified for moving an entire team using buttons. First, the team's formation can be changed by selecting one of the predefined formations in the application [10, 14, 16]. Second, buttons allow the user to shift the entire team horizontally or vertically across the pitch [16]. Third, the spacing between players can be adjusted by narrowing or expanding the team's layout, also by tapping buttons [16].

A tool also offers a function to swap the positions of two selected players. This feature requires the use of the rectangle selection interaction to select the players involved [16].

Finally, some tools include undo and redo functionality through buttons [10,11,14–16,19]. As in most software applications, undo reverts the most recent user action, while redo reapplies an action that was previously undone.

2.1.3 Player Scaling Interactions

Player scaling interactions refer to adjusting the size of players on the pitch. In the reviewed tools, three types of scaling interactions were identified:

- use a slider
- use control handles
- use +/- buttons

Sliders are commonly used in software interfaces to adjust values, and some tools applied this mechanism to control the size of the player [10, 16, 19]. Other tools used control handles, small draggable points around an object, which allow users to resize players directly on the pitch [8, 9, 17, 19]. This method is widely used in many applications to resize visual elements, such as images. A third approach involved tapping buttons labeled '+' and '-' to increase or decrease player size, respectively [8,14].

Most tools do not offer realistic player sizes in proportion to the pitch. Even when a tool supports realistic scaling, it is not the default setting, and users typically need to manually adjust the size using one of the available scaling interactions [8–10,14,16,17,19].

2.1.4 View Manipulation Interactions

View manipulation interactions allow users to adjust their perspective of the tactics board. The following types of interactions were found in the reviewed tools:

- zooming
  - pinch or spread
  - use +/- buttons
  - use a slider
- move view
  - drag with one finger
  - drag with two fingers
- rotation
  - use two-finger-rotation
  - use buttons

Zoom interactions allow users to adjust the level of detail by scaling the view in or out. In many tools, this is achieved by pinching in or spreading out, that is, dragging two fingers toward each other or away from each other [9, 10, 15, 19]. Some tools instead offer buttons to zoom in or out incrementally [8,15], while others provide a slider for more continuous zoom control [10,14].

The view is typically moved by dragging one or two fingers across the screen [8–10,14,15,19].

To rotate the view, some tools support a two-finger-rotation gesture, that is, drag two fingers clockwise or anticlockwise [13,15]. Others provide buttons that rotate the board to the left or right [11,14,16].

2.2 Physical Tools

Despite the increasing digitization of the world, physical tactics boards remain widely used. Most commonly, they are whiteboards that support drawing and use magnetic pieces to represent players.

There are numerous videos on platforms like YouTube showing professional football coaches explaining their tactics using physical boards [20–23]. Since player scaling and view manipulation are not applicable to physical boards, our analysis is limited to player selection and player movement interactions.

The interactions most frequently observed with these boards can be grouped into two main types:

- move players
- pick up and place players

The most common interaction involves moving a player with one or more fingers. This can be understood as a combination of selection (touching the player) and movement (dragging the player to a new position). Coaches often perform this interaction with a single player, but sometimes also with multiple players simultaneously, using different fingers to move each player in a different direction. Another variation involves moving multiple players together by the same distance and in the same direction. This interaction is frequently used to fine-tune the position of players and is sometimes performed even when it is not strictly necessary.

Another widely used interaction is picking up a player and placing them elsewhere on the pitch. This can also be done with more than one player at a time.

One of the key advantages of physical boards is the intuitive, simple, efficient, and fast manipulation. However, they are very limited in their ability to represent and support football-specific concepts, such as formations or player dependencies. In addition, physical tools have a fixed size, players cannot be scaled, and the view cannot be changed.

2.3 Interaction Design Theory

Human-Computer Interaction (HCI) is a field that studies how people interact with computers and digital systems. The key domains within HCI include user-centered design, usability, interface design, and evaluation. User-centered design prioritizes the needs, preferences, and behaviors of users in the design process, ensuring that systems are tailored to real-world usage. Usability is a central quality attribute in HCI, measuring how effectively, efficiently, and satisfactorily users can achieve their goals with a system.

Interface design addresses how systems visually and functionally present information and interactive elements to users. Evaluation methods assess both the usability and performance of systems, providing insights into user experience and identifying areas for improvement [24–27].

User experience (UX) is the totality of the effects felt by a user as a result of an interaction with a system. One key concept in UX is affordance, which refers to the properties of an object that indicate how it can be used. Well-designed affordances help users intuitively understand the interaction possibilities with an interface element. A well-executed HCI design leads to numerous benefits including increased user productivity, reduced error rates, enhanced satisfaction, and broader accessibility [24–27].

The ten usability heuristics of Jakob Nielsen are widely recognized principles for evaluating user interface design. Developed in the early 1990s, they remain fundamental for designing intuitive and user-friendly systems. These heuristics serve as general rules of thumb rather than strict usability guidelines [28, 29]:

1. **Visibility of system status**: Keep users informed about what is going on through appropriate feedback.
2. **Match between system and the real world**: Use familiar language and conventions that match user expectations.
3. **User control and freedom**: Allow users to undo or redo actions and avoid getting stuck in unintended states.
4. **Consistency and standards**: Follow platform conventions to reduce learning effort.
5. **Error prevention**: Design systems that prevent problems before they occur.
6. **Recognition rather than recall**: Minimize users' memory load by making options and actions visible.
7. **Flexibility and efficiency of use**: Allow expert users to speed up interaction through shortcuts and customization.
8. **Aesthetic and minimalist design**: Avoid unnecessary elements that distract from the main tasks.
9. **Help users recognize, diagnose, and recover from errors**: Use clear, plain-language error messages.
10. **Help and documentation**: Provide easy access to helpful information when needed.

In addition to Nielsen's work, Don Norman outlined six fundamental principles of interaction design that focus on making systems understandable and usable [26]:

- **Visibility**: The user should be able to see what actions are possible.
- **Feedback**: The system should provide immediate and informative responses to user actions.
- **Constraints**: Design elements should restrict interactions to reduce user error.
- **Mapping**: There should be a clear relationship between controls and their effects.
- **Consistency**: Similar actions should produce similar outcomes.
- **Affordance**: Interface elements should intuitively suggest their usage.

Many of Norman's principles overlap conceptually with Nielsen's heuristics, supporting key themes in user-centered interaction design.

In addition to heuristics and principles, several empirical laws help explain user performance and behavior during interaction:

Fitts's Law states that the time required to move to a target is a function of the distance and size of the target. This implies that larger and closer targets are faster to interact with, which is essential when designing touch interfaces and interactive elements.

Hick's Law states that the time it takes to make a decision increases with the number and complexity of available choices. Reducing the number of visible options can improve decision-making speed, especially in time-sensitive contexts.

Miller's Law suggests that the average person can hold about seven items (plus or minus two) in their working memory at once. This principle underscores the importance of avoiding information overload in interface designs.

Jakob's Law emphasizes that users spend most of their time on other websites or systems and therefore expect new interfaces to function similarly to those they already know. Consistency with familiar interaction patterns reduces the learning curve and increases usability.

Together, these laws provide designers with practical guidelines for building efficient and cognitively appropriate interfaces [30–33].

Chapter 3

Interaction & Tool Design

In this chapter, we will discuss and propose an interaction design and tool design for digital football tactics boards. It is important that all interactions can be performed using one or more fingers, but using other input devices, such as pens, is also allowed to simulate a finger touching the screen. The intended target devices are multi-touch compatible devices, allowing multi-touch inputs. Devices with larger screens are preferred, as smaller screens, such as those of smartphones, can become cluttered when displaying both the pitch and the interface elements, resulting in reduced visibility and more cumbersome interaction. Larger screens also allow larger elements, reducing the time required for an interaction, according to Fitts's Law [30].

The proposed interactions are structured into four main categories, namely player selection, player movement, player scaling, and view manipulation interactions. Each group serves a different purpose and is designed so that they do not overlap. However, in practice, interactions, especially for selection and movement interactions, often need to be combined to complete a task. In addition, interactions are influenced by their interaction context and their social context, which can affect gesture performance, precision, and collaboration [34]. Our design includes both basic interactions that are intuitive and accessible for casual users, as well as more advanced features and interactions that allow for precise manipulation. These advanced features exploit domain-specific knowledge and are best utilized by experienced users. This aligns with Nielsen's heuristic 7 (flexibility and efficiency of use), which emphasizes the importance of the possibility to speed up interactions.

Optimal use of the tool requires familiarity with its features and corresponding gestures. Therefore, comprehensive documentation in the form of a user manual is essential to support effective use. This is supported by Nielsen's heuristic 10 (help and documentation). To help the user, error messages appear if users try to perform actions that are not allowed, following Nielsen's heuristic 9 (help users recognize, diagnose, and recover from errors) and

Norman's principle of feedback [28,29].

For simplicity, we focus solely on players as interactive objects on the board. Other items, such as balls or training equipment, could be substituted in their place, with similar interaction behavior. The players of the home team are represented by circles, while those of the away team are represented by crosses. Their touch detection zones are the same regardless of the visual representation. Each team is assumed to consist of eleven players, initially positioned on the pitch in a default 4-4-2 formation (one goalkeeper, four defenders, four midfielders, and two strikers). This formation is chosen because it is one of the most basic and widely used formations in football [35]. Each team is assumed to have eleven players on the pitch, as the focus lies on in-game situations rather than training exercises. The players are already placed on the pitch to save time in placing each player individually on the pitch. Each player is also assumed to have an associated position in order to leverage position-dependent interactions.

By default, the pitch is displayed fully and centered on the screen. The pitch is oriented vertically so that, for example, the left winger appears on the left side of the screen.

Touch-based interactions are the most widely used methods across digital interfaces due to their intuitiveness and simplicity. For a football tactics board aimed at touch devices, such interactions must be supported. As the tool should work on any multi-touch compatible device, only touch-based interactions are considered.

3.1 Terminology

Before discussing interactions, we define terms that appear in the following sections, starting with different touch input gestures performed by users interacting with digital devices.

Touch A contact of the screen with a fingertip. Duration and movement are not defined.

Tap A brief touch where the fingertip is released shortly after making contact with the screen. Commonly used to trigger buttons.

Double-tap Two rapid taps in quick succession.

Long press A continuous touch lasting at least 0.5 seconds without movement.

Drag The continuous movement of a fingertip across the screen while maintaining contact.

Drag-and-drop A combination of dragging and then releasing the fingertip to complete the movement.

Flick A quick and short swipe in one direction.

Pinch A gesture involving two fingertips moving toward each other.

Spread The opposite of a pinch, with two fingertips moving away from each other.

Lasso A gesture that involves drawing a free-form, closed shape around an area on the screen and then releasing the touch. The shape is closed by connecting the start and end points with a straight line if they do not meet.

Frame A gesture that involves touching the screen at one point, dragging to another, and releasing, resulting in a rectangular selection box defined by the diagonal corners.

Two-finger-rotation A gesture performed by placing two fingertips on the screen and rotating them clockwise or counterclockwise.

Next, we need to define some other terms that are used in the following sections.

Group A set of at least two players.

Arrangement A set of at least two players, each associated with a specific location on the pitch.

Unit All players in one line of the formation. For example, the defensive unit includes all defenders.

Grid The pitch is divided into a grid consisting of six rows and six columns. Rows are labeled 1 to 6 from bottom to top, and columns are labeled A to F from left to right. These labels appear outside the pitch boundaries. The grid can be toggled using the 'Grid' button.

Concurrent Interactions Interactions that can be performed simultaneously by different touches without interfering with one another.

Locked Selection A selection state that remains active and exclusive until explicitly removed by the user. In contrast, unlocked selections may be automatically cleared depending on the interaction context. Locked selections are preferred for detailed adjustments, whereas unlocked selections are better suited for quick or collaborative interactions.

Touch-assistant-zone A semi-transparent circle linked to a player by a thin line. Each player has its own zone, placed near them but not overlapping. These zones enlarge the touch detection zone of the linked player, increasing the user experience. They also allow players to be manipulated without covering them with the finger. Touch-assistant-zones are disabled by default and can be activated using the 'Touch-Assistant-Zones' button in the menu.

3.2 Player Selection Interactions

The first main category of interactions focuses on the selection of players. We begin by describing interactions that select a single player, followed by those that allow selecting multiple players as a group.

Selection is an essential prerequisite for meaningful interaction with a tactics board. Most existing digital tools offer only basic or limited options for selection, typically restricted to simple touch gestures. To improve both usability and functionality, we introduce a broader set of selection methods. These are adopted from existing tools, inspired by widely adopted interaction paradigms from other applications, as well as novel techniques tailored to the specific needs of football tactics boards.

Each selection interaction is described using five aspects:

- *Usage:* Explains which types of movement interactions it enables.
- *Selection:* Describes what is selected.
- *Selection-Interaction:* Describes the interaction required to perform the selection.
- *Deselection-Interaction:* Describes the interaction required to remove the selection.
- *Concurrent Use:* States whether the interaction can be performed concurrently.

To ensure that the user is always aware of the current selection state, selected players or areas are visually highlighted using a thin, dotted blue outline. This visual feedback reduces ambiguity and supports rapid recognition. It also aligns with Nielsen's heuristic 1 (visibility of system status) and Norman's principle of feedback.

3.2.1 Selection of One Player

Selecting an individual player is one of the most common and essential operations when interacting with a tactics board.

Touch a Player (TP)

Usage: Must be used before the movement interactions *Drag-and-Drop a Player (DaDP)* or *Flick a Player (FP)*.

Selection: The touched player, or the player associated with the touched touch-assistant-zone, is selected. The selection is not locked.

Selection-Interaction: Touch the player to be selected, or touch their associated touch-assistant-zone.

Deselection-Interaction: Release the touch.

Concurrent Use: Can be performed concurrently.

The *TP* interaction forms the basis for individual player manipulation. It supports fast and precise selection and is especially useful in collaborative environments where multiple users may interact with different players simultaneously. The use of touch-assistant-zones enhances accuracy by enlarging the effective touch detection zone, which is especially valuable on smaller screens, but requires an additional step to activate them.

Long-Press a Player (LPP)

Usage: Must be used before the movement interactions *Drag-and-Drop a Player (DaDP)*, *Flick a Player (FP)*, *Draw a Player's Path (DPP)*, *Long-Press a Point on the Pitch for a Player (LPPPP)*, *Swap Players (SP)*, *Linked Movement (LM)* or *Pair Movement (PM)*.

Selection: The long-pressed player, or the player associated with the long-pressed touch-assistant-zone, is selected. The selection is locked.

Selection-Interaction: Long-press the player to be selected or their touch-assistant-zone.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

The *LPP* interaction introduces a method of selecting a player for advanced operations. Locked selection is crucial for chaining multiple interactions or for exclusive or advanced interactions. Locking also gives the option of focusing the attention of the users on the current selection. Long-press is a common gesture for accessing more complex options in mobile UIs, making it familiar to users. This aligns with Nielsen's heuristic 4 (consistency and standards) and with Jakob's law. The inability to perform it concurrently prevents conflicts when multiple players require exclusive manipulation, following Nielsen's heuristic 5 (error prevention) and Norman's principle of constraints [28,29,33]. When players are positioned closely together, accurately long-pressing each one can become difficult. In such cases, enabling touch-assistant-zones can help improve precision.

Double-Tap a Player (DTP)

Usage: Must be used before the movement interaction *Tap a Player's Destination (TPD)*.

Selection: The double-tapped player is selected and the selection is locked.

Selection-Interaction: Double-tap the player to be selected.

Deselection-Interaction: Tap the 'X' button in the top right corner or complete the movement via *TPD*.

Concurrent Use: Cannot be performed concurrently.

The *DTP* interaction mimics the physical gesture of picking up a magnetic piece from a real tactics board, adhering to Nielsen's heuristic 2 (match between system and the real world). Upon selection, the player is temporarily removed from the pitch and displayed in a dedicated queue at the bottom right of the screen, awaiting repositioning. This mechanism is particularly effective for the reformation of player arrangements. By supporting the sequential selection of multiple players through a FIFO (First-In-First-Out) queue, this interaction allows coaches to temporarily remove players from the pitch to clean up the board. They can then reintroduce those players in a new arrangement to illustrate new tactical instructions. The queue system also provides clarity and control over the repositioning process. The player displayed at the rightmost position in the queue is always the next to be placed. If a coach wishes to prioritize a different player, tapping that player within the queue shifts them to the front, making them the next to be placed on the pitch.

Summary

Table 3.1 summarizes the interactions for the selection of one player.

Name	Selection	Interaction	Movement
Touch a Player (TP)	Player (not locked)	Touch a player	*DaDP, FP*
Long-Press a Player (LPP)	Player (locked)	Long-press a player	*DaDP, FP, DPP, LPPPP, SP, LM, PM*
Double-Tap a Player (DTP)	Player (locked)	Double-tap a player	*TPD*

Table 3.1: Interactions for the selection of one player

3.2.2 Selection of a Group of Players

After introducing methods for selecting individual players, it is essential to provide interactions for selecting multiple players as a group. Group selection significantly reduces the time and effort required to manipulate several players simultaneously, improving efficiency when giving tactical instructions. Furthermore, football inherently involves structured groupings, such as formations, units, and spatial positioning, that are naturally suited to grouped interactions. However, group selections can also introduce a small overhead. In cases where only a few players need to be selected, it may actually be quicker and more intuitive to select them individually using single-player interactions.

Lasso (L)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Pinch a Group (PG)* or *Spread a Group (SpG)*.

Selection: All players within the lassoed area are selected and the selection is not locked.

Selection-Interaction: Lasso the area in which all players to be selected are located. To successfully achieve the selection, the initial touch point must not be a player.

Deselection-Interaction: Tap the small 'x' button on the border of the selected area.

Concurrent Use: Can be performed concurrently.

The *L* interaction is inspired by common tools in drawing and graphic applications and is included to offer maximum flexibility in shaping a selection. Unlike rectangular selection tools, the lasso allows users to draw around players using arbitrary shapes, enabling precise inclusion or exclusion of specific individuals. It is consistent with Nielsen's heuristic 4 (consistency and standards) and Jakob's law [28,29,33].

This method is particularly well suited for selecting custom subsets of players during tactical transitions. Since the selection is not locked, it supports quick and temporary selections. Lasso also supports concurrent use, allowing multiple users to draw different selections at the same time, which is an important feature in collaborative coaching or analysis settings.

Lasso and Long-Press (LaLP)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All players within the lassoed area are selected and the selection is locked.

Selection-Interaction: Perform a lasso interaction as described in *L*, but long-press before releasing the touch. After the selection of the first area, additional areas can be added to the selection by tapping the 'Add' button and repeating *LaLP*. When all desired areas have been added, the 'Add' button must be tapped again to confirm the final group.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

While the standard *L* interaction supports fast and temporary selections, *LaLP* extends it by allowing the selection to be locked, making it persistent and eligible for more complex movement interactions. Additionally, it becomes visually emphasized on the board. This ensures that active (e.g., the coach) and passive users (e.g., players during a tactical session) immediately recognize the current tactical focus. As locking requires slightly more time, selection for single and short adjustments may be faster without locking.

Furthermore, the ability to add multiple lassoed areas allows flexible groups, even across unconnected regions of the pitch. This accommodates realistic use cases in football, such as combining players from different units. However, since the selection is locked, *LaLP* is best suited for deliberate single-user interactions rather than fast-paced collaboration.

Frame (F)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Pinch a Group (PG)* or *Spread a Group (SpG)*.

Selection: All players within the framed area are selected and the selection is not locked.

Selection-Interaction: Tap the 'Frame' button and frame the area in which all players to be selected are located. To successfully achieve the selection, the initial touch point must not be a player.

Deselection-Interaction: Tap the small 'x' button on the border of the selected area.

Concurrent Use: Can be performed concurrently.

The *F* interaction provides a quick way to select players within a clearly defined rectangular area. Known from many digital applications, such as image editors, file managers, and graphic design tools, framing is a widely understood and intuitive gesture. This aligns with Nielsen's heuristic 4 (consistency and standards) and Jakob's law [28, 29, 33]. Unlike *L*, which allows free-form selection, *F* is optimized for speed. Users only need to draw a diagonal to complete the frame, which is often faster and easier to perform, especially on touchscreens.

This interaction should not be used if precise selection is needed as the rectangular shape cannot be modified. Using a rectangular boundary is particularly helpful when players are already aligned in structured formations, such as a back four or midfield block. The time overhead introduced by the need to first tap the 'Frame' button may be outweighed in situations where framing needs less time than lassoing. Because *F* is non-locking, it supports fast-paced, collaborative use cases where temporary selections are made for immediate adjustments.

Frame and Long-Press (FaLP)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All players within the framed area are selected and the selection is locked.

Selection-Interaction: Perform a frame interaction as described in *F*, but long-press before releasing the touch. After the selection of the first area, additional areas can be added to the selection by tapping the 'Add' button and repeating *FaLP*. When all desired areas have been added, the 'Add' button must be tapped again to confirm the final group.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

Similarly to *LaLP*, *FaLP* extends the basic *F* interaction by allowing users to lock their selection and combine unconnected areas of the pitch. As *FaLP* is the combination of *LaLP* and *F*, its detailed mechanics are not repeated, as they apply analogously. However, it is worth noting that tapping the 'Frame' button and executing a long-press introduces a time overhead that may outweigh the benefits, making *FaLP* less efficient in some situations.

Repeat LPP (RLPP)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All individually long-pressed players are selected and the selection is locked.

Selection-Interaction: Repeat *LPP* with different players.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

RLPP is essentially an extension of the *LPP* interaction, applying the same interaction repeatedly to select multiple players. Therefore, the details and considerations described for *LPP* also apply to *RLPP*. *RLPP* is a powerful interaction to select non-adjacent players, particularly useful when the desired players are scattered across the pitch. Unlike group-based methods such as *L* or *F*, *RLPP* allows full control over which individual players are included, regardless of their spatial relation.

Despite being slightly more time-consuming than area-based selections, *RLPP* ensures deliberate selection and is ideal when precise grouping matters more than speed.

Select a Team (ST)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Select a Formation (SF)*, *Pinch a Group (PG)*, *Spread a Group (SpG)*, *Mirror Players of one Half of the Pitch (MPHP)* or *Maintain Shape (MS)*.

Selection: All players in the selected team are selected and the selection is locked.

Selection-Interaction: Tap the 'Home' or 'Away' button to select one or both teams.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

Selecting an entire team is a fundamental function, enabling coaches to quickly manipulate all players of that team. With *ST*, an entire team can be selected simply by tapping one of the two team buttons, which is often faster than performing individual or other group selections. Additionally, by tapping both team buttons, users can select all players on the pitch, allowing for simultaneous adjustment of all player positions.

Select an Unit (SU)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All players in the selected unit(s) are selected and the selection is locked.

Selection-Interaction: Tap the 'Unit' button and select an unit. Selecting or deselecting multiple units makes it possible to combine them.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

In football, players are often organized into units based on their tactical roles, such as defenders, midfielders, or forwards. Selecting an entire unit at once is highly beneficial for coaches, as it allows quick adjustments of specific tactical groups rather than individual players. The *SU* interaction leverages this domain-specific structure to improve efficiency and clarity.

Users can select not only one unit, but also multiple units to manipulate them all together. This gives the user even more flexibility in making adjustments. Unlike broader team selections, *SU* targets tactical subgroups, allowing fine-grained control over formations and strategies.

Select a Row/Column (SRC)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All players within the selected rows and columns are selected and the selection is locked.

Selection-Interaction: Tap the 'Grid' button to activate the grid on the pitch and tap the labels of the rows and columns to be selected.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

Football pitches are often mentally and visually divided into zones or grids by coaches to define spatial responsibilities. The *SRC* interaction supports this tactical segmentation by allowing users to select all players within specific rows or columns of a virtual grid. In our system, the pitch is divided into six vertical columns and six horizontal rows. This configuration is chosen to reflect common tactical zoning: the six columns correspond to typical divisions into central and wide channels, while the six rows represent layers of positioning from defense to attack, providing finer vertical granularity [36].

By tapping the labels corresponding to the desired rows and columns, users can quickly and precisely select all players positioned in those areas. This enables zone-based reorganization, such as shifting an entire midfield line forward.

Although *SRC* adds structure to the selection process, it is less flexible than free-form selection methods when players fall outside conventional grid boundaries. However, for standard formations, *SRC* can be a highly efficient method of group selection.

Select a Cell (SC)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All players of the selected cells are selected and the selection is locked.

Selection-Interaction: Tap the 'Grid' button to activate the grid on the pitch and tap the cells to be selected.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

The *SC* interaction builds on the grid concept introduced in *SRC* by allowing more precise selections at the level of individual cells within the pitch grid.

Activating the grid overlay lets users tap specific cells to select only the players positioned within those exact areas. This granular approach offers enhanced flexibility compared to selecting entire rows or columns, enabling coaches to target very specific zones for tactical adjustments. *SC* is especially useful when a coach wants to isolate a small number of players or focus on critical areas of play, such as selecting midfielders positioned in a particular central cell.

Players within a locked selection can be saved as either a group or an arrangement using the 'Save Group' or 'Save Arrangement' buttons. While saving an arrangement also saves the corresponding group, it additionally records the specific positions of the players within that group, allowing users to recall both the selection and the exact player locations later. To distinguish different saved groups and arrangements, names can be given to groups and arrangements.

Select a Group (SG)

Usage: Must be used before the movement interactions *Drag-and-Drop a Group (DaDG)*, *Flick a Group (FG)*, *Draw a Group's Path (DGP)*, *Long-Press a Point on the Pitch (LPPP)*, *Select Arrangement (SA)*, *Pinch a Group (PG)*, *Spread a Group (SpG)* or *Maintain Shape (MS)*.

Selection: All players in the selected group(s) are selected and the selection is locked.

Selection-Interaction: Tap the 'Group' button and select one of the groups in the list that pops up. Selecting or deselecting multiple groups makes it possible to combine them.

Deselection-Interaction: Tap the 'X' button in the top right corner.

Concurrent Use: Cannot be performed concurrently.

Selecting a saved group allows users to quickly reselect previously defined player combinations, significantly reducing the effort required to rebuild complex selections. However, if only a few players need to be selected, alternative methods may be faster since selecting a group involves tapping two buttons and choosing from a potentially long list of saved groups. More than one group can be selected from the list, combining them into the current selection, enabling more flexibility. This reflects Nielsen's heuristic 7 (flexibility and efficiency of use) [28,29].

Summary

Table 3.2 summarizes the interactions for the selection of a group of players.

3.3 Player Movement Interactions

The second main category of interactions we address concerns the movement of players on the tactics board. These interactions are closely linked to the selection interactions, as players generally need to be selected before any movement can take place. We separate the movement instructions into three parts: movements of an individual player, movements affecting several players, and behavior modifiers that automatically influence player movement.

Movement interactions are as essential as selection interactions because they directly manipulate the tactics board. This is the core tool for visualizing player positioning in specific game situations. Since players must frequently change positions based on evolving contexts during a game, it is critical that tactics board tools accurately and intuitively represent these movements. However, our review revealed that most existing tools provide only a limited set of movement interactions, restricting flexibility and efficiency. To address this gap, we present a set of movement interactions designed to enhance functionality and improve the overall user experience. Similarly to our approach with selection interactions, well-established methods from current tools are incorporated, combined with widely recognized interaction patterns, alongside innovative techniques uniquely optimized for football tactics boards.

Each movement interaction is described using four aspects:

- *Usage:* Explains which types of selection interactions must precede it.
- *Movement:* Describes how the players are moved on the pitch.

Name	Selection	Interaction	Movement
Lasso (L)	Area (not locked)	Lasso an area	*DaDG, FG, PG, SpG*
Lasso and Long-Press (LaLP)	Area (locked)	Lasso an area and long-press	*DaDG, FG, DGP, LPPP, PG, SpG, MS*
Frame (F)	Area (not locked)	Frame an area	*DaDG, FG, PG, SpG*
Frame and Long-Press (FaLP)	Area (locked)	Frame an area and long-press	*DaDG, FG, DGP, LPPP, PG, SpG, MS*
Repeat LPP (RLPP)	Players (locked)	Repeat LPP	*DaDG, FG, DGP, LPPP, PG, SpG, MS*
Select a Team (ST)	Team (locked)	Tap 'Home' or'Away'	*DaDG, FG, DGP, LPPP, SF, PG, SpG, MPHP, MS*
Select an Unit (SU)	Unit(s) (locked)	Tap 'Unit' and select the unit(s)	*DaDG, FG, DGP, LPPP, PG, SpG, MS*
Select a Row/Column (SRC)	Rows(s) and Column(s) (locked)	Tap row and column label(s)	*DaDG, FG, DGP, LPPP, PG, SpG, MS*
Select a Cell (SC)	Cell(s) (locked)	Tap cell(s)	*DaDG, FG, DGP, LPPP, PG, SpG, MS*
Select a Group (SG)	Group(s) (locked)	Tap 'Group' and select the group(s)	*DaDG, FG, DGP, LPPP, SA, PG, SpG, MS*

Table 3.2: Interactions for the selection of a group of players

- *Movement-Interaction:* Describes the interaction required to perform the movement.
- *Concurrent Use:* States whether the interaction can be performed concurrently.

'Undo' and 'Redo' buttons are ubiquitous across digital tools, including tactics boards. The 'Undo' button reverses the most recent action, while 'Redo' reapplies an action that was undone. Given their widespread familiarity and consistent functionality across applications, we do not elaborate further on these controls here. However, since they primarily affect player positioning, we include them in this section for completeness. Incorporating these buttons is essential for delivering a user-friendly experience, as users frequently expect the ability to easily correct or reinstate their actions. This design choice is supported by Nielsen's heuristics 3 (user control and freedom) and 4 (consistency and standards) and Jakob's law [28,29,33].

3.3.1 Movement of One Player

Just as we have selection interactions designed for individual players, it is essential to have movement interactions that allow moving one player at a time.

Drag-and-Drop a Player (DaDP)

Usage: Must be used after the selection interactions *Touch a Player (TP)* or *Long-Press a Player (LPP).*

Movement: The selected player moves according to the drag interaction.

Movement-Interaction: Drag the selected player to the desired location and release the touch. For *TP*, the touch must be on the player or its touch-assistant-zone, for *LPP* the touch can be anywhere on the pitch, as the selection is locked. If *LPP* is used for the selection, a second drag can be performed without releasing the initial touch to fine-tune the player's position.

Concurrent Use: Can only be performed concurrently with *TP*.

Drag-and-drop is a fundamental interaction because it closely mimics the natural, physical way coaches manipulate players on a real tactics board with magnets. In addition, drag-and-drop is widely used in digital applications, as well as it was present in all reviewed tools. It is highly intuitive, requiring minimal learning effort, making it accessible for users of all skill levels. Furthermore, drag-and-drop is fast and direct, allowing users to immediately see the effect of their movement, which is essential for real-time tactical adjustments. Because of these advantages, drag-and-drop is an important method for moving individual players efficiently and effectively on football tactics boards. Drag-and-drop also aligns with Nielsen's heuristics 2 (match between system and the real world) and 4 (consistency and standards) and Jakob's law [28,29,33].

DaDP can be used after either *TP* or *LPP*, each supporting different use cases. *TP* allows for fast, simple, and intuitive drag-and-drop movements, requiring only a touch and drag. It also supports multiple simultaneous touches, enabling coaches to move several players at once during a tactic explanation. This mirrors real-world coaching gestures and enhances the fluidity of demonstrations. On the other hand, *LPP* requires a long-press to lock the selection, adding some overhead but offering benefits when focusing on a single player who may be moved repeatedly. *LPP* also allows users to avoid covering the player with their finger during movement, as dragging can start anywhere on the pitch, and provides the option for fine-tuning the player's position with an additional drag gesture.

Flick a Player (FP)

Usage: Must be used after the selection interactions *Touch a Player (TP)* or *Long-Press a Player (LPP)*.

Movement: The selected player moves in the direction of the flick, where the distance of the movement depends on the intensity of the flick.

Movement-Interaction: Flick the selected player in the desired direction. For *TP*, the touch must be on the player or its touch-assistant-zone, for *LPP* the touch can be anywhere on the pitch, as the selection is locked.

Concurrent Use: Can only be performed concurrently with *TP*.

Flicking a player is a quick and expressive way to indicate a player's intended movement direction and approximate distance without the need for precise placement. This interaction is especially useful when the exact endpoint is less important than the general movement idea, such as signaling a forward run or a quick shift in position. Flick gestures are natural on touch devices and require minimal effort, enabling fast and fluid tactical demonstrations.

FP can be used following *TP* or *LPP* selection methods, similar to *DaDP*. *TP* allows for quick, simple flicks, while *LPP* supports more focused and repeated movements. The details are not repeated here as they align closely with those described for *DaDP*.

Draw a Player's Path (DPP)

Usage: Must be used after the selection interaction *Long-Press a Player (LPP)*.

Movement: The selected player moves along the drawn path. The movement starts as soon as the draw touch is released. If the start point of the path is not the player's location, the player jumps to the start point of the path. The player remains at the end point of the path after the movement.

Movement-Interaction: Tap the 'Draw Path' button and drag to draw a path on the pitch.

Concurrent Use: Cannot be performed concurrently.

DPP provides a way to visualize a player's movement path without requiring continuous touch on the screen during movement. This interaction is especially useful when the user wants to focus on demonstrating the player's path clearly. Although it requires tapping an additional button to activate the drawing mode, this overhead can be worthwhile in situations where player movement strategies over time need to be demonstrated. *DPP* is not suited for quick or concurrent movements, but is effective when precision and clarity of movement paths are the priorities.

Long-Press a Point on the Pitch for a Player (LPPPP)

Usage: Must be used after the selection interaction *Long-Press a Player (LPP)*.

Movement: The selected player moves toward the touched point until the touch is released or the player reaches the point.

Movement-Interaction: Long-press the desired point on the pitch.

Concurrent Use: Cannot be performed concurrently.

Pressing is a fundamental tactical concept in football, involving players applying pressure to opponents to regain possession [37,38]. *LPPPP* allows coaches to intuitively demonstrate pressing behavior with a single player by simulating continuous movement toward a target area. The advantage of *LPPPP* over other movement interactions is that using *LPPPP*, the selected player automatically moves slowly in the desired direction.

Tap a Player's Destination (TPD)

Usage: Must be used after the selection interaction *DTP*.

Movement: The first player in the selection queue is placed at the touched location on the pitch.

Movement-Interaction: Tap the desired destination point on the pitch.

Concurrent Use: Cannot be performed concurrently.

TPD completes the pick-and-place interaction inspired by physical tactics boards, allowing for precise positioning of players after they have been picked up. While it introduces an extra step compared to drag-and-drop, it can be especially helpful when the user wants to first clear or clean up the board before deciding where to place the player next.

If several players are picked up, *TPD* can be performed repeatedly to place them. As already introduced, a FIFO queue manages the picked up players. As the selection is locked, all players must be placed before using other interactions.

Summary

Table 3.3 summarizes the interactions for the movement of one player.

Name	Movement	Interaction	Selection
Drag-and-Drop a Player (DaDP)	Player moves according to the drag	Drag-and-drop a player	*TP*, *LPP*
Flick a Player (FP)	Player moves according to the flick	Flick a player	*TP*, *LPP*
Draw a Player's Path (DPP)	Player moves along the path	Tap 'Draw Path' and draw a path	*LPP*
Long-Press a Point on the Pitch for a Player (LPPPP)	Player moves toward the touched point	Long-press a point on the pitch	*LPP*
Tap a Player's Destination (TPD)	Player is placed at the touched point	Tap a point on the pitch	*DTP*

Table 3.3: Interactions for the movement of one player

3.3.2 Movement of a Group of Players

Moving groups of players is at least as important as moving individual players, as it can significantly reduce the time and effort required to reposition multiple players simultaneously. Efficient group movement supports more fluid tactic adjustments and helps coaches quickly visualize team formations or coordinated maneuvers.

Drag-and-Drop a Group (DaDG)

Usage: Must be used after the selection interactions *Lasso (L)*, *Lasso and Long-Press (LaLP)*, *Frame (F)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC) Select a Cell (SC)* or *Select a Group (SG)*.

Movement: The selected players move according to the drag interaction, i.e., each player moves by the same relative distance and direction from its original position.

Movement-Interaction: Drag the selected players to the desired location and release the touch. For *L* or *F*, the touch must be inside the selected area, for *LaLP*, *FaLP*, *RLPP*, *ST*, *SU*, *SRC*, *SC* or *SG*, the touch can be anywhere on the pitch, as the selection is locked. If a locked selection is used for the selection, a second drag can be performed without releasing the initial touch to fine-tune the player's position.

Concurrent Use: Can only be performed concurrently with *L* or *F*.

Just as individual players benefit from drag-and-drop for quick repositioning, groups of players require this interaction to efficiently adjust team shape and tactics. Moving multiple players individually by the same vector is time-consuming and inefficient.

Like its single-player counterpart (*DaDP*), *DaDG* supports both locked and unlocked selections. Unlocked selections enable faster, more fluid movements and allow concurrent manipulation of multiple groups, supporting rapid tactical demonstrations. Locked selections provide greater precision and spatially independent touch, which is useful when detailed positioning is critical.

Flick a Group (FG)

Usage: Must be used after the selection interactions *Lasso (L)*, *Lasso and Long-Press (LaLP)*, *Frame (F)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC)*, *Select a Cell (SC)* or *Select a Group (SG)*.

Movement: The selected players move in the direction of the flick, where the distance of the movement depends on the intensity of the flick.

Movement-Interaction: Flick the selected players in the desired direction. For *L* or *F*, the touch must be inside the selected area, for *LaLP*, *FaLP*, *RLPP*, *ST*, *SU*, *SRC*, *SC* or *SG*, the touch can be anywhere on the pitch, as the selection is locked.

Concurrent Use: Can only be performed concurrently with *L* or *F*.

Flicking a group provides a quick and expressive way to indicate approximate player movements, such as a collective forward push, without specifying exact endpoints.

FG extends the flick gesture from single-player movement (*FP*) to groups, preserving the benefits of fast and intuitive manipulation while allowing for coordinated adjustments of multiple players at once. It supports both locked and unlocked selections, balancing between rapid fluid movement (unlocked) and controlled precision (locked).

Draw a Group's Path (DGP)

Usage: Must be used after the selection interactions *Lasso and Long-Press (LaLP)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC)*, *Select a Cell (SC)* or *Select a Group (SG)*.

Movement: The selected players move along the drawn path. They keep the relative distances between each other. The movement starts as soon as the draw touch is released. If the start point of the path is not the group's location, the players jump to the start point of the path. The players remain at the end point of the path after the movement.

Movement-Interaction: Tap the 'Draw Path' button and drag to draw a path on the pitch.

Concurrent Use: Cannot be performed concurrently.

DGP allows coaches and users to illustrate complex coordinated movements of multiple players along a specified trajectory, which is especially useful for demonstrating planned attacking or defensive runs. Unlike drag-and-drop or flick gestures, which focus on endpoint positioning, *DGP* emphasizes the movement path itself, providing a clear visual representation of timing, spacing, and coordination within the group. *DGP* extends *DPP*, and works analogously.

Long-Press a Point on the Pitch (LPPP)

Usage: Must be used after the selection interactions *Lasso and Long-Press (LaLP)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC)*, *Select a Cell (SC)* or *Select a Group (SG)*.

Movement: The selected players move slowly toward the touched point until the touch is released or the players reach the point.

Movement-Interaction: Long-press the desired point on the pitch.

Concurrent Use: Cannot be performed concurrently.

LPPP extends the concept of pressing from a single player to a coordinated group action, which is a fundamental tactical principle in football. Pressing as a collective effort requires players to move together toward a target area, often applying pressure to regain possession or cut passing lanes [37,38].

This interaction enables users to demonstrate pressing strategies more realistically by allowing an entire group to move slowly and purposefully toward a point. It helps visualize defensive organization and collective response.

Select an Arrangement (SA)

Usage: Must be used after the selection interaction *Select a Group (SG)*.

Movement: The selected players move to the corresponding stored locations of the arrangement.

Movement-Interaction: Tap on the desired arrangement in the saved arrangements list that pops up.

Concurrent Use: Cannot be performed concurrently.

SA is crucial for efficiently managing recurring tactical setups, such as set-pieces, defensive shapes, or attacking formations. By saving and later recalling player arrangements, coaches and analysts can quickly switch between complex positional setups without manually repositioning each player every time. This not only saves time, but also reduces errors or inconsistencies in repositioning players, ensuring accuracy and consistency in tactical demonstrations.

In order to use *SA*, at least one arrangement must be saved beforehand.

Select a Formation (SF)

Usage: Must be used after the selection interaction *Select a Team (ST)*.

Movement: The players of the selected team move to the locations defined by the formation.

Movement-Interaction: Tap on the 'Formation' button and tap on the desired formation in the list that pops up.

Concurrent Use: Cannot be performed concurrently.

SF is fundamental because football formations represent the foundation of a team structure and tactics. Common formations like 4-3-3 or 4-4-2 define player roles and positioning on the pitch. Enabling users to quickly apply these formations saves significant time and effort compared to manually positioning each player.

By providing a list of common formations, *SF* ensures usability across a wide range of tactical scenarios. Since the number of formations used in practice is finite, we include them in the list. These formations are: 4-4-2, 4-4-1-1, 4-1-2-1-2, 4-1-3-2, 4-2-3-1, 4-2-2-2, 4-2-4, 4-3-3, 4-1-4-1, 4-3-2-1, 4-5-1, 4-6-0, 3-4-3, 3-4-2-1, 3-4-1-2, 3-2-4-1, 3-1-3-3, 5-2-2-1, 5-4-1, 3-5-2, 3-5-1-1, and 5-3-2 [39].

Pinch a Group (PG)

Usage: Must be used after the selection interactions *Lasso (L)*, *Lasso and Long-Press (LaLP)*, *Frame (F)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC)*, *Select a Cell (SC)* or *Select a Group (SG)*.

Movement: The selected players move toward the center of the group, where the distance of the movement depends on the intensity of the pinch.

Movement-Interaction: Pinch either inside the selected area if using *L* or *F*, or anywhere on the pitch if using *LaLP*, *FaLP*, *RLPP*, *ST*, *SU*, *SRC*, *SC* or *SG*.

Concurrent Use: Can only be performed concurrently with *L* or *F*.

PG is a vital interaction for adjusting team compactness, especially in defensive scenarios where players need to tighten their formation to reduce space for opponents. By moving players closer to the group's center, coaches can quickly demonstrate strategies such as closing down passing lanes or maintaining a solid defensive block. This interaction enables users to efficiently represent real-world team dynamics without individually moving each player. Its intuitive pinch gesture aligns well with common touch interfaces, making it both natural and efficient, and supporting Nielsen's heuristic 2 (match between system and the real world) [27,28].

Similar to *DaDG* or *FG*, *PG* supports both locked and unlocked selections, making it suitable for quick and concurrent adjustments, as well as more focused and precise movements.

Spread a Group (SpG)

Usage: Must be used after the selection interactions *Lasso (L)*, *Lasso and Long-Press (LaLP)*, *Frame (F)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC)*, *Select a Cell (SC)* or *Select a Group (SG)*.

Movement: The selected players move away from the center of the group, where the distance of the movement depends on the intensity of the pinch.

Movement-Interaction: Spread either inside the selected area if using after *L* or *F*, or anywhere on the pitch if using after *LaLP*, *FaLP*, *RLPP*, *ST*, *SU*, *SRC*, *SC* or *SG*.

Concurrent Use: Can only be performed concurrently with *L* or *F*.

In opposition to *PG*, widening a group of players is essential to create space, stretch the opponent's formation, or open passing lanes. *SpG* enables users to quickly illustrate these tactical adjustments. Just like *PG*, *SpG* allows users to efficiently reflect real-world tactical adjustments without having to reposition each player individually. Its intuitive spreading gesture aligns naturally with familiar touch interactions, making it both natural and efficient and supporting Nielsen's heuristic 2 (match between system and the real world) [27,28].

By supporting both locked and unlocked selections, *SpG* allows for rapid repositioning as well as precise adjustments.

Mirror Players of one Half of the Pitch (MPHP)

Usage: Must be used after the selection interaction *Select a Team (ST)*.

Movement: The players from the selected team and the unselected half of the pitch move to mirrored locations of the players from the same team in the other half of the pitch. After the movement, the players of the selected team are symmetrically located. The matching between players and new locations is determined by minimizing the total distance all players must move.

Movement-Interaction: Tap the 'Mirror Left' or 'Mirror Right' button.

Concurrent Use: Cannot be performed concurrently.

MPHP addresses the common tactical need for symmetrical layouts, which are common in football strategies. By automatically mirroring one half of the pitch to the other, it significantly reduces the effort of adapting or replicating one side of the pitch manually, ensuring spatial consistency and saving time. This is especially useful in scenarios like setting up defensive lines or mirrored attacking patterns.

Note that this interaction requires both halves of the pitch to contain the same number of players. Players positioned near the vertical center line are excluded, as they do not have a clear counterpart and thus cannot be mirrored.

Swap Players (SP)

Usage: Must be used after the selection interaction *Long-Press a Player (LPP)*.

Movement: The selected player changes its location with the tapped player.

Movement-Interaction: Tap the 'Swap' button and then on a player other than the one selected.

Concurrent Use: Cannot be performed concurrently.

SP allows for quick reallocation of specific roles or positions between individual players without disturbing the overall team shape. This is particularly useful when players are uniquely identified, such as by name or number, or when players from opposing teams need to be switched.

Summary

Table 3.4 summarizes the interactions for the movement of a group of players.

3.3.3 Behavior Modifiers

Behavior modifiers affect all movement interactions and are applied after the movement of each moved player. After activating a behavior modifier, its effect remains until the modifier is removed. Behavior modifiers cannot be performed concurrently.

Each behavior modifier is described using four aspects:

Name	**Movement**	**Interaction**	**Selection**
Drag-and-Drop a Group (DaDG)	Players move according to the drag	Drag-and-drop a group	*L, LaLP, F, FaLP, RLPP, ST, SU, SRC, SC, SG*
Flick a Group (FG)	Players move according to the flick	Flick a group	*L, LaLP, F, FaLP, RLPP, ST, SU, SRC, SC, SG*
Draw a Group's Path (DGP)	Players move along the path	Tap 'Draw Path' and draw a path	*LaLP, FaLP, RLPP, ST, SU, SRC, SC, SG*
Long-Press a Point on the Pitch (LPPP)	Players move toward the touched point	Long-press a point on the pitch	*LaLP, FaLP, RLPP, ST, SU, SRC, SC, SG*
Select an Arrangement (SA)	Players move according to the arrangement	Select the arrangement	*SG*
Select a Formation (SF)	Players move according to the formation	Tap 'Formation' and select the arrangement	*ST*
Pinch a Group (PG)	Players move together	Pinch a group	*L, LaLP, F, FaLP, RLPP, ST, SU, SRC, SC, SG*
Spread a Group (SpG)	Players move apart	Spread a group	*L, LaLP, F, FaLP, RLPP, ST, SU, SRC, SC, SG*
Mirror Players of one Half of the Pitch (MPHP)	Players move to mirror the other half	Tap 'Mirror Left' or 'Mirror Right'	*ST*
Swap Players (SP)	Players swap locations	Tap 'Swap' and tap a player	*LPP*

Table 3.4: Interactions for the movement of a group of players

- *Usage:* Explains which types of selection interactions must precede it.
- *Activation-Interaction:* Describes the interaction required to activate the modifier.
- *Deactivation-Interaction:* Describes the interaction required to deactivate the modifier.
- *Movement:* Describes how affected players move.

Note that both the activation and deactivation interactions must be used directly after the selection interaction. To indicate which players are currently affected by behavior modifiers, a small red icon is displayed next to each affected player's symbol, aligning with Nielsen's heuristic 1 (visibility of system status) and Norman's principle of feedback [26,28,29].

Linked Movement (LM)

Usage: Must be used after the selection interaction *Long-Press a Player (LPP).*

Activation-Interaction: Tap the 'x-Axis' or 'y-Axis' button, then tap a player other than the currently selected one. Both axis buttons can be tapped to select both axes.

Deactivation-Interaction: Tap the non-selected player of the link.

Movement: If one of the two players, the selected or the tapped, moves, the other moves the same. Depending on the selected axes, the movement is mirrored along them.

Sometimes, players are meant to mirror each other's movement. For example, two fullbacks who move forward in a coordinated manner or one midfielder who moves forward and one who stays at the back. With *LM*, the motion of one player is reflected in another, either along the x-axis, y-axis, or both. This is useful to simulate tactical symmetry or synchronized defensive/offensive movements.

Note that if both players are moved together with another interaction, *LM* has no effect.

Pair Movement (PM)

Usage: Must be used after the selection interaction *Long-Press a Player (LPP).*

Activation-Interaction: Tap the 'Pair' button, then tap a player other than the currently selected one.

Deactivation-Interaction: Tap the non-selected player of the pair.

Movement: Both players, the selected and the tapped, always stick together, that is, they always touch each other, and if one moves, the other moves as well.

In many real-game scenarios, such as man-marking during set pieces, players operate in tightly coupled pairs. *PM* reflects this by keeping two players physically bound together. When one player moves, the other follows, always remaining in direct contact.

Snap-to-Grid (StG)

Usage: Can be used without a selection.

Activation-Interaction: Tap the 'Snap-To-Grid' button.

Deactivation-Interaction: Tap the ′Snap-To-Grid′ button.

Movement: All moved players move to the closest pitch grid line.

Tactical clarity often benefits from visual precision. *StG* ensures that players align neatly along a predefined pitch grid. It enhances the visual organization of the pitch and removes slight manual inaccuracies in player placement, especially during setup.

Maintain Shape (MS)

Usage: Must be used after the selection interactions *Lasso and Long-Press (LaLP)*, *Frame and Long-Press (FaLP)*, *Repeat LPP (RLPP)*, *Select a Team (ST)*, *Select an Unit (SU)*, *Select a Row/Column (SRC)*, *Select a Cell (SC)* or *Select a Group (SG)*.

Activation-Interaction: Tap the ′Shape′ button.

Deactivation-Interaction: Tap the ′Shape′ button.

Movement: The selected players move to maintain the shape.

In football, certain groups of players, such as the back four in a defense or a midfield block, are expected to move while maintaining a fixed formation (e.g., a straight line or a diamond). *MS* guarantees this by preserving the geometric configuration of the group as it moves.

Summary

Table 3.5 summarizes the behavior modifiers.

Name	**Movement**	**Interaction**	**Selection**
Linked Movement (LM)	Player moves if linked player moves	Tap ′x-Axis′ or ′y-Axis′ and tap a player	*LPP*
Pair Movement (PM)	Players stick together	Tap ′Pair′ and tap a player	*LPP*
Snap-to-Grid (StG)	Players move to the closest pitch grid line	Tap ′Snap-To-Grid′	-
Maintain Shape (MS)	Players move to maintain a shape	Tap ′Shape′	*LaLP*, *FaLP*, *RLPP*, *ST*, *SU*, *SRC*, *SC*, *SG*

Table 3.5: Behavior modifiers

3.4 Player Scaling and View Manipulation Interactions

The third and fourth main categories are discussed in this section. They are about interactions for scaling players and for manipulating the view. We begin with player scaling interactions.

3.4.1 Player Scaling

An issue in many tactical board tools is the disproportionate size of players relative to the pitch. Oversized player symbols can distort the perceived spatial relationships, reducing realism and making tactical planning less intuitive. To counteract this, our system introduces players in a realistic scale, accurately reflecting their size in proportion to the football pitch.

However, this improved realism introduces a challenge: the smaller player symbols result in smaller touch detection zones, making selection and movement more difficult. According to Fitts's law, smaller and more distant targets require more time and effort to acquire. This can negatively impact the user experience by making interactions feel less responsive and less accurate [30].

One solution, already introduced earlier, is the use of touch-assistant-zones, which extend the touch detection zone of each player without changing their visual size. In addition to that, this section presents player scaling interactions that temporarily enlarge players, making them easier to manipulate. Users can scale up players to move or adjust them and then scale them back to restore a realistic tactical view. However, it is important to note that enlarged players are harder to place precisely due to their size. In some cases, using touch-assistant-zones may be preferable to scaling, depending on the interaction context and the required level of accuracy. At the same time, both methods (using touch-assistant zones or scaling players) require additional interactions and therefore introduce time overhead. As a result, neither method holds a definitive advantage over the other, the choice depends on the specific task and user preference.

Each player scaling interaction is described by two aspects:

- *Scaling:* Describes how the players are scaled.
- *Scaling-Interaction:* Describes the interaction required to perform the scaling.

Use the Slider (US)

Scaling: The size of all players is increased if dragged up and decreased if dragged down.

Scaling-Interaction: Drag up or down on the left side of the board.

Sliders are a common and intuitive control for adjusting continuous values such as volume, brightness, or timelines. Users are generally familiar with their function, which makes them suitable for scaling player sizes, aligning with Nielsen's heuristic 4 (consistency and standards) and Jakob's law. [28,29, 33]. To avoid visual clutter on the board, the slider remains invisible and is activated by dragging on the left edge. The user does not have to hit a precise location on the screen to start the drag. The *US* interaction provides a fluid and precise way to adjust the player scaling with minimal visual distraction.

Use Scaling Buttons (USB)

Scaling: The size of all players is incremented or decremented.

Scaling-Interaction: Tap the '+' or '-' button.

An alternative to sliders is the use of buttons, which allow for stepwise control of player scaling. This interaction is especially useful for users who prefer discrete adjustments over continuous ones. The *USB* interaction is straightforward and aligns with common interface patterns, ensuring accessibility even for less experienced users.

Summary

Table 3.6 summarizes the interactions for player scaling.

Name	Scaling	Interaction
Use the Slider (US)	Size of players is scaled	Drag up or down on the left side
Use Scaling Buttons (USB)	Size of players is incremented or decremented	Tap '+' or '-'

Table 3.6: Interactions for player scaling

3.4.2 View Manipulation

In several of the reviewed tools, users can zoom, move, or rotate the view of the pitch. These capabilities are essential for providing flexible navigation and ensuring optimal user experience, especially when working with detailed or complex arrangements. To support such usability, our tool includes intuitive view manipulation interactions that allow users to adjust their perspective on the board as needed.

Each view manipulation interaction is described using two aspects:

- *View:* Describes how the view changes.

- *View-Interaction:* Describes the interaction required to perform the view change.

Pinch-to-Zoom (PtZ)

View: The view is zoomed in or out.

View-Interaction: Pinch to zoom out or spread to zoom in.

PtZ is a widely adopted interaction, commonly found in applications such as maps or image viewers. Therefore, it provides an intuitive mechanism for zooming in and out, aligning with Nielsen's heuristic 4 (consistency and standards) and Jakob's law [28, 29, 33]. It is important to note that if the gesture begins on players, it may trigger player movement instead of zooming.

Use Zoom Buttons (UZB)

View: The view zooming is incremented or decremented.

View-Interaction: Tap the 'Zoom In' or 'Zoom Out' button.

As an alternative to pinch-to-zoom, zoom buttons offer a discrete method for adjusting the view. This interaction is helpful for users who prefer tactile button control over gesture-based input. In addition, having buttons may be useful in situations where using *PtZ* could trigger other interactions.

Move Pitch (MP)

View: The view is moved.

View-Interaction: Tap the 'Move Pitch' button and drag the pitch in the desired direction.

When the pitch is zoomed in, users need a way to navigate across the field. This interaction allows the pitch to be repositioned, supporting better exploration and visibility of specific areas.

Move to an Important Zone (MIZ)

View: The view is moved to the selected important zone.

View-Interaction: Tap the 'Important Zone' button and select an important zone from the list.

Football pitches contain important zones, such as penalty areas, central zones, or attacking thirds, that are critical in tactical analysis. *MIZ* allows the user to quickly shift the view to an important zone. The list of important zones includes the own penalty area, the opponent's penalty area, both left and both right wings (on both own and opponent halves), the central defensive zone, the central midfield zone, the central attacking zone and both final thirds.

Two-Finger-Rotation (TFR)

View: The view is rotated.

View-Interaction: Perform two-finger-rotation on the pitch.

Although the pitch is vertically aligned by default, there are scenarios where rotating the view is beneficial. The widely recognized two-finger-rotation gesture allows for intuitive reorientation of the pitch, following Nielsen's heuristic 4 (consistency and standards) and Jakob's law [28,29,33].

Double-Tap (DT)

View: The view is reset to the default view.

View-Interaction: Double-tap somewhere on the pitch.

Quickly returning to the default view is essential for maintaining orientation and resetting perspective after various interactions. *DT* provides a fast and natural way to do this without requiring additional buttons. Note that double-tapping on a player does not reset the view.

Summary

Table 3.7 summarizes the interactions for view manipulation.

Name	**View**	**Interaction**
Pinch-to-Zoom (PtZ)	View is zoomed in or out	Pinch to zoom out or spread to zoom in
Use Zoom Buttons (UZB)	View zooming is incremented or decremented	Tap 'Zoom In' or 'Zoom Out'
Move Pitch (MP)	View is moved	Tap 'Move Pitch' and drag
Move to an Important Zone (MIZ)	View is moved to the important zone	Tap 'Important Zone' and select the important zone
Two-Finger-Rotation (TFR)	View is rotated	Perform two-finger-rotation
Double-Tap (DT)	View is reset to the default	Double-tap on the pitch

Table 3.7: Interactions for view manipulation

3.5 Buttons, Interface and Configuration

3.5.1 Buttons

In the previous sections, various buttons were introduced. For quick, efficient and intuitive use of the board, these buttons must be placed to ensure a good user experience. In the following, we list all required buttons, organized by their position on the screen. As not every button can be used in every state of the system, conditions in which each button is active are also given. The buttons' functionalities are not explained here, as these have already been explained in the previous sections.

All buttons are represented as icons to maintain a compact and easily recognizable interface and are always visible on the screen. However, only active buttons can be tapped, while inactive ones appear grayed out. Instead of graying out the inactive buttons, an alternative is to remove them from the screen. We chose not to remove inactive buttons, as this could confuse users, hinder recall, or lead to unnecessary searching. Displaying all buttons while indicating availability provides immediate feedback about the current state of the system and the actions that are currently possible [40]. If tapping a button turns on a special mode (e.g., active grid), the button color switches to blue to clearly indicate its active state. This design choice algins with Nielsen's heuristics 1 (visibility of system status), 5 (error prevention), 6 (recognition rather than recall) and 8 (aesthetic and minimalist design), as well as Norman's principles of visibility, constraints, mapping, and affordance [26,28,29].

All buttons are arranged in linear menus. While pie menus are sometimes used in applications, appearing as donut-shaped interfaces around a touch point where dragging in a direction selects an option, they come with trade-offs. Pie menus can reduce finger travel distance since all options are placed around the touch point and the interaction area is not limited to a specific location on the screen. However, they support only a relatively small number of options, may interfere with other interactions, and can overlap or cover important elements on the screen. Although pie menus are used in some applications, linear menus remain more common and are therefore more intuitive for most users. Given that our tool includes many advanced features and options, we chose to use simple linear menus to support a smoother and more familiar user experience [41–43].

Some interactions in our design are standard across many applications, but others are more specialized and may be harder for non-expert users to understand. To support learning, we allow users to long-press a button to open a brief tutorial about its function, helping them become more confident and capable with the tool, reflecting Nielsen's heuristic 10 (help and documentation) [28,29].

The buttons in the top left corner are used for zooming and moving the view:

- Zoom In: Is always active.
- Zoom Out: Is always active.
- Move Pitch: Is always active.
- Important Zone: Is always active.

The buttons in the top center are used for the selection of players via buttons:

- Home: Is active if no selection exists or the other team is selected.
- Away: Is active if no selection exists or the other team is selected.
- Group: Is active if no selection exists.
- Unit: Is active if no selection exists.
- Formation: Is active if no selection exists.

In the top right corner are two buttons:

- X: Is active if a locked selection exists.
- Menu: Is always active.

Note that the 'Menu' button has not been introduced so far. It is not used for an interaction, but for configurations and settings.

On the right side of the screen, there are different buttons:

- Save Group: Is active if a locked selection exists and the selection is not already saved as a group.
- Save Arrangement: Is active if a locked selection exists and the selection is not already saved as an arrangement.
- Frame: Is active if no selection exists and the grid is deactivated.
- Add: Is active if at least one framed or lassoed and locked selection exist.
- Draw Path: Is active if a locked selection exists.
- Mirror Left: Is active if a team is selected and locked.
- Mirror Right: Is active if a team is selected and locked.
- Swap: Is active if one player is selected and locked.
- x-Axis: Is active if one player is selected and locked.
- y-Axis: Is active if one player is selected and locked.
- Pair: Is active if one player is selected and locked.

- Shape: Is active if a locked group selection exists.

In the bottom left corner, there are buttons to redo/undo and to scale the players:

- Undo: Is active if undo is possible.
- Redo: Is active if redo is possible.
- +: Is always active.
- -: Is always active.

The last three buttons are displayed in the bottom center:

- Touch-Assistant-Zones: Is active if no selection exists.
- Grid: Is always active.
- Snap-To-Grid: Is always active.

Each unlocked selection made by L or F also has a small button:

- x: Is always active if the selection exists.

3.5.2 Interface

Some interactions require a pop-up window displaying a list of selectable items. The interactions that use this feature are:

- *Select an Unit (SU):* Displays a list of all available units.
- *Select a Group (SG)/Select an Arrangement (SA):* Displays a list of all saved groups. If arrangements are available for a group, a secondary list with those arrangements is also displayed. To save groups and arrangements, a pop-up window is shown to insert a name.
- *Select a Formation (SF):* Displays a list of all predefined formations.
- *Move to an Important Zone (MIZ)*: Displays a list of all predefined zones.

To support better visualization, we present mock-ups illustrating the tool in different contexts. Figure 3.1 shows the general interface with labeled buttons, while Figure 3.2 and Figure 3.3 demonstrate the interface in different contexts and show the use of some features.

3.5.3 Configuration

Our design provides several configuration options for both players and the board. As most reviewed tools, and applications in general, offer similar configuration options, we do not discuss them in detail, but merely list typically available parameters.

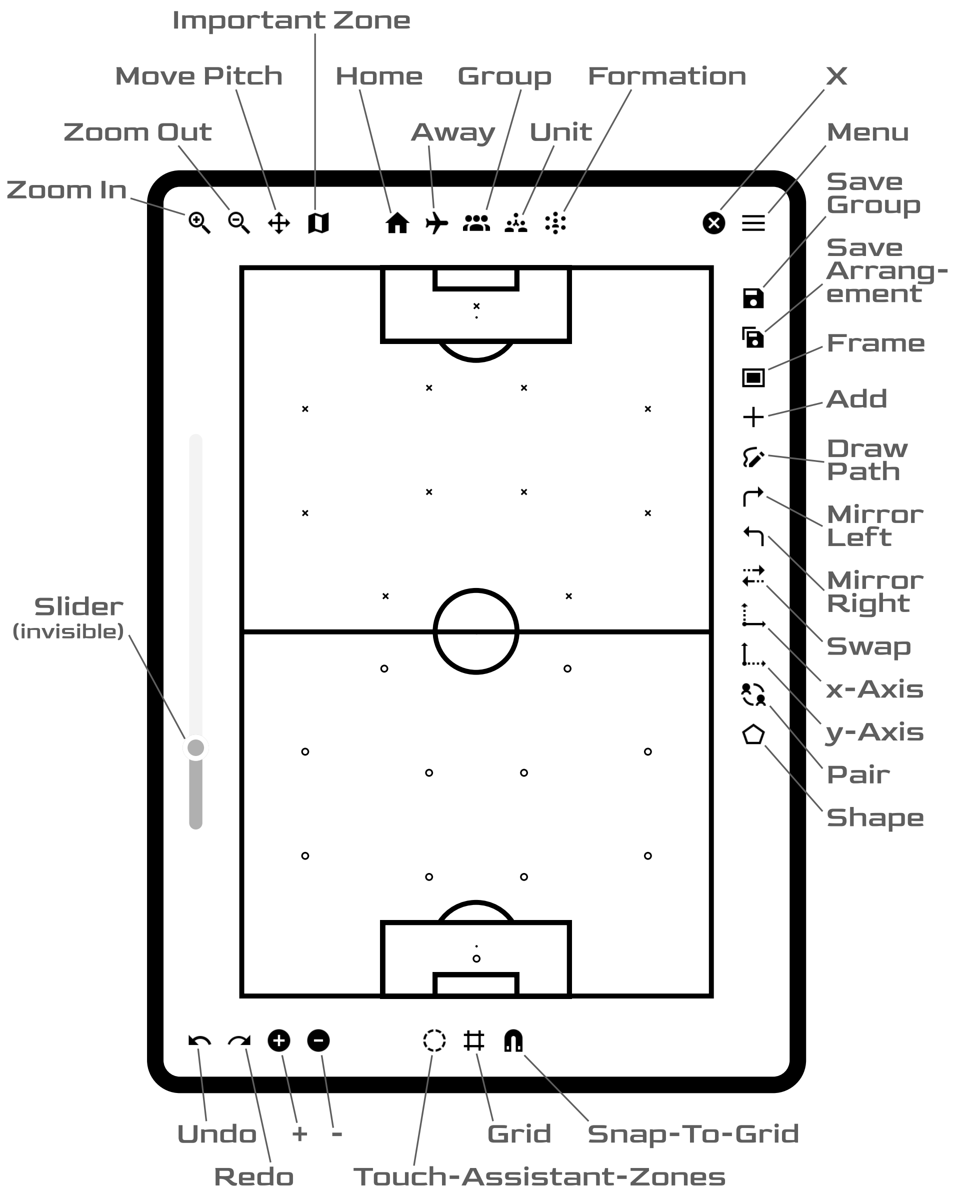

Figure 3.1: Tactics board interface with labeled buttons

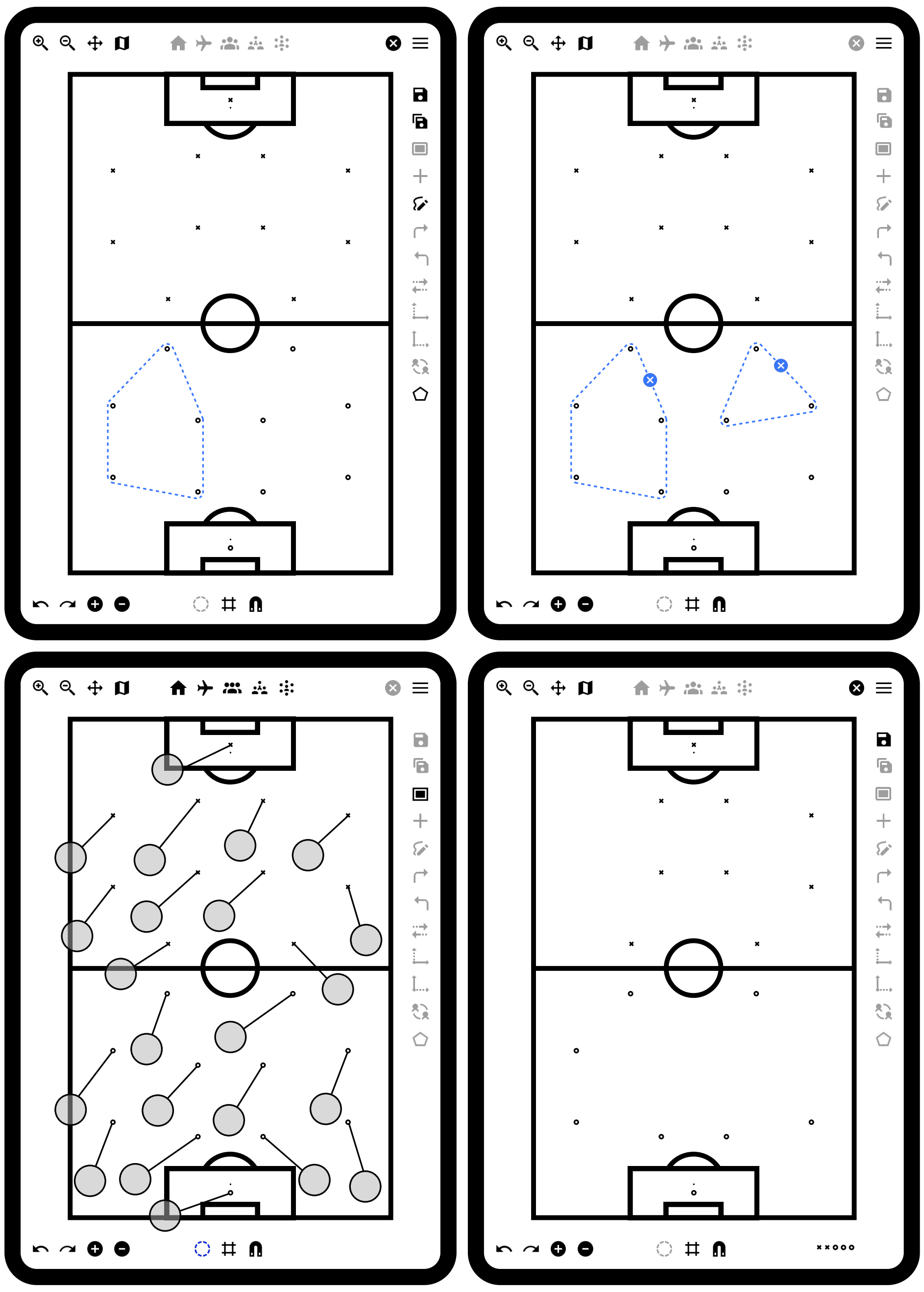

Figure 3.2: Tactics board displaying some features: locked selection with *RLPP* (top left); selections with *L* (top right); active touch-assistant-zones (bottom left); selection with *DTP* (bottom right)

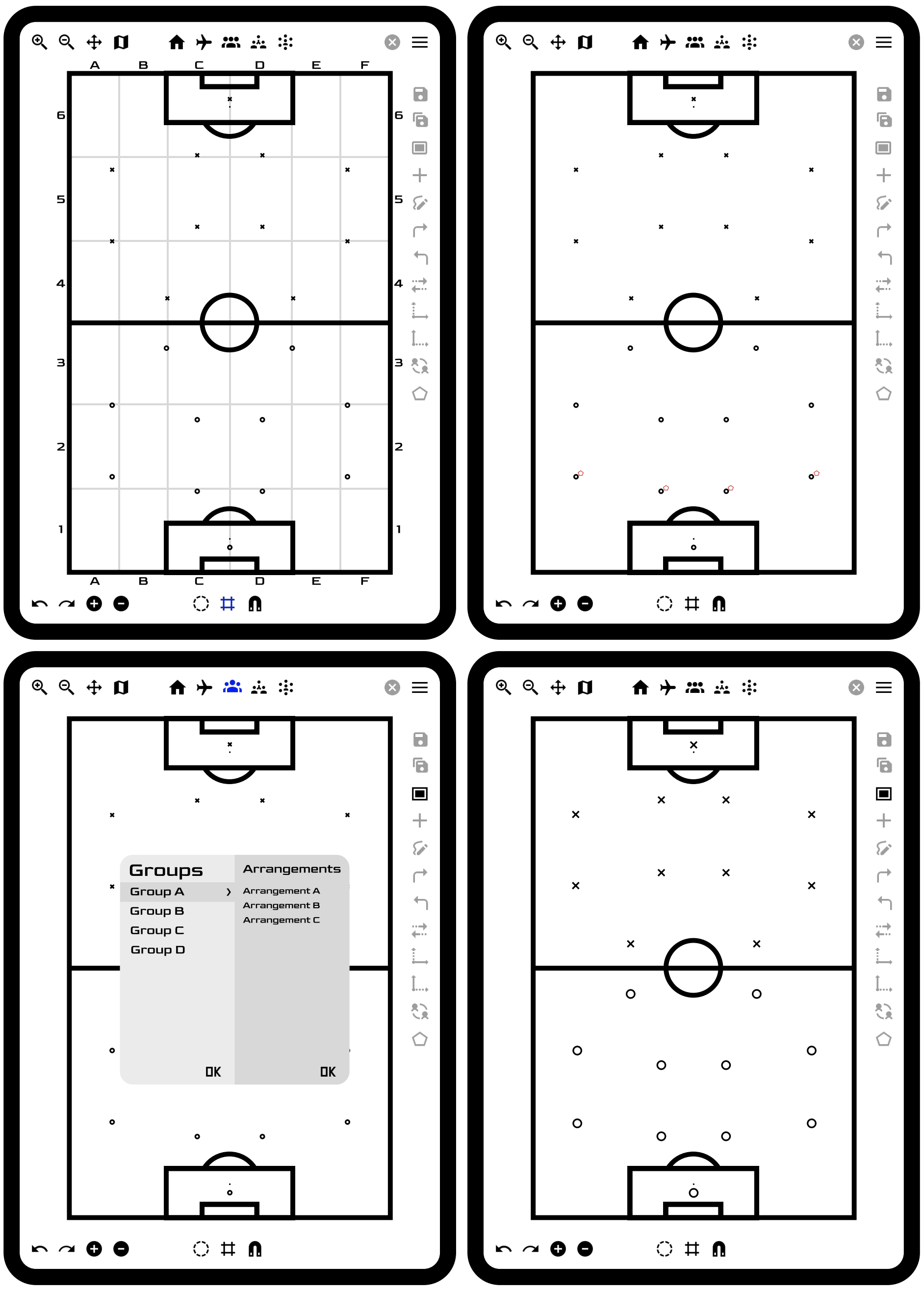

Figure 3.3: Tactics board displaying some features: active grid (top left); active *MS* for defensive line (top right); selection with *SG* or *SA* (bottom left); scaled players after *US* or *USB* (bottom right)

When no selection is active, tapping a player opens a pop-up window displaying the player's name and number, if available. In this window, some configurations can be made:

- Change the player's color.
- Add or edit the player's number.
- Add or edit the player's name.
- Adjust the player's movement speed.

In addition to player-specific configurations, board configurations can be made by tapping the 'Menu' button in the top right corner:

- Adjust pitch dimensions.
- Change pitch colors.

Chapter 4

Implementation

In this chapter, we will present a theoretical implementation of our design. We will not discuss a concrete or fully functional implementation, but will use pseudocode to illustrate the general logic. Although the pseudocode may not be syntactically applicable in a direct implementation, it serves as a guideline for potential real-world development and demonstrates that the interaction design is free of overlaps and inconsistencies. The corresponding pseudocode is provided in the Appendix.

4.1 The Board

First, a class that holds everything together and stores different attributes that can be referenced by different functions is necessary. This class is called TacticsBoard, as shown in Algorithm 1 in Appendix A. Its attributes include:

- players: A list of all 22 players on the pitch, where each player has 'location', 'links', and 'pair' as attributes.
- selectedPlayers: A 2D list containing selected players. Each item in the list is a group or a player. For concurrent selection, multiple items can be in the list. If a selection is locked, all players in the selection are in a list at index 0.
- selectionIsLocked: A boolean indicating whether there is a locked selection at the moment.
- pickedUpPlayers: A list of players picked up.
- frameModeIsActive: A boolean indicating whether the 'framing mode' is active.
- dragPoints: A list of location points that define an area or line.

- addAreaModeIsActive: A boolean indicating whether the 'area adding mode' is active.
- gridIsActive: A boolean indicating whether the grid is active.
- drawPathModeIsActive: A boolean indicating whether the 'draw path mode' is active.
- swapPlayersModeIsActive: A boolean indicating whether the 'swap players mode' is active.
- activeAxes: A tuple of booleans indicating if the axis buttons are tapped and therefore active. The first item in the tuple represents the x-axis, and the second the y-axis.
- pairingModeIsActive: A boolean indicating whether the 'pairing mode' is active.
- snapToGridModeIsActive: A boolean indicating whether the 'snap-to-grid mode' is active.
- shapesToMaintain: A list of groups whose shape must be maintained.
- playerScaling: A value between 0.1 and 10, indicating the scaling of players, where 1 is the default value.
- zoomFactor: A value indicating the zoom factor. The default value is 100.
- movePitchModeIsActive: A boolean indicating whether the 'move pitch mode' is active.
- viewCenter: A tuple indicating the center of the current view, where (0,0) is the default center.
- viewRotation: A value indicating the rotation angle of the current view in degrees, where 0 is the default value.

4.2 Touch Detection Functions

With a board providing all necessary attributes, functions can now be defined to detect input touches and perform the appropriate board manipulations.

The handling of touch detection strongly depends on the programming language and framework. Since this discussion is theoretical, no specific implementation is assumed. Therefore, a touch is treated as a sequence of rapid touch events that repeatedly trigger the detection function. This has the advantage that, for example, drag interactions, can update attributes during the interaction, not only after it. For every function, a touch object is given as input, which has 'location', 'flick', 'intensity', 'rotation', 'hasReleaseEnding', and 'hasLongPressEnding' as attributes. A detailed discussion of these attributes is omitted, as their purpose is self-explanatory based on their names and context.

4.2.1 Handle Player Touches

Firstly, the handling of touches on a player, covering touch, tap, double-tap, long-press, drag, and flick interactions, must be discussed. In addition to the touch input, these functions also get the touched player as input. Algorithm 2 and 3 in Appendix A present these functions:

- HandlePlayerTouch: Selects the touched player and removes the selection if touch is released.
- HandlePlayerTap: Performs a player swap or adds a link or pair on two players, depending on the active mode.
- HandlePlayerDoubleTap: Selects and picks up the touched player.
- HandlePlayerLongPress: Selects the touched player and locks the selection.
- HandlePlayerDrag: Performs drag-and-drop and removes the selection if touch is released.
- HandlePlayerFlick: Flicks the touched player.

4.2.2 Handle Pitch Touches

In addition to touches on players, many touches are made somewhere on the pitch. These include tap, double-tap, long-press, drag, flick, pinch, spread, and two-finger-rotation interactions. The functions used in Algorithm 4, Algorithm 5, Algorithm 6 and Algorithm 7 in Appendix A are:

- HandlePitchTap: Places the first player picked up back on the pitch on the touched point.
- HandlePitchDoubleTap: Resets the view.
- HandlePitchLongPress: Moves all selected players toward the touched point.
- HandlePitchDrag: Calls the following functions or moves the view, depending on the active mode.
- HandlePitchDragLockedSelection: Draws a path or performs drag-and-drop, depending on the active mode.
- HandlePitchDragFrame: Selects players using the frame methods.
- HandlePitchDragLasso: Selects players using the lasso methods.

- HandlePitchFlick: Flicks the selected players.
- HandlePitchPinch: Pinches the selected players or zooms out.
- HandlePitchSpread: Spreads the selected players or zooms in.
- HandlePitchTwoFingerRotation: Rotates the view.

4.2.3 Handle Button Touches

In the interaction design and tool design, buttons were introduced. Touching them also triggers functions. These functions are defined in Algorithm 8, Algorithm 9, Algorithm 10 and Algorithm 11 in Appendix A:

- HandleXButtonTap: Deselects a locked selection and deactivates all modes.
- HandleAreaXButtonTap: Deselects an unlocked lasso or frame selection.
- Handle+ButtonTap: Used to add more areas to the locked selection or to confirm the area as locked selection.
- HandleHomeButtonTap: Selects all home players.
- HandleAwayButtonTap: Selects all away players.
- HandleGroupButtonTap: Adds or removes the players from the tapped group to or from the locked selection.
- HandleUnitButtonTap: Adds or removes the players from the tapped unit to or from the locked selection.
- HandleRowLabelButtonTap: Adds or removes the players from the tapped row to or from the locked selection.
- HandleColumnLabelButtonTap: Adds or removes the players from the tapped column to or from the locked selection.
- HandleCellButtonTap: Adds or removes the players from the tapped cell to or from the locked selection.
- HandleGridButtonTap: Activates or deactivates the grid.
- HandleDrawPathButtonTap: Activates or deactivates the 'draw path mode'.
- HandleArrangementButtonTap: Applies an arrangement to the selected players.
- HandleFormationButtonTap: Applies a formation to the selected team.

- HandleMirrorButtonTap: Mirrors one half of the pitch.
- HandleSwapButtonTap: Activates or deactivates the 'swap players mode'.
- HandleAxisButtonTap: Selects the axes.
- HandlePairButtonTap: Activates or deactivates the 'pair players mode'.
- HandleSnapToGridButtonTap: Activates or deactivates the 'snap-to-grid mode'.
- HandleShapeButtonTap: Adds or removes the selected players to or from the list of shapes to maintain.
- HandleScalingButtonTap: Applies the new scaling to the players.
- HandleZoomButtonTap: Applies the new zooming factor to the view.
- HandleMovePitchButtonTap: Activates or deactivates the 'move pitch mode'.
- HandleImportantZoneButtonTap: Moves the view to the important zone.

4.2.4 Handle Screen Touches

The last interaction is triggered by touching neither players, the pitch, nor any buttons. It is triggered by a drag interaction on the screen. This function, used in Algorithm 12 in Appendix A, describes it:

- HandleDragScreen: Scales the players according to the drag.

4.3 Behavior Modifier Functions

The interaction design includes behavior modifiers that move players without direct interaction. Since these modifiers are also part of an implementation, their discussion is also necessary. They are all called after the movement of each player. As input they get the moved player and some also the movement. The necessary functions are utilized in Algorithm 13 in Appendix A and are described in the following:

- CheckLinkMovement: Iterates over the links of the moved player and moves the linked players according to the movement and the stored axes.
- CheckPairMovement: Iterates over the paired players of the moved player and moves them according to the movement.

- CheckSnapToGrid: Moves the moved player to the grid if the 'snap-to-grid' mode is active.
- CheckMaintainShape: Checks if the moved player belongs to a group that should maintain its shape and, if so, moves the other players of the group to maintain that shape.

Chapter 5

Conclusion

Our review of existing tactics board tools for football revealed a wide range of available solutions. While all offer basic features to manipulate the board, they lack in leveraging football-specific structures or providing advanced features for efficient interaction and manipulation. In this work, we introduced and discussed an interaction design and tool design that combines basic, well-known, and intuitively usable features that must be present for good user experience, such as drag-and-drop, along with more creative and innovative ideas. These include interactions adapted from other domains, such as lasso selection and pinch and spread gestures. Our proposed design aims for high efficiency, intuitive use, and enhanced functionality. The design follows established usability heuristics and principles, such as consistency and feedback, which contribute to its intuitiveness and efficiency. To complement the interaction design, we presented a theoretical implementation using pseudocode, offering guidance for potential real-world development.

Future Work

In future work, to validate the tool against its intended goals, usability studies are essential. This will require the development of working prototypes and the inclusion of both casual and expert users to ensure broad applicability.

Following user testing, an appropriate programming language and framework must be chosen to implement the design, incorporating the feedback from usability evaluations. Additional features should be considered, such as enhanced player details (e.g., name, height, nationality, statistics) and drawing tools (e.g., arrows, lines, shapes). For professional applications, integrating real game data will be crucial. The tool should enable the display and analysis of real scenarios to better visualize and correct in-game situations.

Appendix A

Pseudocode

Algorithm 1 Tactics Board

```
1: Class TACTICSBOARD
2:     players
3:     selectedPlayers ← []
4:     selectionIsLocked ← false

5:     pickedUpPlayers ← []                          ▷ for DTP, TPD
6:     frameModeIsActive ← false                     ▷ for F, FaLP
7:     dragPoints ← []                               ▷ for L, LaLP, F, FaLP
8:     addAreaModeIsActive ← false                   ▷ for LaLP, FaLP
9:     gridIsActive ← false                          ▷ for SC
10:    drawPathModeIsActive ← false                  ▷ for DPP
11:    swapPlayersModeIsActive ← false               ▷ for SP
12:    activeAxes ← (false, false)                   ▷ for LM
13:    pairingModeIsActive ← false                   ▷ for PM
14:    snapToGridModeIsActive ← false                ▷ for StG
15:    shapesToMaintain ← []                         ▷ for MS

16:    playerScaling ← 1                             ▷ for US, USB
17:    zoomFactor ← 100                              ▷ for PtZ, UZB, DT
18:    movePitchModeIsActive ← false                 ▷ for MP
19:    viewCenter ← (0,0)                            ▷ for MP, MIZ, DT
20:    viewRotation ← 0                              ▷ for TFR, DT
```

Algorithm 2 Handle Player Touches (1)

```
function HandlePlayerTouch(player, touch)                                   ▷ TP
    selectionIsLocked ← false
    selectedPlayers.add([player])
    if touch.hasReleaseEnding then
        selectedPlayers.remove([player])
    end if
end function

function HandlePlayerTap(player, touch)
    if selectedPlayers.first.length == 1 then
        selectedPlayer = selectedPlayers.first
        if swapPlayersModeIsActive then                                      ▷ SP
            swapLocations(selectedPlayer, player)
            selectedPlayers.removeFirst()
            swapPlayersModeIsActive ← false
        else if activeAxes != (false, false) then                            ▷ LM
            selectedPlayer.links.add(player, activeAxes)
            player.links.add(selectedPlayer, activeAxes)
            activeAxes ← (false, false)
        else if pairingModeIsActive then                                     ▷ PM
            selectedPlayer.pair.add(player)
            player.pair.add(selectedPlayer)
            player.location ← selectedPlayer.location + calculateOffset()
            pairingModeIsActive ← false
        end if
    end if
end function

function HandlePlayerDoubleTap(player, touch)                               ▷ DTP
    selectionIsLocked ← true
    mergeSelectedPlayers(player)
    pickedUpPlayers.add([player])
end function
```

Algorithm 3 Handle Player Touches (2)

```
function HandlePlayerLongPress(player, touch)                ▷ LPP, RLPP
    selectionIsLocked ← true
    mergeSelectedPlayers(player)
end function

function HandlePlayerDrag(player, touch)                     ▷ DaDP
    player.location ← touch.location
    if touch.hasReleaseEnding then
        selectedPlayers.remove([player])
    end if
end function

function HandlePlayerFlick(player, touch)                    ▷ FP
    flickPlayer(player, touch.flick)
    selectedPlayers.remove([player])
end function
```

Algorithm 4 Handle Pitch Touches (1)

```
function HandlePitchTap(touch)                               ▷ TPD
    if pickedUpPlayers.isNotEmpty then
        pickedUpPlayers.first.location ← touch.location
        selectedPlayers.remove([pickedUpPlayers.first])
        pickedUpPlayers.removeFirst()
        if pickedUpPlayers.isEmpty then
            selectionIsLocked ← false
        end if
    end if
end function

function HandlePitchDoubleTap(touch)                         ▷ DT
    viewCenter ← (0,0)
    zoomFactor ← 100
    viewRotation ← 0
end function

function HandlePitchLongPress(touch)                         ▷ LPPPP, LPPP
    if selectionIsLocked then
        moveTowards(selectedPlayers.first, touch.location)
    end if
end function
```

Algorithm 5 Handle Pitch Touches (2)

```
function HandlePitchDrag(touch)
    if selectionIsLocked then
        HandlePitchDragLockedSelection(touch)
    else if movePitchModeIsActive then                          ▷ MP
        viewCenter ← touch.location
    else if frameModeIsActive then
        HandlePitchDragFrame(touch)
    else
        HandlePitchDragLasso(touch)
    end if
end function

function HandlePitchDragLockedSelection(touch)
    if drawPathModeIsActive then                                ▷ DPP, DGP
        dragPoints.add(touch.location)
        if touch.hasReleaseEnding then
            moveSelectedPlayersAlongPath(dragPoints)
        end if
    else                                                        ▷ DaDP, DaDG
        if touchIsInsideAGroup(touch) then
            move(selectedPlayers[getIndexOfGroup()], touch.location)
        else if touchIsSecondaryTouch(touch) then
            moveSlightly(selectedPlayers.first, touch.location)
        else
            moveGroup(selectedPlayers.first, touch.location)
        end if
    end if
end function
```

Algorithm 6 Handle Pitch Touches (3)

```
function HandlePitchDragFrame(touch)
    dragPoints.add(touch.location)
    if touch.hasReleaseEnding then                                  ▷ F
        selectionIsLocked ← false
        selectedPlayers.add([getPlayersOfFramedArea(dragPoints)])
        dragPoints ← []
    else if touch.hasLongPressEnding then                           ▷ FaLP
        selectionIsLocked ← true
        if addAreaModeIsActive then
            mergeSelectedPlayers(getPlayersOfFramedArea(dragPoints))
        else
            selectedPlayers ← [[getPlayersOfFramedArea(dragPoints)]]
        end if
        dragPoints ← []
    end if
end function

function HandlePitchDragLasso(touch)
    dragPoints.add(touch.location)
    if touch.hasReleaseEnding then                                  ▷ L
        selectionIsLocked ← false
        selectedPlayers.add([getPlayersOfLassoedArea(dragPoints)])
        dragPoints ← []
    else if touch.hasLongPressEnding then                           ▷ LaLP
        selectionIsLocked ← true
        if addAreaModeIsActive then
            mergeSelectedPlayers(getPlayersOfLassoedArea(dragPoints))
        else
            selectedPlayers ← [[getPlayersOfLassoedArea(dragPoints)]]
        end if
        dragPoints ← []
    end if
end function
```

Algorithm 7 Handle Pitch Touches (4)

```
function HandlePitchFlick(touch)
    if selectionIsLocked then                                          ▷ FP, FG
        if touchIsInsideAGroup(touch) then
            flickGroup(selectedPlayers[getIndexOfGroup()], touch.flick)
        else
            flickGroup(selectedPlayers.first, touch.flick)
        end if
    end if
end function

function HandlePitchPinch(touch)
    if selectionIsLocked then                                          ▷ PG
        if touchIsInsideAGroup(touch) then
            pinchGroup(selectedPlayers[getIndexOfGroup()], touch.intensity)
        else
            pinchGroup(selectedPlayers.first, touch.intensity)
        end if
    else
        zoomFactor ← zoomFactor - touch.intensity
    end if
end function

function HandlePitchSpread(touch)
    if selectionIsLocked then                                          ▷ SpG
        if touchIsInsideAGroup(touch) then
            spreadGroup(selectedPlayers[getIndexOfGroup()], touch.intensity)
        else
            spreadGroup(selectedPlayers.first, touch.intensity)
        end if
    else
        zoomFactor ← zoomFactor + touch.intensity
    end if
end function

function HandlePitchTwoFingerRotation(touch)                           ▷ TFR
    viewRotation ← touch.rotation
end function
```

Algorithm 8 Handle Button Touches (1)

```
function HandleXButtonTap                    ▷ LPP, DTP, LaLP, FaLP, RLPP, ST,
    selectionIsLocked ← false                              ▷ SG, SU, SRC, SC
    selectedPlayers ← []
    deactivateAllModes()
end function

function HandleAreaXButtonTap(index)                               ▷ L, F
    selectedPlayers.removeAt(index)
end function

function Handle+ButtonTap                                    ▷ LaLP, FaLP
    addAreaModeIsActive ← !addAreaModeIsActive
end function

function HandleHomeButtonTap                                         ▷ ST
    selectionIsLocked ← true
    selectedPlayers ← [[getAllHomePlayers()]]
end function

function HandleAwayButtonTap                                         ▷ ST
    selectionIsLocked ← true
    selectedPlayers ← [[getAllAwayPlayers()]]
end function

function HandleGroupButtonTap(group, action)                         ▷ SG
    if action == Add then
        selectionIsLocked ← true
        mergeSelectedPlayers(getPlayersOfGroup(group))
    else if action == Remove then
        selectedPlayers.remove([getPlayersOfGroup(group)])
        if selectedPlayers.isEmpty then
            selectionIsLocked ← false
        end if
    end if
end function
```

Algorithm 9 Handle Button Touches (2)

```
function HandleUnitButtonTap(unit, action)                ▷ SU
    if action == Add then
        selectionIsLocked ← true
        mergeSelectedPlayers(getPlayersOfUnit(unit))
    else if action == Remove then
        selectedPlayers.remove([getPlayersOfUnit(unit)])
        if selectedPlayers.isEmpty then
            selectionIsLocked ← false
        end if
    end if
end function

function HandleRowLabelTap(row, action)                   ▷ SRC
    if action == Add then
        selectionIsLocked ← true
        mergeSelectedPlayers(getPlayersOfRow(row))
    else if action == Remove then
        selectedPlayers.remove([getPlayersOfRow(row)])
        if selectedPlayers.isEmpty then
            selectionIsLocked ← false
        end if
    end if
end function

function HandleColumnLabelTap(column, action)             ▷ SRC
    if action == Add then
        selectionIsLocked ← true
        mergeSelectedPlayers(getPlayersOfColumn(column))
    else if action == Remove then
        selectedPlayers.remove([getPlayersOfColumn(column)])
        if selectedPlayers.isEmpty then
            selectionIsLocked ← false
        end if
    end if
end function
```

Algorithm 10 Handle Button Touches (3)

```
function HandleCellButtonTap(cell, action)                          ▷ SC
    if action == Add then
        selectionIsLocked ← true
        mergeSelectedPlayers(getPlayersOfCell(cell))
    else if action == Remove then
        selectedPlayers.remove([getPlayersOfCell(cell)])
        if selectedPlayers.isEmpty then
            selectionIsLocked ← false
        end if
    end if
end function

function HandleGridButtonTap                                        ▷ SC
    gridIsActive ← !gridIsActive
end function

function HandleDrawPathButtonTap                              ▷ DPP, DGP
    drawPathModeIsActive ← !drawPathModeIsActive
end function

function HandleArrangementButtonTap(arrangement)                    ▷ SA
    applyArrangement(arrangement, selectedPlayers.first)
end function

function HandleFormationButtonTap(arrangement)                      ▷ SA
    applyFormation(formation, selectedPlayers.first)
end function

function HandleMirrorButtonTap(halfOfPitch)                       ▷ MPHP
    mirrorHalfOfPitch(halfOfPitch, selectedPlayers.first)
end function

function HandleSwapButtonTap                                        ▷ SP
    swapPlayersModeIsActive ← !swapPlayersModeIsActive
end function
```

Algorithm 11 Handle Button Touches (4)

```
function HandleAxisButtonTap(axes)                              ▷ LM
    activeAxes ← axes
end function

function HandlePairButtonTap                                    ▷ PM
    pairingModeIsActive ← !pairingModeIsActive
end function

function HandleSnapToGridButtonTap                              ▷ StG
    snapToGridModeIsActive ← !snapToGridModeIsActive
end function

function HandleShapeButtonTap                                   ▷ MS
    if shapesToMaintain.contains(selectedPlayers.first) then
        shapesToMaintain.remove(selectedPlayers.first)
    else
        shapesToMaintain.add(selectedPlayers.first)
    end if
end function

function HandleScalingButtonTap(newScaling)                     ▷ USB
    playerScaling ← newScaling
end function

function HandleZoomButtonTap                                    ▷ UZB
    zoomFactor ← newZoomFactor
end function

function HandleMovePitchButtonTap                               ▷ MP
    movePitchModeIsActive ← !movePitchModeIsActive
end function

function HandleImportantZoneButtonTap(importantZone)            ▷ MIZ
    viewCenter ← importantZone.center
end function
```

Algorithm 12 Handle Screen Touch

```
function HandleScreenDrag(touch)                                ▷ US
    if touchIsOnSlider(touch) then
        playerScaling ← getScalingFromTouch(touch)
    end if
end function
```

Algorithm 13 Behavior Modifier Functions

```
 1: function CheckLinkMovement(movedPlayer, movement)        ▷ LM
 2:     for (player, axes) in movedPlayer.links do
 3:         movePlayerWithLink(player, movement, axes)
 4:     end for
 5: end function

 6: function CheckPairMovement(movedPlayer, movement)        ▷ PM
 7:     for player in movedPlayer.pair do
 8:         movePlayer(player, movement)
 9:     end for
10: end function

11: function CheckSnapToGrid(movedPlayer)                    ▷ StG
12:     if snapToGridModeIsActive then
13:         movePlayerToGrid(movedPlayer)
14:     end if
15: end function

16: function CheckMaintainShape(movedPlayer)                 ▷ MS
17:     if shapesToMaintainContains(movedPlayer) then
18:         playersOfShape $\leftarrow$ getPlayersBelongingToShape(movedPlayer)
19:         for player in playersOfShape do
20:             moveToMaintainShape(player, shapesToMaintain, playersOf-
    Shape)
21:         end for
22:     end if
23: end function
```

Bibliography

[1] Tactiq Coaching. Top 5 Soccer Coaching Tools: Why the Magnetic Tactic Board is a Game-Changer, December 2024. Accessed: 2025-06-06. URL: https://tactiqcoaching.com/blogs/news/top-soccer-coaching-tools.

[2] Rise of the Coach. The Tactics Board – Undervalued at Grassroots Level, September 2012. Accessed: 2025-06-06. URL: https://riseofthecoach.wordpress.com/2012/09/09/the-tactics-board-undervalued-at-grassroots-level/.

[3] Oemkiosks. Get to know the 10 benefits of multitouch technology, February 2021. Accessed: 2025-06-06. URL: https://oemkiosks.com/blog/get-know-10-benefits-multitouch-technology/.

[4] Guide to Football. Playing Systems. Accessed: 2025-06-06. URL: https://www.guidetofootball.com/tactics/playing-systems/.

[5] FutbolLab. Internal structure of a football system: how to develop a winning tactic, December 2023. Accessed: 2025-06-05. URL: http://futbollab.com/en/news/internal-structure-of-a-football-system-how-to-develop-a-winning-tactic.

[6] Tactical Board. Accessed: 2025-02-24. URL: https://tactical-board.com/uk/big-football.

[7] Share my Tactics. Accessed: 2025-02-24. URL: https://sharemytactics.com/.

[8] Easy2Coach. Accessed: 2025-02-24. URL: https://www.easy2coach.net/en/online-soccer-tactic-board/.

[9] Mohamed Aberkane. Coach Board: Create Drills. Accessed: 2025-02-24. URL: https://apps.apple.com/ch/app/coach-board-create-drills/id1555645807?l=en-GB.

[10] Gaku Morita. Tacbo - Tactical Board. Accessed: 2025-02-24. URL: https://apps.apple.com/ch/app/tacbo-tactical-board/id1372526196?l=en-GB.

[11] BLUELINDEN. Coach Tactic Board: Football. Accessed: 2025-03-17. URL: https://apps.apple.com/ch/app/coach-tactic-board-football/id834813357?l=en-GB.

[12] TacticalPad. Accessed: 2025-03-17. URL: https://www.tacticalpad.com/new/index.php?lang=de.

[13] The Coaching Manual. Accessed: 2025-03-17. URL: https://www.thecoachingmanual.com/features/session-planner.

[14] Soccer Tutor. Tactics Manager. Accessed: 2025-03-17. URL: https://www.soccertutor.com/pages/tactics-manager.

[15] Coaches' Voice. Sport Session Planner. Accessed: 2025-03-17. URL: https://www.sportsessionplanner.com/.

[16] Tacticalista. Accessed: 2025-03-28. URL: https://tacticalista.com/en/.

[17] Tactical Boards. Accessed: 2025-03-28. URL: https://tacticalboards.com/.

[18] Joris Bekkers. Interactive Digital Tactics Board, March 2022. Accessed: 2025-06-02. URL: https://unravelsports.github.io/2022/03/06/tactics-board.html.

[19] Game Coach. Accessed: 2025-06-02. URL: https://gamecoach.gg/sports/football-soccer/football-soccer.

[20] Fussball Coach. Roberto De Zerbi explains his build-up tactics, February 2024. Accessed: 2025-02-24. URL: https://www.youtube.com/watch?v=MRR1ouiYwC4.

[21] Coaches' Voice. Enzo Maresca • Leicester City tactics, Inverted fullbacks • Masterclass, December 2023. Accessed: 2025-02-24. URL: https://www.youtube.com/watch?v=qyUu1PN8ELA.

[22] Coaches' Voice. Edin Terzić • Tactics, Borussia Dortmund v PSG Champions League Semi Final, Part 1 • Masterclass, October 2024. Accessed: 2025-02-24. URL: `https://www.youtube.com/watch?v=7cVKRuTq2Ss`.

[23] Coaches' Voice. Cesc Fàbregas • Building up to attack • Masterclass, November 2023. Accessed: 2025-02-24. URL: `https://www.youtube.com/watch?v=qLmV-m0SxhI`.

[24] Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale. *Human Computer Interaction*. Pearson, 3 edition, 2004.

[25] Helen Sharp, Yvonne Rogers, Jenny Preece. *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 6 edition, 2023.

[26] Don Norman. *The Design of Everyday Things*. Basic Books, revised and expanded edition, 2013.

[27] Jakob Nielsen. Usability 101: Introduction to Usability, January 2012. Accessed: 2025-06-12. URL: `https://www.nngroup.com/articles/usability-101-introduction-to-usability/`.

[28] Jakob Nielsen, Robert L. Mack. *Usability Inspection Methods*. John Wiley & Sons, 1 edition, 1994.

[29] Jakob Nielsen. 10 Usability Heuristics for User Interface Design, April 1994. Accessed: 2025-06-12. URL: `https://www.nngroup.com/articles/ten-usability-heuristics/`.

[30] Laws of UX. Fitts's Law. Accessed: 2025-06-12. URL: `https://lawsofux.com/fittss-law/`.

[31] Laws of UX. Hick's Law. Accessed: 2025-06-12. URL: `https://lawsofux.com/hicks-law/`.

[32] Laws of UX. Miller's Law. Accessed: 2025-06-12. URL: `https://lawsofux.com/millers-law/`.

[33] Laws of UX. Jakob's Law. Accessed: 2025-06-12. URL: `https://lawsofux.com/jakobs-law/`.

[34] Uta Hinrichs, Sheelagh Carpendale. Gestures in The Wild: Studying Multi-Touch Gesture Sequences on Interactive Tabletop Exhibits. *University of Calgary*, May 2011.

[35] Freek Bijl. Top 10 Best Football (Soccer) Formations and Line-ups, December 2023. Accessed: 2025-06-05. URL: `https://mingle.sport/blog/top-10-best-football-formations/`.

[36] Vivek Kothapalli. Football Pitch Zones, November 2024. Accessed: 2025-06-09. URL: `https://medium.com/@vivek97hills/football-pitch-zones-8fe265d1d3dd`.

[37] AJ Speaks. Understanding "pressing" in the modern game, July 2023. Accessed: 2025-06-15. URL: `https://medium.com/@aj_speaks/understanding-pressing-in-the-modern-game-8aa911ea49c7`.

[38] England Football Learning. What is pressing in football?, February 2022. Accessed: 2025-06-15. URL: `https://learn.englandfootball.com/articles-and-resources/coaching/resources/2022/what-is-pressing-in-football`.

[39] Mark White. Formations in football: Every modern formation and system, February 2025. Accessed: 2025-06-09. URL: `https://www.fourfourtwo.com/features/formations-in-football`.

[40] Vitaly Friedman. Hidden vs. Disabled In UX, May 2024. Accessed: 2025-06-10. URL: `https://www.smashingmagazine.com/2024/05/hidden-vs-disabled-ux/`.

[41] Jack Callahan, Don Hopkins, Mark Weisert, Ben Shneiderman. AN EMPIRICAL COMPARISON OF PIE vs. LINEAR MENUS. *University of Maryland*, 1988.

[42] Pietro Murano, Iram N. Khan. Pie Menus or Linear Menus, Which Is Better? *Journal of Emerging Trends in Computing and Information Sciences*, 2015.

[43] Josh Clark. Touch Means a New Chance for Radial Menus, July 2012. Accessed: 2025-06-10. URL: `https://bigmedium.com/ideas/radial-menus-for-touch-ui.html`.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten schriftlichen Arbeit. Eine der folgenden drei Optionen ist in Absprache mit der verantwortlichen Betreuungsperson verbindlich auszuwählen:

○ Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben, namentlich, dass mir niemand beim Verfassen der Arbeit geholfen hat. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuungsperson. Es wurden keine Technologien der generativen künstlichen Intelligenz[1] verwendet.

[✓ Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben, namentlich, dass mir niemand beim Verfassen der Arbeit geholfen hat. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuungsperson. Als Hilfsmittel wurden Technologien der generativen künstlichen Intelligenz[2] verwendet und gekennzeichnet.]

○ Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben, namentlich, dass mir niemand beim Verfassen der Arbeit geholfen hat. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuungsperson. Als Hilfsmittel wurden Technologien der generativen künstlichen Intelligenz[3] verwendet. Der Einsatz wurde, in Absprache mit der Betreuungsperson, nicht gekennzeichnet.

Titel der Arbeit:

Interactive Display of Football Action

Verfasst von:
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Knill	Thiago

Ich bestätige mit meiner Unterschrift:
- Ich habe mich an die Regeln des «Zitierleitfadens» gehalten.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu und vollständig dokumentiert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Eigenständigkeit überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, 16.06.2025	[illegible]

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie grundsätzlich gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

[1] z. B. ChatGPT, DALL E 2, Google Bard
[2] z. B. ChatGPT, DALL E 2, Google Bard
[3] z. B. ChatGPT, DALL E 2, Google Bard