CIS400/401 Progress Report - Designing Rhythm Game Interfaces for Touchscreen Devices

Dept. of CIS - Senior Design 2011-2012

Philip H. Peng
pengp@stwing.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Stephen H. Lane
shlane@cis.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

As touchscreen devices become increasingly popular, new software applications are expected to support touch-focused interfaces for user interaction. This study focuses on the evaluation of the effectiveness of various rhythm game interface designs on touchscreen devices. This will be accomplished through the development of a rhythm game prototype for Android tablets. The prototype app will demo various game interfaces and collect usage data that will be studied later to evaluate their effectiveness.

1. INTRODUCTION

Over the past few years, touchscreen devices have become increasingly common in the consumer market. According to a report in the March 2010 publication of *Information Display*, consumer-device manufacturers are rapidly adopting touch, with revenues increasing *10x* and unit production *3x* faster than the display industry [8]. With the adoption of this new input technology comes the natural expectation of increased software support for new touch-focused interfaces, allowing for more natural, intuitive, and powerful human-device interactions [4].

One area that touchscreen support can be leveraged is in the design of rhythm games. A *touchscreen* is a specialized display that receives user input through physical contact from a finger or stylus. *Rhythm games* are a genre of music-based games in which the player performs specific actions in response to audio and visual cues. Rhythm games often focus the the player's beat recognition abilities and consequently their timing accuracy, aided through visual patterns that match the rhythm of the song. These visual patterns consist of a series of *note* objects that appear or move across the screen in a manner referred to in this study as the *gameplay mode*. Interaction with these notes would typically involve hand actions occuring in the *hitbox* area, a pre-defined area for interaction. In non-touchscreen rhythm games, such actions may be the pressing of a button; in a touchscreen scenario, such actions would be either *soft button* touches, defined as virtual buttons interacted by through tap, or *touch-input gestures*, defined as touch events with predefined timing and path properties.

Poorly placed user interface elements can lead to degraded performance and lowered response times, especially under environments with secondary physical tasks or factors that demand high attention [1]. In rhythm games, lowering the attention load required for executing the associated note action can result in increased performance in rhythm correctness and timing accuracy. A well designed user interface that yields high gameplay performance is considered efficient in its task, the presentation of notes data to the player. In this study, various simplified game interfaces are designed and comared to find out which game interface is most efficient for rhythm games on touchscreen devices.

2. RELATED WORK

Wiimote + Dance Game

In their study, "Understanding Visual Interfaces for the Next Generation of Dance-Based Rhythm Video Games," Charbonneau et al. presented their experimental study of comparing game interfaces for *RealDance*, a dancing game prototype that uses the Wiimote. Three interfaces were compared: "Timeline", "Motion Lines", and "Beat Circles." The results of their studies showed that both "Motion Lines" and "Beat Circles" were significantly more efficient than the traditional "Timeline" interface in dance games [2].

External Multi-Touch Panel + Turn-based Strategy Game

In their study, "A Study on Multi-Touch Interaction for Game," Yong-Chul Kwon and Won-Hyung Lee created a multi-touch panel using FTIR technology and tested ot with a turn-based strategy game. In doing so, they created guidelines for multi-touch user interface designs and also argues that touch interfaces can be more comfortable and sensitive than traditional mouse and keyboard input given that the game interface is designed for multi-touch technology [6].

iPad + Real-Time Strategy Game

In their study, "One-handed interface for multitouch-enabled real-time strategy games," Crenshaw et al. designed a new touch-based interface for single-handed usage of large-sized touch devices. They first designed a real-time strategy game with a touch-based interface and surveyed participants with it. Their study showed that porting traditional desktop games to iOS require the development of a user interface specifically aimed at touchscreen interfaces.They argue that well designed multi-touch user interfaces can lead to faster and more accurate response times due to the larger area and less targetting precision required of gestures over traditional buttons [3].

3. STUDY OVERVIEW

This study will evaluate rhythm game interfaces through the following three stages:

1. *Design* various simplified rhythm game interfaces with categorized properties
2. *Prototype* app development of a rhythm game that demos the various interfaces
3. *Evaluation* of the interfaces through collecting user data and feedback

These three stages will lead to better understanding of the how the compared game interface properties affect gameplay effectiveness in rhythm games. The results of this study can be used in designing user interfaces for rhythm games as well as other time-critical applications.

3.1 Design

In the *Design* stage of the study, a basic rhythm game prototype was designed to contain demos implementing each target interface under study. These demos will also only recognize the simple, singular "tap" gesture to eliminate the variable of delay from performing complex touch gestures. Just like in most rhythm games, if a hitbox is tapped when a note is overlapping it, the game will register the note as "hit"; if a note passes its destination hitbox, the game will register the note as a "miss". To determine what types of interfaces to design and study, various commercial rhythm games were inspected for common game interface properties. These properties were then used to create simplified rhythm game interfaces.

Designs

The results of the analysis of the commercial rhythm games are shown in Figure 3 in the Appendix. Based on the generalized styles extracted from the analysis, simplified interface designs were drafted and categorized as shown in Figure 4 in the Appendix. Overall, eight designs were created based off object layout and object movement. For layout, the four categories were "column", "corners", "centre", and "grid". For movement, either the multiple notes move toward stationary hitboxes or hitboxes move toward multiple stationary notes. In these categorizations, the larger object is classified as the hitbox while the smaller object is classified as the notes. This clarification is needed due to the independence of movements when outside the column scrolling style.

Simplification

To simplify the interface design, there is only a maximum of four hitboxes for interface demos #1, #3 and #5. For #2 and #4, only one hitbox object is used; once the hitbox reaches the edge, it restarts. These two simplifications are done to reduce the degrading effect of gameplay complexity and interface clutter on player performance. Depending on complexity experienced during testing in the *Prototype* stage, the grids in #7 and #8 may be reduced from to 4x4 to 2x2 or be added as two more demos.

Comparisons

These eight interface designs encompass the majority of interface designs used by the rhythm games analysied in Figure 3. Demo #1 closely matches all the rhythm games under the "Falling Notes" style, particularly *Dance Dance Revolution*. Demo #2 matches *DJMax Technika*'s "Sliding Hitbox" style. Demo #3 is similar to the games under the *Spreading Notes* style but more emphasized in the spreading aspect. In those games, objects approach from a central area in the horizon up top to a row near the bottom; in demo #3, the objects approach from the horizon in the centre of the screen to the four corners. Demo #5 is similar to *Gitaroo Man Lives!*'s "Focusing Notes" style but with four focus points instead of one. There is currently no rhythm game with an interface similar to demo #4 nor #6; however, the two styles should be studied as the reverses of #3 and #5 respectively. Demo #7 closely matches *jubeat*'s "Filling Notes" style. Demo #8 is similar to *Osu! Tatakae! Ouendan!*'s "Shrinking Hitbox" style but restricts objects into a grid instead of allowing any location. This is done to reduce the complexity of the game and the player's possible reaction delay from an object appearing in an unexpected location. The "Streaming Notes" and "Sliding Cursor" styles are not covered in this study as they only operate on one row; the complexity of their gameplay comes from visual recognition of the object subtypes, a factor that is eliminated from this study through only using a single graphic for all notes objects and a single graphic for all hitbox objects.

Added Buttons

In order to test the attention requirement of each interface, a secondary, lower priority task with an maximal potential attention load needed to be added. This was done by adding two red buttons to the side of the screen (since the screen will be kept landscape in these demos) as shown in Figure 1. These buttons will not affect the main task (tapping the tapboxes with the correct timing) but provides a secondary, recordable task with no minimal attention requirement.

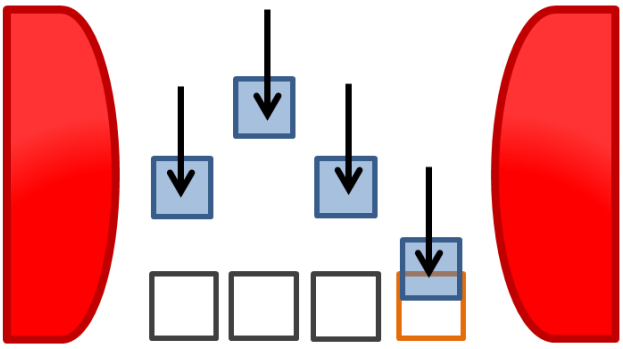


Figure 1: Example interface with side buttons.

3.2 Prototype

In the *Prototyping* stage of the study, the designed rhythm game prototype will be created based on the designs drafted in the previous *Design* stage.

Figure 2 shows the components structure of prototype app and how each component will interact with each other. The player will then go through the following stages of app usage:

App Startup

The Android app starts off with the "Main Menu" Activity, from which the player can choose which demo they want

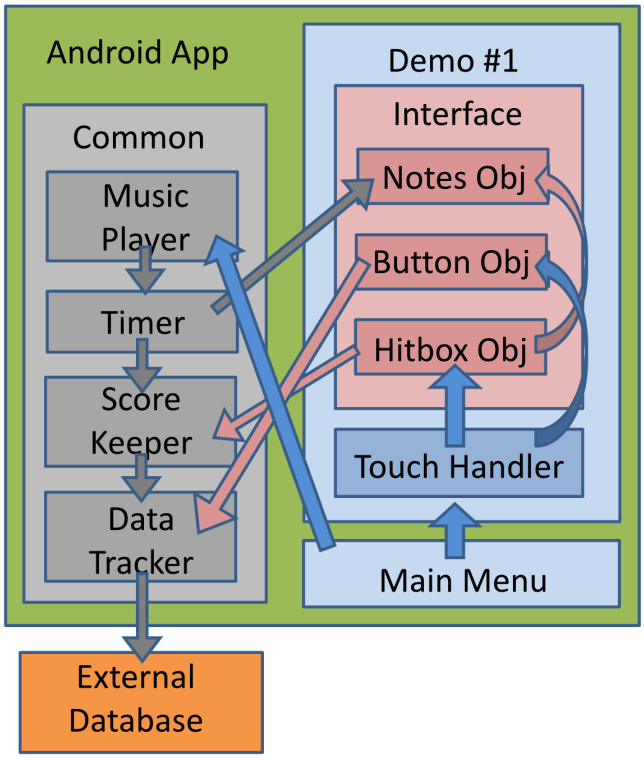


Figure 2: App components interaction diagram.

to test (only one demo is shown in the figure as an example) from a list. Once selected, the demo Activity will launch.

Demo Startup

Upon the start of the demo Activity, the touch handler, notes list, hitbox list, buttons list, and other objects will be initialized. The touch handlers will generally all be the same, but may be tailored to the specific demo. In a normal rhythm game, the notes list would normally be generated from or loaded based on the song that was selected. In this prototype, a hardcoded song's notes data will be loaded instead. The hitbox list would depend on the demo being tested. Because all the interfaces are designed as squares and the tablet's screen is wider than tall, two buttons will always be loaded, set to each side of the main interface box. By placing them on either side, they will not interfere with the main interface but is also easily accessible and always present. Ofter object initialization, the visible interface will start updating with the drawable objects and the music player will start.

Demo Notes Update

The music player will be synchronized with the timer, which holds the central "current time" of the game for the rest of the game to reference. On every update cycle, the timer will be resynced and all notes objects will synchronize with that timer. Based on the time difference between the current game time and the note's expected end time, the note object will calculate its difference from its target hitbox and tell the main interface's View to draw the note's graphics at the corresponding distance from the hitbox (if it is within the drawable area). For the demos with moving hitboxes, the notes will independently update its variables, but it will not move.

Demo Button Tap

Upon a touch event, the touch handler will check if a hitbox was tapped. If no hitbox was tapped, it will check if a button was tapped. If a button is tapped, the button will tell the data tracker that it was tapped. The data tracker will increment the visual counter of taps as well as record the time before the last tap occured.

Demo Hitbox Tap

Upon a touch event, the touch handler will check if a hitbox was tapped. If one is, that hitbox will then cycle through the list of notes objects to see if any were in range of itself. If none were, nothing happens and the game continues. If a note was within the hitbox's range, that note's hit status will be updated and the time difference will be recorded by the score tracker. The score tracker will increment the notes success count as well as the accuracy rating based on the time difference.

Demo Finished

After all the notes are done, some statistical data will be calculated based on the note hits during the game, such as the average accuracy and hit percentage. These numbers will be sent to the data tracker, which will then all be sent to an external database for analysis in the *Evaluate* stage of this study. At this step, the player will also be presented with a survey to fill out to give qualitative feedback on the demo.

3.3 Evaluation

In the *Evaluation* stage of the study, the rhythm game demos will be packaged together as a testable prototype with an added data gathering system. This playable prototype will be built for the Android 3.0 OS due to the facts that 1) Android 3.0 targets tablets, which usually feature large multi-touch displays [5], 2) Android tablet use is on the rise [9], and 3) I own an Android Tablet. This playable prototype will be tested by random samples of UPenn students as well as published online through the Android Market. The prototype's data gathering system will consist of two parts: 1) a data tracker that will provide quantitative values, and 2) a user survey and feedback system for qualitative but relative information.

Quantitative Measurements

There will be four raw data values that will be measured quantitatively:

- Notes hit count
- Notes hit average score
- Button press count
- Button press average frequency

The data values collected on note hits can be used to directly compare the player's performance between different game interfaces. Ideally the higher the hit count and the higher the average score, the more efficient the user interface was in keeping the player on rhythm (relative to the other demoed interfaces). The data collected on button presses can be used to directly compare the attention load experienced by the player between different game interfaces. If an interface requires a lot of attention, the player will press the

buttons fewer times or do so at a slower rate. As a baseline, demo #1 will be used for comparison purposes as "Falling notes" style rhythm games are the most familiar for most players.

Qualitative Surveys

There will be five qualitative aspects of the demo that will be rated in the surveys:

- Challenge (Easy –> Hard)
- Concentration (Low –> High)
- Fun (Boring –> Fun)
- Mastery (Easy –> Hard)
- Uniqueness (Old –> New)

"Challenge" will ask the question of, "Did the demo require a lot of skill (e.g. timing with hand-eye coordination)?". This result will be compared to the quantitative notes hit data to see if there are differences between perceived and actual results. "Concentration" will ask the question of, "Did the demo require a high amount of attention (e.g. less focus on hitting the button)?". This result will be compared to the quantitative button press data in the same way. "Fun" will ask the question of "Did you enjoy playing the demo (relative to the other demos)?", "Mastery" will ask the question, "Did you find the demo's interface intuitive and easy to learn/use?", and "Uniqueness" will ask the question, "Did you find the demo's interface and gameplay new and unique?". Finally, an extra field will be added for additional comments.

While the last three qualitative aspects listed above are unrelated to the efficiency of the rhythm game interface, they have merit in deciding the feasibility of using such an interface in an actual rhythm game. Swetser and Wyeth argues that enjoyment of games does not only depend on the final outcome (notes hitting performance in this case) but also factors such as concentration, mastery, and fun [7]. An interface that achieves high efficiency but is not perceived as unique (such as "Falling Notes") may not be a good choice if the goal is creating a new, innovative rhythm game. Alternatively, an interface with low efficiency but high fun factor may be worth investigating into further and refining.

The methods and statistical analysis approaches of analysing the collected data has not yet been planned out yet. They will most likely be decided upon after seeing the results of the initial sample survey.

4. TECHNICAL RESOURCES

Note: As of the current moment, the prototype is being written in pure Android Java code. After completion of the project, the prototype will be ported over to the *Unity 3* game engine.

Test Device

The target touchscreen device for testing in this project will be the Samsung Galaxy Tab 10.1. The Galaxy Tab is an Android 3.0 tablet running on a 1GHz dual-core processor and has a 10.1-inch capacitive touchscreen supporting up to 10 multi-touch points. It also has a built-in vibrating motor, allowing for additional haptic-feedback support.

Android SDK

`http://developer.android.com/sdk/`

The *Android SDK* is the set of development tools and core libraries required for developing Android applications.

Eclipse

`http://www.eclipse.org/`

Eclipse is the standard IDE (Integrated Development Environment) for developing Android apps. The *Android SDK* is designed to integrate with *Eclipse*.

Google Code

`http://code.google.com/p/beats2/`

Google Code is a group of online resources, tools, and hosting for project development. In this project, *Google Code* will be used for hosting the SVN source code, as well as bug tracking and wiki hosting.

Google Analytics

`http://www.google.com/analytics/`

Google Analytics is a free online service for tracking usage of features in applications. It will probably be used in this project to track the user feedback during the *Evaluation* stage.

Unity 3

`http://unity3d.com/unity/publishing/android.html`

The *Unity 3* development tools consists of the *editor*, the series of tools for developing games, and the *game engine*, the software backend that allows the developed games to run on target platforms. *Unity 3* was chosen due to its cross-platform support of other touchscreen-supporting platforms and large community and professional support base. In this project, the toolkit will be developed as script extentions for the editor, while the prototypes will be run using the game engine. The *Unity 3* license required for this project will be the regular *Unity 3* package with the additional Android add-on to allow for development of Android apps.

Orthello2D

`http://www.wyrmtale.com/products/unity3d-components/orthello`

Orthello2D is a free 2D framework for *Unity 3* game development. Since the demos being built for this study are meant to be simple and 2D, this framework will very likely be used.

FingerGestures

`http://www.fatalfrog.com/?page_id=140`

FingerGestures is a series of scripts for *Unity 3* that adds support for advanced single and multi-finger gestures. The prototype in this study only requires tap actions, but more touch gestures will be needed if it were to develop beyond the prototype stage. The efficiency of different finger gestures with different rhythm game interfaces is also another interesting area to be studied.

KinectWrapper

`http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Open_NI`

KinectWrapper is a series of scripts for *Unity 3* that adds support for the XBOX Kinect via wrapping *Open NI* drivers. With the Kinect, a virtual touch grid can be implemented and supported as the input channel instead of a touchscreen.

Whether or not the same efficiency of the rhythm game interfaces studied here stays the same when switching from a physical touchscreen to a virtual one is also further topic to study.

5. WORK PLAN

5.1 Work Completed

As of the moment, the *Design* stage is complete and the *Prototype* stage has started. The work-in-progress prototype currently has a working menu and demos #1, #3, and #5 are implemented but not fully functional as demos. More specifically, the interface and touch handler portions are done, but none of the common components are complete, nor is the external database ready.

Learning how to use the *Unity 3* engine proved more difficult than originally though, mostly due to the fact that the engine was meant for designing large, complex 3D games instead of simple 2D games such as the demos used for this study. As a result, the prototype will continue being built with standard Android Java libraries, then later ported over to *Unity 3*. The following are completed work items based on the original plan:

1) Setup - *Completed*
- Obtained Unity 3 license with Android add-on
- Set up Google Code project
- Set up Eclipse with Android SDK
- Create a "Hello World" Android app through Unity 3

2) Investigation - *Completed*
- Experimented with FingerGesture scripts
- Experimented with Orthello2D framework
- Decided that it would be faster to write the prototype first with Android Java

3) Drafting - *Completed*
- Drafted designs for rhythm game interfaces
- Drafted designs for rhythm game engine
- Createed a simple prototype demoing touch input and drawing

5) Prototyping - *In Progress*
- Created implentation demos for interfaces #1, #3, and #5

5.2 Remaining Work

The following are remaining work items for next semester.

5) Prototype - *Mar 9th*
- Create implentation demos for interfaces #2, #4, #6, #7 and #8 [3 weeks]
- Complete remaining parts of rhythm game engine [3 weeks]
- Set up Google Analytics for data tracking [2 weeks]
- Create proper graphics and cleanup code [1 week]

6) Evaluation - *Apr 6th*
- Selectively survey students for prototype feedback and fixing [0.5 week]
- Randomly survey students [0.5 week]
- Large-scale survey gamers via Market publishing [3 weeks]
- Perform statistical analysis on collected results [ongoing]

7) Report - *Apr 20th*
- Draw conclusions based on results of feedback analysis
- Write report summarizing findings

6. APPENDIX

See figures on following pages.

7. REFERENCES

[1] Andrew Bragdon, Eugene Nelson, Yang Li, and Ken Hinckley. Experimental analysis of touch-screen gesture designs in mobile environments. http://yangl.org/pdf/gesturestudy-chi2011.pdf.

[2] Emiko Charbonneau, Andrew Miller, Chadwick Wingrave, and Joseph J. LaViola Jr. Understanding visual interfaces for the next generation of dance-based rhythm video games. http://dl.acm.org/citation.cfm?id=1581092.

[3] Nicole Crenshaw, Alexandra Holloway, Scott Orzech, and Wai Son Wong. One-handed interface for multitouch-enabled real-time strategy games. http://ga.fdg2011.org/papers/2.pdf.

[4] Bradley H. Hayes. Software driven multi-touch input display as an improved, intuitive, and practical interaction device. http://www.bradhayes.info/thesis.pdf.

[5] Google Inc. Android 3.0 platform highlights. http://developer.android.com/sdk/android-3.0-highlights.html.

[6] Yong Chul Kwon and Won-Hyung Lee. A study on multi-touch interface for game. fdff.

[7] Penelope Sweetser and Peta Wyeth. Gameflow: A model for evaluation player enjoyment in games. http://dl.acm.org/citation.cfm?id=1077253.

[8] Geoff Walker. The best of times. http://www.informationdisplay.org/issues/2010/03/art3/ar March 2010.

[9] Jun Yang. Android tablets gained on ipad in third quarter. http://www.bloomberg.com/news/2011-10-21/android-tablets-gained-on-ipad-in-third-quarter-researcher-says.html.

Rhythm Game	Layout	Notes	Hitbox	Movement	Style
Dance Dance Revolution	4/6 columns	4/6 arrows	Box at top	Notes scroll up	Falling Notes
In The Groove	4 columns	4 arrows	Box at bottom	Notes scroll up	Falling Notes
Pump It Up NX	5 columns	4 arrows + middle stomp	Box at bottom	Notes scroll up	Falling Notes
Dance Maniax	4 columns	4 motion sensors	Bar at top	Notes scroll up	Falling Notes
Beatmania IIDX	6/8 columns	5/7 bars + 1 scratch	Bar at bottom	Notes fall down	Falling Notes
Pop 'N Music	9 columns	9 buttons	Bar at bottom	Notes fall down	Falling Notes
DJMax	4/5/6/8 columns	4/5/6/8 bars	Bar at bottom	Notes fall down	Falling Notes
GuitarFreaks	3 columns	3 tabs	Bar at bottom	Notes fall down	Falling Notes
Drummania	6 columns	5 drum + 1 foot pedal	Bar at bottom	Notes fall down	Falling Notes
Keyboardmania	24 columns	24 keys	Bar at bottom	Notes fall down	Falling Notes
Guitar Hero	5 columns	5 tabs	Bar at bottom	Notes approach from distance	Spreading Notes
DJ Hero	3 columns	3 buttons on scratch	Bar at bottom	Notes approach from distance	Spreading Notes
Rockband	5 columns	5 tabs or 4 drum + 1 foot pedal	Bar at bottom	Notes approach from distance	Spreading Notes
Taiko no Tatsujin	1 row	2 drum parts	Box on side	Notes stream to single point	Streaming Notes
The iDOLM@STER	1 row	6 buttons	Box on side	Notes stream to single point	Streaming Notes
Hatsune Miku: Project DIVA	Fullscreen	8 buttons	Sequence of hitboxes	Notes focus to corresponding hitbox	Focusing Notes
Gitaroo Man Lives!	Fullscreen	4 buttons	Box in centre	Notes focus to single point	Focusing Notes
DJMax Technika	3/4 rows	3/4 buttons	Moving bar	Hitbox slides across rows	Sliding Hitbox
Parappa The Rapper	1 row	4 buttons	Cursor	Cursor slides across row	Sliding Cursor
Audition Online	1 row	4 arrows	Cursor	Cursor slides across row	Sliding Cursor
Osu! Tatakae! Ouendan	Fullscreen	Anywhere buttons	Shrinking rings	Rings shrink around buttons	Shrinking Hitbox
jubeat	Grid	16 buttons	Collapsing box	Box solid fills grid	Filling Notes

Figure 3: Analysis of the interfaces of various rhythm games.

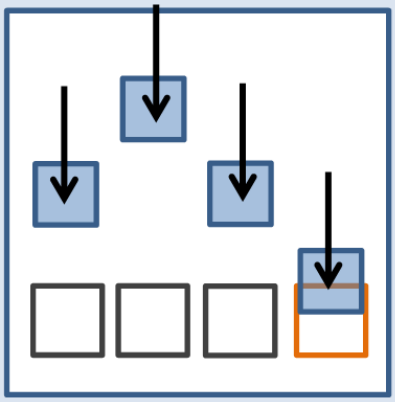
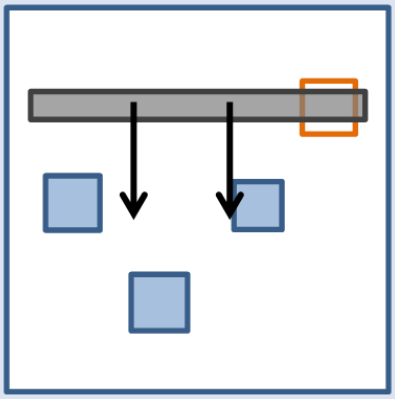
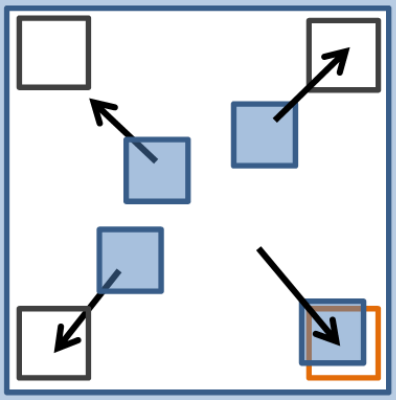
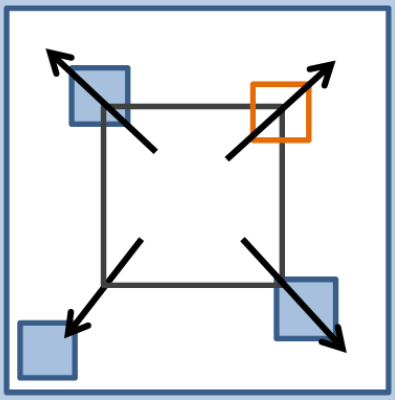
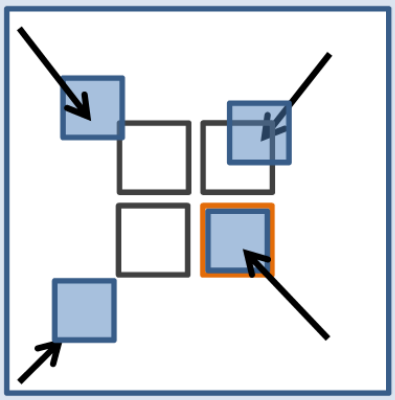
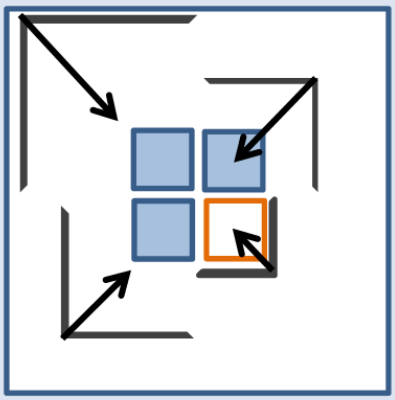
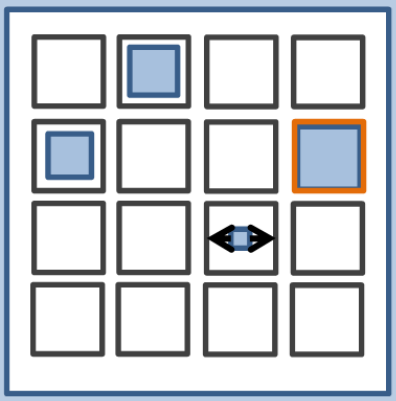
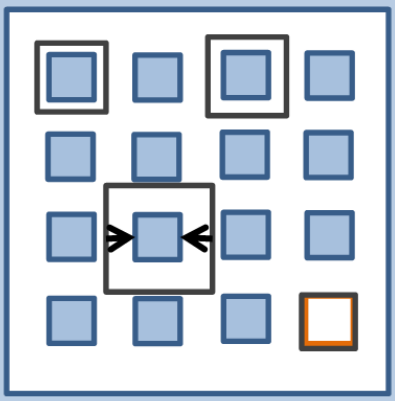
Moving Notes -> Stationary Hitbox		Moving Hitbox -> Stationary Notes	
Diagram	**Description**	**Diagram**	**Description**
	**Falling Notes** Demo #1, Column style: Columns of points move toward a single line		**Sliding Hitbox** Demo #2, Column style: Single line slides down across columns of points
	**Spreading Notes** Demo #3, Corners: Streams of points move diagonally toward corners		**Expanding Hitbox** Demo #4, Corners: Single box expands toward points along diagonals
	**Focusing Notes** Demo #5, Centre Points from corners move toward central area		**Collapsing Hitbox** Demo #6, Centre Box quadrants collapse toward central area
	**Filling Notes** Demo #7, Grid Independent points expand to fill grid areas		**Shrinking Hitbox** Independent boxes shrink to surround points

Figure 4: Interfaces designs to be demoed in the prototype.