Moving a Desktop Physics-Based Programming System to a Tablet

Robert Sheehan
University of Auckland,
Auckland, New Zealand
0064 9 373 7599x88814
r.sheehan@auckland.ac.nz

ABSTRACT
Converting an existing children’s programming environment to a tablet version has many challenges; the most important being that interactions have to be initiated and controlled by touch and that the display area is significantly smaller. Such a platform change can also provide the opportunity for improvements. This paper describes the changes required when moving the Fizz environment to the new platform focusing on considerations of interaction modes. The desktop version was designed to minimise modes of operation, effectively operating with two. The tablet version explicitly provides more modes in order to overcome some of the tablet deficiencies. In doing so it was discovered that the use of more modes helped to structure the programming process in this work in progress.

Categories and Subject Descriptors
D.1.7 [Programming Techniques]: Visual Programming

General Terms
Design, Human Factors, Languages.

Keywords
Programming environment, children, touch, games, simulations.

1. INTRODUCTION
Modern tablets such as the iPad provide power approaching that of desktop machines from only a few years ago. They accompany this with light weight, excellent screens and reliable touch input. Because of their form factor they can be carried easily and held while being used. Many families and schools now have or require such devices for the use of children. Indeed tablets are overtaking personal computers in availability[1]. However, providing compelling and engaging creativity applications on such devices is not a simple matter of converting existing desktop applications to tablet versions. There are several limitations associated with tablets and also some advantages. In this paper we describe and examine changes to a physics-based programming environment usable by children, originally developed as a desktop computer application, as it was remodelled to maximise the advantages of a tablet implementation.

1.1. Tablet Advantages
Ever since the design of the Dynabook[2] two of the important characteristics for a computing device for children have been low weight and ease of portability. These are advantages tablet devices have over desktop computers and even over the lightweight laptops which are now available. Children can easily carry these devices in their hands as they move around homes or classrooms.

Interactions with tablets are mostly carried out using their multitouch displays. This is not necessarily an advantage. Laptop computers with their traditional keyboards are better for some computing tasks, in particular those which require lots of data or text to be entered. But there are some tasks where touch input facilitates control. When it comes to the selection of onscreen objects touch has been shown to be superior to mouse selection as long as the size of the objects is suitable[3]. If we use multitouch to allow bimanual selection then touch is better still[4], more efficient and accurate.

Accuracy of selection is not the only advantage touch has over traditional mouse and keyboard interaction. The now standard techniques of rotating and zooming objects using two fingers are much easier to use than actions with a mouse.

For our purposes it also appears that something else is going on when we select and move objects via touch. Using touch makes using applications more enjoyable and engaging. As Watson et al stated:

“Our results also indicate that touch interaction improves positive affect and feelings of autonomy, which may be valuable in the design of interfaces intended to support creativity.”[5]

This is noticeable when people use the tablet application described here. Drawing a toy with your finger and then immediately being able to interact with that toy and see it move realistically promotes engagement and enjoyment.

1.2. Tablet Disadvantages
The biggest disadvantage of modern tablet computers is the reduction in the amount of display space. Desktop computers are commonly attached to displays with diagonal measurements over 20”, whereas tablet displays range from 7” to 11” on the diagonal. Even though the number of pixels in the displays may be comparable between the tablet and the desktop, the greater pixel density of the tablet gives higher resolution images but the display provides less usable area.
This means that menus, sidebars of information, and floating palettes of tools are less likely to be used in tablet applications. If they are used they have to be very carefully arranged so as not to inhibit access to the areas of the display which the user is currently modifying.

Another interaction disadvantage with tablets is the fat finger problem[6]. As a user goes to select or interact with a screen object the user’s hand and finger obstruct the view of the object and other parts of the display. Since the display is smaller than the corresponding desktop display this problem is exacerbated.

There is also no feedback from touch interfaces until the display is touched. Desktop displays with mouse cursors show the location before any action is taken. One example of this is when selecting a object. Objects under the mouse cursor can be highlighted to correctly identify which object will receive notification when the user clicks the mouse button. This was used in the desktop version of the physics-based programming environment to identify which object was intended even if there were multiple objects in the same area of the screen. This cannot be done with the tablet version. The current solution is to move objects out of the way so that the intended object can be selected.

2. TWO FIZZES
In order to understand the design choices made in developing the tablet version of the programming environment we briefly introduce the desktop version.

2.1. Desktop Fizz
The desktop version of the physics-based programming environment for children is called Fizz[7, 8]. Using Fizz, children can draw any shape which automatically becomes a programmable object with 2D physical properties. These programmable shapes are known as toys. Apart from the original drawing of a toy all subsequent interactions with a toy including programming it to respond to events are conducted through a context sensitive pie menu[9](Figure 1).

Figure 1. The toy menu in desktop Fizz
Collections of toys, along with their programmable actions, are known as scenes and one of the possible actions is to change from one scene to another, thus allowing narrative flow or game levels. The toys in a desktop Fizz program are stored in a sliding drawer called the toy box (Figure 2).

As far as possible desktop Fizz was designed to operate without modes. In other words all user interface actions are possible at all times. This means it is possible to draw new toys while a scene is running, or to drag a toy from the toy box and drop it into a running scene. The only time this changes is when the user right-clicks on a toy to open its menu. When this happens the current scene stops running. Running and stopping a scene is always under user control, either from the main menu or by toggling the space bar.

Figure 2. A scene with the open toy box in desktop Fizz

2.2. iFizz
Because of the lack of display space in a tablet the decision was made to incorporate a number of modes in the app. This way each of the modes helps the user to concentrate on what he or she is doing at the time. This enables the limited screen space to be devoted wholly to the current activity whether it is drawing a toy, positioning toys in a scene, adding event/actions to a toy or playing a scene. The current version of the tablet version of Fizz called iFizz was hence designed to incorporate four different modes.

Generally modes are regarded as problematic in user interfaces. As Jef Raskin says, “Modes are a significant source of errors, confusion, unnecessary restrictions, and complexity in interfaces.”[10]. The problems arise because the same user interface action can lead to different results and because being in one mode restricts users from carrying out actions which are only available in other modes.

In iFizz the modes are implemented as different views which fill up almost all of the display and they are selectable from a tab-bar at the bottom of the display. Touching the corresponding tab icon changes the view shown on the display. The different modes were chosen to reflect the work-flow as a user creates and runs a scene.

3. THE iFIZZ MODES
The modes are:
- Toy Maker
- Scene Maker
- Action Adder
- Scene Player

First a user must create a toy or toys; these are the active objects in scenes (Figure 3). Then a scene must be created. This entails drawing background objects and positioning toys within the scene (Figure 3). Only then can actions be added because the events which cause actions (such as collisions) may require the presence
of more than one toy. When actions have been added the scene is ready to run by changing to the playing mode. Actually the scene can be run before actions are added but in this case the only things the toys do is react to gravity.

Figure 3. The toy (left) and scene (right) maker modes

3.1. Toy Maker

Creating iFizz was also seen as an opportunity to improve on some of the awkward interactions in desktop Fizz.

The way to define a toy in desktop Fizz was to draw a single stroke. This meant it was difficult to create complex shapes. In order to do this the technique was to glue two or more toys together, effectively combining the toys into one toy. This process required opening up the toy menu, clicking the connect button, clicking the glue button and then selecting the toy to be glued to.

Creating a toy with a single stroke was useful if a toy was created when a scene was running. Because of this the stroke was automatically provided with a random colour. So toys were single strokes of a single colour. The automatically chosen colour could be changed from the toy menu after the toy was created. Normally users created toys while the scene was not running and almost always went and changed the automatic colour of the toy immediately.

The Toy Maker mode in iFizz is more powerful because it deals with this single task. In this mode anything drawn is part of the same toy. Toys now naturally consist of multiple strokes; the lines can either be straight using the line tool or follow the user’s finger using the squiggle tool. There is also the ability to add circles. All of these parts are individually coloured and editable whenever the toy gets selected from the toy box (Figure 4).

When a toy gets dropped into a scene the default size is one quarter the original drawn size. This means that the whole display can be used to draw the toy but the toy is a usable size when dropped into a scene. When toys are dropped into a scene they can be repositioned, rotated and resized. Toys in desktop Fizz could be repositioned and rotated but they could not be resized.

3.1.1. The Toy Box

The toy box changed only marginally in the transition to iFizz. Originally it was a panel which appeared over a scene from the left when the toy box handle was clicked (Figure 2). In iFizz it appears when a button is pressed. One difference is how toys end up in the toy box. Desktop Fizz required the user to drag a toy into the toy box; iFizz puts all toys (as they are created in the Toy Maker mode) into the toy box. Once again this was a consequence of using the Toy Maker mode to create toys, because in desktop Fizz every single stroke was a toy and it was intended that the toy box would only contain significant toys. Conversely to place a toy into a scene from the toy box in desktop Fizz required dragging the toy from the toy box. This was acceptable as the toy box occupied only a quarter of the scene area. The toy box in iFizz occupies most of the scene area and so touching a toy places it in the centre of the scene and closes the toy box.

Figure 4. The toy box in iFizz

3.2. Scene Maker

The Scene Maker mode is almost identical to the Toy Maker mode (Figure 3). Both are used to draw objects on the display and share the same tools. The major difference is that toys from the toy box can be dropped into the Scene Maker. But the Scene Maker mode also solves another problem.

In desktop Fizz there was very little difference between toys which move in the environment, dynamic toys, and toys which never move, static toys. Static toys were used to provide the ground or other stationary objects for the dynamic toys to rest on, roll along, bounce off or otherwise interact with. Both types of toy were created in exactly the same way. The difference was that a dynamic toy could be converted to a static toy by selecting the freeze menu item from the toy’s top-level menu. This meant that a static toy could also be turned back into a dynamic toy if desired. In practice children frequently forgot to freeze a toy which they intended to be static. This was only noticed when the child ran the scene and the toy fell down under gravity or moved when hit by another toy. Fortunately, it was easy to correct this common behaviour.

With iFizz the addition of the Scene Maker mode provides an elegant solution to this problem. Any drawing done in this mode is taken to be part of the static background. Users can still understand these drawings as static toys but it is more likely they will perceive them as things attached to the background rather than as toys. As we have seen the two maker modes, Toy Maker and Scene Maker, share the same drawing tools. The differences are in the default background colour, the width of the pen used to draw (because the scene is at full and not 4x size), and a different title. Whether this is enough to convey the difference between the modes has yet to be determined.

3.3. Adding Actions

The menu system in iFizz is very different from that in desktop Fizz. As we have seen desktop Fizz allows most actions to be performed on a toy by opening up a pie menu which by necessity includes many different submenus. One of those submenus allows the addition of event/actions to the toy, thus programming the toy.
Partly because of the display area problem iFizz uses a different style of menu to perform the same task. All menus in iFizz provide their menu items as buttons down the left and right hand edges of the tablet display (Figure 3). The Action Adder mode is the same. In this mode the left edge of the display shows all possible event types which provide the conditions which cause an action to occur. Currently, these include taps, collisions and timing events but the events are being extended to include accelerometer and score events. The right display edge of the mode contains the action buttons. These are the actions produced when the causal event arises. Actions such as applying a force, creating a toy, exploding and jumping to a new position are provided.

3.4. Playing
The mode to play a scene has few buttons. The edges of the screen are special. Scenes are designed to be played in landscape mode and the edges are used for touch interactions. This keeps the user’s fingers off of the play scene itself and also coincides with holding the iPad with two hands using the thumbs of each hand to make the gestures. When a scene is playing there are user definable buttons (only 3 on each side to minimise player button selection error while playing an intense game). These buttons occupy the same area as the normal buttons in the non-playing modes (Figure 5).

Figure 5. The play mode

4. OTHER IMPROVEMENTS
There are several other improvements incorporated into iFizz. Simplifying the process of adding wheels to toys is one of them.

Because adding wheels to toys is such a common requirement the desktop Fizz had already been modified to make this simpler[8]. The technique to add wheels was reduced to three mouse clicks.

In iFizz any circle drawn in the Toy Maker mode is automatically assumed to be a wheel and is pinned to the rest of the body of the toy, this makes adding wheels as simple as possible, no extra interaction is required after the drawing of the wheel. If a circle is not intended to be a wheel e.g. the face of a character it makes effectively no difference to the toy’s interactions.

5. CONCLUSION
Translating a 2D physics-based programming environment from a desktop version to a tablet version necessitated a new design. The design developed ameliorates the disadvantage of the smaller display space of the tablet and attempts to benefit from the advantages the touch interface provides. The solution was to partition the interactions into clearly defined modes of operation where each mode can use the full display area. The chance to redesign the application meant that some of the deficiencies in the desktop version could also be remedied.

The resulting iFizz app is partially complete and we look forward to testing the final version before making it freely available.

6. ACKNOWLEDGEMENTS
Thanks to David Haden and Sara Metz for additional implementation work.

7. REFERENCES