One Touch Composer: Creating a Digital Sheet Music App with Touch, Stylus, and Keyboard Control on Microsoft Windows* 8 Tablets

By Dominic Milano

Introduction

When Dmitriy Golonanov, the CEO of the international company, Maestro Music Software, learned that many musicians would rather use a lightweight tablet than carry a laptop to gigs or a classroom, he decided to enlist a small team of developers, which included Sergey Samokhin, to create music notation software that would run on Microsoft Windows* 8 tablets or Ultrabook™ 2 in 1s running Windows 8.1. The result of this inspiration is One Touch Composer for Microsoft Word*, which gives composers, music educators, and musicians of all levels the ability to create and share great-looking sheet music (Figure 1). Originally Submitted under the name One Touch Notation, One Touch Composer was the winning entry in the Tablets category of the Intel App Innovation Contest 2013 (AIC 2013). During the contest Dmitriy used resources from the Intel® Developer Zone.

![Figure 1: One Touch Composer for Microsoft Word* leverages the Lenovo ThinkPad* Tablet 2's multi-touch screen and supports input from touch, stylus, and keyboards.]

To accomplish their goal, Golonanov's team faced a number of challenges, including having to design a user experience that was both intuitive and practical for professional and amateur musicians, music teachers, and students. Music notation, after all, consists of a complex collection of specialized 2D symbols, and even a relatively simple song can span many pages. Displaying and interacting with that complexity on a small tablet screen challenged the team on a variety of levels, from programming a UI that could handle input from touch, stylus, and two kinds of virtual keyboards to making efficient use of computing resources.

Approach

To meet the six-week development cycle that the AIC 2013 rules specified, Golonanov and his team knew their app wouldn’t be able to meet the needs of all users. So rather than create a program that
combined music notation with extensive audio playback capabilities via MIDI* and sound fonts—an approach that’s common in desktop PC music-notation programs—they restricted audio playback to built-in sounds and focused on giving users the ability to easily create and share digital sheet music.

In doing so, Golonanov and his team encountered their biggest challenge—changing their design philosophy from giving users as many powerful tools as they could pack into a desktop app to providing a simplified, intuitive touch-driven experience for a mobile device.

Golonanov and Samokhin worked with Alexey Bagenov, a music notation expert and conductor of the National Wind Orchestra of the Ukraine, who literally drew the UI on a piece of graph paper during a Microsoft BizSpark* UX Tour workshop (Figure 2).

Figure 2: Alexey Bagenov’s sketch of One Touch Composer’s UI positioned key functions around the perimeter of a touch screen.

Bagenov’s solution provided access to all the app’s functions from a single screen, without traditional menus. Instead, contextual submenus provide access to features such as selecting clefs or entering text.
The UI positions the most commonly accessed functions on the left and right sides as well as along the top of the screen (Figure 3). Notes can be input from a virtual touch-screen music keyboard positioned across the bottom, and if the device has a physical keyboard, it can be used to name files and enter text (Figure 4). If no keyboard is present (Figure 5), the user can select a virtual touch-screen keyboard. If a MIDI keyboard is attached to the device over a USB, the program uses a timed function to determine whether a keyboard is present and automatically connects to it.
The Windows 8 Touch API provided all the necessary functions for implementing multiple input technologies. During early usability tests, the team learned that Microsoft’s recommended minimum of 23x23 pixels for touchable areas was too small for the buttons they wanted to use. Instead, they set 40x40 as the default size for buttons and gave users the ability to change the defaults and customize the interface. Stylus input, the team learned, proved very helpful for handling small UI elements such as staff lines, notes, and dynamics symbols. Finger-based touch interaction is best for simple commands such as scaling, zooming, and scrolling. To avoid unresponsive behavior caused by gaps between touch-targets, they avoided gaps larger than 5 pixels. The team was also careful to make UI elements positioned around the edge of the touch screen large enough to tap easily with a fingertip.

![Figure 5: If no keyboard is detected, a virtual keyboard can be used to input text (left). The multi-touch screen allows users to interact with the software using touch and a stylus at the same time (right).](image)

**Migrating Legacy Code**

Some aspects of One Touch Composer were written in Delphi*. Other aspects, such as performance-critical components, were written using Assembler—the inline assembler for writing x86 machine code directly within Delphi programs. Assembler supports Intel® MMX™ technology and Intel® Streaming SIMD Extensions. Golonanov’s team borrowed heavily from existing resources, which played a large role in being able to create the app in the allotted six weeks. For example, One Touch Composer makes use of Maestro Music Software’s extensive, proprietary vector graphics music notation library, which the app accesses using the Delphi GDI+ API. This high-level, easy-to-use API is modeled after the .NET system. Drawing namespace and provides the ability to both display and print 2D graphics and formatted text.

To draw musical symbols on-screen, Golonanov and Samokhin used the DrawString method, shown here:

```pascal
procedure TMSWindow.WMGesture(var Msg: TMessage); var
  gi: TGestureInfo;
  bResult, bHandled: BOOL;
  dwError: DWORD;
  L: Integer;
  ptZoomCenterX, ptZoomCenterY, K, angle: Single;
  aptSecond: TPoint;
  r: TRect;
```
bHandled := False;
if bTouchGestureAPIPresent then
begin
  ... test coord of symbols

When a user touches a UI button, which in this example is a Bass Clef button, the following code is used:

procedure TFProgramModule.OnButtonClick(Sender: TObject); begin
  if FMode = mdSetting then
  begin
    if Sender is TControl then
      SetActiveButton(TECItem(TControl(Sender).VCLComObject));
  end else if TECItem(TControl(Sender).VCLComObject) is TECButton then
    with TECButton(TControl(Sender).VCLComObject) do
    case Action of
      CLEF_ACTION: AClefExecute(Self); ...

The app also drew on two existing Maestro Music Software products, Maestro Book* and Maestro Book Online*, to provide the ability to publish, print, and share ebook versions of the music created in One Touch Composer.

**Efficiency and Multi-Threading**

GDI+ is by no means the fastest way to display 2D graphics on a screen, but it helped ensure that One Touch Composer would run on what Golonanov calls “classic computers” (older x86 systems).

The developers also took care to ensure that One Touch Composer would run on a minimum of 1 GB RAM. The AIC development system, a Lenovo ThinkPad* Tablet 2 with an Intel® Atom™ processor inside, featured 2 GB RAM, which was more than enough to give users the ability to handle musical scores longer than 20 pages.

One Touch Composer’s code was parallelized and multi-threading was used to enable simultaneous play back and printing of digital sheet music (for example), while a user reads RSS news feeds and so on.
Lessons Learned

By far the biggest challenge in creating One Touch Notation for Word on Windows 8 tablets was simplifying a very complicated application. Golonanov stressed the importance of considering the needs of the audience—in his case, musicians, composers, and educators—in determining the core features of a mobile app.

Golonanov also advises developers to embrace all available input technologies—in this case, touch, stylus, and keyboards—utilizing each for their best-suited functions. He also suggests that developers avoid locking their users into defaults that may not suit the size of their fingers or be the preferred way of interacting with a device or software.

Golonanov’s team continually analyzes the needs of musicians, eliciting and listening to their feedback to improve their applications. Golonanov also continues to participate in the Habrahabra developer community to get ideas from other professionals and to keep pace with platform developments.

Community Help

According to Golonanov, the Habrahabra developer community, one of four international developer communities that participated in AIC 2013, helped his team refine One Touch Composer by providing guidance in the form of written articles about developing software for musicians. The Intel Developer Zone forums were another invaluable resource in providing expert insights.

About the Developer

Dmitriy Golonanov’s interest in music led him to develop music notation software. In 2003, he started at MagicScore Music and in 2013 became CEO of Maestro Music Software.

Helpful Resources

Intel Developer Zone offers tools and how-to information for cross-platform app development, platform and technology information, code samples, and peer expertise to help developers innovate and succeed. Join our communities for the Internet of Things, Android*, Intel® RealSense™ technology and Windows* to download tools; access dev kits; share ideas with like-minded developers; and participate in hackathons, contests, roadshows, and local events.

Related Articles

- Touch Developer Guide for Ultra Mobile Devices
- Ultrabook™ and Tablet Windows® 8 Sensors Development Guide
- Touch and Sensors