Gameplay: Touch Controls and 2 in 1 Awareness for Your Favorite Games

GestureWorks Gameplay is a revolutionary new way of interacting with popular PC games. Gameplay software for Windows 8 lets gamers use and build their own Virtual Controllers for touch, which are overlaid on top of existing PC games. Each Virtual Controller overlay adds buttons, gestures, and other controls that are mapped to input the game already understands. In addition, gamers can use hundreds of personalized gestures to interact on the screen. Ideum's collaboration with Intel gave them access to technology and engineering resources to make the touch overlay and 2 in 1 awareness in Gameplay possible.

Check out this one-minute video that explains the Gameplay concept.

It's All about the Virtual Controllers
Unlike traditional game controllers, virtual controllers can be fully customized, and gamers can even share them with their friends. Gameplay works on Windows 8 tablets, Ultrabooks, 2 in 1 laptops, all-in-ones, and even multitouch tablets and large touch screens.

Figure 1: Gameplay in action on Intel Atom-based tablet

"The Virtual Controller is real! Gameplay extends hundreds of PC games that are not touch-enabled and makes it possible to play them on a whole new generation of portable devices," says Jim Spadaccini, CEO of Ideum, the makers of GestureWorks Gameplay. "Better than a
physical controller, Gameplay's Virtual Controllers are customizable and editable. We can't wait to see what gamers make with Gameplay.”

Several dozen pre-built virtual controllers for popular Windows games come with GestureWorks Gameplay (currently there are over 116 unique titles). Gameplay lets users configure, layout, and customize existing controllers as well. The software also includes an easy to use, drag-and-drop authoring tool allowing users to build their own virtual controller for many popular Windows-based games distributed on the Steam service.
Users can place joysticks, D-pads, switches, scroll wheels, and buttons anywhere on the screen, change the size, opacity, and add colors and labels. Users can also create multiple layout views that can be switched in the game at any time. This allows a user to create unique views for different activities in the game, such as combat versus inventory management functions in a Role Playing Game.

Figure 3: Virtual controller layout view

Figure 4: Virtual controller global gestures view
Powered by the GestureWorks gesture-processing engine, aka GestureWorks Core, Gameplay provides support for over 200 global gestures. Basic global gestures such as tap, drag, pinch/zoom, and rotate are supported by default but are also customizable. This allows extension of overlaid touch controllers, giving gamers access to multi-touch gestures that can provide additional controls to PC games. For example, certain combat moves can be activated with a simple gesture versus a button press in an FPS. Gameplay even includes experimental support for accelerometers so you can steer in a racing game by tilting your Ultrabook™ or tablet, and it detects when you change your 2-in-1 device to tablet mode to optionally turn on the virtual controller overlay.

**Challenges Addressed During Development**

Developing all this coolness was not easy. To make the vision for Gameplay a reality, several technical challenges had to be overcome. Some of these were solved using traditional programming methods, while others required more innovative solutions.

**2 in 1 Transition Support**

Early on in Gameplay development, we decided to include basic support for the 2-in-1 transition (going from clamshell to tablet mode) available on some new Ultrabooks. The vision was to hook into the game as usual but not display the overlay if it was launched in clamshell mode. Then, during game play, if the system was switched to tablet mode, the Gameplay Virtual Controller overlay would immediately appear to allow touch-only game control. You can see this capability in action on any virtual controller run on an Ultrabook with 2-in-1 support. In the virtual controller edit mode, just enable 2-in-1 mode switch support in the experimental section of the settings tab.

![Figure 5: Virtual controller settings view](image-url)
For those interested in learning more about detecting the 2-in-1 transition, there is an excellent guide with a sample application listed in the References section.

**DLL Injection**

DLL injection is a method used for executing code within the address space of another process by getting it to load an external dynamically-linked library. While DLL injection is often used by external programs for nefarious reasons, there are many legitimate uses for it, including extending the behavior of a program in a way its authors did not anticipate or originally intend. With Gameplay, we needed a method to insert data into the input thread of the process (game) being played so the touch input could be translated to inputs the game understood. Of the myriad methods for implementing DLL injection, Ideum chose to use the Windows hooking calls in the SetWindowsHookEx API. Ultimately, Ideum opted to use process-specific hooking versus global hooking for performance reasons.

**Launching Games from a Third-Party Launcher**

Two methods were explored for hooking into a target process's address space. The application can hook into a running process's address space, or the application can launch the target executable as a child process. Both methods are sound; however, in practice, it is much easier to monitor and intercept processes or threads created by the target process when the application is a parent of the target process.

This poses a problem for application clients, such as Steam and UPlay, which are launched when a user logs in. Windows provides no guaranteed ordering for startup processes, and the Gameplay process must launch before these processes to properly hook in the overlay controls. Gameplay solves this issue by installing a lightweight system service during installation that monitors for startup applications when a user logs in. When one of the client applications of interest starts, Gameplay is then able to hook in as a parent to the process, ensuring the overlay controls are displayed as intended.

**Lessons Learned**

**Mouse Filtering**

During development, several game titles were discovered that incorrectly processed virtual mouse input received from the touch screen. This problem largely manifested with First Person Shooter Titles or Role Playing Titles that have a “mouse-look” feature. The issue was that the mouse input received from the touch panel was absolute with respect to a point on the display and thus in the game environment. This made the touch screen almost useless as a “mouse-look” device. The eventual fix was to filter out the mouse inputs by intercepting the input thread for the game. This allowed Gameplay to emulate mouse input via an on-screen control such as a joystick for the “mouse-look” function. It took a while to tune the joystick responsiveness and dead zone to feel like a mouse, but once that was done, everything worked beautifully. You can see this fix in action on games like *Fallout: New Vegas* or *The Elder Scrolls: Skyrim*. 
Vetting Titles for Touch Gaming
Ideum spent significant amounts of time tuning the virtual controllers for optimal gameplay. There are several elements of a game that determine its suitability for use with Gameplay. Below are some general guidelines that were developed for which types of games work well with Gameplay:

<table>
<thead>
<tr>
<th>Good</th>
<th>Better</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Role Playing Games (RPG)</td>
<td>- Simulation</td>
<td>- Platformers</td>
</tr>
<tr>
<td></td>
<td>- Fighting</td>
<td>- Side Scrollers</td>
</tr>
<tr>
<td></td>
<td>- Sports</td>
<td>- Action and Adventure</td>
</tr>
<tr>
<td></td>
<td>- Racing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Puzzles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Real Time Strategy (RTS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Third Person Shooters</td>
<td></td>
</tr>
</tbody>
</table>

While playability is certainly an important aspect of vetting a title for use with Gameplay, the most important criteria is stability. Some titles will just not work with the hooking technique, input injection, or overlay technology. This can happen for a variety of reasons, but most commonly it is due to the game title itself monitoring its own memory space or input thread to check for tampering. While Gameplay itself is a completely legitimate application, it employs techniques that can also be used for the forces of evil, so unfortunately some titles that are sensitive to these techniques will never work unless natively enabled for touch.

User Response
While still early in its release, Gameplay 1.0 has developed some interesting user feedback in regard to touch gaming on a PC. There are already some clear trends to the user feedback being received. At a high level, it is clear that everyone universally loves being able to customize the touch interface for games. The remaining feedback focuses on personalizing the gaming experience in a few key areas:

- Many virtual controllers are not ideal for left-handed people. This was an early change to many of the published virtual controllers.
- Button size and position is the most common change, so much so that Ideum is considering adding an automatic hand-sizing calibration in a future Gameplay release.
- Many users prefer rolling touch inputs vs. discrete touch and release interaction.

We expect many more insights to reveal themselves as the number of user-created virtual controllers increases.
**Conclusion**

GestureWorks Gameplay brings touch controls to your favorite games. It does this via a combination of a visual overlay and supports additional interactions like gesture, accelerometers, and 2-1 transitions. What has been most interesting in working on this project has been the user response. People are genuinely excited about touch-gaming on PCs, and ecstatic they can now play many of the titles they previously enjoyed with touch.

**About the Authors**

Erik Niemeyer is a Software Engineer in the Software and Solutions Group at Intel Corporation. Erik has been working on performance optimization of applications running on Intel microprocessors for nearly fifteen years. Erik specializes in new UI development and micro-architectural tuning. When Erik is not working, he can probably be found on top of a mountain somewhere. Erik can be reached at erik.a.niemeyer@intel.com.

Chris Kirkpatrick is a software applications engineer working in the Intel Software and Services Group supporting Intel graphics solutions on mobile platforms for the Visual & Interactive Computing Engineering team. He holds a B.Sc. in Computer Science from Oregon State University. Chris can be reached at chris.kirkpatrick@intel.com.

**Resources**

https://gameplay.gestureworks.com/

**Related Articles**

*How to Write a 2 in 1 Aware Application:* http://software.intel.com/en-us/articles/how-to-write-a-2-in-1aware-application


**Related Video**

Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document or other Intel literature may be obtained by calling 1-800-548-4725 or go to: http://www.intel.com/design/literature.htm

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel, the Intel logo, and Ultrabook are trademarks of Intel Corporation in the U.S. and/or other countries.

Copyright ©2014 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.