

Google Summer of Code 2008

Haptic Application Programmable Interface for Simple Directmedia Layer



2008
Edgar Simó

Table of Contents

| | |
|--|----|
| About the Author..... | 3 |
| Abstract..... | 4 |
| Introduction..... | 5 |
| What is Google Summer of Code (GSoC)?..... | 5 |
| What is a haptic device?..... | 5 |
| Why is a haptic API important?..... | 5 |
| Project Details..... | 6 |
| Implementation..... | 7 |
| Technical Needs..... | 7 |
| OS API investigation..... | 7 |
| Abstraction..... | 8 |
| Conclusions..... | 9 |
| Results..... | 9 |
| Experience..... | 9 |
| Appendix I: Glossary..... | 10 |



About the Author

Edgar Simó is currently a 3rd year Industrial Engineering student at the UPC-ETSEIB (Universitat Politècnica de Catalunya – Escola Tècnica Superior d'Enginyeria Industrial de Barcelona). He's been using open source for over 7 and enjoys coding in C.



Abstract

Project consisted of making an interface for programmers to interact with haptic (force feedback) devices using the Open Source library Simple Directmedia Layer (SDL). This will allow programmers to work with haptic devices with a platform independent API.

Introduction

What is Google Summer of Code (GSoC)?

Google Summer of Code or GSoC for short is a global open source program that allows university students to get paid for working on open source. This is very important because while open source developers usually don't get paid, it is very important for many other companies including Google to have. The program allows university students that would otherwise be working at jobs not related to their fields to work on real world projects giving them experience and helping improve the entire open source movement. Google Summer of Code is considered part-time work for 3 months.

What is a haptic device?

Haptic is the greek word meaning contact or touch. A haptic device is a device that interfaces with the user's sense of touch. A simple example would be a controller for a console that vibrates when something happens to the player. More complex examples would be joysticks for flight simulation.

Why is a haptic API important?

Haptic devices have recently reached popularity with the modern gaming consoles. They allow to increase the immersion of the user greatly with simple vibrations and rumbles.

Touch is a more primitive than vision or hearing in the sense that the data transmitted with a touch is much simpler. You don't need to process at all, you just feel the input and react on it. Audio and video on the other hand need active processing to be able to understand it. You have to listen to the sound or watch the message then decipher what it means to be able to react on the input. In the area of gaming this is translated to an immediate feeling of immersion with the environment the game is presenting.

Haptic is not only gaming though. Two notable uses of haptic technology would be the joystick in planes which help transmit the way the aerodynamic forces are affecting the plane. This was traditional felt because of the way the control was implemented, later it disappeared with the implementation of servo motors to help control. It was noticed that the lack of information affected the pilot's ability to fly negatively. Since then the aerodynamic forces are emulated with haptic technology.

Another example of important real world haptic technology implementation would be



the field of robotics and especially surgical robots. The controlling doctor can actually feel the incision as the robot is cutting the patient. This information helps him not damage the patient and do a successful operation.

The technology behind haptic interfaces is growing and you can now find it in many devices making it more and more important to take into account ever day.

Project Details

This project consists in creating an open source operating system independent application programmable interface for haptic devices for the library SDL. The end goal is to allow any programmer using the SDL library to be able to control haptic devices on any system that is supported by SDL. This will allow programmers to write less code and have it working on more operating systems without having to know the details of each implementation. This will help programmers to make more haptic aware applications to be able to get the most out of new technologies.

The implementation will be done in C and will form part of SDL 1.3. It will be release as open source under the GPLv2 license and a commercial license for platforms which can't comply to the GPLv2 like the iPhone. The entire implementation will be done during the 2008 summer under the mark of the Google Summer of Code program and will be mentored by Ryan "icculus" Gordon.

Implementation

Technical Needs

The needs of the implementation are simple. It has to expose as much of the haptic API the underlying operating system has to give. The implementation must also be simple and straightforward to use. Last it must have adequate documentation for a programmer to be able to use it easily.

OS API investigation

When starting the OS API investigation, it was first needed to find out which operating systems supported by SDL had haptic functionality. The following three were found:

1. GNU/Linux
2. Mac OS X
3. Windows

Next it was needed to find a common ground between the haptic aware OS to be able to design an OS-independent API around that. It wasn't too hard since it seems like the three OS implementations shared the same root. Windows had the first haptic implementation, which was then pretty much copied into Mac OS X. Finally GNU/Linux did it's own implementation which was the most different, but still very similar. An interesting thing to note is that as each OS copied the Windows API, they cleaned it up and made it better.

The basics of haptic usage is as follows:

1. Open haptic device.
2. Upload haptic effects to the device.
3. Play effects until finished with device.

There are also more advanced features supported by force feedback which will be implemented as allows. On systems which don't support features emulation will be done if possible.

Once the details of each operating system's implementation was studied, the next step was to begin the abstraction.

Abstraction

The base abstraction consisted of exposing the common API to all operating systems. The data is fed in a platform-neutral way and is then converted by each backend to the proper data structures to feed the operating system. Common functionality includes:

1. Detecting devices.
2. Opening/closing devices.
3. Effect types.
4. Detecting supported effects.
5. Uploading supported effects.
6. Playing supported effects.
7. Destroying supported effects.

After the common functionality was implemented work was done on more operating system specific functionality like:

1. Pausing/resuming effects.
2. Custom effects.

Overall all the process was pretty straightforward since the implementations on the different operating systems were very similar.



Conclusions

Results

Project was a success. The SDL haptic API is already merged into the 1.3 branch and is pending on the release. It fully supports all the targeted operating systems and exposes all the needed API in a cross platform way to the programmer.

Experience

Overall I found this project to be a rewarding experience. I've always been very interested in human machine interaction, and enjoyed the opportunity to play around with haptic devices.

The actual programming was very smooth. The only real issues cropped up when I had to do the backends for Windows and Mac OS X. As a GNU/Linux programmer I had quite a few issues adapting to the other OS. This led to some slow downs and other issues that were solved. I was gratefully surprised with the similarity between API and that didn't pose a problem.

The project was also something I looked forward as a programmer, since I do use the SDL library in some of my projects. The ability to be able to interface with haptic devices will be interesting and something I'll soon implement in some of my projects.

With such a great experience it's no surprise that I look forward to repeating the Google Summer of Code next year.

Appendix I: Glossary

- **API:** Application Programmable Interface. Interface for programmers to use when coding with the application.
- **GNU/Linux:** An open source operating system.
- **GPLv3:** The GNU Public License. An open source viral license which forces the developer to always give the source to any user which asks for it.
- **Open Source:** A movement which consists in releasing computer programs with all the code needed to be able to modify the program in every way.
- **OS:** Operating System.
- **SDL:** Simple Directmedia Library. An open source library designed to allow platform independent access to basic operating system functionality.