Using CUDA to accelerate an adaptive game controller

Leonardo Torok, Esteban Clua, Anselmo Montenegro, Daniela Trevisan
ltorok@ic.uff.br, esteban@ic.uff.br, anselmo@ic.uff.br, daniela@ic.uff.br

Federal Fluminense University, Niterói, Rio de Janeiro, Brazil.

The **adaptive game controller** is a **novel concept** that aims to introduce new ways to interact with videogames, **allowing developers to design the joystick used to play** through a simple API provided by our solution. A key component is the **K-means clusterization algorithm**, used to fine tune the controller interface according to the **user's touches** during the gameplay session. Since it is **NP-hard**, K-means is a challenging problem. Currently, our controller is implemented in JAVA for the Android operating system and the machine learning routines are executed by the **mobile device's CPU**, limiting the amount of touches that can be considered. With **CUDA**, we will introduce the **next evolutionary** step in our controller, increasing the amount of points evaluated to include **all touches, in real time**, using the **GPU available on the computer** that is running the game.

Concept

To optimize the interface for each user's needs, the adaptive controller uses the K-means algorithm to evaluate the touches on the mobile device's screen, locating interaction patterns close to each virtual button. Each one of these patterns will be grouped in a cluster, with a centroid that represents the average position for it and the correct position for that button. After that, **each one will be smoothly moved towards the closest centroid**, correcting the interface.

The mobile CPU performance limited the controller to evaluate just some dozens of touches, hardly an ideal numbers. To solve this issue, this work proposes to use a desktop-grade GPU to **accelerate the calculations using CUDA**, allowing to **clusterize all touches** and providing a better representation of an **optimal joypad**.

Solution

Our controller uses a **mobile device to display a virtual joypad to control a game on a regular PC**, with all communication performed by network sockets. Each button press on the device is sent to a desktop application that generates the corresponding key event, performing the correct action in the game.

To accelerate and improve the adaptations, now the controller will also **send each touch coordinates** to the desktop app and the K-means algorithm will be performed with CUDA, using the approach described by Giuroiu (2014) and Liao (2015). After that, **the centroids are sent to the mobile device** and the changes are executed on the interface.

During simple 5 minutes test sessions, **users easily touched the screen 700 to 1500 times**. Instead of analyzing just a small subset of points, **CUDA allowed us to evaluate all points not only from the current gameplay session, but also the entire history of previous interactions**, creating a natural interface that evolves with the user.

References

TOROK, L.; PELEGRINO, M.; TREVISAN, D. G.; CLUA, E.; and MONTENEGRO, A. A Mobile Game Controller Adapted to the Gameplay and User's Behavior Using Machine Learning, 2015. In *Entertainment Computing-ICEC 2015* (pp. 3-16). Springer International Publishing.

LIAO, W. Parallel K-Means Data Clustering, 2005. Available at: http://users.eecs.northwestern.edu/~wkliao/Kmeans/index.html. Accessed in October 1st, 2015

GIUROIU, S. CUDA K-means Clustering, 2010. UC Berkeley. Available at: http://serban.org/software/kmeans/. Accessed in October 14th, 2014.

Results

To determine the real improvements, we compared the performance of the controller running K-means on the mobile CPU and both desktop CPU and GPU. The tests were performed on a Moto X Play smartphone, with an octa-core CPU and on a desktop PC with an AMD FX-8350 CPU (4.5 GHz) and a **NVIDIA GeForce GTX 970**.

For larger subsets, the GPU version achieved significant **speedups ranging between 3.7 and 12,6X**, compared with the desktop CPU version. The improvements are even more dramatic when compared to the previous version, that used a mobile CPU. **The GPU version would allow to use larger subsets in real time cases**, where the controller has to **evaluate the clusters in 1 or 2 seconds** to be able to react fast enough to keep up with the user's needs.

Points	K	Mobile	Desktop	
		CPU (s)	CPU (s)	GPU (s)
1,000	5	0.193	**0.015**	0.234
	10	0.310	**0.016**	**0.016**
	15	0.311	**0.015**	0.016
10,000	5	3.630	0.343	**0.094**
	10	11.319	0.343	**0.062**
	15	17.421	0.515	**0.062**
100,000	5	57.706	3.510	**0.687**
	10	206.220	7.769	**0.889**
	15	861.823	29.672	**2.355**

Instituto de computação
www.ic.uff.br

MEDIALAB

This project was financially supported by CAPES-Brazil and CNPq-Brazil.
Special thanks to Victor Machado (poster revision).