# Parallel Code

To get the code working in the most efficient manner on the PS3 we had to utilize the 6 synthetic processing units found in any PS3. The idea is simple, break down each task in the x86 version into 6 parts that can run in parallel across all SPUs. To make interaction with all SPUs easy we wrote a spu\_manager class which would handle the interaction for us.

## SPU Manager

The spu manager deals with loading a SPU program to run on each SPU independently. Upon the manager’s construction it reads system info so we know how many usable SPUs are available. Once we have chosen how many SPUs we want to utilize we pass the location of an SPU executable that we want to be loaded across the SPUs. We then pass a struct to the SPU manager which then made available to the SPU program when it starts. The struct contains information on the work the SPU should do.



Because running an SPU program via spe\_context\_run is a blocking operation the SPU manager starts all SPU program instances on a separate thread to ensure they all launch at the same time (roughly). We can then tell when all SPU programs have finished by performing a join. In our program we call these methods for each program we want to run, in our case 4 (blur, sobel, detection, overlay)

## SPU Program

From the struct passed to the program on start-up we take a subset of the bitmap to work with before putting it back so it can be written to a file or used in another SPU program. We use MFC (Memory Flow Controller) for this, reading the data chunk by chunk in the largest chunks possible till we have acquired all data required to perform the task. Where available we also take advantage of SIMD instructions on the SPU to make sure we can perform multiple calculation simultaneously. Notably we use this in the blur SPU program and the sobel SPU program as there are is matrix multiplication in these functions.

## PPU Program

Even though the PPU program acts more as a controller for SPUs on the system we do take advantage of SIMD instructions to take our grayscale bitmap and save it as a 24-bit bitmap.