This is miscellaneous PSP tutorial m1

Introduction

Hi everybody,

This is a short tutorial on creating a timer. This is only a tutorial to show you what is possible with timers. It will result in a program showing a timer which can be reset. There will also be given some information about other functions.

We will use the code from tutorial 1: The game loop as the basis for this tutorial.

Timer object

Since we are creating a timer we need to have a timer object. In this paragraph I will explain the object. The timer we will make is a timer that counts up. You can easily convert this in any other way.

Timer.h

```
#ifndef TIMER_H_
#define TIMER_H_

#include "time.h"

class Timer {

protected:
   time_t TimeNow_;
   time_t TimeStart_;
   bool timerStarted_;

public:
   void startTimer();
   void resetTimer();
   int getTimer();

};

#endif
```

Well first we have our header guard. We include the time.h header because we need the time_t objects. The boolean is used to check if the timer has started or has not. The three functions speak for themselves. The function getTimer returns the seconds elapsed.

Lets take a look at the source, that the interesting stuff.

Timer.cpp

```
#include "Timer.h"
#include <psputils.h>
```

Here we include our own header and the psputils header for using the sceKernelLibcTime function.

```
void Timer::startTimer() {

   // getting current time
   sceKernelLibcTime(&this->TimeStart_);

   // set bool timer is started
   this->timerStarted_ = true;

}
```

This function first retrieve the current time and saved it in the time_t object TimeStart_. After this we set the boolean timerStarted_ to true. We can now check with that boolean if the timer has started.

```
int Timer::getTimer() {
   if(!this->timerStarted_)
      return 0;

   // getting current time
   sceKernelLibcTime(&this->TimeNow_);

   // return the time now minus the start time, resulting in difference time.
   return ((int)this->TimeNow_ - (int)this->TimeStart_);

}
```

This function returns the seconds elapsed after the startTimer function has been called. The first part checks whether the timer has started. If not it returns 0. The second part retrieves the current time and saves it in TimeNow. With this we can check how much time has elapsed since the startTimer function has been called. We substract the starting time from the time now and that results in the time elapsed.

```
void Timer::resetTimer() {
   this->timerStarted_ = false;
}
```

The resetTimer function only sets the timerStarted to false so that the getTimer function returns 0. This is not a very nice way of doing things but for now it suffice.

In order to use the timer we have to implement it in the gameApp.

GameApp

We will use the timer in the GameApp. We will use the buttons square and cross to start and reset the timer. Debug information is shown on the screen to see what the timer does.

GameApp.h

```
   #include "Timer.h"
```

```
Timer* timer;
int buttonCross;
int buttonSquare;
```

The first line includes our newly made class. The other three are protected variables. The first is a pointer to the new class so that we can use it. The last two variables are used for the controls. (also see Tutorial m5: Controls revisited)

GameApp.cpp

```
// load timer object
timer = new Timer();
// start the timer
timer->startTimer();
```

This code first creates a timer object on the heap. In the line after that it calls the startTimer function discussed in the previous paragraph. We need to put this code in the Load() function of the GameApp.

```
delete timer;
```

We need to delete the timer from the heap off course.

```
if (pad.Buttons != 0){
   if (pad.Buttons & PSP_CTRL_CROSS){
      if(buttonCross == 0) {
         buttonCross = 1;
         timer->resetTimer();
      }
   }
   if (pad.Buttons & PSP_CTRL_SQUARE){
      if(buttonSquare == 0) {
         buttonSquare = 1;
         timer->startTimer();
      }
   }
}
else {
   buttonCross = 0;
   buttonSquare = 0;
}
```

Here is part of the control function. We have added the timer reset and start calls when we press a button. Not much interesting. The only thing that rests us is that we need to print some info to the screen.

```
void GameApp::Render() {
   // render information.
   pspDebugScreenSetXY(10, 5);
   pspDebugScreenPrintf("%d seconds elapsed\n", timer->getTimer());
   pspDebugScreenSetXY(4, 15);
   pspDebugScreenPrintf("Press Cross to reset timer.\n");
   pspDebugScreenSetXY(4, 20);
   pspDebugScreenPrintf("Press Square to start timer.\n");
   sceDisplayWaitVblankStart();
   sceGuSwapBuffers();
}
```

Here we print some info on screen. It renders the current timer on screen and what controls yo ucan use. If you compile it press the button

and see what happens. The pspDebugScreenSetXY(x, y) sets the position where the print function start to write. With this I just made a layout, not one that can win a beauty contest but that does not matter.

MakeFile

The makefile for tutorial m1

```
TARGET = out
OBJS = $(wildcard *.cpp) $(wildcard *.c)

INCDIR =
CFLAGS = -O2 -G0 -Wall -g
CXXFLAGS = $(CFLAGS) -fno-exceptions -fno-rtti -g
ASFLAGS = $(CFLAGS)

LIBDIR =
LDFLAGS =
LIBS = -lc -g -lpspgum -lpspgu -lstdc++ -lm -lpsppower

PSPSDK=$(shell psp-config --pspsdk-path)
include $(PSPSDK)/lib/build.mak
```

Compile and see what happens :D

Source files

Download the source files.