

He he

Following with it here it is something perhaps better than StreamRaster plugin system.

In StreamRaster it is done a "new" any time a function is called. But is is possible to do it only once:

```
class StaticPlugin {
public:
    StaticPlugin();
    ~StaticPlugin();
    bool Init(const char *name);

    template <class T>
    static void Register(const char *name) {
        PluginData& x = Plugins().Add();
        x.name = name;
        x.New = New<T>;
        x.Delete = Delete<T>;
    }

protected:
    void *data;

private:
    String name;

    template <class T> static void *New() {return new T;};
    template <class T> static void Delete(void *p) {delete static_cast<T *>(p);};

    struct PluginData {
        String name;
        void *(*New)();
        void (*Delete)(void *);
    };

    static Array<PluginData>& Plugins();
};
```

The plugin class is stored in a void *data, with its "new" and "delete", in Register() (thanks to templates).

This way, if a plugin is used, StaticPlugin uses the right "new". And in ~StaticPlugin it is used the right "delete" .

