ParaFrame Plugin Architecture

(v0.0.1.dev0)

- Plugins are comprised of a core plugin module in the form of a dynamic library and a plugin definition file which can be in JSON for basic functionality, or JavaScript for more robust additions.
- ParaFrame is cross-platform with support for Windows (DLL) and Linux (SO).
- While ParaFrame itself is a Qt focused framework, plugins do not need to use or require Qt.
- Plugins can be written in multiple languages.
- Starter plugin templates will be provided for C++, C#, and Python.
- Plugins support combinable interfaces representing groups of functionality. New interfaces can be added to the framework in the future while maintaining backwards compatibility.
- Plugins are comprised of calls made to the plugin from the framework, and calls made from the plugin to the framework.
- Add calls to and from a plugin come baked-in with Qt Signal/Slot linkage courtesy of ParaFrame.
- Each plugin is cordoned into a safe area with respect to the consumer of their events. Through IPC and strict separation, the framework is designed to not go down when a plugin does.
- Every plugin comes baked-in with a pulse signal for checking plugin responsiveness which can be overriden with other timing values if needed.

- Authors of plugins can choose from multiple interfaces from which they can combine to make more complex devices.
- Each interface gaurantees a set of functions that can be called to and from the plugin.

```
interfaces v1a

IBasePara
IKeyboard
IMouse
IMotionControl
IAnalog
ISwitchable
IGenericSensor
IMessaging
IAudioInput
IAudioOutput
IStaticCamera
IStreamingCamera
IGamepad
IColorPresenter
```

Plugin Module Example (DLL)

gamepad.dll

```
#include “paraplugin.h”

class PARA_FRAME_SHARE Plugin
  : public ParaPlugin, public IBasicGamepad, public IAccelerometer,
public ITouchpad
{
  /* from ParaPlugin */
  //baked into interfaces above but can be overriden
  std::stringlist<std::string> supportedFunctions() //plugin<<fw
  void pulseStart(int) //plugin ◄ fw
  void pulseStop () //plugin ◄ fw
  void pulse() //plugin ► fw
  void customPropertyChanged() //QtScript Engine instantiation/runs

  //MUST override
  void deviceEnable() //plugin ◄ fw
    {//enable device}
  void deviceEnabled(bool success) //plugin ► fw

  //optional
  void faultOccured(int, std::string=””) //plugin ► fw

  /* from IBasicGamepad */
  void buttonPressed(int, pressure_state) //plugin ► fw
  void buttonDepressed(int) //plugin ► fw
  void axisLeft[X|Y]Changed(double) //plugin ► fw
  void axisRight[X|Y]Changed(double) //plugin ► fw

  /* from IAccelerometer */
  void accelerometerEvent(IAccelerometer::Event) //plugin ► fw

  /* from ITouchpad */
  void touchpadEvent(ITouchpad::Event) //plugin ► fw
}
```

- Using the integrated ParaExplorer, each plugin’s functionality can be explored and it’s functions executed for testing and development purposes.
- For maximum flexibility, ParaExplorer can be run either from a Plugin Manager library or from a standalone executable.

- Plugin definition files (*.para) describe a specific instance of a ParaFrame plugin.
- Both JSON and ECMAScript definitions support custom properties which can be tracked and “signaled” through Qt without the need for recompilation of the core plugin module (DLL, SO).
- Definition files support custom error/fault reporting to consumers of each plugin.
- Multiple definition files can utilize the same ParaFrame plugin module. For instance, two cameras can utilize the same ParaPlugin.

Plugin Definition Example (JSON)

stage_gamepad.para

```
{
  “name”: “Stage Gamepad”,
  “plugin”: “gamepad.dll”,
  “description”: “Stage Controller”,
  “icon”: “pathToImage.png”,
  “fault_catalog”: [
    “FAULT_POWERLOSS”,
    “FAULT_AXIS_CALIBRATION”
  ],
  “custom_properties”: {
    “stick_acceleration_x”: 1.0,
    “stick_acceleration_y”: 0.6,
    “backlight_enabled”: false
  },
  “excluded_functions”: []
}
```

Plugin Definition Example (JavaScript)

stage_gamepad.para

```
//ParaFrame Boilerplate - begin
function ParaFramePlugin(name, plugin)
{
  //required
  this.name = name;
  this.plugin = plugin;

  //optional
  this.description = “”;
  this.faultCatalog = {}
  this.customProperties = {};
  this.excluded_functions = [];
}

ParaFramePlugin.prototype.toString =
  function() {
              return “ParaFramePlugin(name: “ + this.name + “)”;
}
//ParaFrame Boierplate - end

//Plugin Implementation - begin
var exportedPlugin = new ParaFramePlugin(“Stage Gamepad”, “gamepad.dll”);
exportedPlugin.description = “Stage Controller”;
exportedPlugin.faultCatalog = {
  “1”: “FAULT_POWERLOSS”,
  “2”: “FAULT_AXIS_CALIBRATION”
}
exportedPlugin.customProperties = {
  “stick_acceleration_x”: 1.0;
  “stick_acceleration_y”: 0.6;
  “backlight_enabled”: false;
}
```

Para [Frame]work